



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



ЗБОРНИК РАДОВА ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке - зборници

Година: XXXIII

Број: 2/2018

Нови Сад

Едиција: „Техничке науке – Зборници“

Година: XXXIII

Свеска: 2

Издавач: Факултет техничких наука Нови Сад

Главни и одговорни уредник: проф. др Раде Дорословачки, декан Факултета техничких Наука у Новом Саду

Уредништво:

Проф. др Раде Дорословачки

Проф. др Драгиша Вилотић

Проф. др Срђан Колаковић

Проф. др Владимир Катић

Проф. др Драган Шешилија

Проф. др Миодраг Хаџистевић

Проф. др Растислав Шостаков

Доц. др Мирослав Кљајић

Доц. др Бојан Лалић

Доц. др Дејан Убавин

Проф. др Никола Јорговановић

Доц. др Борис Думнић

Проф. др Дарко Реба

Проф. др Ђорђе Лађиновић

Проф. др Драган Јовановић

Проф. др Мила Стојаковић

Проф. др Драган Спасић

Проф. др Драгољуб Новаковић

Редакција:

Проф. др Владимир Катић, главни
уредник

Проф. др Жељен Трповски, технички
уредник

Проф. др Драган Шешилија

Проф. др Драгољуб Новаковић

Др Иван Пинђер

Бисерка Милетић

Језичка редакција:

Бисерка Милетић, лектор

Софија Рацков, коректор

Марина Катић, преводилац

Издавачки савет:

Савет за библиотечку и издавачку делатност ФТН,
проф. др Радош Радивојевић, председник.

Штампа: ФТН – Графички центар ГРИД, Трг Доситеја Обрадовића 6, Нови Сад

CIP-Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

378.9(497.113)(082)
62

ЗБОРНИК радова Факултета техничких наука / главни и одговорни уредник
Раде Дорословачки. – Год. 7, бр. 9 (1974)-1990/1991, бр.21/22 ; Год. 23, бр 1 (2008)-. – Нови Сад :
Факултет техничких наука, 1974-1991; 2008-. – илустр. ; 30 цм. –(Едиција: Техничке науке –
зборници)

Месечно

ISSN 0350-428X

COBISS.SR-ID 58627591

ПРЕДГОВОР

Поштовани читаоци,

Пред вама је друга овогодишња свеска часописа „Зборник радова Факултета техничких наука“.

Часопис је покренут давне 1960. године, одмах по оснивању Машинског факултета у Новом Саду, као „Зборник радова Машинског факултета“, а први број је одштампан 1965. године. Након осам публикованих бројева у шест година, пратећи прерастање Машинског факултета у Факултет техничких наука, часопис мења назив у „Зборник радова Факултета техничких наука“ и 1974. године излази као број 9 (VII година). У том периоду у часопису се објављују научни и стручни радови, резултати истраживања професора, сарадника и студената ФТН-а, али и аутора ван ФТН-а, тако да часопис постаје значајно место презентације најновијих научних резултата и достигнућа. Од броја 17 (1986. год.), часопис почиње да излази искључиво на енглеском језику и добија поднаслов «Publications of the School of Engineering». Једна од последица нарастања материјалних проблема и несрећних догађаја на нашим просторима јесте и привремени прекид континуитета објављивања часописа двобројем/двогодишњаком 21/22, 1990/1991. год.

Друштво у коме живимо базирано је на знању. Оно претпоставља реорганизацију наставног процеса и увођење читавог низа нових струка, као и квалитетну организацију научног рада. Значајне промене у структури високог образовања, везане за имплементацију Болоњске декларације, усвајање нове и активне улоге студената у процесу образовања и њихово све шире укључивање у стручне и истраживачке пројекте, као и покретање нових мастер и докторских студија, доносе потребу да ови, веома значајни и вредни резултати, постану доступни академској и широј јавности. Оживљавање „Зборника радова Факултета техничких наука“, као јединственог форума за презентацију научних и стручних достигнућа, пре свега студената, обезбеђује услове за доступност ових резултата.

Због тога је Наставно-научно веће ФТН-а одлучило да, од новембра 2008. год. у облику пилот пројекта, а од фебруара 2009. год. као сталну активност, уведе презентацију најважнијих резултата свих мастер радова студената ФТН-а у облику кратког рада у „Зборнику радова Факултета техничких наука“.

Поред студената мастер студија, часопис је отворен и за студенте докторских студија, као и за прилоге аутора са ФТН или ван ФТН-а.

Зборник излази у два облика – електронском на веб сајту ФТН-а (www.ftn.uns.ac.rs) и штампаном, који је пред вама. Обе верзије публикују се више пута годишње у оквиру промоције дипломираних инжењера-мастера.

У овом броју штампани су радови студената мастер студија, сада већ мастера, који су радове бранили у периоду од 15.09.2017. до 09.10.2017. год., а који се промовишу 27.01.2018. год. То су оригинални прилози студената са главним резултатима њихових мастер радова.

Известан број кандидата објавили су радове на некој од домаћих научних конференција или у неком од часописа. Њихови радови нису штампани у Зборнику радова.

Велик број дипломираних инжењера–мастера у овом периоду био је разлог што су радови поводом ове промоције подељени у три свеске.

У овој свесци, са редним бројем 2., објављени су радови из области:

- електротехнике и рачунарства.

У свесци са редним бројем 1. објављени су радови из области:

- машинства,
- грађевинарства,
- саобраћаја,
- архитектуре и
- мехатронике.

У свесци са редним бројем 3. објављени су радови из области:

- графичког инжењерства и дизајна,
- инжењерског менаџмента,
- инжењерства заштите животне средине,
- математике у техници,
- геодезије и геоматике,
- инжењерства третмана и заштите вода (TEMPUS),
- инжењерства информационих система и
- сценске архитектуре и дизајна.

Уредништво се нада да ће и професори и сарадници ФТН-а и других институција наћи интерес да публикују своје резултате истраживања у облику регуларних радова у овом часопису. Ти радови ће бити објављивани на енглеском језику због пуне међународне видљивости и проходности презентованих резултата.

У плану је да часопис, својим редовним изласком и високим квалитетом, привуче пажњу и постане довољно препознатљив и цитиран да може да стане раме-уз-раме са водећим часописима и заслужи своје место на СЦИ листи, чиме ће значајно допринети да се оствари мото Факултета техничких наука:

„Високо место у друштву најбољих“

Уредништво

SADRŽAJ

STRANA

Radovi iz oblasti: Elektrotehnika i računarstvo

1. Davor Nađ, MODBUS INDUSTRIJSKI KOMUNIKACIONI PROTOKOL	149-152
2. Marko Kovačević, INDUSTRIJSKI KOMUNIKACIONI PROTOKOL IEC 60870-5-104	153-156
3. Milan Mitrić, KONCEPT MODULARNOG PROTOKOL ANALIZATORA I NJEGOVA PRIMENA NA DNP3 PROTOKOL	157-160
4. Teodor Malbašić, Živko Bojović, DISTRIBUIRANI FAJL SISTEMI	161-164
5. Milan Radeta, KOMPOZICIJA SPOSOBNOSTI KARAKTERA RPG IGRE	165-168
6. Branislav Stanišić, PRAĆENJE I ANIMACIJA POKRETA TELA	169-172
7. Miloš Mičić, PREZENTACIJA REZULTATA DMS FUNKCIJA NA POJEDNOSTAVLJENOM PRIKAZU MREŽE .	173-176
8. Mihailo Drljača, OPTOELEKTRONSKI GENERATORI NASUMIČNIH BROJEVA REALIZOVANI U HV CMOS 0.35 μ m TEHNOLOGIJI	177-180
9. Nera Vlasisavljević, BCI BAZIRAN NA RAVNOTEŽNIM VIZUELNO EVOCIRANIM POTENCIJALIMA	181-184
10. Zorana Deurić, RAZVOJ ANDROID APLIKACIJA U PROGRAMSKOM JEZIKU KOTLIN	185-188
11. Hunor Kiš, MOBILNA APLIKACIJA ZA DISTRIBUIRANO PRIKUPLJANJE PODATAKA O WI-FI MREŽAMA	189-192
12. Nikola Vukadinov, Stevan Gostojić, SISTEM ZA PRIKUPLJANJE PODATAKA I PREDIKCIJU TRAJANJA VOŽNJE JAVNIM PREVOZOM	193-196
13. Miloš Marić, Branko Milosavljević, RAZVOJ VOĐEN TESTOVIMA	197-200
14. Pilip Pilipović, SISTEM ZA PRIKAZIVANJE PODATAKA O MUZIČKIM FESTIVALIMA	201-204

	STRANA
15. Stefan Karać, UPRAVLJANJE U SCADA SISTEMU POMOĆU DNP3 PROTOKOLA	205-208
16. Nemanja Popović, RAZVOJ ALATA ZA VERIFIKACIJU SCADA KONFIGURACIJE	209-212
17. Јелена Карага, РЕАЛИЗАЦИЈА КОМУНИКАЦИОНОГ ПРОТОКОЛА IССР У SCADA ОКРУЖЕЊУ	213-216
18. Бојан Вујиновић, КРЕИРАЊЕ ГЕНЕРИЧКИХ ИЗВЕШТАЈА О КОМПОНЕНТАМА ЕЛЕКТРО-ЕНЕРГЕТСКОГ СИСТЕМА	217-220
19. Goran Ćutić, Darko Marčetić, KONDOMAT SA UGRAĐENIM FUNKCIJAMA ZA DALJINSKI NADZOR	221-224
20. Aleksa Mirković, BLOCKCHAIN DISTRIBUIRANI SISTEMI I NJIHOVA PRIMENA U FARMACEUTSKOJ INDUSTRIJI	225-228
21. Marina Nenić, BLOCKCHAIN DISTRIBUIRANI SISTEMI I NJIHOVA PRIMJENA U INDUSTRIJI VIDEO IGARA ...	229-232
22. Стефан Немет, Дарко Марчетић, ДИГИТАЛНО УПРАВЉАЊЕ СИНХРОНИМ РЕЛУКАНТНИМ МОТОРОМ СА ДАВАЧЕМ ПОЛОЖАЈА НА ВРАТИЛУ	233-236
23. Vukan Turkulov, MODIFIKACIJA PBRC ALGORITMA SA PRIMENAMA U PROBLEMIMA ADAPTIVNE REGRESIJE	237-240
24. Stefana Jocić, DECENTRALIZOVAN SISTEM ZA DETEKCIJU KVAROVA ASINHRONIH MOTORA	241-244
25. Milica Repac, Vladimir Katić, OPTIMIZACIONA ANALIZA KRITERIJUMA INFRASTRUKTURE EV STANICA	245-248
26. Stanko Kuveljić, PREGLED NEURONSKIH MREŽA SA PRIMERIMA PRIMENE	249-252
27. Душан Вукославовић, Велимир Чонградац, Веран Васић, МОДЕЛ КЛИМА КОМОРЕ СА АКЦЕНТОМ НА УЛОГУ И УПРАВЉАЊЕ ВЕНТИЛАТОРОМ У ЊИМА	253-256
28. Mihajlo Fejsa, Veran Vasić, Đura Oros, PRIMENA GENETSKOG ALGORITMA ZA ODREĐIVANJE PARAMETARA EKVIVALENTNE ŠEME ASINHRONOG MOTORA	257-260
29. Miloš Ivković, Željko Trpovski, SISTEMI VIDEO NADZORA NOVE GENERACIJE	261-264
30. Marko Tot, SINHRONIZACIJA MULTIPLAYER VIDEO IGARA	265-268
31. Đorđe Novaković, IOT MERNO-INFORMACIONI SISTEM ZASNOVAN NA HEXIWEAR PLATFORMI U OKRUŽENJU PAMETNE KUĆE	269-272
32. Aleksandar Manasijević, ALGORITMI ZA GENERISANJE LAVIRINATA U VIDEO IGRAMA	273-276
33. Nikola Todorović, SOFTVERSKA PODRŠKA ZA PREDIKCIJU KOLIČINE VLAGE U ZEMLJIŠTU U SISTEMIMA ZA NAVODNJAVANJE	277-280
34. Vladimir Ivković, ALGORITMI ZA ANALIZU VREMENSKIH SERIJA NA PRIMERU ISPITIVANJA DINAMIKE VLAŽNOSTI ZEMLJIŠTA U POLJOPRIVREDI	281-284
35. Aleksandar Bošnjak, WINDOWS PRESENTATION FOUNDATION CODE GENERATOR FOR SOFTWARE PRODUCT LINES	285-288

	STRANA
36. Aleksandar Lukić, PRIMENA "ROSLYN" KOMPJILERSKE PLATFORME I PROŠIRENOG XML PARSERA NA SOFTVERSKO PROIZVODNE LINIJE	289-292
37. Nenad Todorović, ARHITEKTURA GENERATORA KODA ZA SOFTVERSKO PROIZVODNE LINIJE	293-296
38. Jelica Kapetina, GEJMING U SLUŽBI OČUVANJA ZDRAVLJA	297-300
39. Aleksandar Josić, RAZVOJ RAČUNARSKE IGRE ZA STICANJE OSNOVNIH PROGRAMSKIH VEŠTINA	301-304
40. Marko Razumenić, ŽIVOTNI CIKLUS MODELA PODATAKA ELEKTROENERGETSKOG SISTEMA	305-308
41. Никола Вукајловић, Драган Милићевић, РЕКОНСТРУКЦИЈА ФАЗНИХ СТРУЈА ШЕСТОФАЗНЕ АСИНХРОНЕ МАШИНЕ КОРИШЋЕЊЕМ СТРУЈА ЈЕДНОСМЕРНОГ МЕЂУКОЛА	309-312
42. Simona Đurić, ADAPTACIJA DUBOKE NEURONSKE MREŽE NAMENJENE SINTEZI GOVORA NA KARAKTERISTIKE POJEDINAČNOG GOVORNIKA	313-316
43. Milan Kostić, VIZUELIZACIJA AŽURIRANJA MODELA PAMETNIH MREŽA	317-320

MODBUS INDUSTRIJSKI KOMUNIKACIONI PROTOKOL

MODBUS INDUSTRIAL COMMUNICATION PROTOCOL

Davor Nađ, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Ovaj rad predstavlja primer primene Modbus industrijskog komunikacionog protokola za akviziciono upravljački sistem namenjen radu u realnom vremenu. Nakon prikaza protokola, predstavljen je koncept i implementacija programske podrške za praćenje saobraćaja između SCADA sistema i RTU stanica, za navedeni protokol.

Abstract – This paper represents an example of use of the Modbus communication protocol for a SCADA system, used in realtime. After an overview of the Modbus protocol, concept and implementation of traffic sniffing between SCADA system and RTU stations, using this protocol, were presented.

Ključne reči: SCADA, ISO/OSI, Modbus, Protokol analizator, MEF, MVVM

1. UVOD

Razvojem i unapređivanjem proizvodnih procesa u industriji javili su se zahtevi za boljim i pouzdanijim nadzorom i upravljanjem tim procesima. Vrlo značajan aspekt svih akviziciono upravljačkih sistema namenjenih upravljanju industrijskim postrojenjima u realnom vremenu jeste međusobna komunikacija njihovih komponenti. Zahvaljujući tehnološkom napretku u oblasti računarstva i informacionih tehnologija, uz pomoć računara se upravlja industrijskim postrojenjima. Automatizacija proizvodnih (tehnoloških) funkcija postrojenja realizuje se programski, posredstvom systemske i aplikativne programske podrške.

Osnovni cilj SCADA sistema je obezbeđenje efikasnog nadzora i upravljanja nad proizvoljnim fizičkim procesom, korišćenjem digitalnih računarskih komponenti. U ovom radu je predstavljena ideja rešenja za praćenje komunikacije između SCADA sistema i RTU jedinica, tj. protokol analizatora. Predstavljeno je rešenje zajedno sa teorijskim osnovama SCADA sistema, komunikacionim protokolima, Modbus protokolom, kao i korišćenim tehnologijama potrebnim za implementaciju samog programskog rešenja. Svaka razmenjena poruka (komanda, poll, itd.) prikazana je u *human readable* formatu za Modbus protokol, tako da operateru/korisniku bude približno koje poruke SCADA sistem razmenjuje.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Pavlica, vanr. prof.

2. SCADA SISTEMI I KOMUNIKACIONI PROTOKOLI

2.1 SCADA sistemi

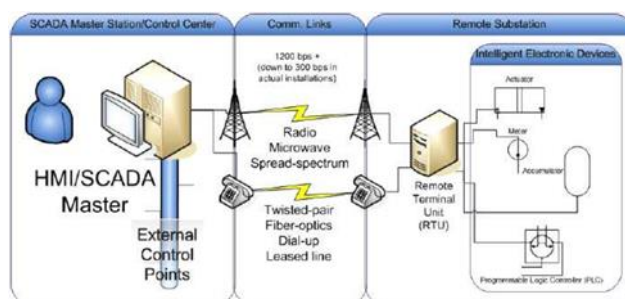
SCADA predstavlja sistem za merenje, praćenje i kontrolu industrijskih sistema. Svaki proces u industriji koji ima smisla automatizovati je odličan kandidat za primenu SCADA sistema i mreža. Osnovni cilj SCADA sistema je obezbeđenje efikasnog nadzora i upravljanje nad proizvoljnim fizičkim procesom uz oslonac na korišćenje digitalnih računarskih komponenti. Zahvaljujući brzom tehnološkom napretku u oblasti mikroelektronike i računarstva, kao i pojava jeftinih mikroprocesora velike procesne moći i operativne memorije, omogućila je široku primenu SCADA sistema u najrazličitijim proizvodnim delatnostima [1].

SCADA mreža se sastoji od jednog ili više MTU (*Master Terminal Unit*) koji su zapravo računarske stanice opremljene odgovarajućim softverom i operativnim sistemom. Ove stanice operatori koriste za praćenje i kontrolu jedne ili više RTU (*Remote Terminal Unit*). RTU je takođe računarski uređaj koji je obično namenjen za rad u industrijskim uslovima. Njegov zadatak je da prikuplja informacije sa raznih digitalnih i analognih senzora i da prosleđuje komande uređajima koji na neki način menjaju stanje upravljanog sistema (razni generatori, step motori, ventili i sl.). Često se za njihovu realizaciju koriste razni tipovi PLC (*Programmable Logic Controller*) [2].

2.1.1. Elementi SCADA sistema

Tri glavne komponente SCADA sistema (koje su prikazane na slici 2.1) su:

- Glavna stanica (*Master station*)
- Komunikaciona infrastruktura
- Udaljene terminalne jedinice (PLC ili RTU)



Slika 2.1. Komponente SCADA sistema

Termin „glavna stanica“ se odnosi na servere i na softver za komunikaciju sa uređajima na terenu (PLC, RTU, senzori i sl.), prikupljanje podataka sa njih i obradu. On takođe treba da prikaže operateru koji se nalazi u kontrolnoj sobi prikupljene podatke i odgovarajuće

statistike u obliku dijagrama. Glavna stanica predstavlja centralno mesto SCADA sistema [3].

Glavna stanica mora da komunicira sa RTU jedinicama koji su udaljeni od centralne lokacije. SCADA sistemi mogu imati i po nekoliko hiljada RTU-a. Sistem za komunikaciju obezbeđuje prenos informacija između udaljenih stanica i dispečerskog centra. Komunikacija između udaljenih stanica i centralne stanice, kao i između samih centralnih stanica, odvija se preko komunikacionog medijuma u zavisnosti od mogućnosti i zahteva korisnika [4].

Udaljene stanice su nezavisni mikroprocesorski uređaji koji obezbeđuju komunikaciju između merne opreme, izvršnih organa i centralne stanice. Prikupljanje podataka počinje na nivou merno-upravljačke opreme, poput RTU-a ili PLC-a. RTU predstavlja mikroprocesorski udaljeni uređaj koji se povezuje sa senzorima i drugim mernom opremom, pretvara merenja sa senzora u digitalne signale i šalje ih ka centralnoj stanici. Merna oprema obuhvata različite instalirane senzore koji mere fizičke veličine (sila, temperatura, broj obrtaja i dr.) i pretvaraju ih u električne (digitalne) signale, tj. u oblik koji je čitljiv udaljenim stanicama. Pomoću ove opreme se dobijaju informacije o funkcionisanju procesa i na osnovu njih se može uticati na poboljšanje poslovnih performansi. Podaci sa merne opreme se prenose ka centralnoj stanici, a RTU uređaji mogu primati digitalne komande poslate od strane centralne stanice (nadzornog sistema) [4].

2.2. Komunikacioni protokoli

Komunikacionim protokolom označava se skup pravila i procedura koji kontrolišu tok komunikacije između serijski spregnutih stanica u okviru računarske mreže. Postoji skup pravila koji definišu sintaksu, semantiku i sinhronizaciju komunikacije.

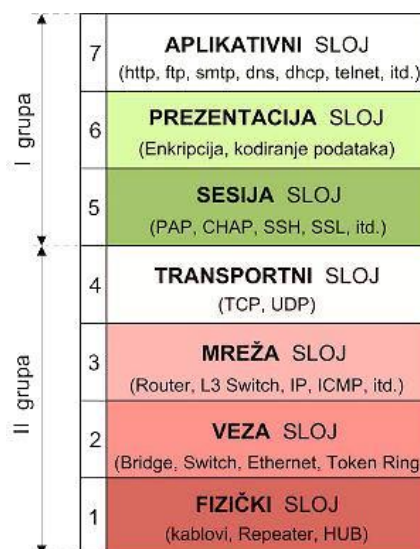
Posredstvom komunikacionih protokola obezbeđuje se uspešna interakcija između udaljenih procesa. Protokol definiše format i redosled poruka koje se razmenjuju, kao i svaku akciju koja se inicira prijemom određene poruke. Sam protokol se opisuje dijagramom promene stanja, ili korišćenjem nekog od viših programskih jezika. Osnovne funkcije komunikacionih protokola su kontrola grešaka i upravljanje tokom podataka u mreži. Kontrola grešaka obezbeđuje pouzdan prenos i integritet poruka. Primarni cilj upravljanja tokom je izbegavanje zagušenja u saobraćaju, i deljenje komunikacionih resursa između više korisnika [1].

2.2.1. ISO / OSI referentni model

Temelj savremenih komunikacionih sistema je standardni ISO/OSI (*Open Systems Interconnection Basic Reference Model*) – referentni model za otvoreno povezivanje sistema koji je usvojen od strane međunarodne organizacije za standarde. Ovaj model je najkorišćeniji apstraktni opis arhitekture mreže, koji opisuje interakciju uređaja, programa, servisa i protokola pri mrežnim komunikacijama. Naziv otvoreni (javni) sistem potiče od toga što ga je moguće stalno modifikovati, a da pri tome učestvuju svi. Svaka izmena koja bi se implementirala i prihvatila postala bi standard. Ovaj model deli protokole u sedam slojeva, pri čemu svaki sloj zahteva usluge od sloja ispod njega, a pruža usluge sloju iznad njega [1].

OSI referentni model se sastoji od sedam različitih nivoa, podeljenih u dve grupe, prikazanih na slici 3.1: [6]

- Prvu grupu sačinjavaju gornja tri sloja, slojevi aplikacije, prezentacije i sesije. Ona ima ulogu da opiše proces interakcije korisnik – računar, rad korisnika sa aplikacijom i proces komunikacije aplikacija među sobom kao krajnjim tačkama.
- Druga grupa je sačinjena od donja četiri sloja, slojevi transporta, mreže, veze i fizički sloj, koji definišu kako se prenose informacije sa jednog na drugi kraj (od jednog do drugog korisnika).



Slika 2.1. ISO / OSI referentni model

2.2.2. Industrijski protokoli i mreže

Uobičajen naziv za klasu protokola specifičnih za SCADA sisteme je industrijski protokoli, ime koje naglašava njihovu primenu u industrijskim postrojenjima i koje ih odvaja od protokola prisutnih u drugim oblastima računarstva i telekomunikacija. Prema aplikativnoj nameni i prirodi podataka u prenosu, industrijski protokoli se mogu podeliti u nekoliko tipičnih kategorija [1]:

- *Senzorski protokoli* koji su dizajnirani u cilju primarne akvizicije podataka iz postrojenja, i omogućavaju povezivanje računara ili izvršnih uređaja i „pametnih“ senzora.
- *Fieldbus protokoli* se koriste za distribuirano upravljanje u realnom vremenu. Oni su primarno orijentisani komunikaciji sa UI modulima i kontrolerima povezanih procesnom magistralom u industrijsku fieldbus mrežu, ali često se i sami senzori i aktuatori opremaju adekvatnom komunikacionom spregom.
- *Telemetrijski protokoli* kontrolišu komunikaciju između procesnih kontrolera i centralne SCADA stanice, koji se primenjuju u sistemima daljinskog nadzora i upravljanja gde je uticaj komunikacionih kanala (brzine i nivoa grešaka) odlučujući.
- *Međusistemski protokoli* su protokoli koji služe za komunikaciju između kontrolnih sistema. Realizuju spregu između dva autonomna ali kooperativna sistema, bez obzira da li se radi o dva SCADA sistema, vezi SCADA sistema sa simulatorom, ili

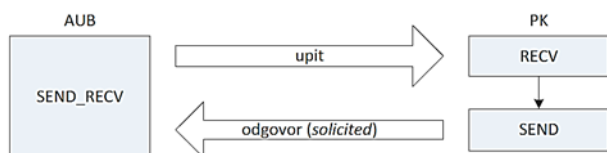
nečemu drugom. Važno je naglasiti da ovi protokoli podrazumevaju komunikaciju u realnom vremenu.

2.2.4. Klasifikacija protokola po režimu komunikacije

Promene koje uređaji dobijaju od senzora kada se te promene dogode se ne prenose odmah do nadzorno-upravljačke jedinice. Uređaj može biti konfigurisan tako da sam obavesti nadzorno-upravljačku jedinicu kada se desi neka promena ili može da bude prozvan od strane nadzorno-upravljačkog softvera i tom prilikom da mu saopšti najnovije stanje neke vrednosti dobijene sa senzora.

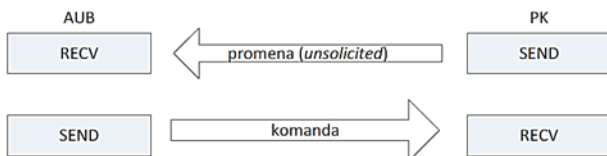
U pogledu organizacije prozivke i načina prenosa podataka sa procesnog nivoa, postoje nebalansirani i balansirani režim rada [1].

Kod nebalansiranog režima rada prozivajuća strana (nadzorno-upravljačka stanica) isključivo slanjem upita izaziva slanje odgovora (*solicited answer*) od prozване stanice (PLC ili RTU). Mana nebalansiranih protokola je sporost u prenosu kritičnih događaja, onih koji zavređuju hitnu obradu. Njihov prenos i procesiranje nisu mogući sve do slanja upita, odnosno do sledeće regularne akvizicije SCADA stanice što može trajati određeni vremenski period. Na slici 2.2 je prikazan nebalansirani režim komunikacije [1].



Slika 2.2. Nebalansirani režim komunikacije

Nedostatak nebalansiranog režima uklanjaju balansirani protokoli, gde svaka strana može samostalno inicirati slanje poruke (*unsolicited answer*, *report by exception*) preko pune dupleksne komunikacione veze. Uslov za implementaciju balansiranih režima rada nekog protokola, jeste taj da format odgovora sadrži sve podatke potrebne i dovoljne za prepoznavanje izvora i tipa prenete informacije, i njenu obradu na određistu. Balansirani režim komunikacije je prikazan na slici 2.3. [1]



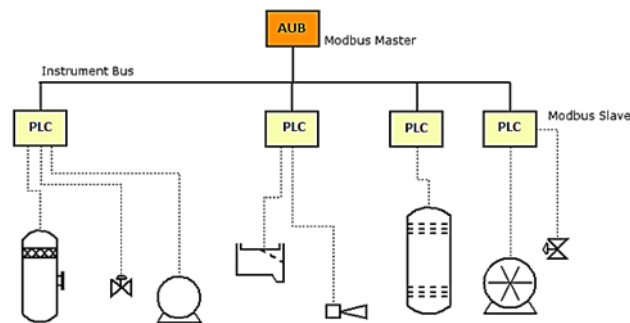
Slika 2.3. Balansirani režim komunikacije

3. MODBUS KOMUNIKACIONI PROTOKOL

Modbus je industrijski komunikacioni protokol, koji omogućava klijent-server komunikaciju između uređaja povezanih na različite vrste magistrala ili mreža. Iako se koristi u vrlo različite svrhe, Modbus je primarno telemetrijski protokol za komunikaciju između akviziciono upravljačkog bloka SCADA stanice i procesnih kontrolera (PLC) [1].

Modbus struktura podrazumeva jedan master i više slave uređaja na jednoj istoj liniji veze. Liniju u potpunosti kontroliše master tako što se po potrebi obraća pojedinim slave uređajima. Ukoliko na primer master hoće da

pročita stanje nekog merila on na liniju šalje poruku sa njegovom adresom. Merilo prepoznaje svoju adresu i u odgovoru šalje stanje svog ulaza, t.j. merene veličine. Po potrebi, odgovor može biti i prosta potvrda prijema (*acknowledgement*). Na primer, ako master želi da otvori neki ventil njemu je kao odgovor dovoljna samo potvrda da je poruka stigla na destinaciju i da je funkcija izvršena, t.j. da je ventil uspešno otvoren. Slika 4.1 prikazuje modbus sistem sa jednim master uređajem i više slave jedinica [6].



Slika 4.1. Modbus sistem sa jednim master uređajem i više slave uređaja

4. PROTOKOL ANALIZATORI

Protokol analizador (često nazvan mrežni ili paket analizador) predstavlja alat, tj. softverski ili hardverski modul, čija je uloga da presreće, loguje i analizira saobraćaj koji se prenosi preko mreže odgovarajućim komunikacionim kanalom. Pojam komunikacionog kanala može se odnositi na lokalnu mrežu između računara, pa sve do satelitske veze i obezbeđuje sredstvo komunikacije koristeći standardne komunikacione protokole. Principijalno bi svaki tip komunikacionog protokola zbog svoje specifične strukture trebalo da ima drugačiji alat za prikupljanje i analiziranje saobraćaja [7].

Kako podaci teku kroz mrežu preko komunikacionog kanala, protokol analizador presreće svaku jedinicu prenosa saobraćaja (paket itd.), i ukoliko je potrebno dekodira sirove podatke paketa, prikazujući vrednosti raznih polja iz paketa u *human readable* formatu, i analizira njihov sadržaj u skladu sa odgovarajućom specifikacijom. Analizatori se mogu koristiti i kao dodaci programima koji se bave zaštitom sistema kao što su *firewall*, *anti-virus* i slično [8].

5. PREDLOG REŠENJA

Koristeći C# programski jezik u *Microsoft Visual Studio* - u 2015 razvojnom okruženju i *.NET framework*, dok je za realizaciju UI dela korišćena WPF tehnologija i *MVVM pattern*.

Predloženo rešenje odnosi se na SCADA sistem gde se razmatra nadgledanje komunikacije sistema sa RTU uređajima za Modbus protokol. Ideja rada je da svaka razmenjena poruka bude prikazana u *human readable* formatu. Treba napomenuti da se sistem ne kači direktno na hardver, nego kao ulaz koristi podatke dobijene od *pubsub-a*. U principu, ovo je urađeno sa namerom jer nije potrebno da se sistem direktno kači na hardver. Za

principijalno rešenje su potrebni podaci, nebitno odakle podaci dolaze. Unutar aplikacije postoje četiri osnovne komponente zadužene za realizaciju rešenja, a to su: *sniffer*, *engine*, *modbus parser* i *UI*.

- **Sniffer**

Sniffer može da prikuplja podatke od raznih izvora (mreža, *PubSub*, *hard disk* i slično) i principijalno predstavlja vezu sa podacima. Ova komponenta je zadužena za prikupljanje podataka, tj. praćenje saobraćaja između SCADA sistema i uređaja u polju za modbus protokol. Svaka razmenjena poruka, biće registrovana od strane ove komponente i dalje se prosleđuje ka *Engine*-u. Ova komponenta je u predloženom rešenju realizovana upotrebom *Publish-Subscribe pattern*-a. Koristeći MEF (*Managed Extensibility Framework*) ova komponenta se učitava, tako da se kasnije može zameniti nekim drugim izvorom podataka. Na taj način je obezbeđena modularnost same aplikacije.

- **Engine**

Centralna komponenta u sistemu je *Engine*. Osnovni zadatak je da povezuje sve ostale komponente tj. Posredstvom ove komponente svi ostali delovi međusobno komuniciraju i razmenjuju podatke. Osnovna uloga je da podatke u *byte* reprezentaciji, dobijene od *Sniffer*-a, prosledi ka parseru koji kao povratnu vrednost vraća *human readable* string format.

- **Modbus parser**

Uloga parsera je da vrši dekodiranje (formatiranje) podataka iz sirovog (*raw bytes*) formata u *human readable* format. Zatim se ova povratna vrednost prosleđuje ka *UI*-u koji služi za prikaz. Preko MEF-a se obavlja i učitavanje ove komponente, tako da je na taj način moguće proširiti aplikaciju sa drugim industrijskim komunikacionim protokolom.

- **UI**

Korisnici/operatori nadgledaju mrežni saobraćaj, tj. analiziraju parsirane podatke koristeći *UI*. Koristeći *UI* korisnici započinju interakciju sa sistemom, nadgledaju i analiziraju mrežni saobraćaj. Takođe komande se izdaju posredstvom *UI* komponente. Pored samog prikaza saobraćaja, u zavisnosti od tipa poruke (*query*, *reply*, *filter*, *unparse*...) vrši se i bojenje teksta. Radi lakšeg praćenja pristiglih poruka omogućeno je i filtriranje samih poruka u *online* režimu, odnosno pretraga u *offline* režimu.

Akcije koje može da zadaje korisnik/operator sistemu:

- Startovanje nadgledanja komunikacije, tj. saobraćaja između SCADA sistema i RTU uređaja
- Prekid nadgledanja komunikacije
- Izbor komunikacionih kanala koje korisnik želi da nadgleda i analizira
- *Import* prethodno sačuvanih podataka
- *Export* prikupljenih podataka u fajl
- Filtriranje dolazećih podataka po odgovarajućem kriterijumu koji zavisi od izabranog protokola
- Pretraga ranije učitanih podataka

7. ZAKLJUČAK

U radu su predstavljeni koncept i realizacija programskog rešenja nadgledanja komunikacije između SCADA sistema i RTU uređaja za Modbus industrijski komunikacioni protokol. Pored toga dati su pregled SCADA sistema, komunikacionih protokola, kao i osnove samog modbus protokola.

Zbog svoje slojevite arhitekture protokol analizatora ostavljen je prostor za veliki napredak ka sledećoj verziji programa.

Trenutna implementacija podržava *PubSub* verziju sniffera, dok bi u narednoj verziji mogao da se koristi i neki drugi tip, u zavisnosti od konteksta upotrebe (npr. direktno nad karticom, ručni unos i slično).

Takođe, trenutno je podržan samo Modbus protokol dok bi se naknadno sistem mogao proširiti drugim protokolima (IEC 104, DNP3, ..).

8. LITERATURA

- [1] Branislav Atlagić, Fakultet tehničkih nauka, Novi Sad, Softver sa kritičnim odzivom, 2015
- [2] SCADA, <https://sr.wikipedia.org/sr-el/SCADA>
- [3] <http://predmet.singidunum.ac.rs/>
- [4] SCADA systems, <http://www.scadasystems.net/>
- [5] OSI model, https://sh.wikipedia.org/wiki/OSI_model
- [6] Modbus komunikacioni protokol, <https://www.automatika.rs/baza-znanja/obrada-signala/modbus-komunikacioni-protokol.html>
- [7] Protocol analyzer, https://en.wikipedia.org/wiki/Protocol_analyzer
- [8] Packet analyzer, https://en.wikipedia.org/wiki/Packet_analyzer

Kratka biografija:

Davor Nađ rođen je u Novom Sadu 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine odbranio je 2008.god

INDUSTRIJSKI KOMUNIKACIONI PROTOKOL IEC 60870-5-104**INDUSTRIAL COMMUNICATION PROTOCOL IEC 60870-5-104**Marko Kovačević, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – Ovaj rad predstavlja primer primene IEC 60870-5-104 industrijskog komunikacionog protokola za akviziciono upravljački sistem namenjen radu u realnom vremenu. Nakon prikaza strukture i formata poruka IEC 60870-5-104 protokola, predstavljan je koncept i implementacija programske podrške za praćenje saobraćaja između SCADA sistema i RTU stanica, upotrebom navedenog protokola.

Abstract – This paper represents an example of use of the IEC 60870-5-104 communication protocol for a SCADA system, used in realtime. After an overview of the IEC 60870-5-104 standards and protocol, concept and implementation of traffic sniffing between SCADA system and RTU stations, using this protocol, were presented.

Ključne reči: SCADA, IEC 60870-5-104, Protokol analizator

1. UVOD

Sa razvojem i unapređivanjem proizvodnih procesa u industriji javili su se zahtevi za što boljim i pouzdanijim nadzorom i upravljanjem tim procesima. Sistemi za transport energenata, bez obzira da li se radi o prirodnom gasu, nafti, električnoj energiji, vodi za grejanje, složeni su i prostorno raspodeljeni.

Efikasna isporuka energenata zahteva poznavanje trenutnog tehnološkog stanja ukupnog transportnog sistema, odnosno poznavanje velikog skupa podataka o topologiji sistema, stanju elemenata sistema, stanju mernih i regulacionih elemenata, trenutnoj potrošnji, meteorološkim uslovima, prognozi potrošnje i sl.

Zahvaljujući tehnološkom napretku u oblasti računarstva i informacionih tehnologija, upravljanje uz pomoć računara je standardna osobina savremenih industrijskih postrojenja. Automatizacija proizvodnih (tehnoloških) funkcija postrojenja realizuje se programski, posredstvom systemske i aplikativne programske podrške.

Osnovna ideja ovog rada je predlog rešenja za praćenje komunikacije između SCADA sistema i RTU stanica. U radu je predstavljeno rešenje zajedno sa teorijskim osnovama SCADA sistema, IEC 60870-5-104 protokola kao i korišćenim tehnologijama potrebnim za implementaciju samog programskog rešenja.

NAPOMENA:

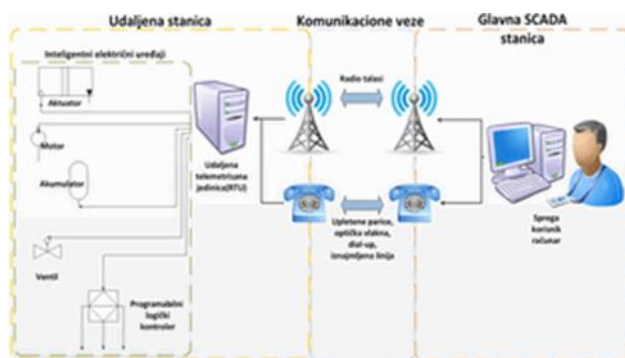
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Pavlica, docent.

Svaka razmenjena poruka (komanda, unsolicited promena, poll) prikazana je u human readable formatu upotrebom IEC 104 parsera, koji omogućava parsiranje niza bajtova u objektni model radi lakšeg rukovanja porukama.

2. OSNOVE SCADA SISTEMA I MREŽNIH PROTOKOLA

SCADA sistem (*Supervisory Control And Data Acquisition*) predstavlja kontrolni sistem koji služi za automatizaciju opštih procesa, odnosno sistem koji se koristi za prikupljanje podataka u realnom vremenu sa senzora i instrumenata lociranih na udaljenim stanicama, i za prenos i prikazivanje tih podataka u centralnoj stanici ili računarskom centru u svrhu nadzora i upravljanja. SCADA sistem u realnosti može da prati i upravlja velikim brojem ulazno-izlaznih vrednosti [1].

RTU je računarski uređaj koji je najčešće namenjen za rad u industrijskim uslovima. Njegov zadatak je da prikuplja informacije sa raznih digitalnih i analognih senzora i da prosleđuje komande uređajima na terenu. Takođe može primati i digitalne komande poslate od strane nadzornog sistema ali poseduje skromnu automatsku upravljačku logiku. PLC za razliku od RTU-a može da realizuje složenije upravljačke procese, preuzimajući na sebe značajan deo od ukupne aplikativne programske podrške SCADA sistema. Dok PLC-ovi omogućavaju automatsko vođenje procesa, oni su obično raspoređeni po celom postrojenju, čime je ručno prikupljanje podataka sa njih otežano. SCADA sistem prikuplja informacije sa njih i drugih kontrolera preko neke vrste mreže, obrađuje ih i prikazuje u realnom vremenu i omogućuje dalju manipulaciju nad njima. Na primer, PLC može da kontroliše protok vode kroz deo industrijskog procesa, ali SCADA sistem može da dozvoli operateru da promeni zadatu vrednost protoka i može da snima i prikazuje bilo koja alarmna stanja, kao što su gubitak pritiska ili visoka temperatura. Glavne komponente malog SCADA sistema prikazane su na slici 2.1. [1]



Slika 2.1. Glavne komponente malog SCADA sistema

Akviziono upravljački sistem se definiše kao distribuiran računarski sistem za rad u realnom vremenu. Njegovo ukupno funkcionisanje primarno zavisi od kvaliteta komunikacije, odnosno od uspešnosti i pouzdanosti prenosa poruka između pojedinih komponenti u njegovom sastavu. Predmet razmene su podaci vezani za nadzor i upravljanje fizičkim sistemom, a tok komunikacije je kontrolisan odgovarajućim komunikacionim protokolima akviziciono upravljačkog sistema [1].

Komunikacioni protokoli predstavljaju skup pravila i procedura koji kontrolišu tok komunikacije između serijski spregnutih stanica u okviru računarske mreže. Posredstvom komunikacionih protokola obezbeđuje se uspešna interakcija između udaljenih procesa. Protokol definiše format i redosled poruka koji se razmenjuju, kao i svaku akciju koja se inicira prijemom određene poruke. Osnovne funkcije komunikacionih protokola su kontrola grešaka i upravljanje tokom podataka u mreži. Kontrola grešaka obezbeđuje pouzdan prenos i integritet poruka. Primarni cilj upravljanja tokom je izbegavanje zagušenja u saobraćaju, i deljenje komunikacionih resursa između više korisnika [1].

Klase protokola koji se koriste u SCADA sistemima najčešće se nazivaju industrijskim protokolima i industrijskim mrežama (Industrial protocols and Industrial networks). Prema tipu učesnika u komunikaciji i njihovim ulogama u akviziciono upravljačkom sistemu, može se izdvojiti nekoliko kategorija industrijskih protokola: senzorski protokoli, Fieldbus protokoli, telemetrijski protokoli i međusistemski protokoli [1].

3. SKUP STANDARDA IEC 60870-5

Međunarodna tehnička komisija IEC (*International Electrotechnical Commission*) je svetska organizacija za standardizaciju koja se sastoji od nacionalnih elektrotehničkih komiteta (*IEC National Committees*). Jedan od ciljeva koji je pred njega postavljen jeste standardizacija sprege SCADA stanice i PLC kontrolera, pa je u tu svrhu uloženi veliki napor da bi se definisali i usvojili opšte prihvaćeni standardi koji bi bili orijentisani ka pojedinim granama industrije. Skup standarda IEC 60870-5 nastao je kao rezultat rada radne grupe (WG03: *Telecontrol protocols*) koja je delovala u sklopu IEC tehničkog komiteta koji se bavi upravljanjem energetske sistemima (TC57: *Power system control and associated communications*) i pruža komunikacioni profil za slanje osnovnih telekontrolnih poruka između dva sistema, gde se koriste trajne, direktne mreže za razmenu podataka između sistema [2].

3.1. IEC 60870-5-104

Standard IEC 104 definiše telemetrijski protokol za komunikaciju akviziciono upravljačke stanice i procesnih kontrolera. Njime je standardizovana upotreba aplikativnih funkcija i podataka kod protokola sa različitim mehanizmima za prenos poruka. Fizički i mrežni sloj koji su definisani IEC 60870-5-101 standardom su u potpunosti zamenjeni, pa IEC 60870-5-104 praktično predstavlja kombinaciju aplikativnih funkcija IEC 60870-5-101 protokola i transportnih funkcija podržanih TCP/IP

(Transmission Control Protocol/Internet Protocol) protokolom. U tabeli 3.1 prikazana je struktura IEC 104 protokola [3].

Tabela 3.1 - Prikaz strukture IEC 104 protokola

Preuzete aplikacione funkcije iz IEC 60870-5-5 prema IEC 60870-5-101	Inicijalizacija	Korisnički proces
Definisani ASDU-i u IEC 60870-5-104 i IEC 60870-5-101	APCI (<i>Application Protocol Control Information</i>)	Aplikacioni sloj
Preuzeto iz TCP/IP protokola		Transportni sloj
		Mrežni sloj
		Sloj veze
		Fizički sloj

Osnovnu jedinicu prenosa predstavlja APDU (*Application Protocol Data Unit*). U zavisnosti od formata poruka, APDU se sastoji iz dva dela: [4]

- APCI (*Application Protocol Control Information*) koje je obavezan i sadrži početni bajt poruke, dužinu poruke i kontrolne bajtove.
- ASDU (*Application Service Data Unit*) koji imaju samo poruke tipa I-Format. Sadrži podatke kao što su uzrok prenosa, tip informacionih objekata koji se prenose kao i same informacione objekte.

3.1.1 APCI deo poruke

Startni bajt označava početak IEC 104 poruke.

Dužina APDU definiše broj bajtova u poruci koji se sastoji iz 4 kontrolna bajta iz APCI i dužine ASDU ako postoji. Prvi bajt koji se broji je prvi kontrolni bajt, a poslednji je poslednji bajt ASDU (ako je u pitanju I-FORMAT poruke). Maksimalna dužina ASDU je 249 zato što je maksimalna dužina APDU 255, početni i bajt dužine APDU se ne računaju pa je to 253, i od toga se oduzimaju 4 kontrolna bajta [5].

Kontrolni bajtovi definišu kontrolne informacije u cilju sprečavanja gubitka ili dupliranja poruka, početak i kraj razmene poruka i nadzor nad konekcijama za prenos. U zavisnosti od načina na koji se koriste kontrolni bajtovi, razlikuju se 3 tipa poruka: [5]

- Poruke tipa S-FORMAT-a, koje vrše nadzorne funkcije
- Poruke tipa U-FORMAT-a, koje vrše upravljačke funkcije
- Poruke tipa I-FORMAT-a, koje vrše prenos informacija

3.1.2 ASDU deo poruke

Ovaj deo se pojavljuje samo u I-FORMAT porukama. Predstavlja strukturu kojom se prenose aplikativni podaci, odnosno rezultati merenja ili nadgledanja [5].

ASDU se sastoji iz identifikatora podataka odnosno DUI (*Data Unit Identifier*) i jednog ili više informacionih objekata. DUI ima uvek istu strukturu i sastoji se iz: [5]

- Identifikatora tipa (*Type identification*)

- Promenljivog kvalifikatora strukture (*Variable Structure Qualifier*)
- Uzroka prenosa (*Cause of Transmission*)
- ASDU adrese (*Common Address of ASDU*)

Što se tiče informacionih objekata, moguće je poslati više takvih u jednoj poruci, sa time da moraju da budu istog tipa. Informacioni objekti sa vremenskom oznakom (*time tag*) i bez vremenske oznake se računaju kao različiti tipovi [5].

4. PROTOKOL ANALIZATORI

Protokol analizatori (često nazvani i mrežni ili paket analizatori) predstavljaju računarski program ili deo hardvera, koji ima mogućnost da presreće i loguje saobraćaj koji se prenosi preko mreže. Dok se podaci prenose preko mreže, zadatak analizatora je da „presretnu“ svaki paket i da analizira i prikaže sadržaj prema određenim specifikacijama.

Analizatori takođe mogu da se koriste kao dodaci programima koji se bave zaštitom sistema, kao što su firewall, anti-virus, spyware detection i slično.

Neke od mogućnosti koje pružaju su: analiza problema na mreži, detektovanje pokušaja upada na mrežu, nadgledanje korišćenja mreže, nadgledanja trenutnih podataka u transportu, filtriranje sumnjivog sadržaja iz saobraćaja u mreži [6].

Neki od najbitnijih poslova koje protokol analizatori obavljaju su dekodiranje sirovih podataka i njihova analiza. Većina protokol analizatora razlikuje protokole i prepoznaje ih, po njihovom najvišem sloju. Postoje primeri protokol analizatora koji mogu da analiziraju preko 300 različitih protokola. Jedan od primera protokol analizatora je Wireshark [7].

Principijelna ideja kod razvoja ovog programskog rešenja je da se razvije protokol analizator kao programsko proširenje za SCADA sistem. Na ovaj način izbeglo bi se korišćenje *third party* programa i eventualni problemi koji se mogu javiti u pogledu sigurnosti.

Naime, svetske organizacije za sigurnost (NERC CIP, ENTSO-E..) postavljaju ograničenja prilikom korišćenja ovakvih alata u vidu dodeljenih prava pristupa osetljivim podacima i prikupljanju podataka sa mreže.

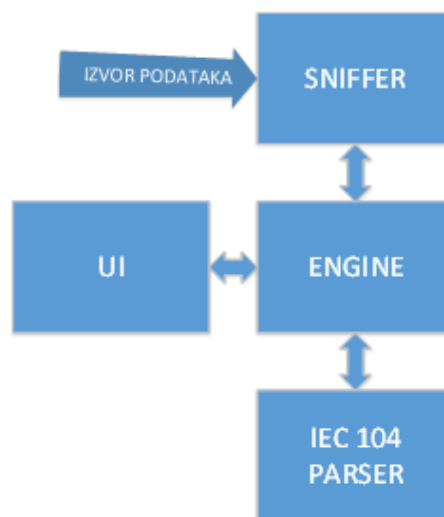
Predloženo rešenje podržava modularan način prikupljanja podataka, te se umesto sa *driver*-a mrežnih kontrolera, podaci mogu prikupljati nakon što stignu u SCADA sistem ili korišćenjem nekog drugog izvora podataka, npr. odgovarajućeg *Publish/Subscribe* servisa.

5. PREDLOG REŠENJA

Predloženo rešenje odnosi se na deo SCADA sistema gde se razmatra nadgledanje komunikacije sistema sa RTU uređajima, upotrebom IEC 104 protokola.

Osnovna ideja rada jeste da svaka razmenjena poruka bude prikazana u *human readable* formatu.

Dijagram sa slike 5.1 prikazuje pojednostavljenu logičku arhitekturu.

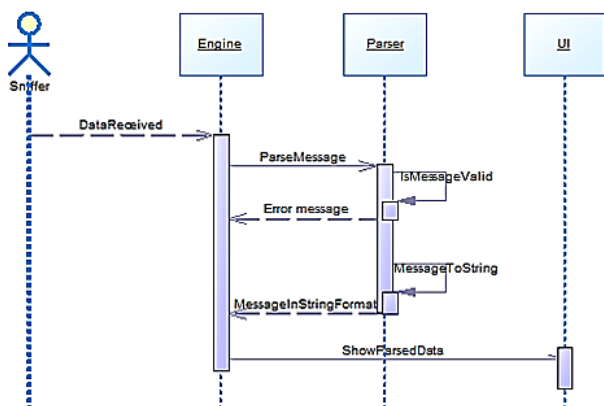


Slika 5.1. Arhitektura rešenja

Unutar rešenja mogu se izdvojiti četiri osnovne komponente.

- *Sniffer* komponenta je zadužena za praćenje saobraćaja između SCADA sistema i uređaja u polju. Svaka razmenjena poruka, bez obzira da li je u pitanju komandovanje, *poll* ili *unsolicited* promena, biće registrovana od strane ove komponente i dalje prosledjena ka *Engine*-u.
- *Engine* predstavlja spregu i veznu tačku za sve ostale komponente. Osnovna uloga je da podatke u *byte* reprezentaciji, dobijene od *Sniffer*-a, prosledi ka parseru koji kao povratnu vrednost vraća *human readable string* format. Takav format *Engine* dalje prosleđuje ka *UI*-u, koji služi za sam prikaz podataka. Sa druge strane, *Engine* prima komande od strane *UI* dela, kao što su *export*, *import*, *start sniffing*, *stop sniffing*.
- *IEC 104 Parser* je zadužen za pretvaranje niza bajtova u objekte i obrnuto. Uloga mu je da pruži objektni model za poruke koje se razmenjuju između *master* i *slave* strane. Kada *engine* dobije niz bajtova od strane *Sniffer*-a, on poziva metode parsera da bi dobio odgovarajući objektni model predstavljen u *human readable string* formatu.
- *UI* deo služi za sam prikaz parsiranih poruka korisnicima/operaterima. Kada *Engine* dobije od parsera formatiranu poruku dalje je prosleđuje ka *UI*-u. Osnovni zadatak *UI*-a, pored samog prikaza, je i bojenje teksta u zavisnosti od tipa poruke (*query*, *reply*, *filter*). Radi lakšeg praćenja pristiglih poruka omogućen je i filter samih poruka u *online* režimu, odnosno *search* poruka u *offline* režimu. Posredstvom *UI* komponenti moguće je i izdavanje komandi kao što su *start/stop sniffing*, *import/export*...

Unutar rešenja može se izdvojiti nekoliko tokova podataka. Osnovni tok podataka je od *Sniffer* komponente, koja prati saobraćaj između SCADA sistema i RTU uređaja, ka *Engine*-u koji prosleđuje niz bajtova ka *IEC 104* parseru, koji ima zadatak da konvertuje poruku u *human readable* format. Takva poruka se sada ponovo preko *Engine*-a prosleđuje ka *UI* delu koji služi za sam prikaz. Ovakav tok podataka prikazan je na slici 5.2.



Slika 5.2. Osnovni tok podataka

Pored ovog toka podataka, postoje i drugi tokovi kada se komande izdaju preko UI dela i prenose preko *Engine*-a ka ostalim komponentama, u zavisnosti od namene same komande.

6. IMPLEMENTACIJA REŠENJA

Programsko rešenje implementirano je korišćenjem C# programskog jezika u *Microsoft Visual Studio 2015* razvojnom okruženju. *Framework* koji je korišćen je .NET 4.6.1, a za realizaciju UI dela je korišćena WPF tehnologija i MVVM (*Model-View-ViewModel*) šablon.

Sniffer komponenta, koja predstavlja izvor podataka aplikacije, realizovana je upotrebom *Publish/Subscribe patterna*. Ipak, radi modularnosti same aplikacije, učitavanje *Sniffer* komponente se obavlja upotrebom MEF-a (*Managed Extensibility Framework*), tako da je u svakom trenutku ovakav izvor podataka moguće zameniti sa nekim drugim (čitanje sa *driver*-a mrežne kartice, *hard disk*-a..). Pored ove komponente preko MEF-a se obavlja i učitavanje IEC 104 parsera, radi mogućnosti proširenja programa sa nekim drugim industrijskim komunikacionim protokolima (*Modbus*, *DNP3*..). Učitavanje ovih komponenti obavlja *Engine* komponenta, koja predstavlja veznu tačku za sve ostale komponente.

Rad aplikacije je moguć u dva režima, *online* i *offline*. U *online* režimu svaka razmenjena poruka između SCADA sistema i RTU uređaja se odmah parsira i prikazuje operateru, dok u *offline* režimu operater može da vrši analizu prethodno pristiglih poruka. Prelazak iz jednog u drugi režim obavlja se startovanjem, odnosno zaustavljanjem snimanja saobraćaja.

7. ZAKLJUČAK

U ovom radu predstavljeni su koncept i realizacija programskog rešenja za nadgledanje komunikacije između SCADA sistema i RTU uređaja, upotrebom IEC 104 industrijskog komunikacionog protokola. Pored toga dat je i uvid u osnove samog protokola kao i SCADA sistema.

Cilj rada je bio praćenje komunikacije i prikaz poruka u *human readable* formatu tako da operateru/korisniku bude približno koje poruke SCADA sistem razmenjuje. Zbog svoje slojevite arhitekture ostavljen je prostor za napredak ka sledećoj verziji programa.

Trenutna implementacija podržava *Publish/Subscribe* verziju *Sniffer*-a, dok bi u narednoj verziji mogao da se koristi i neki drugi tip, u zavisnosti od konteksta upotrebe (čitanje sa mrežne kartice, *hard disk* – a, dobavljanje podataka prilikom ulaza u SCADA sistem..). Takođe, pored IEC 104 protokola, rešenje je moguće proširiti dodavanjem nekog drugog industrijskog komunikacionog protokola, kao što su *DNP3*, *Modbus*..

8. LITERATURA

- [1] Branislav Atlagić, Fakultet tehničkih nauka, Novi Sad, Softver sa kritičnim odzivom, 2015
- [2] International standard IEC 870-5-1: Telecontrol equipment and and systems – Part 5: Transmission protocols. Section 1 – Transmission frame formats, 1990.
- [3] International standard IEC 870-5-2: Telecontrol equipment and and systems – Part 5: Transmission protocols. Section 2: Link transmission procedures, 1992.
- [4] International standard IEC 870-5-3: Telecontrol equipment and and systems – Part 5: Transmission protocols. Section 3: General structure of application data, 1992.
- [5] International standard IEC 870-5-4: Telecontrol equipment and and systems – Part 5: Transmission protocols. Section 4: Definition and coding of application information elements, 1993.
- [6] Protocol Analyzer, https://en.wikipedia.org/wiki/Packet_analyzer
- [7] Wireshark Introduction, https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs

Kratka biografija:

Marko Kovačević rođen je 30.11.1993. godine u Rumi. Diplomski rad iz oblasti Elektrotehnike i računarstva – Računarske nauke i informatika odbranio je na Fakultetu tehničkih nauka 2016. god. Iste godine upisao je master studije Primenjenog softverskog inženjeringa na Fakultetu tehničkih nauka. Položio je sve ispite predviđene planom i programom.

**KONCEPT MODULARNOG PROTOKOL ANALIZATORA
I NJEGOVA PRIMENA NA DNP3 PROTOKOL****CONCEPT OF A MODULAR PROTOCOL ANALYZER
AND ITS USE ON DNP3 PROTOCOL**

Milan Mitrić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**2. TEORIJSKE OSNOVE**

Kratak sadržaj – Ovaj rad sadrži opis protokol analizatora, njegove primene i mesta u SCADA sistemima. Pored toga, opisani su osnove SCADA sistema, komunikacionih protokola i DNP3 protokola. Cilj rada predstavlja opis koncepta modularnog programskog rešenja koje može da nadgleda i analizira saobraćaj u SCADA sistemu.

Abstract – This paper represents a description about protocol analyzers, and its use and place in a SCADA system. Further more, some basics about SCADA systems, communications protocols and DNP3 protocol are described. Purpose of this paper is to present the idea of a modular software solution that can supervise and analyze data traffic in a SCADA system.

Ključne reči: SCADA, DNP3, Protokol analizator, MEF, MVVM

1. UVOD

Zahvaljujući napretku tehnologije, upravljanje procesima u industrijskim postrojenjima uveliko se obavlja uz pomoć računarskih sistema, koristeći softversku i hardversku podršku. Računarski sistemi koji kontrolišu industrijske procese se nazivaju Akviziciono-upravljački ili SCADA sistemi. Vrlo značajan aspekt svih SCADA sistema namenjenih upravljanju u realnom vremenu jeste međusobna komunikacija njihovih komponenti. Tu komunikaciju definišu komunikacioni protokoli, koji se u ovoj sferi nazivaju industrijskim protokolima. Hardverske ili softverske komponente koje se bave nadgledanjem i analizom saobraćaja komunikacionih protokola se nazivaju protokol (paket) analizatori. Kvalitet ovih protokol analizatora zavisi od broja podržanih protokola i mogućnosti koje daju prilikom analize saobraćaja.

U ovom radu je predstavljena ideja modularnog protokol analizatora, koji je proširiv u smislu načina prikupljanja podataka kao i broja podržanih protokola za analiziranje podataka. Prednost koje ovo rešenje daje jeste mogućnost izmene izvora podataka (sa mreže, hard disk-a, odgovarajućih aplikativnih servisa npr. PubSub i slično).

Takođe, dodavanje novih protokola za analizu, ne zahteva izmene celokupnog rešenja, već samo razvijanje modula koji će se baviti konkretnim protokolom.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Pavlica, vanr. prof.

2.1 SCADA sistemi

SCADA (Supervisory Control And Data Acquisition) predstavlja računarski sistem čija je uloga upravljanje i kontrola industrijskim procesima. SCADA sistem se sastoji od prostorno distribuiranih, međusobno povezanih namenskih računarskih modula čiji je zadatak da ostvare funkciju nadzora i upravljanja procesom u realnom vremenu. Jednostavan SCADA sistem se sastoji iz sledećih komponenti [1]:

- Glavna stanica (Master station)
- Komunikaciona infrastruktura
- Udaljene terminalne jedinice (PLC ili RTU)

Osnovni zadaci koji se nameću SCADA sistemu su sledeći:

- **Rad u realnom vremenu** – kako bi se adekvatno reagovalo na promene (regularne i neregularne) u fizičkom procesu kojim se upravlja [1].
- **Distribucija računarskih resursa u okviru sistema** – jer proizvodni proces kao objekat nadzora, sam po sebi može biti prostorno dislociran, kako u okviru samog postrojenja, tako i na mnogo širem geografskom području. Stoga, struktura SCADA sistema se mora odlikovati prostornom distribucijom autonomnih računarskih komponenti, koje su međuračunarskom komunikacijom spregnute u jedinstven sistem [1].
- **Pouzdanost** – visok kvalitet ugrađenih fizičkih komponenti i programske podrške kojim se treba minimizovati srednje vreme između ispada sistema [1].
- **Raspoloživost** – u slučaju odgovarajućih kvarova ili neispravnih komponenti (kako hardverskih tako i softverskih), sistem i dalje treba da bude dostupan i funkcionalan [1].

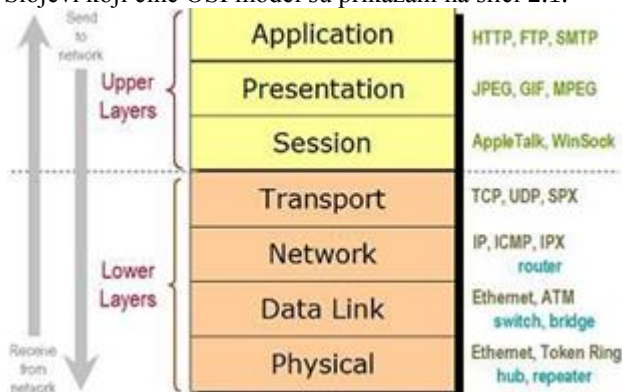
2.2 Mrežni protokoli

Mrežni (komunikacioni) protokol je skup pravila koji omogućavaju da dva ili više entiteta komunikacionog sistema razmenjuju podatke preko nekog vida fizičke veze. Protokoli definišu sintaksu, semantiku i sinhronizaciju komunikacije. 1979. godine je definisan OSI (Open Systems Interconnection) referentni model koji predstavlja temelj savremenih komunikacionih sistema jer je njegovom pojavom omogućena

kompatibilnost opreme različitih proizvođača, a samim time je i primena komunikacionih sistema postala lakša. OSI referentni model se sastoji od sedam različitih slojeva, podeljenih u dve grupe [2]:

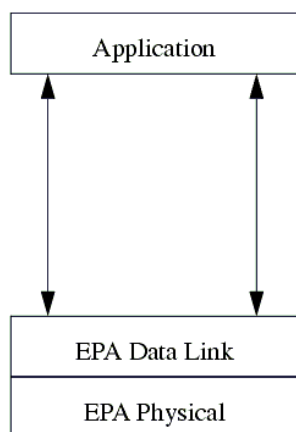
- Prvu grupu sačinjavaju gornja tri sloja, slojevi aplikacije, prezentacije i sesije. Ona ima ulogu da opiše proces interakcije korisnik-računar, rad korisnika sa aplikacijom i proces komunikacije aplikacija među sobom kao krajnjih tačaka [3].
- Druga grupa je sačinjena iz donja četiri sloja koja definišu kako se prenose informacije sa jednog kraja na drugi (od jednog do drugog korisnika) [3].

Slojevi koji čine OSI model su prikazani na slici 2.1.



Slika 2.1 Slojevi OSI modela

Neki komunikacioni protokoli, kao što su DNP3 i IEC 60870-5 familija protokola su bazirani na troslojnom EPA (Enhanced Performance Architecture) modelu. Ovaj model predstavlja uopštenu formu sedmoslojnog OSI modela u cilju pružanja optimalnih performansi za aplikacionu telekontrolu [4]. Na slici 2.2 je prikazan EPA model.



Slika 2.2 Slojevi EPA modela

2.3 DNP3 Protokol

DNP3 (*Distributed Network Protocol*) je razvila kompanija Westronic Incorporated između 1992. i 1994. IEEE je 1994. DNP3 protokol, zajedno sa protokolom IEC 60870-5-101, proglasio za najefikasnije protokole u SCADA sistemima, pri tome uzimajući u obzir preko 140 protokola koji su u tom trenutku bili u upotrebi [5]. Neke od najboljih osobina DNP3 protokola su:

- *Broadcasting* – slanje jedne poruke ka više uređaja.
- Vremenski obeleženi podaci (*timestamp*).
- Mogućnost da se unese dodatna sigurnost kada se radi sa izlaznim podacima, tako što je selektovanje pre upravljanja nekim entitetom.
- Označavanje kvaliteta podataka – da li su podaci validni, ako nisu zašto.
- Akvizicione grupe – mogućnost da se više nepovezanih podataka, zahteva jednim zahtevom.
- Prijavljivanje samo promena u podacima.
- Spontani odgovori – udaljena stanica ne mora da šalje podatke samo kada to glavna stanica zatraži. Spontani odgovori se nazivaju još i nezahtevani odgovori (*unsolicited*).

DNP3 se oslanja na troslojni EPA model, uz odgovarajuće izmene. DNP3 uvodi transportne funkcije, koje se nalaze između aplikacionog i sloja veza. Uloga ovog sloja je da pruži podršku aplikacionom sloju tako što će podatke pristigle sa aplikacionog sloja podeliti na manje delove, na veličinu kojom sloj veze može da manipuliše. Zbog toga se transportna funkcija često naziva pseudo-sloj [5].

Uloge pojedinačnog sloja DNP3 protokola su navedene u tabeli 2.1.

Tabela 2.1: Kratak pregled slojeva DNP3 protokola

Sloj ili funkcija	Osnovna jedinica prenosa	Informacije
Aplikacioni sloj	Aplikacioni fragment	Komunikaciju sa korisničkim slojem. Maksimalna dužina 2048 bajta.
Transportne funkcije (pseudo-sloj)	Transportni segment	Rastavljanje fragmenata na manje delove, segmente. Maksimalna veličina 250 bajta.
Sloj veze	Frejm sloja veze	Služi za kontrolu toka i otkrivanje grešaka u porukama. Maksimalna veličina 292 bajta.

Svaki sloj u protokol pruža usluge sloju iznad, odnosno koristi usluge sloja ispod.

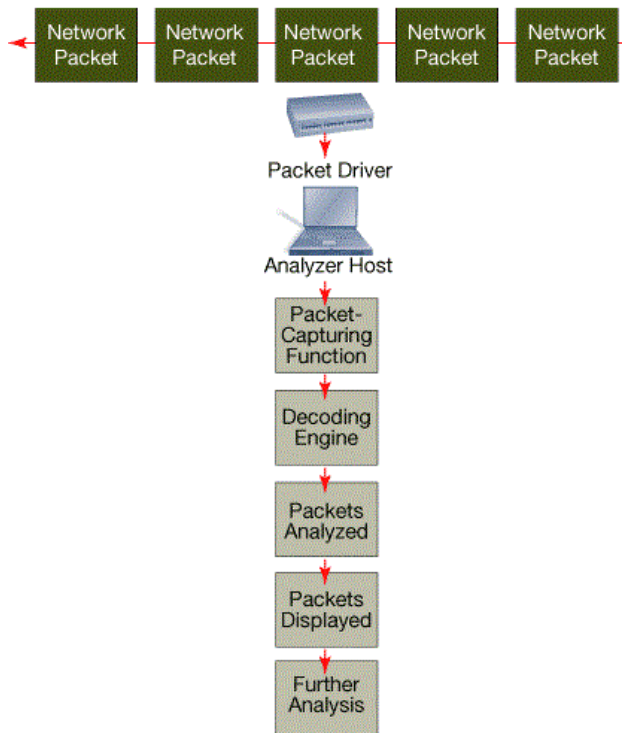
2.4 Protokol analizatori

Protokol analizatori (često nazivani mrežni ili paket analizatori) predstavljaju softverske i/ili hardverske module čija je uloga nadgledanje i analiziranje saobraćaja preko odgovarajućeg komunikacionog kanala. Neke od mogućnosti koje pružaju analizatoru su:

- Analiziranje problema na mreži.
- Detektovanje pokušaja upada na mrežu.
- Nadgledanje korišćenja mreže.

- Nadgledanje podataka trenutno u transportu.
- Utvrđivanje efikasnosti kontrolnog sistema (*proxy*, *firewall*, kontrola pristupa, *spam filter*).

Dok podaci teku preko nekog komunikacionog kanala, uloga analizatora je da „presreće“ svaku jedinicu prenosa tog saobraćaja, i ako je potrebno da dekodiraju te podatke i prikažu ih u drugoj formi (npr. *human readable form*). Takođe je potrebno da pruže podršku za analizu i filtriranje poruka tokom nadgledanja saobraćaja. Princip rada protokol analizatora je dat na slici 2.3.



Slika 2.3 Princip rada protokol analizatora

3. KONCEPT REŠENJA

Predloženo rešenje se odnosi na komponentu koja se koristi da bi se nadgledala komunikacija u SCADA sistemu između glavne i udaljenih (RTU) stanica. Ideja je da ova komponenta bude sastavljena iz zajedničkog dela, nezavisnog od protokola (koji uključuje i korisnički interfejs, i više nezavisnih modula. Te module možemo podeliti u dve grupe:

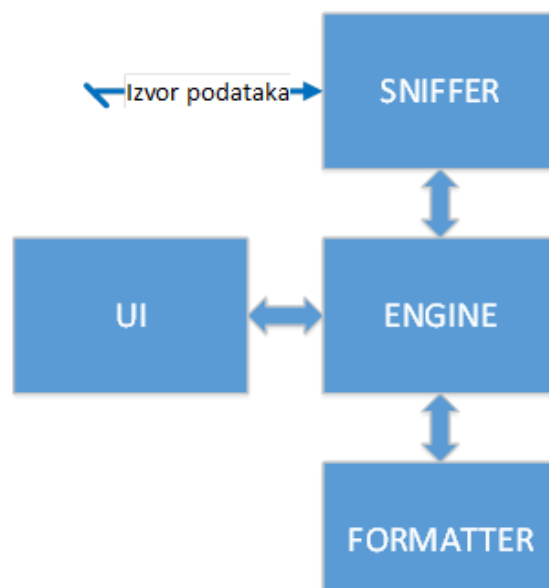
- Moduli za prikupljanje podataka
- Moduli za dekodiranje i analiziranje

Modul za prikupljanje podataka (*Sniffer*) omogućava prikupljanje podataka sa raznih izvora (*driver*-a mrežnih kontrolera, nakon što stignu u SCADA sistem, iz fajlova, odgovarajućih aplikativnih servisa npr. *PubSub*) i slično.

Bitno je naglasiti da u jednom trenutku može da bude aktivan samo jedan izvor podataka, samim time i samo jedan *Sniffer*.

Moduli za dekodiranje i analiziranje podataka se u rešenju nazivaju *Formatter*-ima. Uloga im je da nakon što *Sniffer* prikupi podatke, da ih dekodira u skladu sa pravilima protokola i da omogući dalju analizu nad tim podacima (kao npr. filtriranje podataka po raznim osnovima).

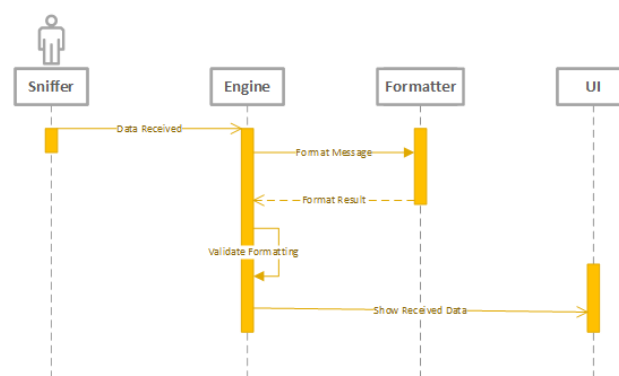
Postoji po jedan *Formatter* za svaki podržani protokol. Arhitektura rešenja je data na slici 3.1.



Slika 3.1 Arhitektura rešenja

Engine predstavlja zajedničku komponentu koja sadrži logiku koja nije *protocol specific*. *Engine* dinamički učitava module i pruža sve usluge korisničkom interfejsu.

Osnovni tok podataka je od izvora – *Sniffer* modula pa do korisničkog interfejsa – UI. Kada *Sniffer* dobije podatke, on ga asinhrono predaje do *Engine*-a. *Engine* prepoznaje protokol kojim je enkodiran podataka i poziva odgovarajući *Formatter* koji dekodira poruku i vraća je *Engine*-u. *Engine* je nakon toga prikazuje na UI. Glavni tok podataka je prikazan na slici 3.2.



Slika 3.2. Glavni tok podataka

4. IMPLEMENTACIJA REŠENJA

Rešenje je implementirano koristeći programski jezik C#. Implementirane su zajedničke komponente *Engine* i UI kao i dva modula. Implementiran je *Sniffer* koji omogućava prikupljanje podataka sa odgovarajućeg aplikativnog *Pub-Sub* servisa. Takođe je implementiran i *Formatter* za DNP3 protokol. Dinamičko učitavanje *Sniffer*-a i *Formatter*-a je urađeno koristeći MEF (*Managed Extensibility Framework*). Korišćen je .NET Framework 4.6.1 i MVVM (*ModelView-ViewModel*) šablon.

4.1 Use Case dijagrami

Akcije koje korisnik izvršava posredstvom UI su sledeće:

- Izbor komunikacionih kanala koje korisnik želi da nadgleda i analizira.
- Početak snimanja saobraćaja.
- Zaustavljanje snimanja saobraćaja.
- Import prethodno snimljenih podataka iz fajla u aplikaciju radi dalje analize.
- Export trenutno snimljenih podataka u fajl.
- Filtriranje podataka (prethodno snimljenih i dolazećih) po odgovarajućih kriterijumima koji zavise od samog protokola.

Aplikacija može da radi u režima:

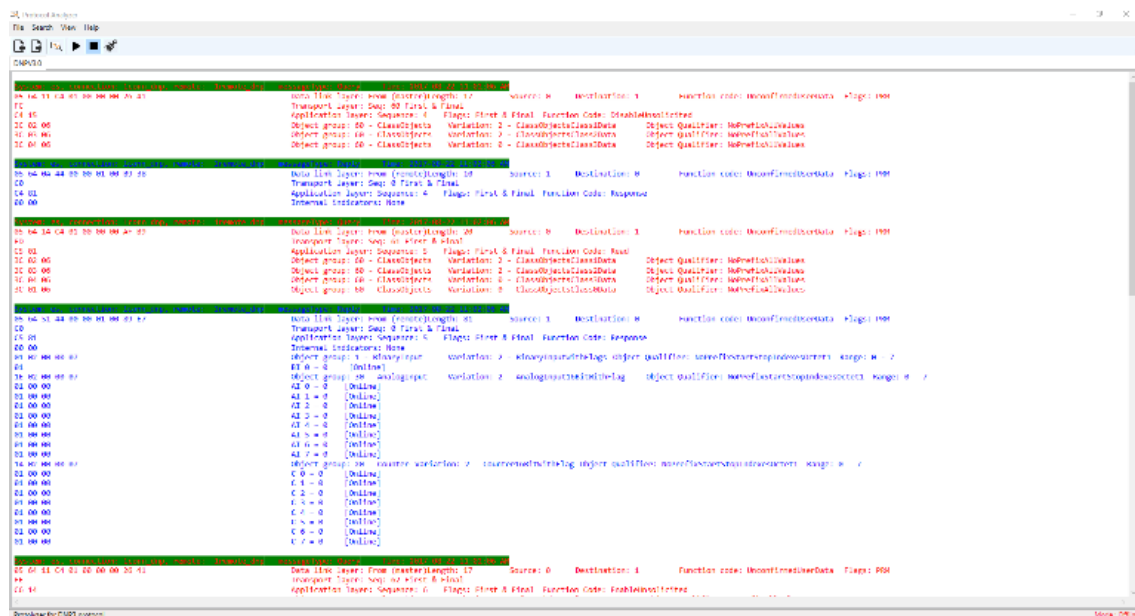
• Online

Tokom *online* režima, korisnik bira konekcije sa kojih će da prati saobraćaj i prispele DNP3 poruke će biti odmah dekodirane i prikazane. Tokom ovog režima sav saobraćaj se snima i moguće je da se *export*-uje u odgovarajući fajl u binarnom obliku. Takođe je moguće zaustaviti snimanje saobraćaja i prelazak u *offline* režim.

• Offline

U *offline* režimu korisnik može uradi *import* prethodno exportovanih podataka, ili analizu trenutno zatečenog stanja.

Primer rada aplikacije se nalazi na slici 4.1.



Slika 4.1. Primer snimljenog saobraćaja DNP3 protokola

5. ZAKLJUČAK

U radu je predstavljen kratak opis SCADA sistema u DNP3 protokola, kao i koncepta i realizacije programske podrške za nadgledanje i analizu saobraćaja između SCADA sistema i udaljenih uređaja.

Objašnjen je koncept generičkog i modularnog rešenja koji treba da dekodira podatke u komunikaciji između SCADA sistema i udaljenih uređaja i da ih prikaže u *human readable form*. Zbog modularnosti rešenja postoji dosta prostora za proširenje implementiranog rešenja u smislu dodavanja podrške za dekodiranje i analizu drugih industrijskih protokola (npr. IEC 104 i Modbus). Takođe, predložena arhitektura podržava i promenu komponente za prikupljanje podataka (*Sniffer*).

6. LITERATURA

- [1] Branislav Atlagić, Fakultet tehničkih nauka, Novi Sad, Softver sa kritičnim odzivom, 2015 [2] A.E. Bryson, Y.C. Ho, "Applied Optimal Control", New York, Wiley, 1975.
- [2] "Publicly Available Standards", Standards.iso.org
- [3] https://en.wikipedia.org/wiki/OSI_model, OSI model

- [4] Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems, Gordon R. Clarke, Deon Reynders, Edwin Wright

- [5] DNP3 Specification Volume 1, Version 2.02, 15 December 2007

Kratka biografija:

Milan Mitrić je rođen 21.10.1993. godine u Mrkonjić Gradu. Završio je srednju školu SŠC „Petar Kočić“ u Šipovu 2012. god. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu je upisao 2012. godine i završio 2016. godine. Iste godine je upisao master studije na smeru Primenjeno softversko inženjerstvo na Fakultetu tehničkih nauka.

DISTRIBUIRANI FAJL SISTEMI**DISTRIBUTED FILE SYSTEMS**

Teodor Malbašić, Živko Bojović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu su opisane prednosti korišćenja distribuiranih fajl sistema. Predstavljena je i jedna konkretna realizacija u cloud-u: Gluster fajl sistem. Ovaj virtuelni fajl sistem obezbeđuje visok nivo redundanse. Umesto koncepta meta-podataka za skladištenje informacija o fajlovima koristi Distributed Hash Table (DHT) algoritam koji se bazira na informacijama o klijentima. Klijent uz pomoć DHT-a lako izračunava adrese i pristupa fajlovima. Osnovu koncepta Gluster fajl sistema čini implementacija brick-ova i volume-a čime se izoluju podaci različitih korisnika. U radu je prikazan proces testiranja funkcionalnosti dva osnovna tipa Gluster fajl sistema: distribuiranog i repliciranog i izvršena evaluacija dobijenih rezultata.

Abstract – This paper considers the characteristics and advantages of distributed file systems. It also presents a concrete solution of a distributed file system in the cloud: Gluster file system. This virtual file system provides a high level of redundancy. Instead of using the concept of metadata for storing information about files, it uses a Distributed Hash Table (DHT) algorithm, which is based on information about clients. A client easily calculates addresses and accesses files with the help of the DHT. The basis of the concept of Gluster file system is consisted of implementing bricks and volumes, which isolates the data of different users. This paper presents the process of testing the functionality of two main types of Gluster file system: distributed and replicated and it evaluates the given results.

Ključne reči: distribuirani fajl sistem, Gluster fajl sistem, trusted storage pool, brick, volume.

1. UVOD

Kupovina novog hardvera za proširivanje sistemskih resursa računara je oduvek bila relativno skupa aktivnost, a kako potrebe za raznim proširenjima sistema postaju sve veće, ovo može predstavljati ozbiljan problem za korisnike. Još jedan problem bi bio instaliranje mnogobrojnog softvera na novim komponentama sistema. Dakle, prvi problem bi bio trošenje finansijskih resursa, a drugi vremenskih resursa.

Ovi problemi se rešavaju uvođenjem distribuiranih računarskih sistema [2]. Distribuirani računarski sistemi predstavljaju mrežu više računara čije se performanse poboljšavaju zahvaljujući međusobnoj saradnji svih računara koji joj pripadaju.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Živko Bojović, docent.

Ova mreža može biti lokalnog karaktera (LAN mreža), a može biti i mreža sa IP komunikacijom (računarstvo u oblacima, engl. *cloud computing*).

Mreže lokalnog karaktera ne podrazumevaju čuvanje podataka na cloud-u. Podaci (fajlovi) se skladište unutar LAN mreže u lokalne repozitorijume (deljeni direktorijumi). Dakle, korisnici ne moraju imati vezu sa internetom. Kod mreža sa IP komunikacijom, deljeni direktorijumi se nalaze na cloud-u i zbog toga jeste potrebna veza sa internetom. Naglasak u ovom radu stavljen je na distribuirane sisteme u cloud-u.

2. DISTRIBUIRANI FAJL SISTEM

Distribuirani fajl sistemi su deo distribuiranih računarskih sistema, koji korisnicima pružaju mogućnosti skladištenja i deljenja podataka [3] [4]. Osnovni koncept primene ovakvih fajl sistema zasniva se na primeni tehničkog rešenja koje korisnicima pruža mogućnost da podatke rasprostranjene po čitavoj mreži (čak i na različitim geografskim lokacijama), doživljavaju kao da su smešteni na njihovim lokalnim mašinama. Dakle, distribuirani fajl sistem bi trebalo zamisliti kao jedan zajednički fajl sistem svih računara jedne mreže ili bolje rečeno kao fajl sistem koji se „rasprostire“ po celoj mreži računara.

Jedna od bitnih karakteristika distribuiranih fajl sistema je mogućnost **replikacije podataka**. Za razliku od tradicionalnih rešenja kod kojih se svi podaci koje korisnik pokušava dobiti nalaze na jednom serveru (koji mora biti moćan računar), kod implementacije distribuiranih fajl sistema podaci se smeštaju na različite servere, a u najvećem broju slučajeva se i repliciraju (kopiraju) na više servera koji se nazivaju i čvorovima (engl. *node*).

Replikacija podataka ima za cilj poboljšanje performansi sistema i veću **pouzdanost sistema**. Veća pouzdanost sistem podrazumeva da sistem bude **dostupan** korisniku tokom celokupnog trajanja njihove interakcije. Sam termin dostupnosti se odnosi na mogućnost sistema da opsluži korisnika na njegov zahtev u trenutku uspostavljanja njihove interakcije. Dostupnost i pouzdanost sistema predstavljaju „meru“ za otpornost sistema na eventualne otkaze u radu svojih čvorova. Drugim rečima, otkaz jedne mašine podrazumeva mogućnost da preostale preuzmu njene funkcije i obezbede normalno funkcionisanje sistema.

Administracija sistema je veoma važna funkcionalnost, koja kroz visok nivo **transparentnosti** treba da obezbedi podjednaku opterećenosti čvorova saobraćajem, ali i da omogućiti efikasno i gotovo „nevidljivo“ premeštanje fajlova sa jednog čvora na drugi. To znači da korisnici ne bi trebalo da zapažaju promenu fizičke lokacije fajlova, odnosno podataka koje pokušavaju da dobave od servera (koji čine više umreženih čvorova), a i prenos podataka sa

lokalne na udaljenu mašinu bi trebalo da bude potpuno transparentan.

Još jedna važna osobina ovih sistema jeste **skalabilnost**. Nadogradnja infrastrukturnih kapaciteta za potrebe skladištenja novih podataka mora biti urađena na lak način, bez velikih zahvata i većih investicija.

3. GLUSTER FAJL SISTEM

Gluster fajl sistem (*GlusterFS*) je konkretan primer distribuiranog fajl sistema u *cloud*-u [1] [5] [6] [7]. To je mrežni fajl sistem, skalabilan i predviđen za skladištenje ogromnih količina podataka (nekoliko PB). Dizajniran je da opslužuje hiljade klijenata. *GlusterFS* je besplatan softver otvorenog koda i primenljiv je nad uobičajenom fizičkom arhitekturom za servere. Moguće ga je primeniti i na fizičkim i na virtuelnim serverima.

U jednom skalabilnom sistemu, jedan od najvećih izazova jeste praćenje logičkih i fizičkih lokacija podataka i meta-podataka. Za razliku od tradicionalnih rešenja za skladištenje podataka na *cloud*-u, *Gluster*-u nije potreban poseban server za meta-podatke, već on locira fajlove korišćenjem *hash* algoritma. Ovo obezbeđuje bolje performanse, jednostavnu skalabilnost i pouzdanost sistema.

Serveri se sastoje od više umreženih čvorova (računara). Mreža ima jedan ili više *trusted storage pool*-ova. U okviru jednog *pool*-a definisani su direktorijumi pojedinačnih čvorova, koje ti čvorovi dele međusobno (svi čvorovi vide te deljene direktorijume kao jedan jedini zajednički direktorijum).

Brick je osnovna jedinica skladištenja podataka, to je izvozni direktorijum na serveru koji pripada bazenu raspoloživih *storage*-a (*trusted storage pool*). Dakle, *brick* se može protumačiti i kao jedan zaseban fajl sistem, pri čemu je on sam početni direktorijum.

Ukoliko se *brick* kreira na istom fizičkom disku kao operativni sistem, preporučuje se da *brick* bude na posebnoj logičkoj particiji fizičke memorije na kojoj se nalazi, kako u slučaju greške pri korišćenju *Gluster*-a, to ne bi uticalo na sistemske i ostale korisničke podatke na dotičnom čvoru.

GlusterFS je prvenstveno predviđen za *Linux* platforme. Sva konfiguracija (pre svega kreiranje *brick*-ova i njihovo grupisanje u *volume*-e) se obavlja radom u komandnoj liniji (engl. *command line*, CLI). Postoji i mogućnost kreiranja konfiguracionih fajlova, koji administratoru sistema znatno ubrzavaju posao time što su ti fajlovi u obliku šablona.

Još jedna ključna karakteristika *Gluster*-a jeste „samoizlečenje” (engl. *self-healing*). U okviru *GlusterFS*-a postoji mogućnost kreiranja repliciranih *volume*-a. (*Volume* je logička grupa *brick*-ova rasprostranjenih po različitim čvorovima jednog servera.) Reč je o stvaranju rezervnog *volume*-a, koji služi kao *backup* za sve podatke smeštene na posmatranom *volume*-u. Jasno je da ovo može obezbediti sistemu pre svega redundansu, ali i brže operacije čitanja. Kada se dogodi isključenje jednog od *brick*-ova iz mreže (*brick* dospe u *offline* stanje iz nekog razloga), *Gluster* automatski počinje da prati izmene unete na aktivnim *brick*-ovima, kako bi se pokrenuli mehanizmi za *self-healing* čim se isključeni *brick* vrati nazad u mrežu i postane aktivan. Nakon *self-healing*-a svi podaci na ponovo aktivnim *brick*-ovima su ažurirani.

Postoje dve vrste ovog „samoizlečenja” (engl. *self-healing*) i one se izvršavaju paralelno. Prvi vid samoizlečenja je na klijentskoj strani (engl. *client-side self-healing*) i ono, kao što i sam naziv ukazuje na to, biva pobuđeno pokušajem klijenta da izvrši neku operaciju nad fajlom ili direktorijumom koji je označen kao nepotpun (jer je prethodno došlo do isključenja tog fajla, odnosno direktorijuma iz mreže). Pod nepotpunim elementom se smatra onaj koji nije ažuriran u najnoviju verziju. Pored *samoizlečenja na klijentskoj strani* postoji i *samoizlečenje na strani servera* (engl. *server-side*). Ono se ne pobuđuje nekim konkretnim događajem, već se ostvaruje pomoću pozadinskih procesa na svakom od *Gluster* čvorova. Naime, ovi procesi periodično proveravaju da li postoje elementi koji su označeni kao nepotpuni, i ako postoje oni ih „leče” (ažuriraju). Kao rezultat ovog lečenja, dobijaju se potpuni, odnosno ažurirani elementi na svim čvorovima.

GlusterFS omogućava i rebalans podataka. Kada se fajl sistem proširuje horizontalno, dodavanjem novih *brick*-ova, dolazi do promene celokupnog fajl sistema u njegovoj osnovi, što utiče na distribuirani *hash* algoritam koji skladišti novonastale fajlove trudeći se da ih ravnomerno raspodeljuje na *brick*-ove. Novododati *brick*-ovi su inicijalno prazni, dok na starim *brick*-ovima već postoji neki sadržaj. To znači da dodavanjem novih fajlova, *brick*-ovi neće biti podjednako opterećeni, jer će svi prvobitni fajlovi (oni koji su bili prisutni pre proširivanja sistema) ostati na njihovim dotadašnjim lokacijama, dok će novi *brick*-ovi tek početi da se pune novim podacima. Administrator sistema mora da pokrene rebalans podataka, kako bi se svi *brick*-ovi unutar jednog *volume*-a podjednako opteretili.

Pokretanjem rebalansa podataka *Gluster* vrši procenu raspoloživog prostora na *brick*-ovima datog *volume*-a i pozadinskim procesom vrši preraspodelu fajlova, tako što fajlove prenosi sa starih na nove *brick*-ove. Ovo je prilično zahtevan posao i iscrpljuje mnogo sistemskih resursa, pa je zato od najveće važnosti izbor trenutka u kojem treba započeti ovu aktivnost.

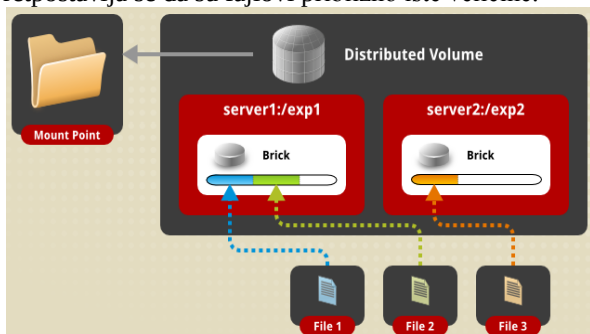
4. TIPOVI GLUSTER FAJL SISTEMA

Tip *Gluster* fajl sistema odgovara tipu *volume*-a. Bitno je prisetiti se da je *volume* jedna logička grupa *brick*-ova, mnoge operacije u okviru konfigurisanja fajl sistema se vrše upravo nad *volume*-om. Tipovi *volume*-a zavise od njihove namene, prema tome, karakteristike određenog tipa *volume*-a možemo tumačiti kroz redundansu koju on pruža sistemu i kroz poboljšanje performansi sistema kojem on doprinosi. U nastavku slede opisi dva osnovna tipa *volume*-a, a to su **distribuirani** i **replicirani**.

4.1. Distribuirani volume

Podrazumevani tip *Gluster volume*-a je distribuirani. Podrazumevan je u smislu da ukoliko se ne naglasi konkretan tip *volume*-a, on će biti kreiran kao distribuirani. Fajlovi se raspodeljuju po svim *brick*-ovima distribuiranog *volume*-a, pri čemu se teži ka što ravnomernijoj raspodeli. Svrha ovakve arhitekture jeste da se što jednostavnije i jeftinije rasporede podaci. Dakle, u ovoj konfiguraciji nema redundanse podataka. Ne postoje kopije podataka, tako da je pouzdanost sistema veoma mala, iako se performanse donekle podižu usled ravnomernog opterećenja na čvorovima. S obzirom na to

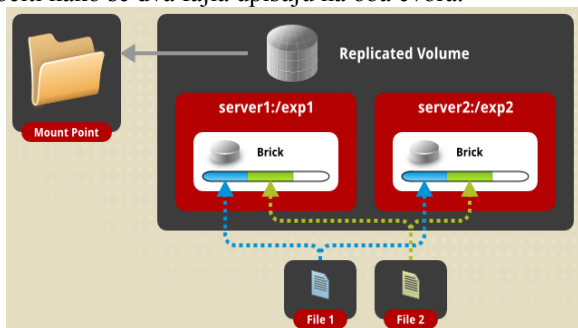
da ne postoje replike (kopije) podataka, nisu prisutni ni *self-healing* mehanizmi. Na slici 1 prikazana su dva *brick*-a sa različitim čvorova. Dok se prva dva fajla smeštaju na prvi čvor (server1), treći fajl se smešta na drugi (server2). Pretpostavlja se da su fajlovi približno iste veličine.



Slika 1. Distribuirani volume [5]

4.2. Replicirani volume

Ko što sam naziv to govori, odlika ovog tipa je ta da on sadrži kopije (replike). Naime, replicirani *volume* sadrži potpune kopije svih podataka na njemu. U replicirani *volume* je potrebno uključiti najmanje dva *brick*-a, kako bi se obezbedile bar dve replike ili bolje reći original i jedna kopija. Na oba *brick*-a se uvek paralelno upisuju svi podaci koji su namenjeni jednom od njih. Ukoliko su neophodne tri replike (original i dve kopije) u *volume* se uključuje tri *brick*-a. Ovakva konfiguracija prevazilazi probleme u vezi sa gubicima podataka, zastupljene kod distribuiranih *volume*-a. Replicirani *volume* obezbeđuje redundansu sistema, pa samim time i pouzdanost podataka na serveru, a ispunjeni su i preduslovi za funkcionisanje *healing*-mehanizama. Na slici 2 se može uočiti kako se dva fajla upisuju na oba čvora.



Slika 2. Replicirani volume [5]

5. IMPLEMENTACIJA GLUSTER FAJL SISTEMA

Za implementaciju *Gluster* fajl sistema korišćen je program *Terminal* operativnog sistema *CentOS 7*, koji predstavlja jednu varijantu *Linux* operativnog sistema. *Terminal* služi za unošenje *Linux* komandi.

U nastavku slede ključni koraci pri realizaciji distribuiranog i repliciranog *volume*-a. Oba *volume*-a će posedovati po dva *brick*-a raspoređena na dva servera.

5.1. Podešavanje *trusted storage pool*-a

Unošenjem naredne komande na jednom serveru (server1), oba servera (server1 i server2) bivaju uključena u *trusted storage pool*:

```
# gluster peer probe <ip-address-server2>
```

Rezultat izvršavanja prethodne komande se može prikazati komandom:

```
# gluster pool list
```

Ova komanda u datom slučaju daje odgovor prikazan na slici 3.

```
[root@localhost teodor]# gluster pool list
JUID                               Hostname      State
959f0e5e-2037-4b8d-a45c-e37cbffc3d52 192.168.0.102 Connected
5d353a66-7aba-4616-b8e0-32b8b9c4334a localhost     Connected
[root@localhost teodor]#
```

Slika 3. Odziv komande # *gluster pool list*

Na slici 3 prikazano je da su oba servera u *trusted pool*-u. Serveru server1 odgovara *Hostname* sa IP adresom, dok je *Hostname* servera server2 označen sa localhost.

5.2. Kreiranje *Gluster volume*-a

Za kreiranje *Gluster volume*-a, potrebno je kreirati direktorijum koji će predstavljati *volume* (npr. *gv0*) i to na *mount*-ovanoj putanji *brick*-a:

```
# mkdir /bricks/brick1/gv0
```

Prethodnu komandu je potrebno izvršiti na oba servera.

Zatim se u zavisnosti od toga koji se tip *volume*-a želi kreirati unese odgovarajuća komanda. Od sada je sve komande potrebno pokretati na samo jednom od servera.

U slučaju kreiranja distribuiranog *volume*-a, komandom:

```
# gluster volume create gv0
<ip-address-server1>`:/bricks/brick1/gv0
<ip-address-server2>`:/bricks/brick1/gv0
```

podigne se distribuirani *volume* sa dva čvora (server1 i server2), dok se replicirani *volume* sa ista ta dva čvora podigne sledećom komandom:

```
# gluster volume create gv0 replica 2
<ip-address-server1>`:/bricks/brick1/gv0
<ip-address-server2>`:/bricks/brick1/gv0
```

Rad *volume*-a se pokreće komandom:

```
# gluster volume start gv0
```

5.3 Funkcionalni testovi

Za potrebe testiranja potrebno je *mount*-ovati *volume* na jedan od servera. Taj server će imati ulogu klijenta. Razlog zbog kojeg se koristi jedan od servera u ulozi klijenta, a ne posebna spoljašnja mašina (tipičan slučaj) jeste taj što bi korišćenje posebne mašine zahtevalo instaliranje dodatnih softverskih paketa na toj mašini.

Mount-ovanje *volume*-a se može izvršiti komandom:

```
# mount -t glusterfs
<ip-address-server1>`:/gv0 /mnt
```

Ova komanda se može uneti na bilo kom od dva servera, bez obzira na to što putanja koja se *mount*-uje odgovara serveru server1.

Naredna *for* petlja kreira 100 tekstualnih fajlova na lokaciji */mnt* sa nazivom *copy-test- $\$i$* , gde *i* označava redni broj kreiranog fajla. Pored samog kreiranja fajlova, u njih se takođe upisuje tekstualni sadržaj fajla *messages* na lokaciji: */var/log/messages*.

```
# for i in `seq -w 1 100`; do cp -rp
/var/log/messages /mnt/copy-test- $\$i$ ; done
```

Sadržaj direktorijuma *gv0* se proverava narednom komandom:

```
# ls /bricks/brick1/gv0
```

Na slikama 4 i 5 je prikazan sadržaj direktorijuma *gv0* u slučaju distribuiranog *volume*-a, na mašinama server1 i server2, respektivno. Slika 6 predstavlja prikaz sadržaja

direktorijuma `gv0` u slučaju repliciranog `volume-a` i na jednoj i na drugoj mašini.

```
[root@localhost teodor]# ls /bricks/brick1/gv0
copy-test-001 copy-test-021 copy-test-034 copy-test-060 copy-test-086
copy-test-004 copy-test-022 copy-test-038 copy-test-063 copy-test-087
copy-test-006 copy-test-023 copy-test-039 copy-test-065 copy-test-088
copy-test-008 copy-test-024 copy-test-041 copy-test-073 copy-test-090
copy-test-011 copy-test-028 copy-test-046 copy-test-077 copy-test-093
copy-test-012 copy-test-029 copy-test-048 copy-test-078 copy-test-094
copy-test-015 copy-test-030 copy-test-051 copy-test-079 copy-test-095
copy-test-016 copy-test-031 copy-test-052 copy-test-081 copy-test-098
copy-test-017 copy-test-032 copy-test-054 copy-test-082 copy-test-099
copy-test-019 copy-test-033 copy-test-057 copy-test-083 copy-test-100
```

Slika 4. Sadržaj direktorijuma `gv0` – server1

```
[root@localhost teodor]# ls /bricks/brick1/gv0
copy-test-002 copy-test-025 copy-test-045 copy-test-062 copy-test-075
copy-test-003 copy-test-026 copy-test-047 copy-test-064 copy-test-076
copy-test-005 copy-test-027 copy-test-049 copy-test-066 copy-test-080
copy-test-007 copy-test-035 copy-test-050 copy-test-067 copy-test-084
copy-test-009 copy-test-036 copy-test-053 copy-test-068 copy-test-085
copy-test-010 copy-test-037 copy-test-055 copy-test-069 copy-test-089
copy-test-013 copy-test-040 copy-test-056 copy-test-070 copy-test-091
copy-test-014 copy-test-042 copy-test-058 copy-test-071 copy-test-092
copy-test-018 copy-test-043 copy-test-059 copy-test-072 copy-test-096
copy-test-020 copy-test-044 copy-test-061 copy-test-074 copy-test-097
```

Slika 5. Sadržaj direktorijuma `gv0` – server2

```
[root@localhost teodor]# ls /bricks/brick2/gv1
copy-test-001 copy-test-021 copy-test-041 copy-test-061 copy-test-081
copy-test-002 copy-test-022 copy-test-042 copy-test-062 copy-test-082
copy-test-003 copy-test-023 copy-test-043 copy-test-063 copy-test-083
copy-test-004 copy-test-024 copy-test-044 copy-test-064 copy-test-084
copy-test-005 copy-test-025 copy-test-045 copy-test-065 copy-test-085
copy-test-006 copy-test-026 copy-test-046 copy-test-066 copy-test-086
copy-test-007 copy-test-027 copy-test-047 copy-test-067 copy-test-087
copy-test-008 copy-test-028 copy-test-048 copy-test-068 copy-test-088
copy-test-009 copy-test-029 copy-test-049 copy-test-069 copy-test-089
copy-test-010 copy-test-030 copy-test-050 copy-test-070 copy-test-090
copy-test-011 copy-test-031 copy-test-051 copy-test-071 copy-test-091
copy-test-012 copy-test-032 copy-test-052 copy-test-072 copy-test-092
copy-test-013 copy-test-033 copy-test-053 copy-test-073 copy-test-093
copy-test-014 copy-test-034 copy-test-054 copy-test-074 copy-test-094
copy-test-015 copy-test-035 copy-test-055 copy-test-075 copy-test-095
copy-test-016 copy-test-036 copy-test-056 copy-test-076 copy-test-096
copy-test-017 copy-test-037 copy-test-057 copy-test-077 copy-test-097
copy-test-018 copy-test-038 copy-test-058 copy-test-078 copy-test-098
copy-test-019 copy-test-039 copy-test-059 copy-test-079 copy-test-099
copy-test-020 copy-test-040 copy-test-060 copy-test-080 copy-test-100
```

Slika 6. Sadržaj direktorijuma `gv0` na oba servera

Rezultati su sasvim očekivani. U slučaju distribuiranog `volume-a` fajlovi su ravnomerno raspodeljeni (po 50 na oba servera), a u slučaju repliciranog su svih 100 fajlova upisani na oba servera.

6. ZAKLJUČAK

Gluster fajl sistem se pokazao kao relativno jednostavan za upotrebu kroz dovoljno intuitivan rad u komandnoj liniji. Uspešno su kreirana i testirana dva tipa distribuiranog fajl sistema: distribuirani `volume` koji odgovara RAID 0 arhitekturi, i replicirani `volume`, koji zadovoljava uslove RAID 1 arhitekture.

Rebalans podataka kod distribuiranih `volume-a` mogao bi se automatizovati, kako administrator sistema ne bi morao svaki put kada dođe do ponovnog uključenja prethodno otkazale mašine ručno inicirati preraspodelu podataka po `brick`-ovima datog `volume-a`.

Dobrodošla je i *Gluster*-ova primena u okviru *Internet of Things* rešenja. Naime, mnogi namenski računarski sistemi, poput onih koji sadrže razne vrste senzora, prikupljaju ogromne količine izmerenih podataka.

Te podatke bi bilo pogodno skladištiti na distribuirani, ali i replicirani `volume Gluster-a`, u zavisnosti od toga da li su tim sistemima preče performanse ili pak redundansa celog sistema.

Ukoliko se želi postići kompromis između ove dve karakteristike, moguće je iskoristiti i kombinaciju distribuiranog i repliciranog `volume-a`, kod koje se distribuirani `volume` sastoji iz `brick`-ova logički organizovanih u više repliciranih `volume-a`.

7. LITERATURA

- [1] Živko Bojović: *Virtualizacija storage-a*, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2016
- [2] Pavel Bžoch: *Distributed File Systems*, University of West Bohemia, Department of Computer Science and Engineering, 2012
- [3] Ondrej Faměra: *Cloud backend deployment at Faculty of informatics MU*, Masarykova univerzita, Faculta informatiky, 2013
- [4] Philip Yeager: *A distributed file system for distributed conferencing system*, University of Florida, 2003
- [5] *GlusterFS Documentation*, <https://Gluster.readthedocs.io/en/latest/>
- [6] *Gluster-Quickstart*, <https://wiki.centos.org/SpecialInterestGroup/Storage/Gluster-Quickstart>
- [7] *Red Hat Gluster Storage*, <https://www.redhat.com/en/technologies/storage/Gluster>

Kratka biografija:



Teodor Malbašić rođen je u Beogradu 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Komunikacione tehnologije i obrada signala odbranio je 2017. god.



Živko Bojović rođen je u Peći 1967. god. Završio je Elektrotehnički fakultet u Prištini, a magistarsku tezu iz oblasti obrade govornog signala i doktorsku disertaciju na temu testiranja kvaliteta servisa u živoj IP MPLS mreži, odbranio je na Fakultetu tehničkih nauka u Novom Sadu gde od 2015. god. radi u zvanju docenta.

KOMPOZICIJA SPOSOBNOSTI KARAKTERA RPG IGRE COMPOSITION OF CHARACTER'S ABILITIES IN RPG

Milan Radeta, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Ovaj rad opisuje fleksibilno korišćenje sposobnosti karaktera igrača korišćenjem podsistema za kompoziciju sposobnosti u jednostavnoj RPG igri.

Abstract – This paper presents flexible use of character's abilities with subsystem for composition of abilities in a simple RPG.

Ključne reči: RPG, multimedija, video igre, Unreal Engine, kompozicija sposobnosti karaktera

1. UVOD

Ovaj rad razmatra šta je potrebno da bi se napravila video igra i analizira sposobnosti i akcije koje karakteri poseduju u *roleplay* igrama, kao i kompoziciju sposobnosti, koja omogućuje izvršavanje više akcija istovremeno. Autor je istraživao, eksperimentisao i radio na implementaciji podsistema kompozicije sposobnosti u okruženju *Unreal Engine*-a, softvera za pravljenje igara. Rad je podeljen u pet glavnih poglavlja, koja opisuju:

- celokupan proces razvoja RPG igre, pri čemu je fokus na sposobnostima karaktera
- sposobnosti RPG karaktera i njihovu kategorizaciju po više svojstava
- programski model podsistema za kompoziciju sposobnosti i dizajn interfejsa kompozitora sposobnosti
- kreiranje, spajanje i svojstva animacija specifičnih sposobnosti
- tehnike kompresije u *Unreal Engine*-u

2. PROCES RAZVOJA RPG IGRE SA NAGLASKOM NA SPOSOBNOSTI KARAKTERA

Bilo da se radi o velikom naslovu ili o nezavisnoj (*indie*) video igri, proces njihovog razvoja je sličan. Video igra mora proći kroz nekoliko faza i potrebno je kreirati različite resurse različitog tipa.

2.1. Faze u procesu razvoja video igre

Proces razvoja igre ne prati sve standarde procesa razvoja softvera, iako postoje sličnosti. Uglavnom se primenjuju metode agilnog razvoja, baziranog na pravljenju iteracija, prototipova. Postoje tri ključne faze procesa razvoja: preprodukcija, produkcija i postprodukcija. Preprodukcija, tj. dizajn faza je faza planiranja projekta u kojoj se osmišljava ideja, koncept igre. Pišu se inicijalni dizajn dokumenti, prema kojima se razvijaju prvi prototipovi igre, a na kraju se vrši tehnički pregled igre. Glavna faza je produkcija, kad učestvuje ceo tim *developer*-a, praveći

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

resurse i programirajući funkcionalnosti. Sastoji se od implementacije i testiranja. Resursi su tekture, modeli i animacije koje stvaraju artisti. Izvršna verzija igre sa novim i ispravljenim funkcionalnostima se testira. Testiranje počinje od prvog igrivog build-a i traje sve dok se igre ne izda. Faza postprodukcije obuhvata održavanje proizvoda, ispravljanje *bug*-ova koji su otkriveni na izdatoj verziji i poboljšavanje igre.

2.2. Specifičnosti razvoja igre RPG žanra

U *role-playing* igrama tok igre zavisi od igračevih odluka u igri. Svoje karaktere igrači stvaraju, birajući rasu, klasu i sposobnosti. Žanr igre određuje i mehanika igre (jedan i/ili više igrača, potezna ili akciona igra...). Programiraju se različiti podsistemi vezani za karaktere (inventar, osobine), kao i veštačka inteligencija karaktera koje ne kontrolišu igrači.

2.3. Razvoj sposobnosti karaktera

Karakter poseduje set sposobnosti koje koristi, pomoću kojih utiču na sebe i okolinu. Dizajnerski zadatak je definisati svojstva sposobnosti na osnovu čega se pravi programski model sistema sposobnosti. Programeri implementiraju funkcionalnost sposobnosti. Animatori prikazuju karaktere kako izvršava neku akciju. Igračima informacije o sposobnostima pruža interfejs, koji osmišljavaju i prave 2D artisti, UI dizajneri i UI programeri.

3. KATEGORIZACIJA SPOSOBNOSTI AVATARA VIRTUELNOG SVETA

Postoji nekoliko kategorizacija sposobnosti prema različitim svojstvima.

3.1. Cena korišćenja

Karakter poseduje 3 atributa, resursa: životna (*health*), magična (*mana*) i fizička energiju (*stamina*). Sposobnosti troše određenu količinu resursa po aktivaciji i/ili dužini trajanja kako bi se izvršile. Mogu trošiti nijedan, jedan ili više tipova energije.

3.2. Aktivnost

Sposobnosti se dele na aktivne i pasivne. Aktivne sposobnosti se izvršavaju korišćenjem dugmeta. Pasivne sposobnosti se stalno izvršavaju, od momenta njihovog otključavanja. Sposobnosti mogu biti i responzivne, tj. da ne mogu da se izvrše dok se ne ispuni neki uslov.

3.3. Dužina trajanja efekta

Efekti sposobnosti se dele na: jednokratne, dugotrajne, privremene i kanališuće. Jednokratni se izvrše jednom nakon aktivacije sposobnosti (npr. udarac). Dugotrajni efekti su stalni (pasivni). Privremeni efekti (engl. *buff*) traju neko vreme nakon što se primene. Kanališući efekti

traju onoliko dugo koliko se sposobnost izvršava. Sposobnost može imati više efekata različitog tipa.

3.4. Ulazni događaji

Sposobnosti se mogu vršiti na pritisak, držanje i/ili otpuštanje dugmeta. Postoje sposobnosti koje zahtevaju jedan pritisak dugmeta za aktivaciju, što započinje mod ciljanja, i drugi pritisak koji će izvršiti sposobnost nad metom. Sposobnosti mogu da se pune držanjem dugmeta, jačajući svoj efekat, a otpuštanjem dugmeta da se konačno izvrše.

3.5. Mogućnost ponovnog korišćenja

Sposobnost može biti onemogućena nakon korišćenja, iako postoji dovoljno resursa. Prema ovom svojstvu, dele se u 3 grupe: 1) stalno omogućene sposobnosti; 2) privremeno onemogućene sposobnosti (poseduju "vreme hlađenja" (engl. *cooldown time*)); 3) rezonivne sposobnosti (onemogućene do ispunjenja određenog uslova).

3.6. Blokiranje ulaza

Aktivacija sposobnosti igrača može da privremeno onemogući *input* igrača, što zavisi od njene animacije i efekta. Animacije mogu zahtevati od igrača da se ne kreće. Određene akcije mogu promeniti položaj igrača, stoga onemogućuju kretanje i napadanje. Igrač može da prekine vršenje sposobnosti na standardnu akcija blokiranja napada.

3.7. Meta

Sposobnosti karaktera ciljaju jednog ili više prijateljskih (uključujući i izvršioca) i/ili neprijateljskih karaktera. Ako ne ciljaju, često generišu nove aktore u prostoru. Sposobnosti mogu ciljati i područja, ciljajući sve karaktere u području.

3.8. Znanje sposobnosti

Napredovanjem karaktera i njegovog nivoa igrač može da otključa i unapređuje sposobnosti u stablu sposobnosti, trošenjem dobijenih poena.

Korenski elementi stabla su bazične sposobnosti, koje rastu i granaju se u razvijenije oblike. Sposobnost može da poseduje uslov za otključavanje.

Što je veći nivo sposobnosti to je njen efekat već. Stablo je često podeljeno u više klasa, tj. može postojati više (pod)stabala. Primer stabla prikazuje Slika 3.1.



Slika 3.1 *Diablo 2 Talent Tree*

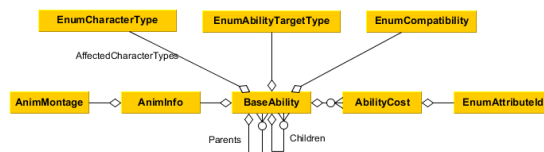
3.9. Kompatibilnost sa drugim sposobnostima

Kompatibilne sposobnosti su one koje se mogu izvršavati istovremeno, što zavisi od sličnosti njihovih animacija. Sposobnosti bez animacija su kompatibilne sa svim ostalim sposobnostima.

4. MODEL I DIZAJN PODSISTEMA ZA KOMPOZICIJU SPOSOBNOSTI

Ovo poglavlje sadrži opis modela sistema sposobnosti, kompozicije sposobnosti i predlog dizajna korisničkog interfejsa za kompoziciju sposobnosti.

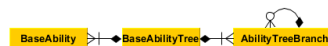
4.1. Sposobnost



Slika 4.1 Sposobnost karaktera - dijagram klasa

BaseAbility (Slika 4.1) je klasa za sposobnost karaktera, model u *Model-View-Controller* sistemu sposobnosti. Sadrži: naziv, nivo, maksimalni nivo, listu kompatibilnih sposobnosti, kategoriju aktivnosti, vreme "hlađenja", tip ciljanja, cenu korišćenja, referencu na roditeljske i dečje sposobnosti u stablu, ikonicu i informacije o animaciji. Poseduje nekoliko metoda koje se pozivaju od strane kontrolera: proveru da li se može izvršiti, da li se njen nivo može povećati/smanjiti, akcije koje se izvršavaju na pritisak/otpuštanje dugmeta.

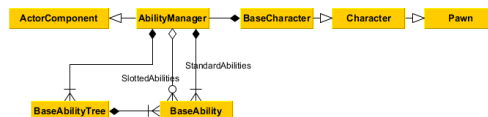
4.2. Stablo sposobnosti



Slika 4.2 Stablo sposobnosti - dijagram klasa

Sposobnost se uglavnom nalazi u nekom stablu sposobnosti (*BaseAbilityTree*) kao čvor (Slika 4.2). Stablo sadrži dva glavna svojstva: listu grana i listu referenci na instance sposobnosti. Grana sadrži klasu određene sposobnosti i listu grana koje se nastavljaju od te klase, odnosno čvora. *BaseAbility* može da sadrži referencu na svog vlasnika, što bi u ovom slučaju bilo stablo *BaseAbilityTree*.

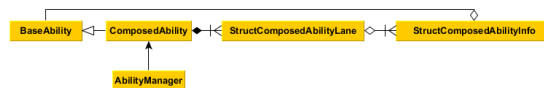
4.3. Upravljač sposobnostima



Slika 4.3 Upravljač sposobnostima - dijagram klasa

Karakter poseduje sposobnosti i upravlja njima pomoću *AbilityManager* (Slika 4.3) komponente karaktera, kontrolera u MVC šablonu. *Pawn* - aktor koji se može instancirati u sceni i prima input. *Character* - *Pawn* koji se kreće. *ActorComponent* - objekat koje se ugrađuje u aktora. *BaseCharacter* je karakter sa *AbilityManager*-om. *AbilityManager* sadrži listu klasa stabala (*AbilityTreeClasses*) i njihove instance, broj skill poena, listu standardnih (napad i blok) i slotovanih sposobnosti, kao i metode za unapređivanje i slotovanje sposobnosti.

4.4. Kompozicija sposobnosti

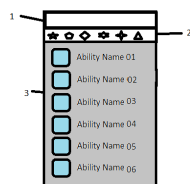


Slika 4.4 Kompozicija sposobnosti - dijagram klasa.

Izvršavanja više sposobnosti pritiskom jednog dugmeta zahteva da se kombinacija sposobnosti predstavi kao jedna sposobnost - *ComposedAbility* (Slika 4.4). Ova klasa nasleđuje *BaseAbility* i niz sekvenci sposobnosti. Sama kompozicija se vrši pomoću metoda *AbilityManager*-a.

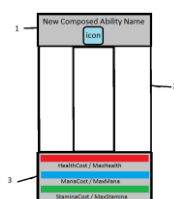
4.5. Korisnički interfejs

Podmeni za kompozitor sposobnosti se sastoji od selektora sposobnosti (engl. *Ability Picker*) i samog kompozitora sposobnosti (engl. *Ability Chooser*), sa dugmadima za kreiranje i uklanjanje kompozicija. *Ability Picker* (Slika 4.5) je pretraživač sposobnosti koje se mogu koristiti u kompoziciji i prevući u *Ability Composer*.



Slika 4.5 *Ability Picker Mockup* - 1) pretraživač sposobnosti po nazivu i opisu; 2) filteri za pretragu; 3) rezultat pretrage, sposobnosti

Ability Composer (Slika 4.6) se sastoji iz zaglavlja, podnožja i glavnog. Zaglavlje sadrži naziv i ikonicu sposobnosti, koje igrač zadaje. Podnožje sadrži podatke o ceni korišćenja kompozicije sposobnosti. Glavni deo je površina za kompoziciju sposobnosti. Površina sadrži 3 trake sposobnosti na koje se mogu prevući sposobnosti iz selektora. Svaka traka se popunjava od gore ka dole, a broj traka je maksimalni broj kompatibilnih sposobnosti koje mogu istovremeno da se vrše, dok dužina iskorišćenog dela traka predstavlja vreme potrebno za izvršavanje sposobnosti. Sposobnosti čije trajanje nije fiksno mogu da se razvlače u vertikalnom pravcu. U slučaju da se dve nekompatibilne sposobnosti paralelno uklopljene te sposobnosti će biti označene kao nevalidne i pokušaj kreiranja kompozicije rezultiraće prijavom greške.



Slika 4.6 *Ability Composer Mockup* - 1) zaglavlje; 2) glavni deo; 3) podnožje

5. ODNOS SPOSOBNOSTI I ANIMACIJA KARAKTERA

5.1. Karakter i animacija

U 3D softveru karakteri se modeliraju, teksturiraju i prave im se skeleti pomoću kojih se animiraju. Skelet je hijerarhija "kostiju". Navezivanje skeleta na model je rigovanje (engl. *rigging*, *rig*) karaktera.

5.2. Priprema i korišćenje animacija

Unreal Engine kosturske modele (engl. *Skeletal Mesh*) i animacije karaktera koje su u *FBX* formatu *export*-ovane iz eksternog 3D softvera može da uvozi, kreirajući *uasset* datoteke koje *Unreal* koristi.

5.2.1. Kosturski model i materijali

Kosturski model se sastoji iz dva dela: modela (kože) i skeleta. Model je skup poligona koji čine površinu karaktera. Skelet je hijerarhija "kostiju" pomoću kojih se animira model. Model bez skeleta je statički model (engl. *Static Mesh*). Uvozom *FBX* datoteke kosturskog modela kreiraju se zasebni *asset*-i za skelet, model, materijali i teksture modela. Modeli mogu deliti zajednički skelet

5.2.2. Animacije

Uvoz animacija se uglavnom vrši odvojeno od uvoza modela karaktera. Svaka animacija se čuva kao zasebna *FBX* datoteka. Uvozom animacija nastaju *asset*-i tipa *Animation Sequence* - sekvence. Skeleti mogu deliti animacije.

5.2.3. Sekvence, montaže i notifikacije

Sekvenca čuva frejmove koji sadrže podatke o transformacijama kostiju. Više sekvenci se može spojiti u montažu (engl. *Animation Montage*), koje mogu ponavljati delove animacije, pustiti neku sekvencu na osnovu nekog uslova. Sposobnost se izvršava na dugme i pušta određenu montažu, koja može da traje određeno (od početka do kraja) ili neodređeno vreme (ponavljajući neku sekvencu). Sekvence mogu sadržati notifikacije, koje okidaju događaje u određenim trenucima animacije. Događaji mogu biti estetskog tipa (vizualni efekti, zvuci) ili ručno isprogramirani. Mogu da obaveste sposobnosti i/ili menadžer sposobnosti koje funkcije da pozovu.

5.2.4. Animacije stanja karaktera

Karakter može biti u određenim stanjima i da pređe u drugo stanje ako je tranzicioni uslov ispunjen. Skup stanja i prelaza se predstavlja preko konačnog automata. Svako stanje sadrži animacionu pozu, konstruisanu od montaža i/ili sekvenci u grafu. Tri osnovna stanja karaktera su neaktivno stanje, stanje šetanja i stanje trčanja, koja se mogu spojiti u jedno, gde se puštaju animacije u zavisnosti od brzine i smera kretanja. Postoji i terminalno stanje smrti karaktera, za koju se pušta animacija umiranja i to samo jednom.

5.2.5. Istovremeno izvršavanje više sposobnosti i spajanje animacija

U šemi animacije karaktera mogu se kreirati grafovi događaja i animacija, spajati animacije i kontrolisati kosti. Da bi se omogućilo igraču da vrši više sposobnosti istovremeno, spajaju se animacije. Montaže se mogu pustiti nad određenom podhijerarhijom kostiju. Za dve različite akcije koje poseduju istu animaciju, pušta se jedna animacija, a vrši dve funkcionalnosti. Nakon analize sposobnosti i njihovih kompatibilnosti sa drugim sposobnostima, menadžer sposobnosti mora da vodi računa o tome nad kojim kostima će puštati montažu.

6. TEHNIKE KOMPRESIJE RPG IGRE U UNREAL ENGINE OKRUŽENJU

Zahtevne igre mogu zauzeti više memorije, ali se teži ka tome da igra bude što je moguće manja, pri tom ne gubeći preterano na kvalitetu.

6.1. Kompresija podataka

Korišćenjem kompresije podataka je moguće datoteku predstaviti u drugom formatu koji zauzima manje memorije. Dva osnovna tipa kompresije su: bez gubitka podataka (engl. *lossless*) i sa gubitkom podataka (engl. *lossy*).

6.2. Teksture

Same teksture se kreiraju u eksternim aplikacijama. Te teksture se potom uvoze u projekat i kompresuju se. Rezolucije tekstura podržane su od 1x1 do 8192x8192. Radi bržeg prikaza, generiše se nekoliko manjih duplikata teksture - mipovi, koji se prikazuju u zavisnosti od toga koliko je tekstura daleko od kamere. Teksture se podrazumevano konvertuju u *DXT* format, *lossy* format kompresije, korišćen uglavnom za difuzne teksture velike rezolucije. Vršiti kompresiju blokova od 4x4 piksela, (4 bita za svaki piksel). Daje izlaz od 64 bita - 2x16 bita se koristi za čuvanje boje, a ostali se koriste u dvobitnoj 4x4 tabeli. *DXT2/3* koriste još 4 bita za alfa kanal svakog piksela. *DXT4/5* koriste za ceo 4x4 blok 2x8 bitova alfa vrednosti i trobitnu 4x4 tabelu, interpolirajući vrednost alfe. Odnos kompresije za *DXT1* je 8:1 za *RGBA* slike, a 6:1 za *RGB*. Za ostale *DXT* tipove je 4:1. *Unreal* nudi i druge tehnike kompresije.

6.3. Audio

Unreal Engine podržava uvoz samo nekompresovanih 16-bitnih WAV audio datoteka. *Wave Audio File Format* (*WAVE*, *WAV*) format je razvio *Microsoft* i *IBM* za čuvanje i razmenu audio podataka na računarima. Mogu da koriste jedan i više kanala. Prilikom uvoza se kompresuju tako da održe sličnost kvaliteta na većini platformi.

6.4. Animacije

Animacije se čuvaju u sekvencama koje sadrže ključne frejmove koje opisuju transformacije kostiju model. Kompresija animacija se vrši analizom pomenutih frejmova i izbacivanjem suvišnih. Ponuđene opcije su: 1) kompresija samih bitova (osnovna opcija, ne izbacuje same frejmove, već menja reprezentaciju u bitovima); 2) najmanje štetna opcija (poništava kompresije animacije, vraća podatke u originalan oblik); 3) uklanjanje svakog drugog frejma (jednostavna, efektna, potencijalno štetna); 4) uklanjanje linearnih frejmova (jednostavna, manje štetna, uklanja frejmove čija vrednost je linearna interpolacija vrednosti okolnih frejmova); 5) uklanjanje trivijalnih frejmova (uklanja frejmove u kojima se transformacija modela ne menja); 6) zasebna kompresija traka (koristi 4) i 1), posmatrajući trake animacije nezavisno); 7) automatska kompresija (podrazumevana, bira najbolju od svih pomenutih kompresija).

6.5. Modeli i nivoi detalja

Modeli objekata u igri da bi se brže prikazali sadrže relativno mali broj poligona. Ukoliko je model dosta udaljen od kamere, nema potrebe prikazati sve poligone. Stoga se pravi više modela sa različitim brojem poligona, kako bi se dalji objekti prikazali sa manje zahtevnim modelom.

6.6. Pakovanje projekta i distribucija

Da bi se napravila izvršna aplikacija projekat prvo mora da se zapakuje. Pakovanje je proces provere da li su kod i sadržaj igre u pravilnom formatu za pokretanje na ciljnoj platformi. Nakon pakovanja, testira se cela igra. Pakovanje priprema igru za distribuciju. Proces se sastoji od 3 osnovna koraka: kompajliranje koda, konverzija sadržaja u ciljni format, grupisanje kompajliranog koda i

pripremljenog sadržaja u instaler. Postoji izbor *build* konfiguracije: *debug* (slabije performanse, podrška za debug), *development* (bolje performanse, minimalna podrška za debug) i *shipping* (bez podrške za debug, opcija za finalni build, sprema za distribuciju).

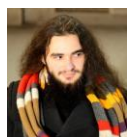
7. ZAKLJUČAK

Jedno parče igre može da se razvija duže vreme i da prođe kroz nekoliko različitih iteracija. Opisani model podsistema nije finalno rešenje, ali dobru početnu tačku. Izvršavanje različitih sposobnosti u isto vreme je predstavljalo najveći izazov, zbog spajanja različitih animacija, koje predstavljaju najranjiviju tačku modela. Prototipski projekat na osnovu kog je rad pisan je rađen samo od strane autora od šablonskih elemenata. Potreban je tim ljudi za pravljenje igre kompleksnog žanra.

8. LITERATURA

- [1] *Character Skills and Abilities*, Stanley Handschuh, *Developing by Me*, 2012
- [2] *Let's Spec Into Talent Trees: A Primer for Game Designers*, Jeff Berry, *envatotuts+*, 2013
- [3] *Dev Diary #2 - The Basic Game Systems*, Brian Heins, *Tyranny Blog*, 2016
- [4] *Skill Scores and Perks aka: Skill Tree*, *TvTropes*, 2016
- [5] *Design Patterns Explained Simply*, *SourceMaking*
- [6] *Game Development Essentials, Third Edition*, Jeannie Novak, *Delmar, Cengage Learning*, 2012
- [7] *From the 80s to Now: The Evolution of Animation in Video Games*, *Pluralsight, Pluralsight*, 2014
- [8] *How Animation for Games is Different from Animation for Movies*, *Pluralsight, Pluralsight*, 2014
- [9] *Making Sense of the 3D Production Pipeline*, Brandon Hix, *Blender Unleashed*, 2016
- [10] *Character Creation and Pipeline in: Ryse, Son of Rome*, Lars Martinsson, *Crytek*
- [11] *Game Character Creation with Blender and Unity*, Chris Totten, *Sybes*
- [12] *Multiple channel audio data and WAVE files*, *Microsoft Developer Network*, 2007
- [13] *WAVE Audio File Format, Library of Congress*, 2012
- [14] *Audio File Format Specifications*, Prof. Peter Kabal, *Multimedia Signal Processing*, 2014
- [15] *Texture Compression*, Philipp Klaus Krause, *ColecoVision*, 2007
- [16] *Texture Compression Techniques and Tips*, Aleen Sherrod, *Gamasutra*, 2005
- [17] *Data Compression*, Debra A. Lelewer, Daniel S. Hirschberg, *UCI Donald Bren School of Information & Computer Sciences*
- [18] *Data-Compression.com*
- [19] *Unreal Engine Community Wiki*

Kratka biografija:



Milan Radeta rođen je u Rumi 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Multimedija i računarske igre odbranio je 2017. god.

PRAĆENJE I ANIMACIJA POKRETA TELA MOTION TRACKING AND ANIMATION OF BODY

Branislav Stanišić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Tema ovog rada je implementacija softverskog rešenja, koje omogućava praćenje pokreta delova tela i animaciju pokreta u realnom vremenu. Praćenje pokreta omogućeno je upotrebom YEI 3- Space Wireless Sensor-a. Animacija pokreta, implementiranog matematičkog modela dela tela, urađena je upotrebom grafičkog jezika OpenGL. Softversko rešenje ima mogućnost povezivanja sa drugim aplikacijama upotrebom Socket-a. Aplikacija na taj način ima ulogu komunikacionog kanala za prosleđivanje podataka dobijenih od senzora drugim aplikacijama.

Abstract – Subject of this paper is implementation of software solution that enables motion tracking of human body parts and motion capture in real time. Motion tracking is enabled using YEI 3 – Space Wireless Sensors. Motion capture of implemented mathematical model of human body parts is done using graphical language OpenGL. Software solution have possibility to connect with other applications using Sockets, that way have purpose of communication channel for sending data received from sensors to other applications.

Ključne reči: praćenje pokreta, animacija pokreta, matematički model tela, YEI 3 – Space Wireless Sensor, OpenGL, Socket

1. UVOD

Praćenje pokreta predstavlja proces snimanja pokreta ljudi i objekata. Koristi se u vojnim tehnologijama, filmskoj industriji, video igrama, medicinskim aplikacijama, robotici itd. U filmskoj industriji i video igrama, odnosi se na snimanje pokreta glumca i korišćenje dobijenih informacija za animaciju digitalnih 2D i 3D modela likova. Prve tehnike praćenja pokreta koristile su snimke sa nekoliko kamera za računanje 3D pozicije. Ove tehnike se zasnivaju na snimanju siluete praćenog objekta. Novije tehnike, koje se takođe zasnivaju na obradi slike, koriste pasivne ili aktivne markere fiksirane na praćeni objekat. Tehnologija inercijalnog praćenja pokreta zasniva se na upotrebi minijaturnih inercijalnih senzora. Senzori koriste podatke sa kompasa, žiroskopa, magnetometra i akcelerometra da bi odredili orijentaciju u prostoru. Podaci od senzora, najčešće se do kontrolnog računara prenose bežičnom komunikacijom.

U ovom radu detaljno je opisan inercijalni senzor YEI 3 – Space Wireless Sensor, koji se koristi za praćenje pokreta delova tela. Sistem se sastoji od komunikacionog adaptera (eng. Dongle) i bežičnih senzora (eng. Wireless Sensors).

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Nikola Jorgovanović, red. prof.

Opisani su mehanizmi koje senzor koristi da bi filtrirao sirove podatke i izračunao svoju orijentaciju u prostoru. Takođe su obuhvaćeni komunikacioni aspekti, kao i režimi komunikacije koji su ostvareni sa senzorom u aplikaciji. Za potrebe vizuelizacije matematičkog modela dela tela i animaciju snimljenog pokreta u realnom vremenu korišten je grafički jezik OpenGL. Funkcionalnosti OpenGL- koje se koriste u ovom rešenju: funkcije za opis primitiva (tačka, linija), funkcije za opis izvora svetla, funkcije zadavanja atributa, funkcije vizuelizacije, skup funkcija za geometrijske transformacije. Matematički model delova tela implementiran je korištenjem sferičnih koordinata i transformacijom vektora nagore. OpenGL koristi vektor nagore kao osnovni pravac prilikom crtanja kupastog oblika.

U aplikaciji je implementirana mogućnost povezivanja sa drugim aplikacijama upotrebom Socket – a. Povezivanje je realizovano na principu server/klijent modela. Podaci dobijeni od senzora u režimu strimovanja se prosleđuju klijentskoj aplikaciji u formi paketa.

2. UPOZNAVANJE SA SISTEMOM YEI 3 – SPACE WIRELESS SENSOR

YEI 3 – Space Wireless Sensor integriše minijaturni, visoko precizni, visoko pouzdani, ARHS (*Attitude and Heading Reference System*), 2.4 GHz DSSS (eng. *Direct Sequence Spread Spectrum*) komunikacioni interfejs i dopunjivu litijumsku bateriju u jedan uređaj. ARHS koristi sledeće senzorne komponente sa tri stepena slobode: žiroskop, akcelerometar i kompas. Ove komponente se koriste u konjukciji sa ugrađenim procesirajućim i filtrirajućim algoritmima da bi odredile apsolutnu i relativnu orijentaciju u realnom vremenu. [1] Ugrađeni multi-referentni vektorski režim povećava preciznost senzora i u velikoj meri smanjuje i kompenzuje greške senzora. Podaci dostupni sa senzora i konfiguracijski parametri senzora mogu da se dobiju korištenjem bežičnog komunikacionog protokola 2.4 GHz DSSS ili USB 2.0 interfejsa. Komande omogućuju pristup neobrađenim podacima, normalizovanim podacima, kao i filtriranim apsolutnim i relativnim podacima. Podaci od senzora, dostupni su u nekoliko formata: kvaternioni (X, Y, Z, W), Ojlerovi uglovi (eng. *Pitch/Roll/Jaw*), rotacione matrice, dva vektora pravca (napred, nagore).

Na slici 1. prikazan je izgled YEI 3 – Space Wireless Sensor – a. Na slici su označeni sledeći delovi senzora:

1. USB konektor – senzor koristi 5 – pinski mali konektor za konekciju sa računarom preko USB porta i za punjenje interne baterije.
2. Ugrađeni naponski prekidač – pomoću ovog prekidača se uključuje i isključuje napajanje senzora iz interne baterije.

3. Ulazni taster 1 – senzor poseduje dva ulazna tastera koji se mogu koristiti u konjukciji sa ostalim senzornim opcijama.
4. LED lampica – senzor koristi RGB LED lampicu kao indikator stanja senzora.
5. Ulazni taster 2.



Slika 1. YEI 3 – Space Wireless Sensor

Na slici 2. prikazan je izgled komunikacionog adaptera. Označeni su sledeći delovi adaptera:

1. USB konektor – adapter koristi 5 – pinski mali konektor za povezivanje sa računarom preko USB porta i za napajanje.
2. LED lampica – adapter koristi RGB LED lampicu kao indikator stanja adaptera.



Slika 2. Wireless Dongle

3. KOMUNIKACIJA

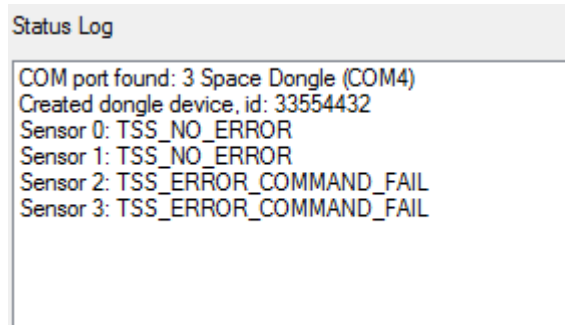
Ovde su navedeni osnovni komunikacioni koncepti i njihove definicije:

- Pan ID predstavlja 16 – bitni broj koji se može dodeliti svakom senzoru i adapteru. Osnovna uloga je grupisanje uređaja u klastere ili mreže.
- Kanal predstavlja frekvenciju na kojoj određeni uređaj komunicira. Postoji ukupno 16 kanala u rasponu 11 – 26. Uređaji koji nisu na istom kanalu ne mogu međusobno komunicirati.
- Svaki uređaj ima fabrički ugrađenu, nepromenljivu adresu (hardverski ID), koja se može naći na poleđini senzora. U komunikaciji, ove adrese se ne koriste direktno već se mapiraju na logičke ID – je adaptera.
- Logički ID ima ulogu da olakša komunikaciju između uređaja. Svaki adapter poseduje ukupno 15 logičkih identifikatora na koje se mogu mapirati hardverske adrese senzora.

3.1 Bežična identifikacija uređaja

U radu, interakcija između programa i senzora realizovana je bežičnom komunikacijom. Proizvođač senzora, firma Yost Labs na svojoj zvaničnoj internet strani nudi mogućnost besplatnog skidanja YEI 3 – Space Sensor C API – ja. Ovaj API je iskorišten u radu za realizaciju komunikacije i interakciju sa sensorima. API se sastoji od zaglavlja u kome se nalaze definicije funkcija, struktura i .dll (eng. *Dynamic Link Library*) u kome se nalazi implementacija.

Da bi se ostvarila bežična komunikacija sa sensorima neophodno je da adapter bude povezan sa host računarom preko USB porta i da naponski prekidači senzora budu uključeni. Priključeni adapter, operativnom sistemu će se predstaviti kao virtuelni COM port, pa je neophodno prvo identifikovati broj COM porta na kome je adapter priključen. Kada dobijemo broj COM porta, identifikujemo adapter tako što ćemo da dobijemo hardverski ID priključenog adaptera. Aplikacija u nastavku će da pokuša da pronađe sve uključene senzore koji se nalaze u dometu adaptera. Enumerator TSS_NO_ERROR koristi se kao identifikator uspešno pronađenog uređaja. TSS_ERROR_COMMAND_FAIL enumerator koristi se kao identifikator neuspešno pronađenog uređaja. Na slici 3. prikazan je izgled statusnog prozora aplikacije nakon identifikacije uređaja.



Slika 3. Statusni prozor

3.2 Režimi bežične komunikacije

U aplikaciji su implementirana dva režima bežične komunikacije:

- Režim prozivanja (eng. *invoke – response*)
- Režim strimovanja (eng. *stream*)

Režim prozivanja podrazumeva, da se senzoru pošalje određena komanda na koju on daje odgovor. Komande u ovom režimu se zadaju senzoru pozivanjem odgovarajućih funkcija API – ja. Funkcije se mogu podeliti, na one koje ne zahtevaju povratne podatke od senzora i funkcije na koje senzor daje odgovor slanjem traženih podataka. Obe vrste funkcija, kao povratnu vrednost daju strukturu TSS_Error, koja je identifikator uspešnosti izvršene komande. Funkcije koje zahtevaju povratne podatke od senzora, prosleđuju referencu u koju se upisuju traženi podaci. Ovaj režim komunikacije koristi se za zadavanje podešavanja senzoru ili za dobijanje podataka vezanih za trenutno stanje senzora.

Komunikacija u režimu strimovanja, podrazumeva dobijanje unapred definisanih podataka od senzora repetitivno. Svaki senzor ima ukupno osam slotova, koji se mogu koristiti u ovom režimu. U aplikaciji zauzet je jedan slot, tako što je definisano da se od senzora traže relativni Ojlerovi uglovi u odnosu na referentnu orijentaciju. Pre

zadavanja komande za početak strimovanja, potrebno je podesiti vremenske intervale. Podešavaju se sledeći vremenski parametri:

- Dužina trajanja sesije strimovanja
- Vremensko kašnjenje početka strimovanja od trenutka zadavanja komande za početak
- Učestalost strimovanja predstavlja vremenski interval između dva dobijena podatka od senzora

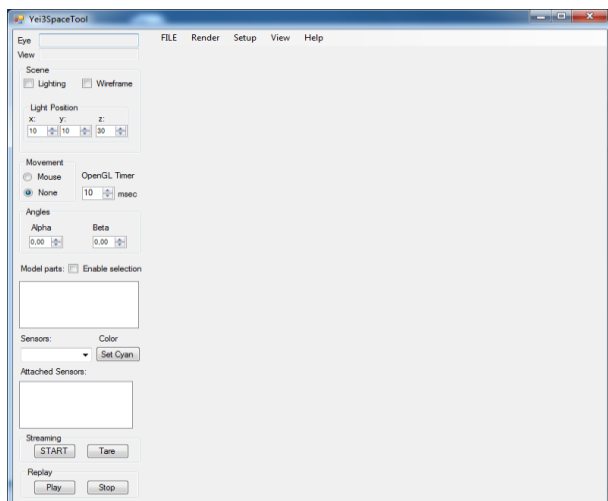
Kada se definišu vrsta podataka u slotovima i vremenski intervali, moguće je zadati komandu za početak strimovanja. Senzori će ostati u ovom režimu, do trenutka kada istekne vreme trajanja sesije ili dok se eksplicitno ne zada komanda za prekid strimovanja.

4. IMPLEMENTACIJA SOFTVERSKOG REŠENJA

Aplikacija je izrađena u razvojnom okruženju Microsoft Visual Studio 2012, koristeći .NET razvojni okvir, koji je vodeći za Windows operativni sistem, za potrebe grafičkog interfejsa. Jezik koji je korišćen za izradu je C++, tj. C++/CLI (eng. *Common Language Infrastructure*) što predstavlja sponu između klasičnog C++-a i .NET programskog okvira. Koristeći ovaj jezik moguće je pisati klasičan C++ kôd, dok su svi objekti koji dolaze iz .NET rukovani pomoću sakupljača otpada (eng. *Garbage Collector*).

Razvojni okvir .NET u radu korišten je za realizaciju grafičkog interfejsa, pomoću koga su definisane forme, paneli za crtanje scene, dijalози za otvaranje i čuvanje fajlova, kao i kontrole, tj. grafički elementi neophodni za interakciju sa aplikacijom. Na slici 4. prikazan je izgled osnovnog prozora aplikacije.

Za potrebe vizuelizacije scene, matematičkog modela delova tela i animaciju pokreta u realnom vremenu korištene su funkcionalnosti OpenGL biblioteka. U projekat aplikacije uvezane su sledeće biblioteke (eng. *Object File Library*): glew32, GIU32, glut32, assimp i OpenGL32. Da bi se omogućila komunikacija između aplikacije i senzora u projekat je uvezana biblioteka: ThreeSpace_API. Zaglavlje: threespace_api.h je uključeno u klasu u kojoj je implementirana komunikacija sa uređajima. U aplikaciji, čuvanje modelovanog dela tela realizovano je u XML formatu. Za potrebe čuvanja i očitavanja XML fajlova u projekat je uključen tinyXML, koji predstavlja C++ XML parser.



Slika 4. Osnovni prozor aplikacije

5. GRAFIČKI JEZIK – OPENGL

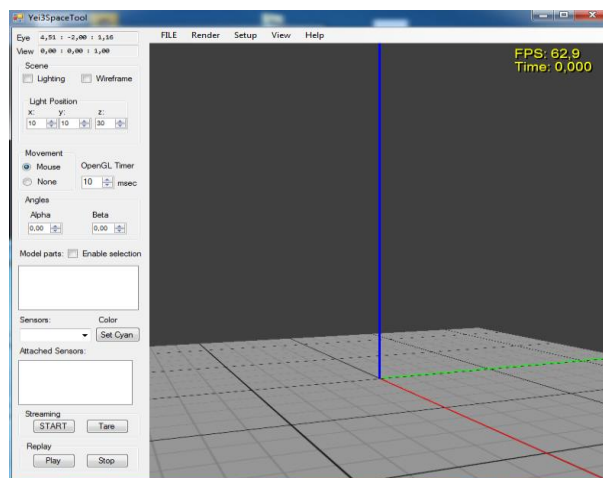
OpenGL predstavlja primarno okruženje prilikom razvoja interaktivnih 2D i 3D aplikacija. Naziv OpenGL predstavlja akronim od eng. *Open Graphic Language*, što znači da je reč o otvorenom grafičkom jeziku, tj. o jeziku koji je dostupan svima. OpenGL podržava većina proizvođača hardvera i softvera. OpenGL moguće je kompajlirati na većini operativnih sistema.

5.1 Crtanje scene

U ovom radu korištene su funkcionalnosti OpenGL-a da bi se omogućilo crtanje scene, kretanje u svetu, crtanje modela tela, animacija pokreta u realnom vremenu i prikazivanje prethodno snimljene animacije pokreta u oflajn režimu. Funkcionalnosti OpenGL-a [2] opisane su preko funkcija njegove biblioteke i mogu se podeliti u sledeće kategorije:

- Funkcije za opisivanje primitiva definišu objekte nižeg nivoa hijerarhije (primitive), koje je grafički podsistem sposoban da prikaže. U OpenGL-u u primitive spadaju tačka, linija, poligon (mnogougao) itd.
- Funkcije za opisivanje izvora svetla služe za definisanje položaja i parametara izvora svetla, koji su raspoređeni u trodimenzionalnoj sceni.
- Funkcije zadavanja atributa, pomoću kojih se definiše kako će da na ekranu izgleda prikazani objekat. Drugim rečima, ako se pomoću primitiva definiše šta se pojavljuje na ekranu, to atributi definišu način prikazivanja na ekranu. OpenGL dozvoljava zadavanje boja, karakteristike materijala, texture, parametre osvetljenja itd.
- Funkcije vizuelizacije omogućavaju da se definiše položaj posmatrača u virtuelnom prostoru, kao i parametri objektiva kamere. Znajući te parametre, sistem može ne samo da pravilno napravi prikaz, već da i iseče objekte koji su van vidnog polja.
- Skup funkcija za geometrijske transformacije omogućavaju da se obave različite transformacije objekata: rotiranje, pomeranje (translacija), skaliranje (promena dimenzija).

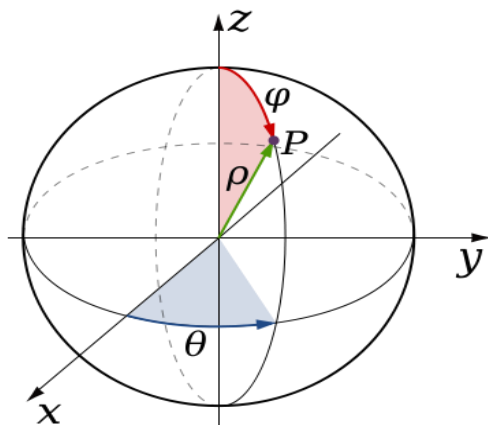
Na slici 5. prikazan je izgled scene tj. virtuelnog sveta u kome se crta modelovani deo tela, animacija pokreta u realnom vremenu i animacija ranije snimljenog pokreta.



Slika 5. Scena

6. MODELOVANJE I ANIMACIJA

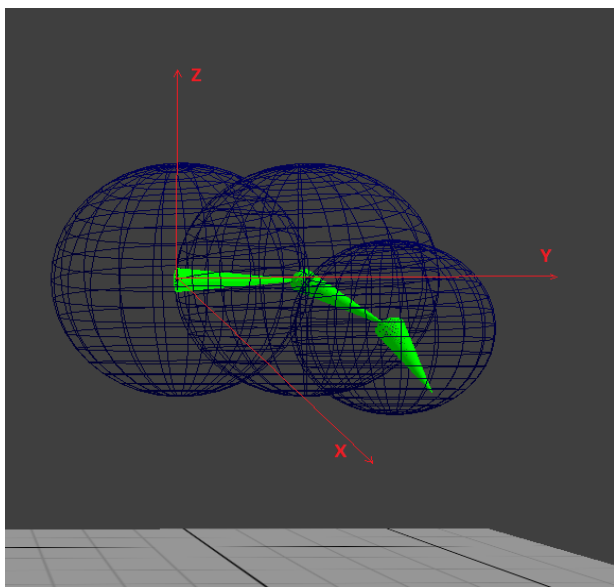
U matematici, sferične koordinate određuju položaj tačke u trodimenzionalnom prostoru pomoću tri varijable (ρ , φ , θ). Na slici 6. prikazan je sferični koordinatni sistem.



Slika 6. Sferični koordinatni sistem

Radijalna udaljenost ili radijalni poteg predstavlja udaljenost tačke od fiksiranog koordinatnog početka i obeležava se sa ρ . Polarni ugao φ , je ugao koji se zatvara između pravca koji prolazi kroz koordinatni početak i tačku u prostoru P, i zenitnog pravca, tj. pravca Z ose sa smerom nagore. Azimutni ugao θ , je ugao koji ortogonalna projekcija na X – Y ravan, pravca koji prolazi kroz koordinatni početak i tačku u prostoru P, zatvara sa pozitivnim pravcem X ose [3].

U ovom radu modelovanje delova tela realizovano je korištenjem sferičnih koordinata. Prilikom modelovanja uvek se kreće od korenskog dela tela tj. dela čijim pomeranjem se pomeraju svi delovi potomci. Na primer, kada se modeluje leva ruka, korenski deo predstavlja leva nadlaktica. U procesu modelovanja korenskog dela zadaje se ofset (X, Y, Z) tj. položaj zgloba u prostoru, dužina dela u centimetrima, uglovi φ i θ koji predstavljaju osnovnu orijentaciju dela. Kada se modeluju delovi potomci, zadaju se isti atributi, osim ofseta koji je određen direktnim delom pretkom, na čiji vrh dodajemo novi deo. Na slici 7. vidimo da se model leve ruke sastoji od tri sferična koordinatna sistema.



Slika 7. Model leve ruke

Pre početka animacije pokreta, potrebno je da deo tela čiji pokret animiramo bude u istom položaju kao i model u OpenGL sceni. Taj položaj se senzorima zadaje kao referentni pravac. Pokret predstavlja ugaoni pomeraj u odnosu na referentni pravac. Uglovi dobijeni od senzora se aritmetički sabiraju sa uglovima osnovne orijentacije modela dela tela, čiji pokret snimamo senzorom. Funkcija koja je zadužena za transformaciju koordinata realizovana je kao rekurzivna funkcija i poziva se nad korenskim delom modela. U funkciji, uglovi dobijeni sabiranjem koriste se za transformaciju sferičnih koordinata u Kartezijanske (X, Y, Z) koordinate. Dobijene Kartezijanske koordinate predstavljaju ofset koordinatnog početka dela potomka. Na ovaj način koordinatni sistemi potomaka se kreću u prostoru a njihov položaj je određen direktnim delom pretkom.

7. POVEZIVANJE SA APLIKACIJAMA

U ovom radu povezivanje i komunikacija sa drugim aplikacijama realizovana je upotrebom Socket tehnologije. Socket-i se najčešće koriste u komunikaciji između dva računara u mreži, ali mogu, kao što je implementirano u ovom radu i za komunikaciju između dva procesa. Komunikacija je realizovana upotrebom server/klijent modela [4]. Prilikom startovanja servera definiše se broj porta na kome server čeka (sluša) pristigle zahteve. Operacija slušanja je blokirajuća pa je zbog toga server realizovan u zasebnoj programskoj niti. Nakon pristizanja zahteva klijenta, konekcija se uspostavlja. Komunikacija između aplikacija se koristi isključivo u režimu strimovanja, gde se podaci dobijeni od senzora prosleđuju klijentu.

8. ZAKLJUČAK

Ovaj rad predstavlja jedno od rešenja koje omogućava praćenje pokreta upotrebom inercijalnih senzora i animaciju pokreta u realnom vremenu.

Primena ovog rešenja je raznovrsna i od posebnog je značaja za oblast medicine i biomedicinskog inženjerstva. Može da se koristi u rehabilitaciji, sa ciljem praćenja procesa oporavka nakon teških povreda i moždanih udara. U sportskoj medicini moguća je primena sa ciljem praćenja efikasnosti trenajnog procesa. Takođe jedna od važnih oblasti primene je praćenje pokreta veštačkih delova tela – proteza.

9. LITERATURA

- [1] Yost Labs, „3- Space Wireless Sensor User's Manual“
- [2] Dragan Cvetković, Dragan Marković, Nenad Dulanović, „OpenGL Praktikum“, Računarski Fakultet, Beograd, 2006.
- [3] John Vince, „Mathematics for Computer Graphics“, 2nd Edition, Springer – Verlag, 2006.
- [4] Andrew Krowczyk, Vinod Kumar, Nauman Laghari, „NET Network Programming“, Wrox, 2009.

Kratka biografija:



Branislav Stanišić rođen je u Sarajevu 1986. god Master rad na Fakultetu tehničkih nauka iz oblast Elektrotehnike i računarstva – Automatika upravljanje sistemima odbranio je 2017.god.

PREZENTACIJA REZULTATA DMS FUNKCIJA NA POJEDNOSTAVLJENOM PRIKAZU MREŽE**RESULTS PRESENTATION OF DMS FUNCTIONS ON SIMPLIFIED NETWORK VIEW**

Miloš Mičić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je obrađen problem prezentacije rezultata funkcija distributivnog menadžment sistema (DMS) na pojednostavljenom prikazu elektroenergetske distributivne mreže. Na osnovu identifikovanih problema dat je predlog njihovih rešenja. Rezultati DMS funkcija koji obuhvataju identifikovane probleme prikazani su na potpunom i pojednostavljenom prikazu test mreže.

Abstract – This paper presents problem of presentation of distribution management system (DMS) functions results on simplified power distribution network view. After problems identifications the problems solutions are proposed. Results of DMS functions which include identified problems are represented on complete and simplified view of test network.

Cljučne reči: *Distributivne mreže, model mreže, grafički prikaz mreže, sekcije, supersekcije.*

1. UVOD

Model elektroenergetske mreže predstavlja osnov za sve elektroenergetske proračune i grafički prikaz mreže [1]. Prilikom prikaza distributivne mreže ili nekog njenog većeg dela potrebno je prikazati veliki broj elemenata.

U nekim primerima, u cilju jednostavnijeg prikaza i bolje preglednosti, bolje je prikazati samo najznačajnije elemente. U tu svrhu u radu je izložen algoritam za kreiranje minimalno potrebnog prikaza mreže, na kojem su obuhvaćeni i prikazani samo najbitniji elementi i elementi koji njih povezuju. Ovaj prikaz mreže je nazvan pojednostavljeni prikaz, dok je prikaz na kojem su vidljivi svi elementi mreže nazvan potpuni prikaz. Model mreže na osnovu kojeg se generiše pojednostavljeni prikaz se naziva supermodel, dok je osnovni element supermodela supersekcija. Ona predstavlja sintezu redno povezanih elemenata kao što su sekcije, čvorovi i rasklopni uređaji.

Većina DMS funkcija rezultate proračuna ili njihovu obrađenu varijantu prikazuje na prikazu mreže na elementima mreže za koje je izvršena, pružajući korisniku vizuelnu i intuitivnu informaciju. Funkcija prilikom proračuna uvažava sve modelovane elemente mreže od interesa i za njih vraća određene rezultate.

Pri pojednostavljenom prikazu, značajan broj elemenata mreže nije vidljiv, pa je veliki broj informacija (rezultata DMS proračuna) potisnut. U tom slučaju nameće se pitanje kakvi su rezultati na tim elementima.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je dr Goran Švenda, red.prof.

Može se desiti da rezultati na potisnutim, neprikazanim elementima imaju veću važnost od rezultata na prikazanim elementima i da u tom slučaju korisnik dobija nepotpunu, ili čak pogrešnu informaciju, odnosno da pojednostavljeni prikaz mreže gubi smisao. Upravo ovi problemi predstavljaju osnovnu temu ovog rada.

2. POJEDNOSTAVLJENI PRIKAZ MREŽE

Povezanost elemenata elektroenergetske mreže (topologija), bilo prenosne, bilo distributivne, se modeluje pomoću grafa [1]. Graf se sastoji od čvorova koji su povezani granama [2]. Grane grafa elektroenergetske mreže su sekcije, rasklopna oprema, transformatori [1]. Što se tiče čvorova modela distributivne mreže, postoje čvorovi koji su tipa: "connectivity node" i čvorovi tipa "bus node" [1]. Prvi predstavlja mesto gde su krajevi provodne opreme povezani nultom impedansom, a drugi izvedeni čvor connectivity node-a, koji modeluje važniji čvor, kao što su npr. sabirnice [1].

Da bi se od potpunog modela mreže formirao supermodel, potrebno je na potpuni model mreže primeniti algoritam segmenta izvoda [3]. Algoritam obrađuje svaki izvod posebno, tako da je od interesa za razmatranje predstava izvoda pomoću grafa. Dva osnovna koraka algoritma su:

1. formiranje minimalnog grafa koji sadrži sve važne elemente;
2. formiranje supersekcija.

Rezultat primene algoritma na potpuni model izvoda je podela izvoda na segmente [3]. Postoje 3 vrste segmenata izvoda:

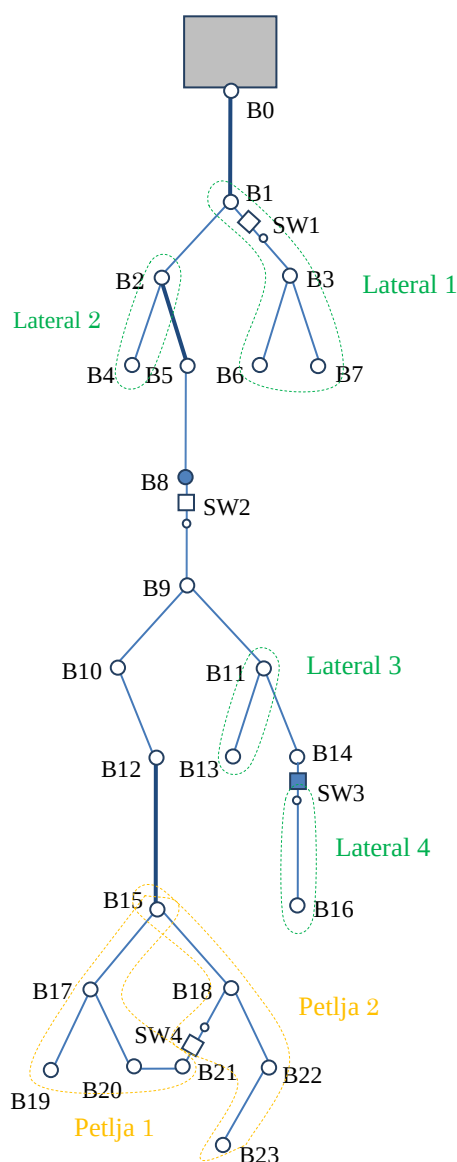
- supersekcije;
- petlje;
- laterali.

Na pojednostavljenom prikazu mreže su vidljive samo supersekcije. One predstavljaju grane supermodela. Supersekcija se sastoji od dva granična čvora i sekcija i rasklopne opreme na putanji između ta dva čvora. Čvorovi supermodela se nazivaju intersekcije [3].

Na slici 1 je predstavljen potpuni prikaz izvoda, na slici 2 njegov pojednostavljeni prikaz, a na slici 3 zajednička legenda.

Na potpunom prikazu izvoda sa slike 1 prikazani su identifikatori čvorova (bus node-ova) i prekidačke opreme. Identifikatori sekcija nisu prikazani zbog preglednosti, ali se smatra da se sastoje od slova „S“ i broja čvora koji posmatrana sekcija napaja (na taj način su identifikatori sekcija jednoznačno određeni).

U tabeli 1 su prikazane supersekcije sa slike 2 kao i elementi sa slike 1 koji ulaze u njihov sastav.



Slika 1. Potpuni prikaz izvoda

Tabela 1. Supersekcije izvoda sa slike 1

Supersekcija	Elementi supersekcije
SS1	B0, S1, S2, S5, S8, B8
SS2	B8, SW2, S9, B9
SS3	B9, S10, S12, S15, B15
SS4	B9, S11, S14, B14

3. VERIFIKACIJA REŠENJA PROBLEMA

Osnovni problem ovog rada, može da se podeli na dva potproblema koji se rešavaju nezavisno:

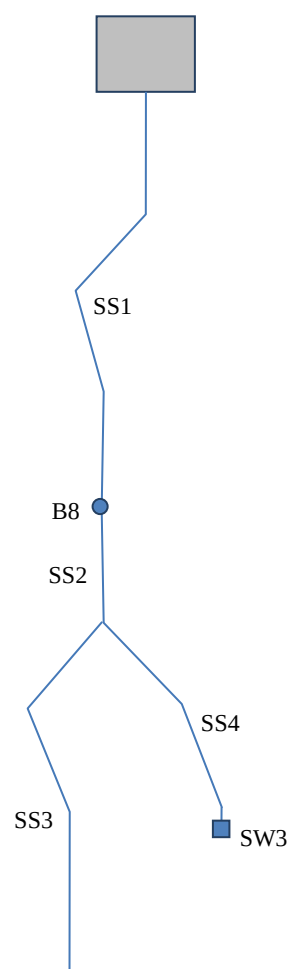
1. Prikaz rezultata na supersekcijama;
2. Prikaz rezultata za elemente koji nisu vidljivi na pojednostavljenom prikazu.

3.1. Prikaz rezultata na supersekcijama

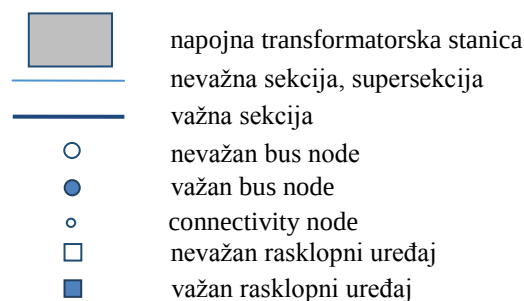
Rezultati na elementima mreže na potpunom prikazu mogu da se prezentuju na sledeća 3 načina:

- prikaz brojnih vrednosti;
- bojenje i markiranje elemenata;
- prikaz zastavica.

Koristeći se ovim rezultatima potrebno je formirati



Slika 2. Pojednostavljeni prikaz izvoda



Slika 3. Legenda slike 1 i 2

rezultate koji će se prezentovati na pojednostavljenom prikazu. Značajan broj elemenata sa potpunog prikaza nije vidljiv na pojednostavljenom i na pojednostavljenom prikazu postoje supersekcije koje nisu modeli realnih elemenata mreže, nego predstavljaju sintezu elemenata modela.

U slučaju prikaza brojnih vrednosti, svaka sekcija u okviru supersekcije ima svoju vrednost razmatrane veličine. Na osnovu tih vrednosti potrebno je odrediti vrednost iste veličine supersekcije. Za vrednost određene veličine supersekcije može se proglašiti:

- vrednost razmatrane veličine topološki prve sekcije koja čini supersekciju;
- najkritičnija vrednost razmatrane veličine na nivou supersekcije (maksimalna ili minimalna vrednost);

- suma vrednosti razmatrane veličine sekcija koje čine supersekciju.

Na slikama 4 i 5 prezentovani su rezultati tokova aktivnih snaga u vidu brojnih vrednosti na potpunom i pojednostavljenom prikazu mreže. Za brojne vrednosti aktivnih snaga supersekcija su proglašene brojne vrednosti topološki prvih sekcija koje čine supersekciju.

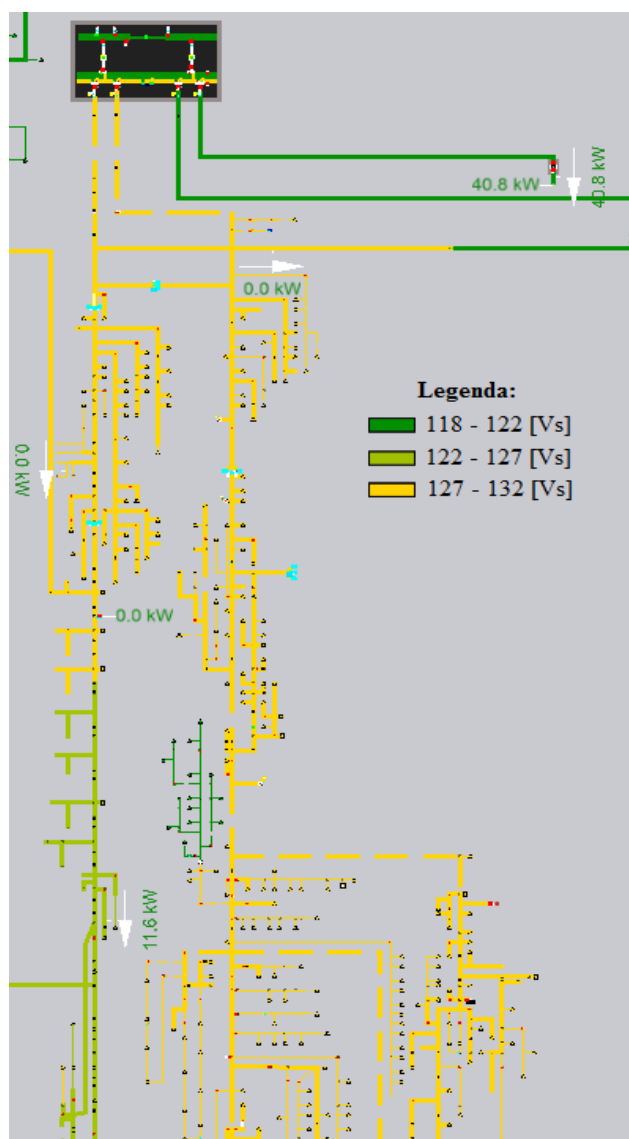
Što se tiče bojenja i markiranja elemenata izvoda, moguća su dva slučaja:

- obojene (markirane) su sve sekcije izvoda, jer se bojenje (markiranje) vrši na osnovu vrednosti

određene veličine;

- markirane su samo pojedine sekcije izvoda, jer se na njima predlaže određena akcija ili su afektovane predloženim prekidačkim akcijama.

U prvom slučaju potrebno je odrediti brojne vrednosti supersekcija. To se može uraditi na neki od načina izloženih prilikom obrade prikaza brojnih vrednosti. Na slikama 4 i 5, dat je prikaz rezultata tokova snaga (TS) na potpunom i pojednostavljenom prikazu, respektivno. Supersekcije su obojene na osnovu najkritičnijih vrednosti modula napona.

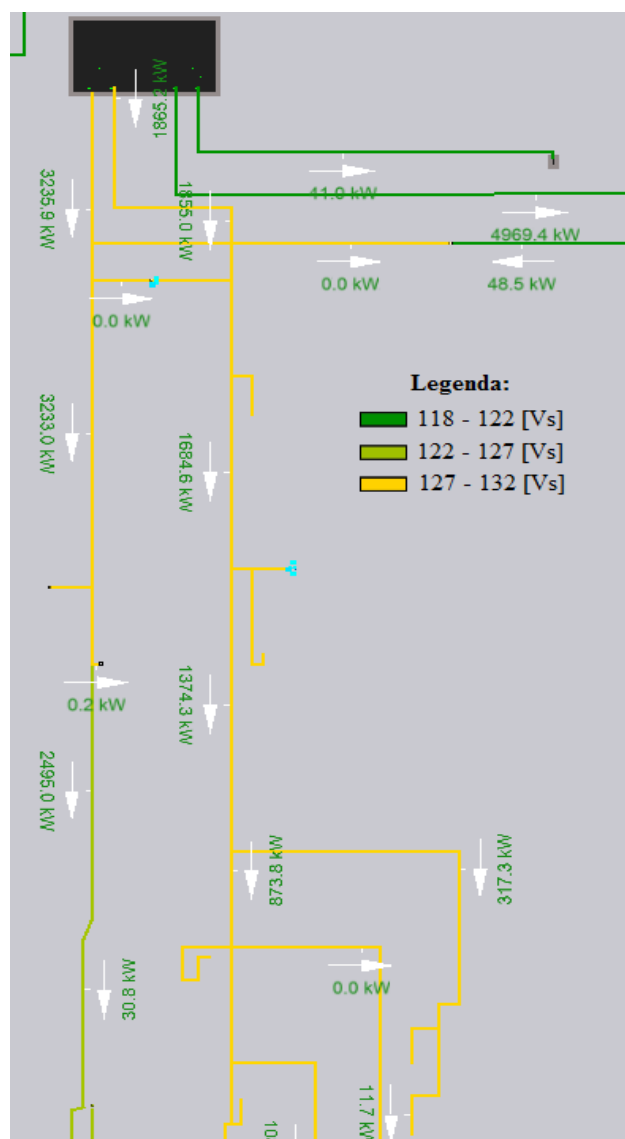


Slika 4. Rezultati TS na potpunom prikazu mreže

U slučaju kada su markirane samo pojedine sekcije izvoda, jer se na njima predlažu određene akcije, ili su afektovane predloženim prekidačkim akcijama, najlogičnije rešenje je markirati supersekcije koje sadrže barem jednu markiranu sekciju.

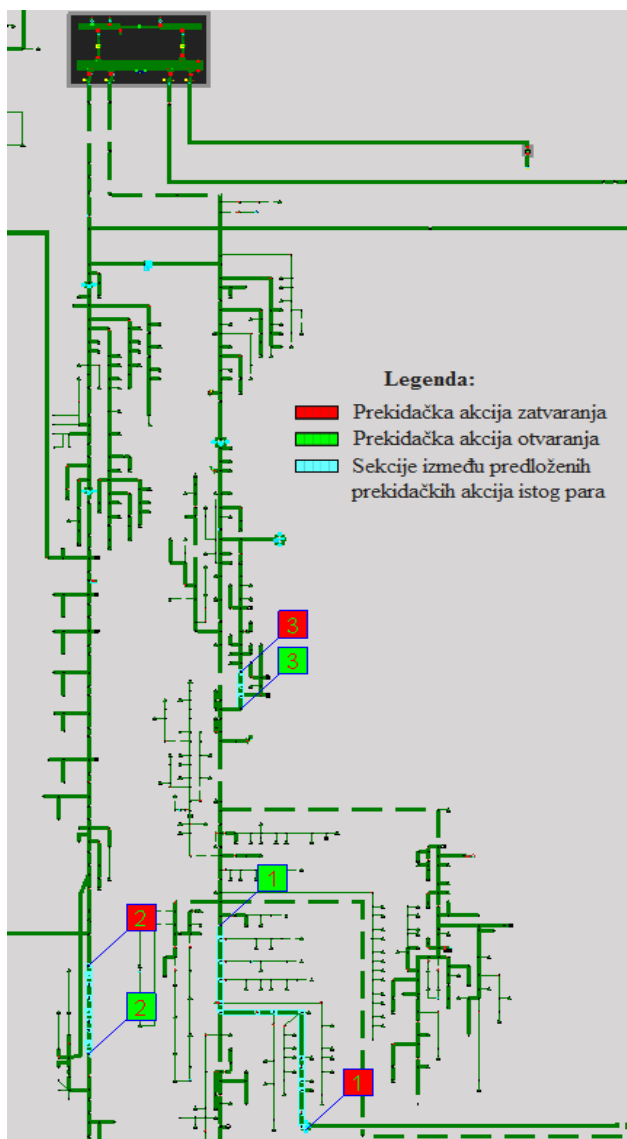
Na slikama 6 i 7 predstavljeni su rezultati funkcije rekonfiguracija mreže na potpunom i pojednostavljenom prikazu mreže, respektivno. Na pojednostavljenom prikazu, markirane su supersekcije koje sadrže barem jednu markiranu sekciju na potpunom prikazu.

Za rasklopnu opremu i releje je potrebno prikazati



Slika 5. Rezultati TS na pojednostavljenom prikazu mreže

zastavice na pojednostavljenom prikazu ukoliko se nalaze na supersekcijama. Kada je postavljeno više zastavica na sekcije ili čvorove jedne supersekcije, potrebno ih je objediniti u jednu zastavicu na nivou supersekcije. Što se tiče funkcije rekonfiguracija mreže, na potpunom prikazu mreže se postavljaju zastavice na rasklopnu opremu, slika 6. Zastavice se postavljaju u paru i jedna od njih označava prekidačku akciju zatvaranja, a druga prekidačku akciju otvaranja, kako bi se zadržala radikalna konfiguracija mreže. Na pojednostavljenom prikazu zastavice para su objedinjene u jednu zastavicu koja je postavljena na markirane supersekcije, slika 7.



Slika 6. Rezultati funkcije rekonfiguracija mreže na potpunom prikazu

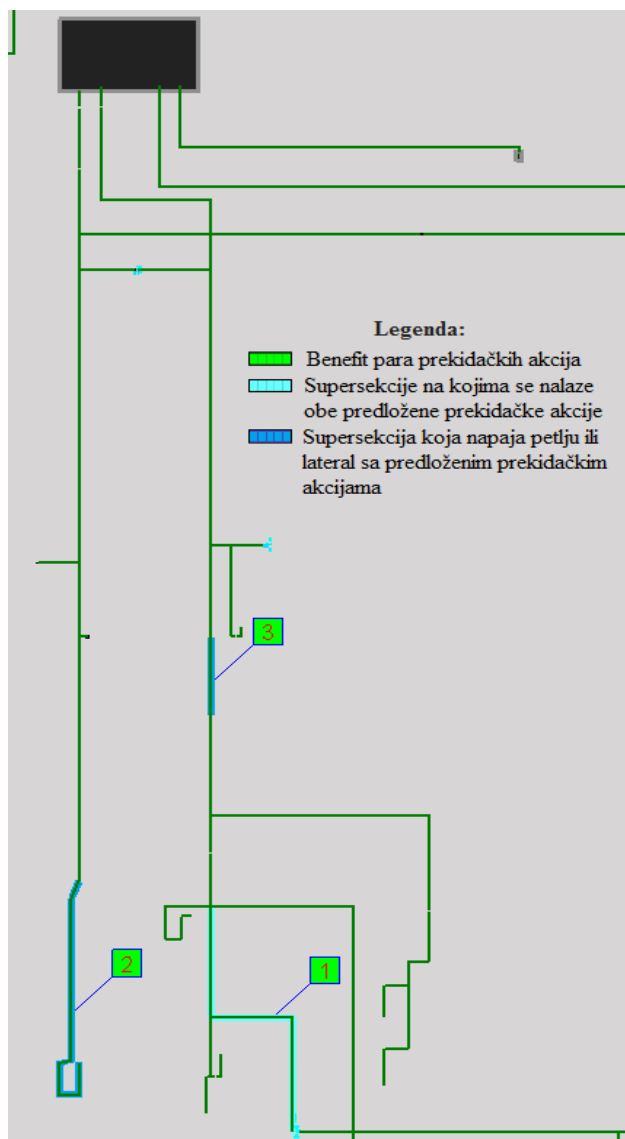
3.2 Prikaz rezultata za elemente koji nisu vidljivi na pojednostavljenom prikazu mreže

Ovaj deo problema se odnosi na funkcije koje označavaju pojedine elemente izvoda, bilo markiranjem, bilo postavljanjem zastavica na elemente. Što se tiče funkcija koje prikazuju rezultate za sve elemente (ili za većinu elemenata) izvoda, rezultati za elemente petlje i laterala ne treba da budu predstavljeni na pojednostavljenom prikazu mreže. Funkcije koje označavaju pojedine elemente izvoda, mogu da se podele u dve grupe:

- funkcije koje računaju benefit predloženih akcija;
- funkcije koje ne računaju nikakav benefit.

Za funkcije prve grupe, rezultati petlje i laterala treba da se prikažu na pojednostavljenom prikazu mreže, posebno u slučaju kada akcije na tim segmentima donose veći benefit od akcija na supersekcijama. Jedno od mogućih rešenja je označavanje supersekcija koje napajaju petlje ili laterale na kojima se nalaze predložene akcije koje donose benefit. Ovo je predstavljeno na slici 7.

Rezultati na petljama i lateralima funkcija koje ne računaju benefit ne moraju da se prikazuju na pojednostavljenom prikazu mreže.



Slika 7. Rezultati funkcije rekonfiguracija mreže na pojednostavljenom prikazu

4. ZAKLJUČAK

U radu je predstavljen algoritam pomoću kojeg se generiše pojednostavljeni prikaz distributivne mreže koji sadrži sve važne elemente. Zatim je obrađen problem prezentacije rezultata DMS funkcija na pojednostavljenom prikazu, tako što su identifikovani načini prezentacije rezultata na potpunom prikazu mreže i prilagođeni pojednostavljenom prikazu. Pri tome, za korisnika su sačuvani i prikazani najznačajniji rezultati.

5. LITERATURA

- [1] SE DMS NS, *Network Model*, Novi Sad, 2017.
- [2] Reinhard Diestel, *Graph Theory*, Springer, New York, 2005.
- [3] SE DMS NS, *Decluttering – Feeder Segment Algorithm*, Novi Sad, 2017.

Kratka biografija:



Miloš Mičić rođen je u Sremskoj Mitrovici 1992. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektroenergetski sistemi odbranio je 2017. god.

**OPTOELEKTRONSKI GENERATORI NASUMIČNIH BROJEVA REALIZOVANI
U HV CMOS 0.35 μm TEHNOLOGIJI****RANDOM NUMBER OPTOELECTRONIC GENERATORS IMPLEMENTED
IN HV CMOS 0.35 μm TECHNOLOGY**

Mihailo Drljača, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Četiri integrisana optoelektronska kola su realizovana u HV CMOS 0.35 μm tehnologiji. Sastoje se od izvora svetlosti, kružne diode koja generiše fotone, i fotodetektora, prstenaste lavinske diode oko izvora svetlosti. Vremenski intervali između detekcija se mogu posmatrati kao nasumični događaji. U programskom paketu Comsol Multiphysics je simulirano ponašanje oblasti osiromašenja i jačina električnog polja. Na RF ispitnoj stanici su merene strujno-naponske karakteristike optičkog uređaja. Simulacije i eksperimentalni podaci pokazuju da se karakteristike uređaja mogu poboljšati korišćenjem jama pri projektovanju lavinske diode. Najbolje performanse je pokazala dioda sa p jamom, dok performanse diode sa dubokom jamom moraju da se poboljšaju udaljavanjem fotodetektora od diode. Karakteristike uređaja realizovanih bez jama nisu zadovoljavajuće.

Ključne reči: Integrisana optoelektronika, Si-CMOS-LED, Lavinska dioda.

Abstract – Four different integrated optoelectronic circuits are implemented in HV CMOS 0.35 μm technology. Each circuit consists of light source, circular diode that generates photons and photodetector, ring shaped avalanche diode around the light source. Time intervals between the detections can be considered as random events. Using Comsol Multiphysics software package depletion region and electric field are simulated. Wafer probe station is used to measure I-V characteristics of devices. Simulations and experimental results showed that using wells to design avalanche diode enhances device performance. Best performance showed diode with p well, while diode designed using deep p well had to be on greater distance. Characteristics of devices designed without wells are not satisfactory.

Key words: Integrated optoelectronics, Si-CMOS-LED, Avalanche diode.

1. UVOD

U poslednje vreme merenje vremena detekcije fotona je korišćeno za projektovanje kvantnih generatora slučajnih brojeva (QRNG – quantum random number generator). Struktura ovog tipa QRNG se sastoji od laserske diode koja ima ulogu izvora svetlosti i lavinske diode koja ima ulogu detektora pojedinačnih fotona (SPAD – single photon avalanche diode). Vremenski intervali između uzastopnih

detekcija se čuvaju kao „sirovi“ podaci koje je potrebno dalje obraditi. Svi sistemi su zasnovani na korišćenju LED (LED – light emitting diode) dioda za generisanje izvora svetlosti. Izvori svetlosti do pre par godina nisu bili fabrikovani u silicijumu, odnosno u standardnoj CMOS tehnologiji [1]. Grupa za mikroelektroniku sa Tehničkog univerziteta u Beču je proteklih godina objavila više radova u kojima su predstavljeni generatori slučajnih brojeva, realizovani u potpunosti u standardnoj 0.35 μm HV CMOS tehnologiji. Sistem se sastoji od Si-CMOS-LED diode koja generiše fotone i SPAD diode koja radi kao fotodetektor [1], [2]. Si-LED je inverzno polarisana i radi u lavinskom režimu. Izvor svetlosti je kružan i emituje fotone u svim pravcima koje detektuje SPAD prsten oko diode.

U ovom radu su ispitivane performanse četiri različite strukture optičkih uređaja projektovanih od strane grupe za mikroelektroniku sa Tehničkog univerziteta u Beču.

2. STRUKTURA OPTOELEKTRONSKOG UREĐAJA

Na slikama 1, 2, 3 i 4 su prikazane polovine poprečnih preseka optoelektronskih uređaja. Uređaji su simetrični u odnosu na centar.

Inverzno polarisana kružna LED dioda poluprečnika 15.4 μm se nalazi u centru. Vidljiva i infracrvena svetlost se emituju iz spojeva p++/n jama koji su na udaljenosti 12.35 μm od centra. Oko diode se nalazi duboka n jama koja obezbeđuje izolaciju diode od supstrata. Izvor svetlosti je u svim slučajevima isti.

Oko LED se nalazi lavinska dioda, SPAD koja ima ulogu fotodetektora. SPAD je inverzno polarisana iznad napona proboja. Pri ovom naponu električno polje u oblasti osiromašenja je toliko veliko da čak i jedan injektovani nosilac može da pokrene lavinu. Posledica je veliko pojačanje jako malih signala, što na izlazu daje struju reda mA. Pokretanje lavine zavisi od nasumičnih sudara nosilaca naelektrisanja i mogu je izazvati termički generisani nosioci i nosioci generisani tunelovanjem u oblasti osiromašenja. Osim zbog pomenutih parazitnih efekata, do pokretanja lavine može doći i zbog oslobađanja nosilaca naelektrisanja prethodno zarobljenih u defektima kristalne rešetke [2].

U prve dve realizacije prikazane na slikama 1 i 2 prstenasta SPAD je realizovana direktno između n++ katode i supstrata. Katoda širine 1.2 μm se nalazi oko izvora svetlosti na rastojanju 27.9 μm i 22.9 μm od centra uređaja. Anodni kontakt SPADa koji je ujedno i kontakt za polarizaciju supstrata se nalazi na 10.4 μm od n++ katode.

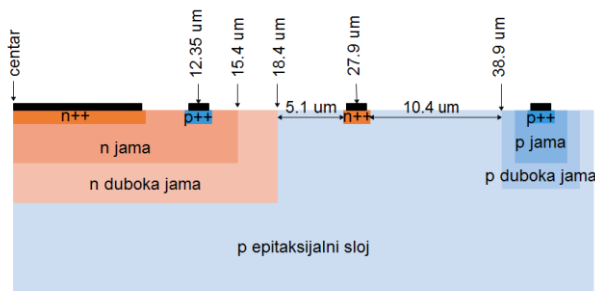
NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Jelena Radić, docent.

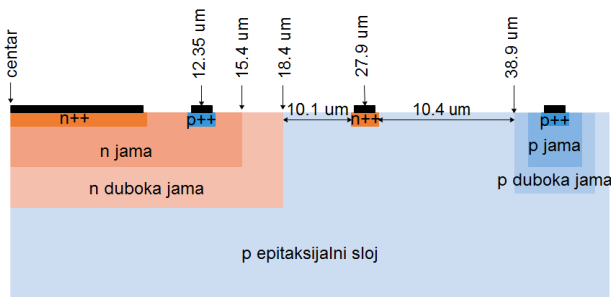
Kontakt je realizovan slojevima p ++/ p jama/ p duboka jama da bi se smanjila njegova otpornost.

U drugim realizacijama prikazanim na slikama 3 i 4 SPAD je realizovana između n++ katode i p jame/duboke p jame koje treba da poboljšaju performanse SPADa.

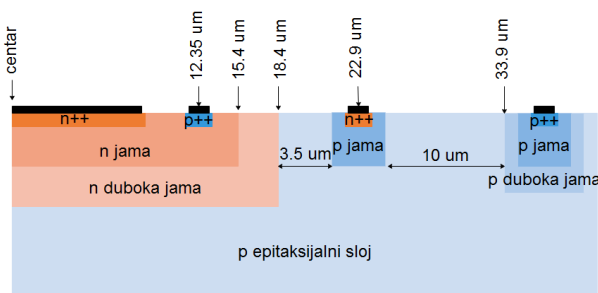
Pošto je SPAD polarisana iznad napona proboja kada se lavina jednom pokrene ona se ne zaustavlja i dalje detektovanje fotona je onemogućeno. Potrebno je realizovati brzo digitalno kolo koje detektuje rastuću ivicu fotostruje, generiše standardan izlaz i smanjuje polarizaciju ispod napona proboja, a zatim vraća napon iznad napona proboja, na taj način omogućavajući dalje detekcije [2].



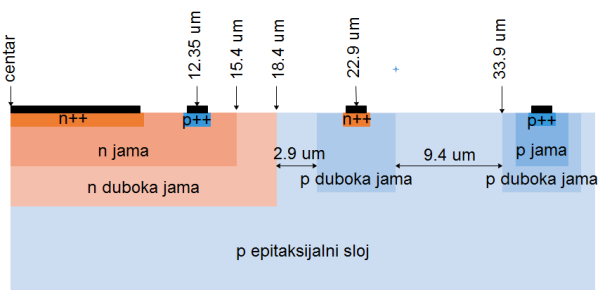
Slika 1. SPAD realizovan bez p jame na manjem rastojanju od LED



Slika 2. SPAD realizovan bez p jame na većem rastojanju od LED



Slika 3. SPAD realizovan sa p jamom



Slika 4. SPAD realizovan sa dubokom p jamom

3. REZULTATI SIMULACIJA

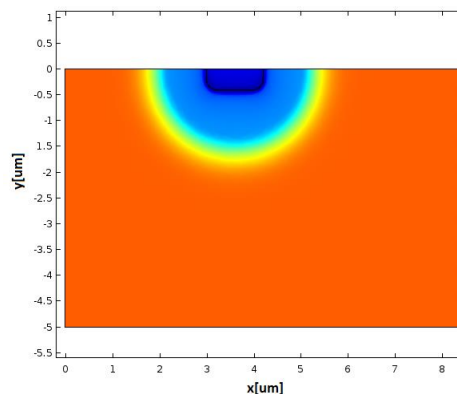
Simulacije uticaja strukture SPADa na oblast osiromašenja i električno polje su rađene u programskom paketu Comsol Multiphysics.

U slučaju strukture prva dva optoelektronska uređaja multiplikaciona zona SPADa se nalazi između n++ i epitaksijalnog sloja p tipa, dok se zona apsorpcije nalazi u epitaksijalnom sloju. Razdvajanje multiplikacione zone i zone apsorpcije je postignuto formiranjem plitke n++ regije, koncentracije 10^{20} cm^{-3} i dubine $0.4 \mu\text{m}$ i korištenjem epitaksijalnog sloja koncentracije akceptora 10^{15} cm^{-3} . Pošto je koncentracija primesa u epitaksijalnom sloju manja od koncentracije u supstratu, oblast osiromašenja u kojoj je električno polje jako, lakše se širi i fotogenerisani nosioci u oblasti osiromašenja se jakim driftom usmeravaju ka spoju lavinske diode gde je verovatnoća pokretanja lavine najveća. Fotogenerisani nosioci generisani van oblasti osiromašenja procesom difuzije stižu do oblasti osiromašenja. Difuzija je relativno spor proces uslovljen slučajnim kretanjem nosilaca i ograničava brzinu uređaja.

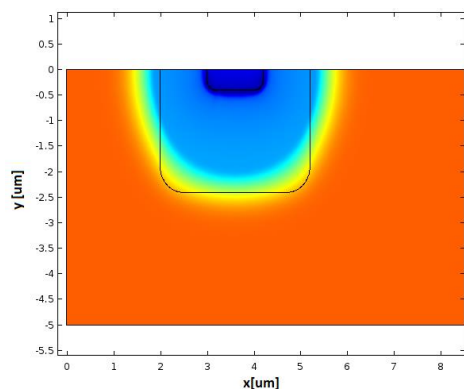
Veću kontrolu nad oblasti osiromašenja možemo postići formiranjem multiplikacione zone između spoja n++/p jama. Koncentracija akceptora p jame je 10^{17} cm^{-3} , dok je dubina $2.5 \mu\text{m}$. Pošto je koncentracija p jame veća od koncentracije epitaksijalnog sloja oblast osiromašenja se širi verikalno ka dole. Na taj način više fotona generisanih u LED koji se emituju na sve strane će generisati nosioce u oblasti osiromašenja. U slučaju kada je SPAD realizovana između spoja n++/p duboka jama oblast osiromašenja se za isti napon polarizacije dublje prostire jer je koncentracija akceptora p duboke jame manja od koncentracije p jame, a i sama jama je dublja što ubrzava prostiranje oblasti osiromašenja.

Na slikama 5 i 6 je prikazana oblast osiromašenja SPADa bez i sa p jamom. Polarizacija supstrata je -3V , dok je katoda SPADa uzemljena. Električno polje multiplikacione zone sa i bez jame je prikazano na slikama 7 i 8.

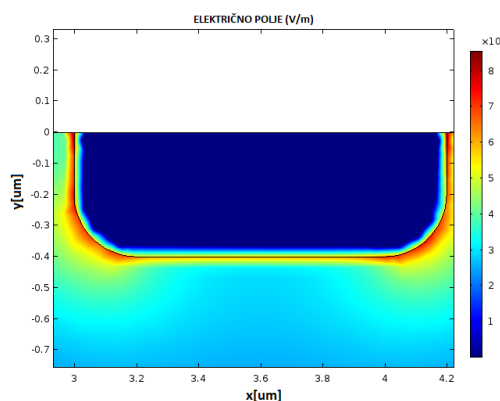
Električno polje SPADa sa jamom je 10 puta veće u odnosu na realizaciju bez jame. Posledica je manji napon proboja SPADa što je ključno za njihovu integraciju.



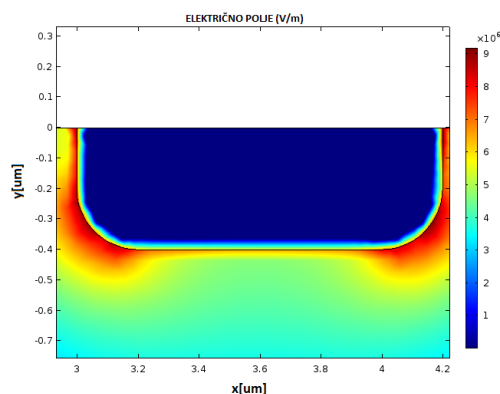
Slika 5. Tamnoplava (većinski nosioci elektroni), narandžasta (većinski nosioci šupljine), svetloplava (oblast osiromašenja) za polarizaciju SPADa bez jame na -3 V



Slika 6. Tamnoplava (većinski nosioci elektroni), narandžasta (većinski nosioci šupljine), svetloplava (oblast osiromašenja) za polarizaciju SPADa sa jamom na -3V



Slika 7. Jačina električnog polja za polarizaciju SPADa sa jamom na -3V



Slika 8. Jačina električnog polja za polarizaciju SPADa sa jamom na -3V

4. STRUJNO-NAPONSKE KARAKTERISTIKE SPAD DIODA

Rezultati merenja strujno-naponskih karakteristika LED i SPAD dioda su dati u ovom poglavlju. Merenja su vršena direktno na poluprovodničkoj pločici (vejferu) korišćenjem RF ispitne stanice prikazane na slici 9.

LED dioda je inverzno polarisana uzemljenjem katode i polarizacijom anode negativnim naponom korišćenjem RIGOL DP832 Programmable DC Power Supply generatora. Struja diode je merena multimetrom Keithley 2000. SPAD dioda je polarisana korišćenjem elektrometra Keyseight B2987A. Anoda SPADa je povezana zajedno sa

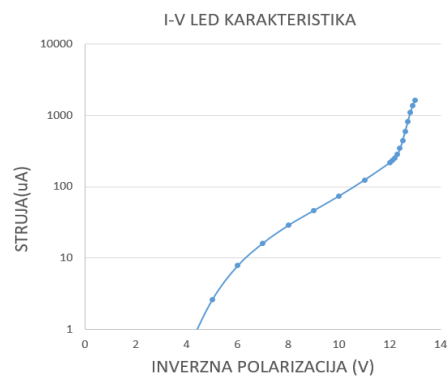
supstratom. Na anodu je dovođen negativan napon dok je katoda uzemljena.



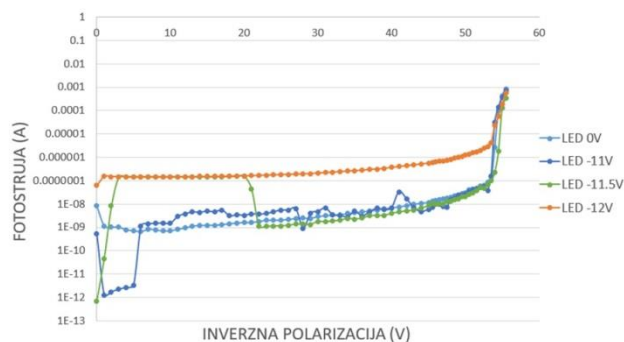
Slika 9. RF ispitna stanica

Na slici 10 je prikazana strujno-naponska karakteristika LED diode. Smatra se da dioda počinje da emituje svetlost kada je struja veća od 100 μA . Sa grafika se vidi da dioda počinje da generiše fotone pri vrednosti napona većim od 11 V.

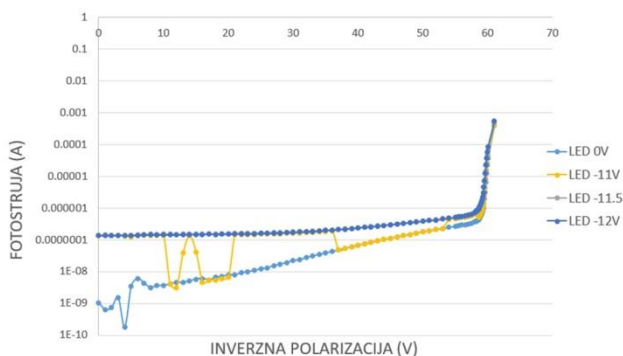
Strujno-naponske karakteristike SPAD realizovanih bez jama su prikazane na slikama 11 i 12. Ove strukture ne pokazuju dobre performanse. Naponi proboja su im preveliki, -52 V i -58 V. Pored toga fotostruja ovih dioda nije konstantna što se vidi na graficima. Do smanjenja struje dolazi jer u oblast osiromašenja ne dospeva približno jednak broj fotona u vremenu. SPAD na većem rastojanju pokazuje nešto bolje performanse jer je odziv sistema dobar za napone polarizacije iznad 11 V dok u slučaju SPADa na manjem rastojanju taj napon mora biti veći od 11.5 V. Međutim dioda na većem rastojanju pokazuje veći nagib rasta fotostruje kada je LED uzemljena. Ovo ponašanje je opravdano većim zagrevanjem dioda koje generišu nosioce naelektrisanja i na taj način povećavaju struju kroz SPAD. Ovo je velika mana i zbog toga ove strukture ne treba koristiti.



Slika 10. Strujno-naponska karakteristika LED



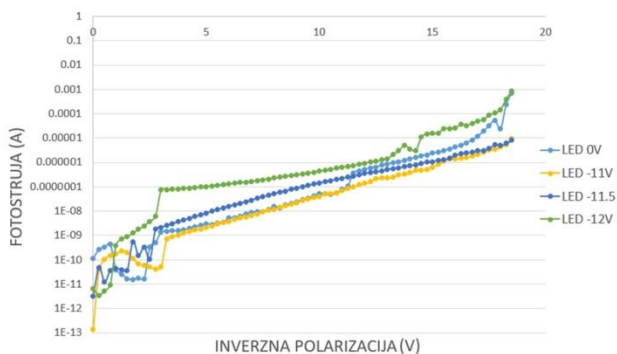
Slika 11. Strujno-naponske karakteristike SPAD bez jame na manjem rastojanju od LED



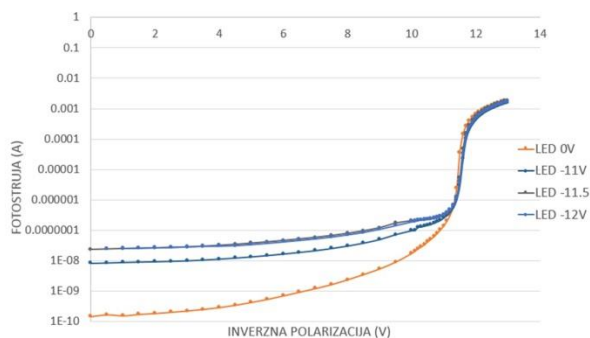
Slika 12. I-V karakteristike SPAD bez jame na većem rastojanju od LED

Strujno-naponske karakteristike SPAD realizovanih sa dubokom p jamom i p jamom su prikazane na slikama 13 i 14. Ove strukture pokazuju bolje performanse. Otklonjen je problem skokova fotostruje. Naponi proboja su značajno niži, -18 V i -11.5V. Rezultati simulacija jačine električnog polja se slažu sa rezultatima merenja.

Problem ipak postoji kod SPADa realizovanog spojem n++ i p duboke jame jer je duboka jama šira u odnosu na p jamu, što znači da je SPAD bliži LED. Korišćeno je rastojanje od svega 2.9 μm . Kako se oblast osiromašenja širi i manjim delom bočno, kao što je pokazano simulacijama, dolazi do proširenja oblasti osiromašenja u LED diodu odakle se jakim driftom ti nosioci usmeravaju ka katodi n++ tranzistora. Ovaj fenomen se jasno vidi na strujno-naponskoj karakteristici SPADa kada je LED uzemljena. Fotostruja naglo skače sa vrednosti reda 10^{-11} A na vrednosti reda 10^0 A i nastavlja da raste sa konstantnim nagibom. Isti skokovi se mogu primetiti i pri drugim polarizacijama diode. Čak šta više jasan napon proboja nije moguće odrediti. Ova dioda je neupotrebljiva i problem se



Slika 13. Strujno-naponske karakteristike SPAD sa dubokom p jamom



Slika 14. Strujno-naponske karakteristike SPAD sa p jamom

rešava udaljavanjem SPADa.

SPAD realizovan sa p jamom pokazuje najbolje performanse. Uspešno su implementirane teorijske pretpostavke potvrđene simulacijama da oblast osiromašenja treba da bude dublja i uža u odnosu na realizaciju bez jama. Takođe je i implementacijom jame značajno smanjen napon proboja generisanjem većeg električnog polja u multiplikacionoj zoni. Prevaziđen je problem prostiranja oblasti osiromašenja u LED jer je SPAD udaljen 3.5 μm . Struja curenja kada je uređaj uzemljen je reda 10^{-10} A i sa polarizacijom raste do 10^{-8} A pre proboja. Kada LED emituje fotone jasno se vidi porast struje pri povećanju fluksa fotona. Razlika između struje curenja i fotostruje za polarizaciju LED na -11 V je oko 100 puta. Daljim povećanjem struje LED diode foto struja raste do 50 puta, nakon toga dolazi do zasićenja.

5. ZAKLJUČAK

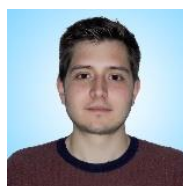
U ovom radu su predstavljena četiri načina projektovanja osnovnog dela QRNG optoelektronskog uređaja. Posmatran je uticaj strukture SPADa na performanse uređaja. Simulacijama je pokazano da korištenjem jama možemo da utičemo na brzinu prostiranja oblasti osiromašenja, kao i njenu širinu i dubinu. Koncentracije ovih spojeva su manje od epitaksijalnog sloja te je slabije i električno polje u oblasti osiromašenja čime se smanju napon proboja ovih dioda što je ključno za integraciju. Rezultati merenja su potvrdili ove pretpostavke. SPAD bez jame je u potpunosti neupotrebljiv zbog diskontinualnog odziva kola. SPAD sa jamama imaju stabilne odzive, ali je do izražaja došla udaljenost SPADa do dioda. SPAD sa dubokom jamom je preblizu LED i oblast osiromašenja prodire u LED diodu čime je čini neupotrebljivom. SPAD realizovan p jamom je pokazao zadovoljavajuće karakteristike. Rastojanje između LED i SPED je dovoljno veliko.

Struja curenja je reda 10^{-10} A dok se jasno povećanje struje 100 puta vidi kada LED emituje svetlo. Od -11 V do -11.5 V fotostruja raste sa fluksom, a za veće napone ona je u saturaciji.

6. LITERATURA

- [1] Abbas Khanmohammadi, Reinhard Enne, Michael Hofbauer, Horst Zimmermann, "A Monolithic Silicon Quantum Random Number Generator Based on Measurement of Photon Detection Time" *IEEE Photonics Journal* 14, Vol. 7, no. 5, ISSN: 1943-0655, October 2015.
- [2] Abbas Khanmohammadi, Reinhard Enne, Michael Hofbauer, Horst Zimmermann, "Monolithically Integrated Optical Random Pulse Generator in High Voltage CMOS Technology", *45th European Solid State Device Research Conference (ESSDERC)*, 2015.

Kratka biografija:



Mihailo Drljača rođen je u Banja Luci 1993. god. Bečlor rad na Fakultetu tehničkih nauka iz oblasti Mikrofluidike odbranio je 2016.god.

BCI BAZIRAN NA RAVNOTEŽNIM VIZUELNO EVOCIRANIM POTENCIJALIMA BCI BASED OD STEADY-STATE VISUALLY EVOKED POTENTIALS

Nera Vlaisavljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – AUTOMATIKA I UPRAVLJANJE SISTEMIMA

Kratak sadržaj – Zadatak ovog rada jeste da opiše razvoj mozak-računar interfejsa baziranog na ravnotežnim vizuelno evociranim potencijalima, u Laboratoriji za biomedicinsko inženjerstvo pri Fakultetu tehničkih nauka u Novom Sadu. Nakon kratkog teorijskog uvida opisani su koraci u razvoju aplikacije i krajnji rezultati.

Abstract – This paper aims to describe the development of a brain-computer interface based on steady-state visually evoked potentials, in the Laboratory for Biomedical Engineering, Faculty of Technical Sciences, Novi Sad. After a short introduction, the steps of the development of the applications are described, as well as the end results.

Ključne reči: brain-computer interfejs, ravnotežni vizuelno evocirani potencijal, BCI, SSVEP

1. UVOD

Mozak-računar interfejs (brain-computer interfejs - BCI) prevodi ljudske namere u kontrolne signale u cilju ostvarivanja direktnog komunikacionog kanala između ljudskog mozga i spoljašnjeg uređaja. Pošto BCI ne zavisi od sposobnosti izlaznih putanja mozga, perifernih nerava i mišića, on može da obezbedi novi komunikacioni kanal ljudima sa smanjenim motornim sposobnostima [1].

Ravnotežni vizuelno evocirani potencijali, najčešće zvani *steady-state* vizuelno evocirani potencijali (SSVEP) su odgovor mozga na ponavljajući vizuelni stimulus konstantne frekvencije. *Steady-state* odgovor se u toku stimulacije pojavljuje u EEG-u u talasnom obliku koji je periodičan i skoro sinusoidalan, koji je iste frekvencije kao ponavljanje stimulusa. Osim frekvencije ponavljanja stimulusa često su u ovaj talas uključeni i viši harmonici ove frekvencije.

SSVEP se koristi kao kontrolni signal u BCI jer može da ponudi korisniku da odabere između nekoliko komandi, time koristeći BCI bazirani meni.

Svaka opcija/komanda u takvom meniju je povezana sa jednim od nekoliko ponuđenih ponavljajućih vizuelnih stimulusa koji su prikazani korisniku. Stimulusi se smeju razlikovati samo po njihovoj frekvenciji ponavljanja. Pošto su svi stimulusi prikazani istovremeno, korisnik može da se fokusira na određeni stimulus i time evocira odgovarajući SSVEP i time odabere odgovarajuću opciju menija.

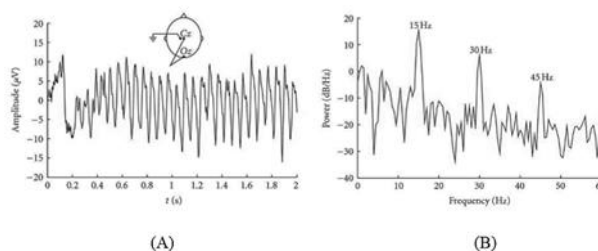
NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Nikola Jorgovanović, red.prof.

Ovo je bazirano na činjenici da je amplituda SSVEP-a veća za stimulus na koji se fokusira, čak i ako se u istoj oblasti vizuelnog polja nalaze drugi stimulusi na koje se ne fokusira.

Za dobijanje SSVEP odgovora najčešće se koriste sledeća tri tipa vizuelne stimulacije: svetlosna stimulacija, jednostruki grafički stimulus i stimulus sa obrtanjem obrasca. Svetlosni stimulusi su treptajući izvori svetlosti kao što su LED diode, fluorescentna svetla ili *Xenon* svetla, čije se treptanje modulira određenom frekvencijom.

Jednostruki grafički stimulus sastoji se od slike koja se pojavljuje i nestaje, najčešće na ekranu računara. Stimulus sa obrtanjem obrasca sastoji se od 2 obrasca koja se zamenjuju određenom frekvencijom. Kada se koristi ekran računara za stimulaciju, treba voditi računa o brzini osvežavanja ekrana monitora koja utiče na prezentaciju stimulusa. Tipičan SSVEP odgovor u vremenskom i frekvencijskom domenu vide se na slici 1.



Slika 1. Talasni oblik EEG signala snimljenog bipolarnom konfiguracijom (Oz-Pz) tokom stimulacije svetlosnim izvorom frekvencije treptanja 15 Hz (A), i njegov frekvencijski spektar (B) preuzeto iz [2]

2. ANALIZA RAVNOTEŽNIH VIZUELNO EVOCIRANIH POTENCIJALA

U ovoj studiji istraživane su metode za indukciju SSVEP-a u laboratorijskim uslovima. Ciljevi ove studije su:

- indukovanje i snimanje SSVEP signala pri stimulaciji LED diodom i stimulaciji na računaru
- poređenje snaga indukovanih signala radi izbora pogodnije stimulacije za BCI
- pronalaženje najboljeg opsega frekvencija za stimulaciju
- ispitivanje različitih metoda obrade SSVEP signala

2.1 Metoda

2.1.1 Subjekti

Troje odraslih ispitanika bez istorije neuroloških poremećaja učestvovali su u ovoj studiji. Ispitanici su imali normalan ili korigovan vid.

Eksperiment je izveden u Laboratoriji za biomedicinsko inženjerstvo na Fakultetu tehničkih nauka Univerziteta u Novom Sadu.

2.1.2 Instrumentacija

Za snimanje EEG-a korišćen je Smarting uređaj kompanije mBrainTrain. Kapa sa elektrodama postavljena je na glavu ispitanika, a elektrode sa kojih se snimao EEG signal su bile na pozicijama O1, O2 i Pz (po internacionalnom "10-20" standardu), dok je referentna elektroda bila na poziciji Fz. Frekvencija odabiranja signala je bila 500 Hz. Nije bilo hardverskog filtriranja signala.

Korišćena su dva tipa vizuelne stimulacije—stimulacija LED diodama i stimulacija na ekranu računara. Za stimulaciju LED diodama korišćene su 3 blisko pozicionirane LED diode koje su se palile jedinstvenim signalom generisanim pomoću NI USB-6211 ploče. Preko dioda postavljen je materijal za difuziju svetlosti koji je činio da svetlost sa tri diode izgleda kao da dolazi sa jednog izvora.

Ekran računara koji je korišćen za stimulaciju je 23.0" LCD monitor sa frekvencijom osvežavanja ekrana od 60 Hz.

2.1.3 Protokol eksperimenta

Eksperiment se sastojao iz dve sesije za svakog ispitanika, jedna za stimulaciju LED diodom a druga za stimulaciju na ekranu računara. Kompletan eksperiment izveden je u programskom okruženju Matlab.

U toku prve sesije ispitanici su sedeli 90 cm ispred LED izvora svetlosti. Svaka stimulacija trajala je 10 sekundi, a periodi između stimulacija bili su promenljive dužine (najmanje 10 sekundi). Ispitnici su bili auditorno obavешteni kada će početi stimulacija kako bi se fokusirali na izvor svetlosti. Vremena svih početaka i krajeva stimulacije su softverski zapamćena. Frekvencije stimulusa su opsega 12-100 Hz (23 frekvencije ukupno), i one su: 12-26 Hz, 30, 35, 40, 50, 70, 80, 90 i 100 Hz.

U toku druge sesije ispitanici su sedeli ispred ekrana računara, na udaljenosti od 50 cm i na računaru je prikazan kvadrat veličine 6x6 cm koji je treperio određenim frekvencijama. Pozadina je bila crna, a kvadrat naizmenično beo i crn. Za kontrolu stimulacije korišćen je Psychtoolbox interfejs između Matlaba i računarskog hardvera. Kako je na ekranu sa frekvencijom osvežavanja 60 Hz moguće tačno generisati samo određene frekvencije stimulacije i one ne smeju biti celobrojni umnošci jedna drugoj, generisane su samo 4 frekvencije: 20, 15, 12 i 8.57 Hz. Svaka stimulacija trajala je 10 sekundi.

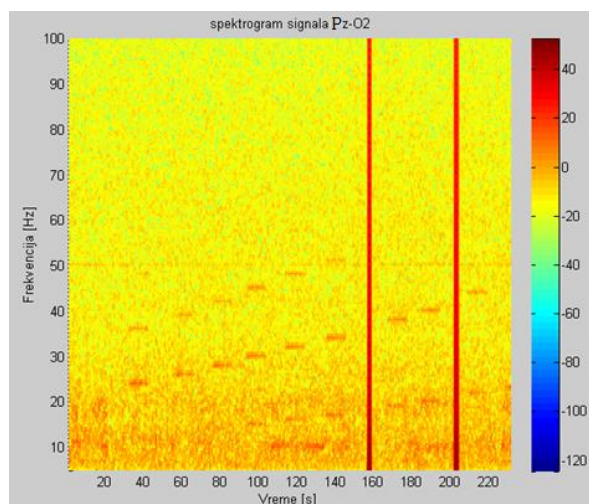
Nakon završetka sesija ispitanici su prijavili osećaj neprijatnosti pri nižim frekvencijama (<25 Hz), a posebno za stimulaciju na računaru.

2.1.4 Obrada i vizualizacija signala

Signali su analizirani nakon snimanja. Prvi korak u analizi bilo je filtriranje filtrom propusnikom opsega. Signali su filtrirani Batervortovim filtrom 4. reda sa odsečnim frekvencijama od 5 i 120 Hz. Kako bi se smanjio uticaj spontane EEG aktivnosti dalje su se obrađivali signali dobijeni oduzimanjem signala sa elektroda O1 i O2 u odnosu na signal sa elektrode Pz (Pz-O1, Pz-O2).

Slika 2 prikazuje spektrogram signala Pz-O2, subjekta 2, u toku stimulacije LED diodom za frekvencije stimulacije 12-22 Hz. Spektrogrami su izračunati korišćenjem prozora dužine 2 sekunde sa 90% preklapanja susednih prozora.

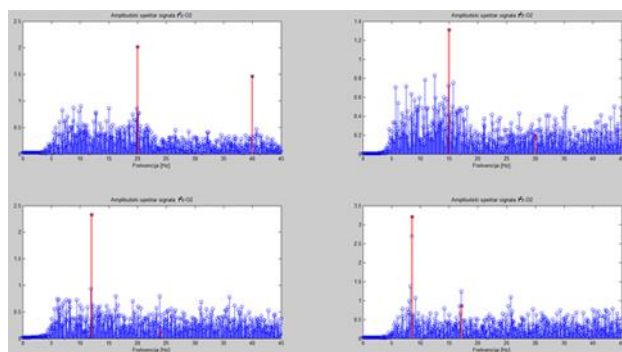
Kod subjekta 2 primećeni su jači SSVEP odgovori nego kod subjekta 1 i 3, i za niže frekvencije (12-14 Hz) odgovori su primetniji na harmonicima frekvencije stimulacije nego na samoj frekvenciji stimulacije. Ovo se može objasniti zašumljenošću tog opsega frekvencija spontanom EEG aktivnošću, koja se preklapa sa frekvencijama stimulacije. Za stimulacije veće od 14 Hz u spektrogramu se jasno uočava pojačanje snage i na frekvenciji stimulacije i na prvom i drugom harmoniku. Za stimulacije veće od 19 Hz uočeni su SSVEP odgovori na frekvenciji stimulacije i prvom harmoniku.



Slika 2. Spektrogram sa SSVEP odgovorima za signal Pz-O2, subjekta 2, u toku stimulacije frekvencijama 12-22 Hz LED diodama

Spektar signala dobijen je računanjem brze Furijeove transformacije. Posmatran je spektar u toku stimulacije i pre nje. Slika 3 prikazuje spektar signala Pz-O2, subjekta 3, u toku stimulacije na ekranu računara.

Spektrogram ovog signala ne prikazuje vidljive promene u SSVEP-u u toku stimulacije, ali spektar signala izračunat za segment signala u toku stimulacije prikazuje povećanje snage na frekvenciji stimulacije i prvom harmoniku.



Slika 3. Amplitudski spektar signala Pz-O2, subjekta 3, u toku stimulacije na ekranu računara, frekvencijama 20, 15, 12 i 8.57 Hz. Crvenom bojom označena je amplituda SSVEP odgovora.

2.1.5 Rezultati i diskusija

Kao što je rečeno, svrha ovog ispitivanja jeste poređenje snaga indukovanih signala radi izbora pogodnije stimulacije za BCI, pronalaženje najboljeg opsega frekvencija za stimulaciju i ispitivanje različitih metoda obrade SSVEP signala.

Veća amplituda SSVEP odgovora kod sva tri subjekta izmerena je na elektrodi O2. Kod subjekata 1 i 3 u spektrumu se nisu jasno uočavali vremenski trenuci kada ima stimulacije i kada je nema, ali se posmatrajem spektra signala u toku stimulacije i pre primetilo pojačanje snage na frekvencijama stimulacije u odnosu na snagu kada nema stimulacije. Najveće amplitude SSVEP odgovora dobijene su okvirno na frekvencijama 15-25 Hz.

Stimulacija na računaru bila je znatno neprijatnija od stimulacije LED diodama, a i zbog ograničenosti u izboru frekvencija za BCI predstavljen u ovom radu će se koristiti stimulacija LED diodama. Takođe se nastoji da se koriste frekvencije stimulacije iz višeg opsega (>25 Hz) jer takve stimulacije nisu nimalo neprijatne za gledanje.

Odnos signal-šum (SNR), definisan kao snaga na frekvenciji stimulacije u odnosu na snagu susednih frekvencija u spektru signala, pokazao se višim za frekvencije iznad 15 Hz, jer se ispod 15 Hz pojavljuje pojačana spontana EEG aktivnost. Frekvencije stimulacije iznad 50 Hz se nisu pojavljivale u spektru signala.

Odlučeno je da se za razvoj BCI kao komunikaciona platforma koristi stimulacija LED diodama, na frekvencijama od 25 Hz do 30 Hz, jer je na tim frekvencijama SSVEP odziv i dalje primetan, a u tom opsegu nema spontane EEG aktivnosti i neprijatnosti pri gledanju. Time se postiže visok SNR i zaključeno je da je računanje SNR pomoću brze Furijeove transformacije pogodan način za izdvajanje obeležja pri obradi SSVEP signala.

3. BCI BAZIRAN NA RAVNOTEŽNIM VIZUELNO EVOCIRANIM POTENCIJALIMA

U ovom poglavlju predstavljen je jednostavan asinhroni egzogeni *brain-computer* interfejs zasnovan na ravnotežnim vizuelno evociranim potencijalima. Razvoju ove aplikacije prethodila je studija opisana u prethodnom poglavlju. Nakon obavljene studije odlučeno je da se koristi LED stimulacija za izazivanje SSVEP odgovora, odnosno kao komunikaciona platforma, i da ona bude opsega frekvencija 25-30 Hz, zato što takve frekvencije ne predstavljaju opterećenje ispitanicima a obezbeđuju dovoljno dobar SSVEP odgovor. Ovaj BCI podrazumeva postojanje 2 mete, i korisnik gledanjem u željenu metu odabira pomeranje kursora na ekranu gore ili dole.

3.1 Metoda

3.1.1 Subjekti

Troje odraslih ispitanika bez istorije neuroloških poremećaja učestvovali su u ovoj studiji. Ispitanici su imali normalan ili korigovan vid. Eksperiment je izveden u Laboratoriji za biomedicinsko inženjerstvo na Fakultetu tehničkih nauka Univerziteta u Novom Sadu.

3.1.2 Instrumentacija

Za snimanje EEG-a korišćen je Smarting uređaj kompanije mBrainTrain. Kapa sa elektrodama postavljena

je na glavu ispitanika, a elektrode sa kojih se snimao EEG signal su bile na pozicijama O1, O2 i Pz (po internacionalnom "10-20" standardu), dok je referentna elektroda bila na poziciji Fz. Frekvencija odabiranja signala je bila 500 Hz. Nije bilo hardverskog filtriranja signala.

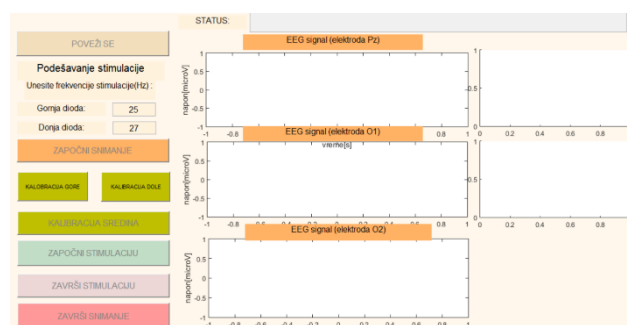
Stimulacija za dobijanje vizuelnog potencijala generisana je LED diodama. Za stimulaciju LED diodama korišćene su 3 blisko pozicionirane LED diode koje su se palile jedinstvenim signalom generisanim pomoću NI USB-6211 ploče. Preko dioda postavljen je materijal za difuziju svetlosti koji je činio da svetlost dioda izgleda kao da dolazi sa jednog izvora. Postojala su dva identična sistema sa po tri diode, koji su predstavljali mete, gornju i donju. Mete su postavljene iznad i ispod ekrana računara. Za kontrolu prikaza na ekranu, tj. povratnu spregu BCI korišćen je Psychtoolbox interfejs između Matlaba i računarskog hardvera.

3.1.3 Protokol eksperimenta

Kompletni eksperiment izveden je u programskom okruženju Matlab. Ispitanici su sedeli 50 cm ispred LED izvora svetlosti (meta) i ekrana računara. Frekvencije stimulusa su opsega 25-30 Hz.

Pre početka eksperimenta postojala je kratka kalibracija za svakog korisnika. Nakon kalibracije na računaru je prikazan kursor veličine 4x4 cm koji se pomerao gore ili dole pri gledanju u gornju ili donju metu BCI. Ukoliko se ne gleda ni u jednu metu kursor ostaje na svom mestu.

Izgled interfejsa kontrolne aplikacije može se videti na slici 4. Preko interfejsa se vršila kontrola frekvencija stimulacije, trenutaka kalibracije, početka i kraja stimulacije i snimanja signala. U toku eksperimenta na graficima su se prikazivali EEG signali sa elektroda Pz, O1 i O2, kao i amplitudski spektri signala Pz-O1 i Pz-O2.



Slika 4. Izgled interfejsa kontrolne aplikacije

3.1.4 Obrada signala

Pre početka eksperimenta izvodi se kratka kalibracija. Ona podrazumeva snimanje EEG signala u toku 2 sekunde gledanja gornje mete, donje mete, kao i pri gledanju između te dve mete. Ova kalibracija služi za izdvajanje obeležja signala koji se kasnije koriste pri klasifikaciji.

Radi eliminisanja EEG-a nastalog spontanom moždanom aktivnošću formirali su se signali dobijeni oduzimanjem signala sa elektroda O1 i O2 od signala Pz, odnosno signali Pz-O1 i Pz-O2. Pri kalibraciji ovi signali su filtrirani eliptičkim filtrom 6. reda sa graničnim frekvencijama od 20 i 40 Hz. Nakon toga izdvajale su se

amplitude SSVEP odgovora na frekvencijama stimulacije i formirao se klasifikator koji predstavlja odnos tih amplituda u toku gledanja gornje i donje mete, kao i između meta. Amplitude SSVEP odgovora dobijale su se korišćenjem brze Furijeove transformacije.

Nakon kalibracije počinje eksperiment, odnosno stimulacija koja rezultuje pomeranjem kursora. Ovaj BCI sistem je asinhron, jer ne postoji vremenski prozor u kojem korisnik donosi odluku, već to može činiti bilo kad i obrada signala se odvija kontinualno. Signali su u toku eksperimenta analizirani na svakih 100ms, odnosno na svakih 100ms donosi se odluka da li korisnik gleda u gornju ili donju metu, ili nijednu, i vrši odgovarajuće pomeranje kursora. Proces obrade signala počinje formiranjem signala Pz-O1 i Pz-O2 za poslednje 2 sekunde snimanja. Zatim se vrši isto filtriranje kao u procesu kalibracije. Potom se izdvajaju amplitude SSVEP odgovora na frekvencijama stimulacije i računa njihov odnos. Upoređivanjem ovog dobijenog odnosa sa odnosima dobijem u kalibraciji sistema donosi se odluka da li korisnik gleda u gornju metu, donju metu ili nijednu od njih dve.

3.1.5 Rezultati i diskusija

Ovaj BCI pokazao je uspešnu detekciju mete u koju se gleda kao i ako se ne gleda ni u jednu metu, što predstavlja značajan problem u BCI zajednici, pošto se i u tom slučaju mete nalaze u vidnom polju korisnika.

Postignuta je visoka tačnost detekcije, s tim što ako se prelazi sa pogleda na kursor na pogled na jednu od meta potrebno je u proseku gledati u metu 1 sekund pre nego što se to detektuje. Ovaj rezultat objašnjen je dužinom vremenskog prozora koji je potreban brzom Furijeovoj transformaciji za dobru frekvencijsku rezoluciju spektra. Dok god se gleda u metu kursor se pomera kontinualno, a pri prestanku gledanja u metu detekcija je skoro trenutna i kursor odmah prestaje sa pomeranjem.

4. ZAKLJUČAK

Cilj ovog rada je razvijanje jednostavnog BCI sistema zasnovanog na ravnotežnim vizuelno evociranim potencijalima. Razvijanju sistema prethodila je studija o ovim potencijalima. Cilj ove studije bilo je poređenje snaga indukovanih signala SSVEP signala radi izbora pogodnije stimulacije za BCI, pronalaženje najboljeg opsega frekvencija za stimulaciju i ispitivanje različitih metoda obrade SSVEP signala.

Najveće amplitude SSVEP odgovora dobijene su okvirno na frekvencijama 15-25 Hz. Stimulacija na računaru bila je znatno neprijatnija od stimulacije LED diodama. Takođe se nastoji da se koriste frekvencije stimulacije iz višeg opsega (>25 Hz) jer takve stimulacije nisu nimalo neprijatne za gledanje.

Na kraju predstavljen je jednostavan asinhroni egzogeni *brain-computer* interfejs zasnovan na ravnotežnim vizuelno evociranim potencijalima. Razvoju ove aplikacije prethodili su eksperimenti sa generisanjem SSVEP odgovora kod ispitanika.

Nakon obavljene studije odlučeno je da se koristi LED stimulacija za izazivanje SSVEP odgovora, odnosno kao komunikaciona platforma, i da ona bude opsega frekvencija 25-30 Hz, zato što takve frekvencije ne predstavljaju opterećenje ispitanicima a obezbeđuju dovoljno dobar SSVEP odgovor. Ovaj BCI podrazumeva postojanje 2 mete, i korisnik gledanjem u željenu metu odabira pomeranje kursora na ekranu gore ili dole.

Ovaj BCI pokazao je uspešnu detekciju mete u koju se gleda kao i ako se ne gleda ni u jednu metu. Dok god se gleda u metu kursor se pomera kontinualno, a pri prestanku gledanja u metu detekcija je skoro trenutna i kursor odmah prestaje sa pomeranjem.

5. LITERATURA

- [1] Andrej M. Savić "Electroencephalographic control signals of braincomputer interface in neurorehabilitation", University of Belgrade, School of electrical engineering, 2014.
- [2] D. Zhu, J. Bieger, G. G. Molina, and R. M. Aarts, "A survey of stimulation methods used in SSVEP-based BCIs," Computational intelligence and neuroscience, vol. 2010, p. 1-12, 2010.

Kratka biografija:



Nera Vlaisavljević je rođena 1.10.1993. godine u Zrenjaninu. Završila je Zrenjaninsku gimnaziju u Zrenjaninu, 2012. godine. Fakultet tehničkih nauka u Novom Sadu je upisala 2012. godine. Osnovne akademske studije Računarstva i automatike završila je 2016. godine. Ispunila je sve obaveze i položila je sve ispite predviđene master studijskim programom Računarstvo i automatika (Automatika i upravljanje sistemima).

RAZVOJ ANDROID APLIKACIJA U PROGRAMSKOM JEZIKU KOTLIN ANDROID APPLICATIONS DEVELOPMENT USING KOTLIN

Zorana Deurić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu opisan je razvoj Android aplikacije u programskom jeziku Kotlin. Ukratko su navedene osnovne osobine jezika. Dati su specifikacija zahtjeva i dizajn aplikacija koje su implementirane u Java i Kotlin programskim jezicima, kako bi se uočile njihove sličnosti i razlike u oblasti Android programiranja. Opisani su ključni elementi implementacije, te je na kraju demonstriran rad aplikacija i iznesen zaključak istraživanja.

Abstract – This paper presents the study about developing Android applications using Kotlin programming language, uses the small application as a case study to compare Java and Kotlin in Android programming. It describes Kotlin's main properties, application requirements specification, design specification, implementation's key details, and applications demo. At the end gives a brief conclusion based on the research.

Ključne reči: Mobilne aplikacije, Android, Kotlin, Java

1. UVOD

Zadatak ovog rada je razvoj aplikacije upotrebom Kotlin programskog jezika s ciljem da se izvrši analiza samog jezika, kao i poređenje sa Java programskim jezikom u oblasti razvoja mobilnih aplikacija. Istraživanje je vršeno paralelnim razvojem dvije aplikacije, pisane u ova dva jezika, koje imaju istu specifikaciju zahtjeva i dizajn, kako bi se uporedile osobine jezika.

2. PROGRAMSKI JEZIK KOTLIN

Kotlin je relativno mlad programski jezik nastao 2011. godine. Predstavlja statički tipiziran programski jezik, što znači da su tipovi poznati u vrijeme kompajliranja i sve provjere tipova se dešavaju u vrijeme kompajliranja, za razliku od dinamički tipiziranih jezika kod kojih se provjere tipova vrše u vrijeme izvršavanja (en. *runtime*). Razvijen je od strane tima *JetBrains* [1] programera kao jezik opšte namjene, prije svega za upotrebu u industriji.

Vodi porijeklo iz Rusije, tačnije Sankt Peterburga, a naziv je dobio po istoimenom ostrvu u blizini Sankt Peterburga (rus. *Комлион*). Kotlin se izvršava na Java virtuelnoj mašini (*Java Virtual Machine, JVM*), a može biti kompajliran u Java Script izvorni kod. Primarni cilj pri razvijanju ovog jezika je bio da se dobije objektno-orijentisani jezik, koji će prevazići ograničenja na koja se nailazi u programskom jeziku Java. Ističu se četiri osnovne vrijednosti koje stoje iza dizajna Kotlina:

- interoperabilnost – može lako biti korišćen zajedno sa Javom,
- sigurnost – statičke provjere najčešćih grešaka u vrijeme kompajliranja (npr. *null pointer exception* i sl.),
- alati – obezbjeđeni su adekvatni alati za precizno izvršavanje uz dobre performanse,
- otvorenost („demokratičnost“) – svi dijelovi jezika su dostupni programerima za prilagođavanje sopstvenim potrebama, uključujući restrikcije postojećih karakteristika.

2.1. Glavne osobine jezika

U Kotlinu je sve objekat u smislu da se funkcije članice i atributi mogu pozvati za bilo koju varijablu. Neki tipovi su ugrađeni i njihova implementacija je optimizovana, ali za korisnika izgledaju kao sasvim obične klase. Osnovni tipovi su: brojevi, karakteri, *boolean* vrijednosti i nizovi. Svaka klasa u Kotlinu nasljeđuje klasu *Any* koja predstavlja podrazumijevanu super klasu za sve klase koje nemaju definisan super tip.

Kotlin ne koristi koncept statičkih članova kao Java, već uvodi sopstveni koncept objekata za opisivanje atributa i metoda povezanih singleton stanjem. Prijateljski objekat neke klase predstavlja *singleton* objekat namijenjen čuvanju vrijednosti vezanih za klasu i prvenstveno se koristi kao zamjena za statičke deklaracije koje postoje u Javi. Na jednom mjestu grupiše funkcije ili vrijednosti kojima je moguć pristup bez instance klase. Iako jako podsjećaju na statičke članove u Javi, objekti su u Kotlinu instancirani u realnom vremenu i mogu, na primjer, da implementiraju interfejse. Ukoliko je ipak potrebno da budu statički ili da sadrže statičke metode to se vrlo lako može postići upotrebom *@JvmStatic* anotacije.

Vrijednost promjenljive u Kotlinu može biti označena kao nepromjenljiva upotrebom ključne riječi *val* (en. *immutable*), kao *final* promjenljive u Javi, ali je bolja praksa da se koriste promjenljive čija vrijednost može da se mijenja (en. *mutable*), a koje se označavaju sa *var*.

Funkcije u Kotlinu su najvišeg nivoa (en. *top level*) što znači da mogu biti definisane bilo gdje. Takođe, podržane su funkcije višeg reda koje kao parametar primaju funkciju ili je njihova povratna vrijednost funkcija. Lambda izrazi dodatno pojednostavljaju upotrebu funkcija višeg reda, a vrlo su slični *Linq* upitima u C# programskom jeziku.

Često je potrebno definisati klase čija je jedina namjena da “drže” podatke, kao što su objekti za transfer podataka (DTO, *Data Transfer Object*). U Kotlinu je moguće ispred ključne riječi *class* dodati *data* i time naglasiti da se radi o DTO klasi. Ovakve klase podrazumijevano sadrže metode: *equals()*, *hashCode()*, *toString()*,

NAPOMENA:

Ovaj rad proistekao je iz master rada, čiji mentor je bio dr Nikola Jorgovanović.

`componentN()` i `copy()` i nije potrebno definisanje getera i setera, dovoljno je navesti nazive polja u konstruktoru unutar definicije klase.

U Kotlinu nije podržana implicitna konverzija manjih tipova u veće, kao što je na primjer konverzija `Byte` u `Int`, zbog toga što manji tipovi nisu podtipovi većih. Osnovni razlog za ovo potiče od toga što se prilikom *boxing*-a¹ tipova ne čuva identitet, već samo jednakost. Odsustvo eksplicitne konverzije je rijetko primjetno zbog toga što se prilikom aritmetičkih operacija dešavaju odgovarajuće konverzije. Na primjer, moguće je sabrati `Long` i `Int` vrijednosti, a tip rezultujuće vrijednosti će biti `Long`.

Slično kao C#, Kotlin nudi mogućnost da se klase prošire dodatnim funkcionalnostima, bez upotrebe dizajn šablona ili nasljeđivanja. Ekstenzije su posebne deklaracije kojima se mogu proširiti funkcionalnosti klase, njihovi atributi i prijateljski objekti. Ono što je bitno naglasiti za ekstenzije je da one zapravo ne modifikuju klasu koju proširuju. Definisanjem ekstenzije se ne dodaje nova članica klase već se definiše dodatna funkcija koja će se nad svakom instancom klase moći pozvati nakon navođenja tačke.

Jedan od ciljeva programera koji su pisali Kotlin je bio da se riješi problem *Null Pointer Exception*-a, poznatog kao „greška od milijardu dolara“ (*The Billion Dollar Mistake* [2]). Kotlin je implementiran tako da NPE može da se desi samo kada to programer eksplicitno navede, tj. u sljedećim situacijama: eksplicitnim pozivom *NullPointerException* klase, navođenjem operatora `!!`, ukoliko ga uzrokuje neki eksterni Java kod ili ako se na primjer polje *this*, koje je dostupno u konstruktoru, koristi prije nego što se izvrši inicijalizacija (što generalno nije u skladu sa pojmom inicijalizacije). U Kotlinu razlikujemo tipove koji mogu da imaju vrijednost `null` od onih koji ne mogu. Nalabilne tipove deklariramo navodeći `?` nakon naziva tipa. Varijabla kojoj je dodijeljen ne-nalabilan tip ni u jednoj situaciji ne može imati vrijednost `null` i nikada ne može uzrokovati NPE. Nakon što je neka promijenljiva deklarirana kao nalabilana, Kotlin neće dozvoliti da se pristupi njenoj vrijednosti bez provjere da li je ta vrijednost `null`, upravo kako bi NPE bio izbjegnuto. U Kotlinu postoje različiti načini za provjeru i tretiranje `null` vrijednosti:

- Klasična provjera: `if (b != null) b.length else -1`
- Sigurni pozivi (*safe calls*, `?.`) - obezbjeđuju se navođenjem `?.` nakon naziva promjenljive, a prije pristupanja vrijednostima njenih atributa ili pozivanja funkcija: `b?.length`
- Elvis operator (`?:`) - koristi se u situacijama koje imaju scenario: “ako je vrijednost reference različita od `null` onda je treba iskoristiti, a ako nije umjesto te vrijednosti treba iskoristiti neku drugu vrijednost ili vrijednost navedenog izraza, (eventualno baciti *exception*)”: `val l = b?.length ?: -1`
- Operator `!!` - navođenjem ovog operatora kažemo: “ako je vrijednost reference različita od `null` onda je upotrijebi, a ako nije baci NPE”:

`val l = b!!?.length`

¹ *Boxing* - konverzija primitivnih tipova u odgovarajuće *wrapper* reference, kao što je, na primjer, konverzija primitivnog tipa `int` u `Integer` u Java programskom jeziku

Jedan od čestih izvora problema pri razvoju Android aplikacija je svakako pozivanje funkcije `findViewById`. Poznato je da već postoje biblioteke koje omogućavaju izbjegavanje poziva ove funkcije, ali korišćenje tih biblioteka već samo po sebi znači zavisnost u vrijeme izvršavanja tj. neophodno je postojanje biblioteka u *classpath*-u projekta kako bi se pristupalo korišćenim instancama i njihovim metodama u vrijeme izvršavanja programa. Nasuprot bibliotekama u Kotlinu je ova ekstenzija već dio Kotlin IDEA *plugin*-a i ne zahtjeva dodatnu zavisnost, niti dodatne anotacije. Sve što je potrebno je da se importuje odgovarajući `view` i moguće je svakoj njegovoj komponenti pristupiti bez pozivanja `findViewById` funkcije.

3. ANALIZA ZAHTJEVA

Kako bi se uporedili Kotlin i Java programski jezici, implementirane su aplikacije za pretraživanje aktualnih događaja u svijetu muzike, filma, sportskih i kulturnih događaja i događaja namijenjenih za djecu i porodično druženje. Pretraga nije ograničena na specifičan geografski region, već pronalazi sve događaje koji su dostupni u okviru *Eventful API*-a. U daljem tekstu su ukratko opisani slučajevi korišćenja aplikacije.

Primarna namjena aplikacije je pretraga događaja koji su razvrstani u pomenute kategorije. Na početnom ekranu aplikacije se nalazi lista dostupnih kategorija. Korisnik nakon pregleda kategorija može da izabere bilo koju i da pregleda listu svih događaja koji joj pripadaju. Događaji se mogu filtrirati unosom ključne riječi u predviđeno polje, a razvrstani su po datumu u tri taba.

Unutar liste događaja u *navigation drawer*-u je dostupna lista katego-rija te se korisnik vrlo lako može prebacivati sa jedne liste događaja na drugu, odabirom željene kategorije. Odabirom nekog događaja iz liste otvara se prikaz detalja tog događaja koji sadrži osnovne informacije, kao što su datum početka i završetka, opis, slika, mjesto itd. Pored toga korisniku su raspoloživa dva dugmeta, od kojih ga prvo vodi na prikaz lokacije događaja na mapi, a drugo do linka događaja na *Eventful* sajtu gdje korisnik može potražiti ulaznice ukoliko su dostupne.

Korisnik iz *navigation drawer*-a glavnog ekrana može da otvori formu za pretragu događaja. Nakon unosa vrijednosti kriterijuma pretrage (ključna riječ, datum, kategorija i kriterijum sortiranja) korisniku se prikazuju svi pronađeni događaji. Osim toga, iz *navigation drawer*-a korisnik može da zahtjeva automatsku sinhronizaciju aplikacije ili da pregleda informacije o aplikaciji.

Korisnik može da podesi lokaciju po kojoj se pretražuju događaji. Na raspolaganju su mu pretraga po njegovoj trenutnoj lokaciji i unos grada u predviđeno polje za koji se događaji pretražuju. Ove dvije opcije su međusobno isključive, a da bi se vršila pretraga po korisničkoj lokaciji aplikacija će zahtjevati odobren pristup lokaciji uređaja, u suprotnom pretraživanje se vrši po poslednjoj poznatoj lokaciji. Korisnik, takođe, može da podesi period ili da isključi sinhronizaciju aplikacije sa API-em.

4. DIZAJN

Model podataka aplikacije se sastoji od 4 entiteta: *Category*, *Event*, *Location* i *Venue*. Klasom *Event* je predstavljen entitet događaj kao centralni entitet aplikacije.

Klasom *Category* predstavljen je entitet kategorija. Svakoj kategoriji pripada nula ili više događaja, a jedan događaj pripada jednoj ili više kategorija.

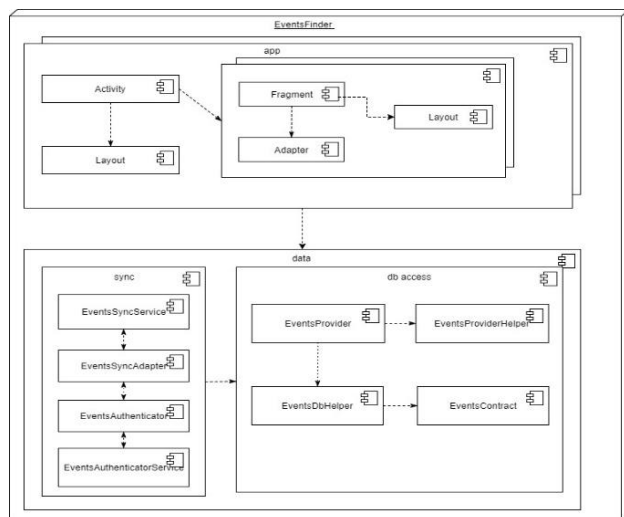
Klasom *Venue* predstavljeno je mjesto odvijanja događaja. Jedan događaj se odvija na jednom mjestu, a na jednom mjestu može se odvijati nula ili više događaja.

Klasom *Location* predstavljena je tačna lokacija odvijanja događaja. Jedan događaj ima tačno jednu lokaciju odvijanja. Na jednoj lokaciji može da se odvijati više događaja, a ne mora ni jedan.

Na dijagramu 4.1 prikazane su dvije glavne komponente *data* i *app*, sačinjene od grupa komponenti, od kojih se prva odnosi na dobavljanje i skladištenje podataka, dok druga obuhvata komponente same aplikacije koje se prije svega odnose na prikaz i manipulaciju sadržajem.

Komponentu podataka, *data*, čine *sync* komponenta za dobavljanje podataka sa eksternog servisa i sinhronizaciju sa istim, i *db access* komponenta zadužena za skladištenje dobavljenih podataka u SQLite bazi, kao i interakciju sa bazom u skladu sa potrebama aplikativnog sloja. Sinhronizacija sa eksternim servisom se vrši pomoću adaptera za sinhronizaciju, a za komunikaciju sa bazom je odgovoran dobavljač sadržaja *ContentProvider*.

Aplikativni sloj čini više komponenti od kojih svaka sadrži aktivnost, *layout* date aktivnosti i komponenta koju čine fragment, odgovarajući *layout* i adapter (za svaku aktivnost postoji više ovakvih komponenti, stoga su na dijagramu predstavljene simbolom za grupu komponenti).



Slika 4.1 Dijagram komponenti

5. IMPLEMENTACIJA

Aplikacije su implementirane u oba jezika i izvršeno je njihovo poređenje.

Za skladištenje podataka u oba slučaja je korišćena SQLite [3] baza podataka. U bazi su smješteni podaci o entitetima opisanim u prethodnom poglavlju. Za dobavljanje podataka koristi se javno dostupni izvor podataka, što je u ovom slučaju bio Eventful API [4].

Podaci su dobavljani pomoću javno dostupnih REST [5] (eng. *REpresentational State Transfer*) *endpointa* u JSON (eng. *JavaScript Object Notation*) formatu, te su skladišteni u internoj bazi kako bi rad aplikacije bio brži, ali i moguć kada internet konekcija nije dostupna. Komponenta koja je korišćena za preuzimanje podataka sa javnog API-a je adapter za sinhronizaciju (en. *sync adapter*).

Nakon što su podaci uspješno dobavljeni sa API-a, oni se čuvaju u SQLite bazi. Interakcija sa bazom omogućena je kroz implementaciju tri osnovne komponente. Definisan je ugovor (en. *contract*) između aplikacije i baze gdje je za svaku relaciju u bazi definisana klasa koja implementira interfejs *BaseColumns* i u kojoj su navedeni nazivi relacija i njihovih kolona.

Contract je u Java implementaciji obična klasa sa po jednom unutrašnjom statičkom klasom za svaku relaciju, dok je u Kotlinu objekat koji za svaku relaciju sadrži klasu u čijem prijateljskom objektu su navedeni pomenuti nazivi. Druga komponenta je *Helper* koji nasljeđuje *SQLiteOpenHelper* i redefiniše metode *onUpgrade* i *onCreate* gdje su kreirane relacije.

Prilikom definisanja iskaza za kreiranje relacija u Kotlinu su korišćeni šabloni za definisanje stringova. Treća komponenta je dobavljač sadržaja *EventsProvider* koji nasljeđuje *ContentProvider* i definiše sve metode koje su dostupne aplikacijama za komunikaciju sa bazom.

Svaki pojedinačni ekran je implementiran kao aktivnost. Unutar aktivnosti su grupisana određena ponašanja, koja predstavljaju dijelove GUI-a, i za njihovu implementaciju su korišćeni fragmenti. Za svaki fragment je definisan odgovarajući adapter koji je zadužen za prikaz podataka u definisanom *layout*-u.

Fragmenti u kojima se prikazuju podaci iz baze implementiraju *LoaderCallbacks* iz *Loader Manager*-a, te se podaci iz baze dobavljaju kada *Content Resolver* razriješi dobijeni URI pozivom odgovarajuće metode *EventsProvidera*. Kada su podaci učitani, okida se *callback* metoda *Loader Manager*-a i obavještava se adapter o tome. Adapteru se proslijeđuju dobavljeni podaci i prikaz se osvježava pozivom metode *notifyDataSetChanged*.

6. DEMONSTRACIJA

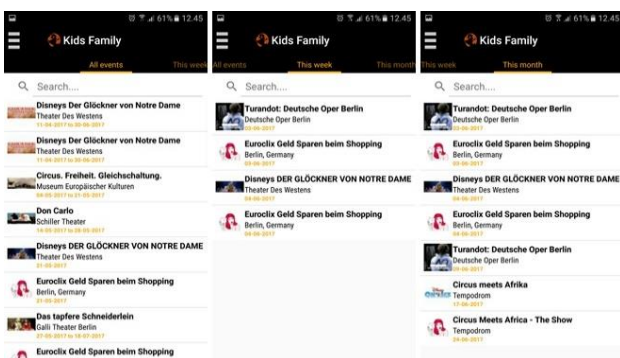
Prilikom definisanja UI-a za svaku aktivnost je kreiran isti *layout*, s tim da je u aplikaciji implementiranoj u Javi primjenjena tema sa primarnom zelenom bojom, dok je u Kotlin verziji primarna boja narandžasta. Sve predviđene funkcionalnosti su uspješno implementirane u oba jezika.

Na slici 6.1 prikazani su početni ekrani aplikacija sa izlistanim kategorijama događaja.



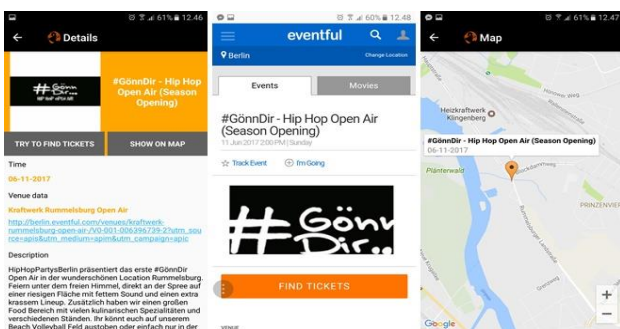
Slika 6.1 Početni ekran, Kotlin, Java

Na slici 6.2 je dat prikaz listi događaja izabrane kategorije (*Kids and Family*) u Kotlin aplikaciji.

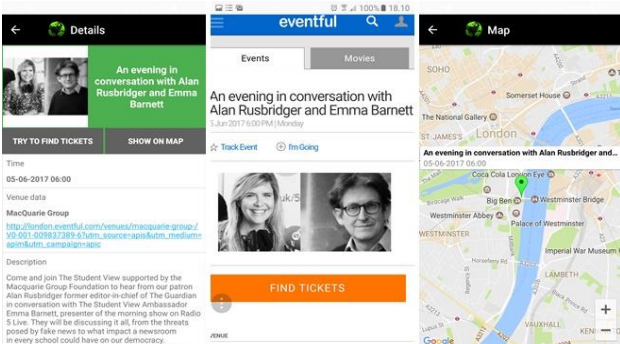


Slika 6.2 Lista događaja, Kotlin

Na slikama 6.3 i 6.4 je dat prikaz detalja izabranog događaja, njegovih detalja na Eventful sajtu i prikaz lokacije na mapi.



Slika 6.3 Detalji događaja, Kotlin



Slika 6.4 Detalji događaja, Java

7. ZAKLJUČAK

Da bi se izvršilo detaljno poređenje Kotlina sa Javom u kontekstu razvoja Android aplikacija, implementirane su aplikacije koje zadovoljavaju istu specifikaciju zahtjeva, te je izvršeno poređenje njihovog razvoja u ova dva jezika. Na osnovu urađenog, moguće je zaključiti da je Kotlin jezik vrlo sličan Javi i nije težak za učenje. Dizajniran je tako da bude intuitivan i lak za čitanje i pisanje, te je prelazak sa Jave na Kotlin lak.

Kao jezik nove generacije sadrži dobre osobine funkcionalnog i proceduralnog programiranja. Iako u zadnje vrijeme Javu proširuju različitim funkcionalnim elementima kao što su lambda izrazi ili funkcionalni interfejsi, Java je prije svega proceduralni jezik. Izlazak Jave 8 je ogroman pomak u procesu modernizacije jezika, ali ovo nije od velikog značaja za razvoj Android aplikacija dok Java 8 ne postane neizostavan dio Android svijeta.

Ako sumiramo dobre i loše strane Kotlina, može se reći da je Kotlin zaista odličan jezik. S obzirom da je odnedavno postao i jedan od zvaničnih jezika za razvoj Android aplikacija, Kotlin zajednica će bez sumnje nastaviti da raste. Međutim, treba imati u vidu da se Kotlin i dalje razvija kao jezik opšte namjene, a ne samo za Android. *JetBrains* tvrdi da će se i u budućnosti Kotlin razvijati nezavisno od Android Studio-a, ali će svakako poboljšavati i Android Studio *plugin*. Takođe će se nastaviti i razvoj podrške u *IntelliJ IDEA* razvojnom okruženju. Pravac daljeg razvoja ovog jezika je trenutno usmjeren na kreiranje Kotlin/Native verzije za druge platforme kao što su *macOS*, *iOS* and *IoT/embedded* sistemi.

8. LITERATURA

- [1] JetBrains, <https://www.jetbrains.com/>
- [2] The billion dollar mistake, <https://www.infoq.com/presentations/Null-References-The-Billion-Dollar-Mistake-Tony-Hoare>
- [3] SQLite, <https://www.sqlite.org/>
- [4] Eventful API, <http://api.eventful.com/>
- [5] REST, https://en.wikipedia.org/wiki/Representational_state_transfer

Kratka biografija:

Zorana Deurić je rođena 18.09.1992. godine u Milićima. Osnovnu školu „Vuk Karadžić” završila je 2007. godine. Gimnaziju „Milorad Vlačić” u Vlasenici završila je 2011. godine. Zvanje diplomirani inženjer elektrotehnike i računarstva stekla je 2015. godine

MOBILNA APLIKACIJA ZA DISTRIBUIRANO PRIKUPLJANJE PODATAKA O WI-FI MREŽAMA

MOBILE APPLICATION FOR DISTRIBUTED COLLECTION OF DATA ABOUT WI-FI NETWORKS

Hunor Kiš, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Rešenje koje je prikazano u ovom radu omogućava pronalaženje otvorenih bežičnih mreža. Za implementaciju sistema je korišćen crowdsensing pristup. Sistem se sastoji od Android aplikacije i od veb servisa koji je implementiran u Spring okviru.

Abstract – The solution presented in this paper enables the search for open wireless networks. Crowdsensing approach was used to implement the system. The project consists of an Android application and a web service, which is implemented in the Spring framework.

Ključne reči: Android, crowdsensing, GPS, Wi-Fi

1. UVOD

Zadatak ovog rada bio je dizajn i implementacija mobilne aplikacije za prikupljanje podataka o lokaciji bežičnih mreža i prikaz tih podataka korisnicima da bi im se olakšao pristup internetu kada su mogućnosti za pristup internetu ograničene.

Za rešavanje ovog problema aplikacija koristi crowdsensing pristup. Crowdsensing [1] je pristup po kome grupa uređaja automatski prikuplja podatke pomoću senzora i šalje te podatke serveru koji ih agregira. Svaki uređaj snima karakteristike bežičnih mreža uključujući i njihove GPS (Global Positioning System) koordinate, podatke šalje na server i čini ih dostupnim ostalim uređajima.

Slične aplikacije već postoje. Dve najpopularnije Android aplikacije sa Play Store-a su Wifi Finder i Wifi Mapper. Novorazvijeni softver ujedinjuje pozitivna svojstva obe aplikacije i ne ograničava se na prikaz javnih bežičnih pristupnih tačaka nego pokazuje sve dostupne bežične mreže.

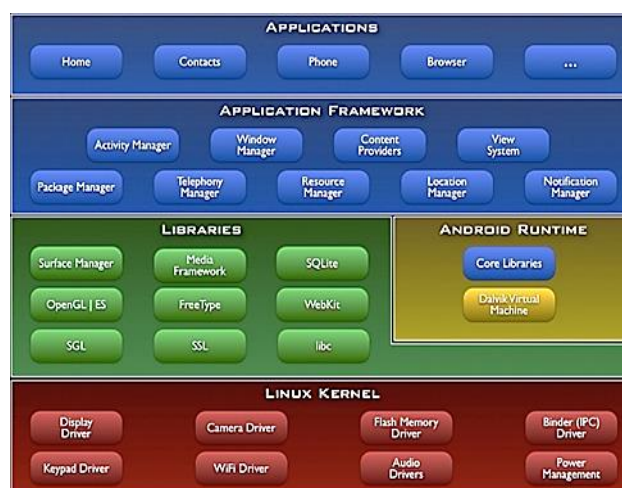
2. KORIŠĆENE TEHNOLOGIJE

Razvoj mobilnih aplikacija se razlikuje od razvoja desktop aplikacija i zahteva drugi pristup u razvoju od strane programera.

Programer treba da se fokusira na optimizaciju aplikacije, da bi aplikacija bila funkcionalna na uređajima različitih karakteristika. U ovom odeljku su predstavljeni osnovni elementi Android platforme (slika 1) i elementi Android aplikacija, vodeće platforme za razvoj mobilnih aplikacija.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio Stevan Gostojić.



Slika 1. Struktura Android operativnog sistema

Osnova Android operativnog sistema jeste Linux kernel. Glavni zadatak kernela jeste rukovanje memorijom i upravljanje nitima. Na ovom nivou Android platforme se nalaze i upravljački programi za hardverske komponente kao što su kamera, ekran osetljiv na dodir, senzori, itd.

Iznad kernela se nalaze biblioteke kao što su: libc, SSL, SQLite, OpenGL/ES, itd. Ove biblioteke služe za pokretanje Android okruženja i pisane su u C/C++ programskom jeziku. Android aplikacije su pisane u Java programskom jeziku i pokreću se pomoću Dalvik virtualne mašine koja je Java virtualna mašina optimizovana za mobilne uređaje sa procesorom slabijeg takta i sa manjom količinom memorije.

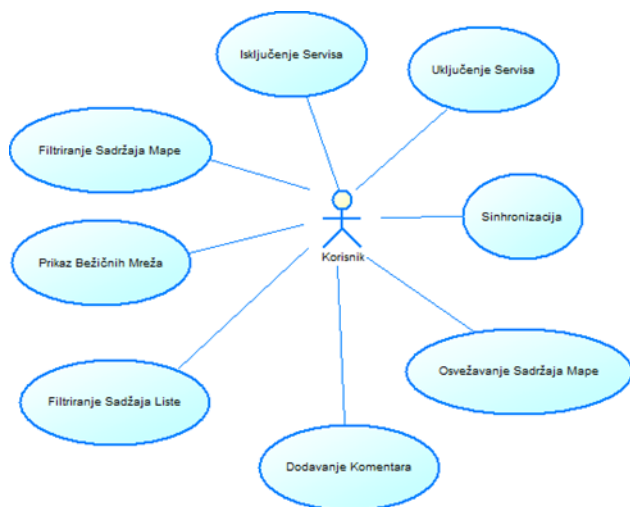
Android aplikacija može da se sastoji od četiri tipa komponenti: aktivnosti, servisa, prijemnika poruka i dobavljača sadržaja. Za implementaciju Android aplikacije korišćen je i Google Maps API koji pojednostavljuje preuzimanje podataka sa Google Maps servisa i olakšava implementaciju mapa na veb stranicama i mobilnim aplikacijama.

Za implementaciju veb servisa je korišćen Spring (tj. Spring Boot) okvir. Spring je veoma popularan okvir za razvoj veb aplikacija baziran na programskom jeziku Java. Obezbeđuje fleksibilnost u konfigurisanju JavaBeans na više načina: XML deployment descriptors, Annotations i JavaConfig.

Spring Boot ubrzava razvoj tako što pojednostavljuje konfiguraciju Spring okvira i omogućava da se Spring bazirana veb aplikacija samostalno pokrene.

3. SPECIFIKACIJA ZAHETVA

U ovom odeljku opisani su funkcionalni zahtevi koji su prikazani i u dijagramu slučajeva korišćena (slika 2).



Slika 2. Slučajevi korišćenja

Sistem treba da podrži: prikaz bežičnih mreža na mapi, prikaz bežičnih mreža u listi, filtriranje sadržaja mape, filtriranje sadržaja liste, dodavanje opisa mreže, osvežavanje opisa mreže, uključivanje ili isključivanje servisa i sinhronizaciju podataka. Android aplikacija treba da na mapi iscrta sve bežične mreže koje se nalaze u bazi podataka.

Filtriranje mreža na mapi treba omogućiti na tri načina: izbor prikaza svih mreža, samo otvorenih mreža ili samo mreža koje sadrže opis. Bežične mreže koje se nalaze u bazi podataka treba da se prikažu i u listi. Filtriranje mreža u list po nazivu se vrši preko tekstualnog polja.

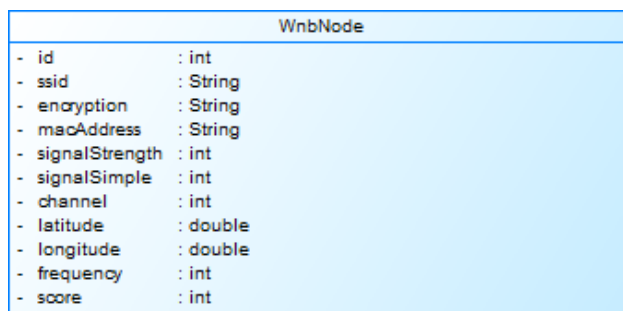
Korisniku treba omogućiti opisivanje mreže i izmenu opisa mreže. Takođe treba omogućiti i uključivanje i isključivanje pozadinskog servisa. Kada je servis isključen, ne prikupljaju se podaci o bežičnim mrežama. Podatke je potrebno sinhronizovati između mobilnih aplikacije i veb servisa.

4. SPECIFIKACIJA DIZAJNA

U ovom odeljku su pomoću UML dijagrama specificirani bitni aspekti sistema. Sistem sadrži mobilnu aplikaciju i veb servis.

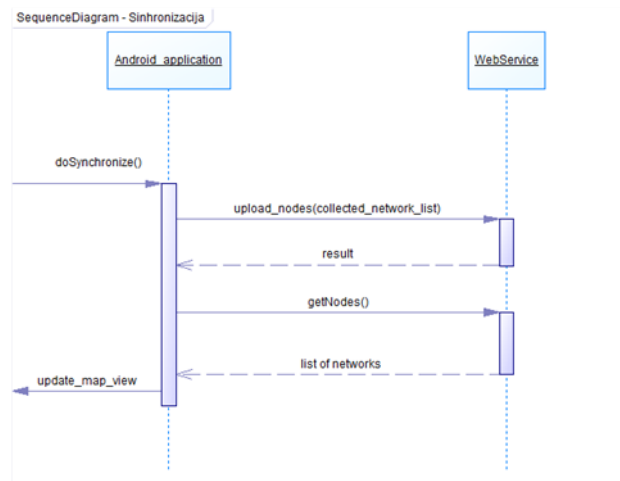
Mobilna aplikacija sadrži dve komponente: aktivnost i servis. Aktivnost se koristi za komunikaciju sa korisnikom, a servis za izvršavanje pozadinskih operacija. Mobilna aplikacija preko TCP/IP i HTTP protokola komunicira sa veb servisem koji je dostupan na javnoj IP adresi. U okviru veb servisa se nalazi RestController koji obrađuje zahteve i čuva podatke u bazipodataka.

Slika 3 prikazuje WnbNode klasu koja predstavlja model podataka. Promenljiva *id* služi za jedinstvenu identifikaciju, *ssid* je ime mreže, *macAddress* MAC adresa uređaja, *channel* kanal, *frequency* frekvencija, *signalStrength* jačina signala, *latitude* i *longitude* geografska širina i geografska dužina, *encryption* tip enkripcije i *score* ocena mreže.



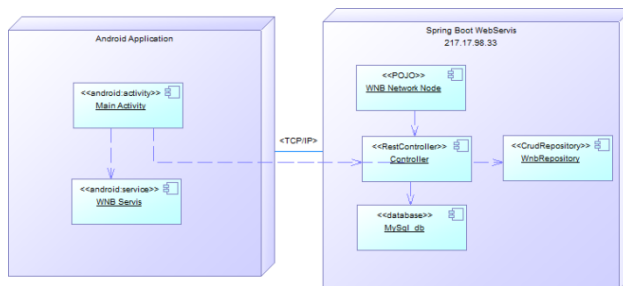
Slika 3. Model podataka

Slučaj korišćenja „Sinhronizacija“ je opisan pomoću dijagrama sekvence (slika 4).



Slika 4. Dijagram sekvence za sinhronizaciju

Kada se pokrene sinhronizacija, sistem inicijalizuje listu sa sakupljenim podacima o bežičnim mrežama koju šalje veb servisu. Veb servis obrađuje podatke i unosi nove podatke u bazu podataka. Kada se obrada podataka završi, šalje se JSON poruka o rezultatu obrade. Ako je slanje podataka uspešno završeno, počinje preuzimanje podataka sa veb servisa. Android aplikacija kao rezultat dobija listu novih podataka i novi podaci se snimaju u bazu podataka aplikacije.



Slika 5. Dijagram rasporeda sistema

Dijagram rasporeda (slika 5) se sastoji od: Android aplikacije i veb servisa. Android aplikacija sadrži aktivnost i servis. MainActivity je aktivnost koja definiše izgled aplikacije. Servis je nadležan za funkcionalnost aplikacije. Android servis preko TCP/IP i HTTP protokola komunicira sa Spring Boot veb servisem koji je dostupan na javnoj IP adresi. U okviru veb servisa se nalazi RestController koji obrađuje zahteve i čuva podatke u MySql bazi.

5. IMPLEMENTACIJA SISTEMA

U ovom odeljku opisana je implementacija sistema. Sistem se sastoji od mobilne aplikacije implementirane u Android okviru, veb servisa koji je implementiran u Spring okviru i MySQL sistema za upravljanje bazama podataka.

Sve klase mobilne aplikacije se nalaze unutar `com.wnb.wnb_tab` paketa. Klasa `WnbMainActivity` je nadležna za inicijalizaciju aplikacije i implementaciju korisničkog interfejsa koji se sastoji od nekoliko fragmenata. Svaki fragment odgovara jednoj kartici u aplikaciji i sadrži poseban `layout.xml` fajl u kome je definisan njegov izgled. Svaka kartica ima svoju ulogu: prva je za prikaz mapa, druga daje pregled sakupljenih podataka, na trećoj se nalazi lista dostupnih mreža i četvrta je nadležna za kontrolu nad servisom.

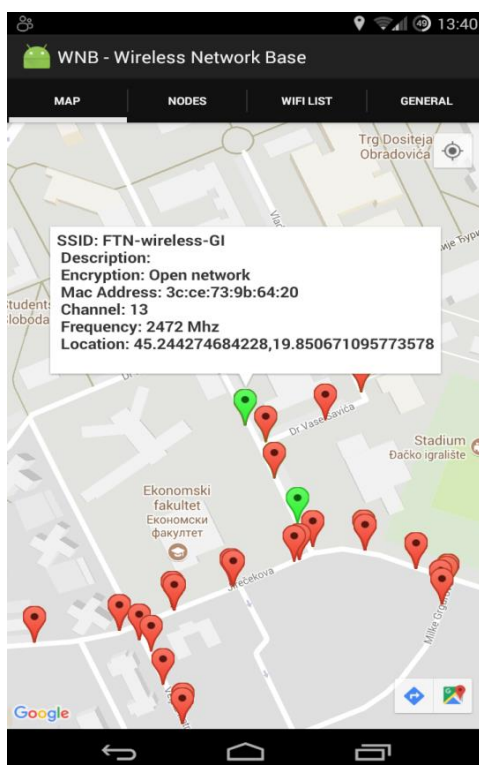
Servis se instancira kod pokretanja aplikacije. `WnbService` je komponenta koja u pozadini skuplja podatke o Wi-Fi mrežama i GPS lokacijama. Istovremeno, zadatak ove komponente je da čuva podatke u bazi podataka datog uređaja i da ih kasnije sinhronizuje sa veb servisom.

U ovoj klasi je definisana unutrašnja klasa `LocalBinder` koja omogućava pristup servisu.

`WnbWebServis` je Spring Boot aplikacija koja omogućava da prikupljeni podaci budu dostupni svim uređajima na kojim je pokrenuta aplikacija. Spring Boot aplikacija se sastoji od kontrolera, modela (POJO klasa) i redefinisane repozitorijuma. Kontroleri su nadležni za rukovanje HTTP zahtevima, repozitorijum olakša upravljanje entitetima, a POJO (*Plain Old Java Object*) klasa sadrži model podataka u bazi.

6. DEMONSTRACIJA SISTEMA

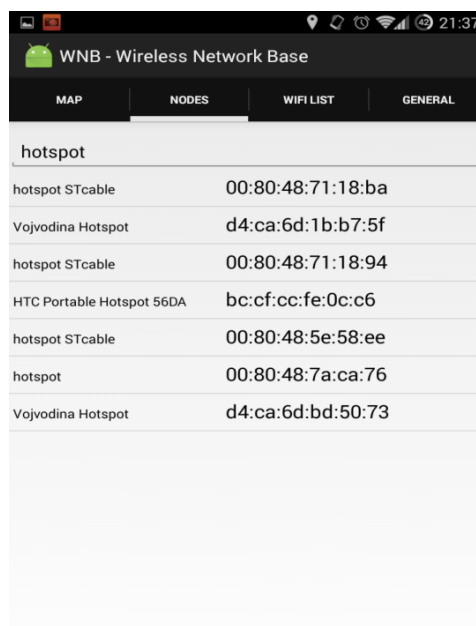
U ovom odeljku je demonstrirana korisnička sesija. Posle pokretanja aplikacije prikazuje se glavni prozor koji sadrži četiri kartice (slika 6).



Slika 6. Interfejs aplikacije

Svaka kartica ima svoju ulogu. Prva kartica služi za prikaz mape na kojoj su bežične preče prikazane kao markeri. Crveni markeri predstavljaju zatvorene bežične mreže, a zeleni markeri otvorene bežične mreže. Svaki marker sadrži informacije o izabranoj mreži: SSID (*Service Set Identifier*), MAC (*Media Access Control*) adresa, kanal, frekvencija, geografska širina, geografska dužina i tip enkripcije. Pored toga, moguće je videti i komentare korisnika.

Druga kartica prikazuje bežične mreže u obliku liste sa mogućnošću filtriranja (slika 7).



Slika 7. "Nodes" kartica sa filtriranjem

Treća kartica prikazuje bežične mreže koje su dostupne uređaju.

Četvrta kartica se koristi za podešavanje aplikacije. Iz ove kartice se može isključiti ili uključiti prikupljanje podataka o bežičnim mrežama.

Pored toga, omogućeno je i pokretanje sinhronizacije podataka između mobilne aplikacije i veb servisa pomoću „Synhronize“ dugmeta. Tada se prikazuje prozor koji označava da je sinhronizacija u toku.

Prozor nestaje nakon završetka sinhronizacije. Kartica takođe omogućava filtriranje mape na prvoj kartici dugmičima „Show All Networks“ (prikazuje sve mreže), „Show Only Open Networks“ (prikazuje sve otvorene mreže) i „Show Networks with Description“ (prikazuje samo mreže koje imaju komentar).

7. ZAKLJUČAK

U ovom radu je prikazana Android aplikacija koja koristi crowdsensing kao metod prikupljanja podataka o bežičnim mrežama. Aplikacija sakuplja podatke o

bežičnim mrežama sa trenutnim GPS koordinatama i snima ih u bazu podataka veb servisa.

Implementirana aplikacija može se unaprediti na više načina. U budućnosti je moguće poboljšati aplikaciju tako da podržava dva režima rada: *offline* i *online*. Trenutno aplikacija učitava sve snimljene podatke sa servera i to značajno usporava njen rad. Bilo bi poželjno implementirati funkciju koja omogućava da klijent unapred preuzme listu mreža za određeni grad.

Za lakše pronalaženje mreže, dobro bi bilo implementirati modul pomoću koga aplikacija može da planira put od trenutne lokacije klijenta do izabrane bežične mreže.

Takođe, aplikacija može da posluži kao osnova za formiranje GeoIP baze podataka. GeoIP baza podataka služi za određivanje mesta na osnovu IP adrese klijenata. Ako bi aplikacija snimila IP adrese bežičnih pristupnih tačaka i GPS koordinate, mogla bi se formirati GeoIP baza podataka.

8. LITERATURA

- [1] Mobile Crowdsensing, <https://goo.gl/ATzwcS>
- [2] „Android (operating system)“, <https://goo.gl/24aK4v>
- [3] „Google says there are now 1.4 billion active...“, <https://goo.gl/Ceiipd>
- [4] „Android Architecture“, <https://goo.gl/3bRpn3>
- [5] „Android rendszer működése, felépítése“, <https://goo.gl/D2P4lr>
- [6] „Android version history“, <https://goo.gl/bgscLx>
- [7] „Activity Lifecycle“, <https://goo.gl/slioUg>
- [8] „Activity“, <https://goo.gl/MeoEXv>
- [9] E.Peter, F.Marcell, F.Bertalan, K.Imre, “Android-alapú szoftverfejlesztés”, SZAK Kiadó Kft. 1977.
- [10] „Service Lifecycle“, <https://goo.gl/Pae6aB>
- [11] “Services”, <https://goo.gl/AC4MRw>
- [12] “Broadcast Receiver”, <https://goo.gl/N2oGYR>
- [13] “Broadcast Receiver”, <https://goo.gl/j6ZPm2>
- [14] “Globalni pozicioni sistem”, <https://goo.gl/BsqzV3>
- [15] “Global Positioning System”, <https://goo.gl/IDMyKY>
- [16] “Wi-Fi”, <https://goo.gl/wFY8pV>
- [17] „Why Spring Boot?“, <https://goo.gl/j5c9Ss>
- [18] “Key Components of Spring Boot Framework”, <https://goo.gl/QVCDXN>
- [19] “SpringBootComponents”, <https://goo.gl/WR1sdt>

Kratka biografija:



Hunor Kiš rođen 15.03.1988. godine u Subotici. 2007. godine upisao je studentski program Računarstvo i automatika na Fakultetu tehničkih nauka u Novom Sadu. Nakon završenih osnovnih akademskih studija, upisao je master akademske studije na istom fakultetu i na istom smeru.

**SISTEM ZA PRIKUPLJANJE PODATAKA I PREDIKCIJU TRAJANJA VOŽNJE
JAVNIM PREVOZOM****SYSTEM FOR COLLECTING PUBLIC TRANSPORT DATA AND PREDICTING PUBLIC
TRANSPORT TRAVEL TIMES**

Nikola Vukadinov, Stevan Gostojić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je prikazana specifikacija i implementacija sistema za prikupljanje, obradu i prikazivanje podataka o vožnjama javnim prevozom. Sistem se sastoji od mobilne aplikacije, koja prikuplja i prikazuje podatke, i serverske aplikacije, koja obrađuje podatke. Prikazan je proces prikupljanja i analize podataka o javnom prevozu i određivanje najboljih modela za klasifikaciju aktivnosti korisnika i predikciju vremena trajanja vožnje javnim prevozom.

Abstract – This paper presents the specification and the implementation of a system for collecting, processing and presenting public transport data. The system consists of a mobile application, which collects and presents data, and a server application, which processes data. It also presents the process use to collect and analyze public transport data and determine the best machine learning models for classification of user activities and prediction of travel time.

Ključne reči: javni prevoz, crowdsensing, klasterizacija, klasifikacija, regresija, Android

1. UVOD

Transport je aktivnost koja ima brojne ekonomske, socijalne i ekološke uticaje. U poslednjih nekoliko decenija čovečanstvo je svedok velikih promena – sve veća urbanizacija, promena klime izazvana ljudskim faktorom, očekivani vrhunac proizvodnje nafte (eng. *peak oil*) – koje utiču da se preispita način na koji se organizuje transport [1]. Uticaj transporta na zagađenje je velik – on je odgovoran za četvrtinu emisija gasova staklene bašte i postoji trend uvećanja emisija koji je veći u odnosu na bilo koji drugi sektor. Dobrom organizacijom javnog prevoza u gradovima može se doprineti rešavanju prethodno navedenih problema. Informacione tehnologije mogu doprineti većoj popularnosti javnog prevoza, ali i unapređenju samog sistema javnog prevoza. Korisnicima se može omogućiti korišćenje veb sajtova i mobilnih aplikacija koje pružaju informacije, kao i korišćenje pametnih kartica za plaćanje karata. Informacije mogu uključivati redove vožnje, informacije o linijama, stanicama i cenama karata, kreiranje ruta za korišćenje javnog prevoza i podatke o vremenima polaska vozila sa određenih stanica. Dok se postojanje jednostavnih veb sajtova ili mobilnih aplikacija sa osnovnim informacijama može očekivati, za napredne funkcionalnosti,

koje zahtevaju određenu infrastrukturu, kao što je plaćanje putem pametnih kartica ili praćenje vozila, potrebno je ulaganje od strane preduzeća za javni prevoz koje nije uvek ostvarivo.

Sa druge strane, velika popularnost pametnih telefona mogla bi da omogući prikupljanje podataka o javnom prevozu koji bi se zatim mogli iskoristiti za ekstrakovanje informacija o javnom prevozu. Ovaj rad opisuje implementaciju sistema za skupljanje, obradu i prezentaciju podataka koji koristi mobilne uređaje.

Sistem omogućava prikazivanje osnovnih informacija o linijama, stanicama i redu vožnje javnog prevoza i naprednih informacija kao što su vremena dolaska vozila na stanice. Za implementaciju ovog sistema neophodno je implementirati mobilnu aplikaciju koja će omogućiti prikupljanje i prikazivanje podataka i serversku aplikaciju koja će omogućiti njihovu obradu. Prikupljanje podataka obuhvata problem detekcije vožnje u javnom prevozu, dok obrada podataka podrazumeva korišćenje podataka skupljenih koristeći mobilne aplikacije za predviđanje trajanja vožnje.

2. PREGLED SLIČNIH APLIKACIJA

Postoji veliki broj aplikacija koje pružaju informacije o javnom prevozu. To mogu biti aplikacije za javni prevoz koje pokrivaju veći broj gradova, kao što su *Moovit*, *Citymapper* i *Transit*, aplikacije za javni prevoz koje pokrivaju jedno preduzeće za javni prevoz ili jedan grad, kao što su *Los Angeles Metro and Bus* i *Subway Paris*, i aplikacije za navigaciju koje sadrže funkcionalnost za javni prevoz, kao što je *Google Maps*. Ove aplikacije obuhvataju sličan skup funkcionalnosti, koji uključuje pregled stanica i linija javnog prevoza, pregled vremena polazaka javnog prevoza sa stanica, kreiranje ruta između različitih lokacija i prikazivanje uputstava za navigaciju po ruti. Aplikacije mogu dobijati podatke direktno od javnog prevoznika, putem javno objavljenih podataka od strane preduzeća za javni prevoz i skupljanjem podataka od korisnika aplikacije.

3. ANDROID

Android predstavlja operativni sistem za mobilne uređaje razvijen od strane Gugla. Prvenstveno je namenjen za mobilne uređaje sa ekranom osjetljivim na dodir, pametne telefone i tablete, ali su vremenom razvijeni i Android operativni sistemi za televizore, automobile i pametne satove. Gugl je predstavio prvu verziju Androida 2007. godine zajedno sa kreiranjem Open Handset Alliance konzorcijuma čiji je cilj kreiranje otvorenih standarda za

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Stevan Gostojić.

mobilne uređaje. Trenutna verzija Android je 8.0 (Oreo), objavljena u maju 2017. Android predstavlja najpopularniji operativni sistem za mobilne uređaje sa preko dve milijarde aktivnih korisnika i 2,7 miliona aplikacija na Gugl Pleju (eng. *Google Play*), oficijelnom repozitorijumu Android aplikacija. Otvorena priroda i široka upotreba kreirala je veliku zajednicu programera koja doprinosi daljem razvoju funkcionalnosti i modifikovanih verzija operativnog sistema.

4. ISTRAŽIVANJE I ANALIZA PODATAKA

Istraživanje i analiza podataka je interdisciplinarno polje informatike koje se bavi otkrivanjem obrazaca u skupovima podataka i nalazi se na preseku mašinskog učenja, statistike i baza podataka. Osnovni cilj procesa istraživanja i analize podataka predstavlja izvlačenje informacija iz skupa podataka i njihovo transformisanje u strukturu (tj. model za kasniju upotrebu).

Klasifikacija predstavlja problem smeštanja podataka u prethodno definisane klase na osnovu obučavajućeg skupa, gde su podaci već smešteni u klase.

Klasterizacija predstavlja problem nenadgledanog učenja koji odgovara klasifikaciji. Klaster predstavlja grupu sličnih podataka, gde se sličnost drugačije definiše za različite klasterizacione modele.

Regresiona analiza predstavlja statistički proces određivanja veze između zavisnog atributa i jednog ili više nezavisnih atributa (tj. načina na koji nezavisni atributi utiču na zavisni atribut).

U sklopu rešavanja postavljenog problema metode za istraživanje i analizu podataka su korišćene u sklopu rešavanja dva problema:

- detektovanja pešačenja, ulaska u vozilo i vožnje i
- predikcije trajanja vožnje na osnovu sakupljenih podataka o aktivnostima i lokacijama tokom vožnji.

4.1. Skup podataka o vožnjama javnim prevozom

Paralelno sa razvojem aplikacije kreiran je skup podataka kako bi se omogućila implementacija. Skup podataka je formiran koristeći senzore na korisničkim uređajima kojima je omogućeno merenje podataka o aktivnostima i lokacijama korisnika uređaja i sadrži podatke izmerene tokom 97 vožnji javnim prevozom. Za kreiranje skupa podataka implementiran je servis na Android uređaju za čuvanje podataka o aktivnostima i lokacijama, koji je ručno uključivan neposredno pre i isključivan neposredno nakon vožnje javnim prevozom.

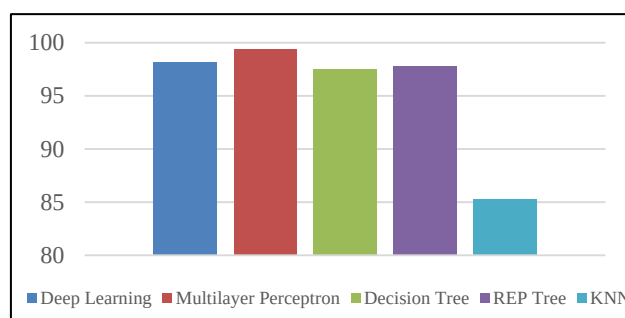
Paralelno sa snimanjem vožnji javnog prevoza koristeći Android uređaj, skupljani su podaci o vremenu, skladištenjem podataka dobijenih od *OpenWeatherMap API*-ja u bazu podataka.

Za rešavanje problema predikcije vremena vožnje potrebno je prethodna obrada podataka koja na osnovu prikupljenih podataka kreira podatke o vremenima putovanja između parova stanica (tj. segmenata linije javnog prevoza).

4.2. Detektovanje ulaska u vozilo i pešačenja

Problem detektovanja ulaska u vozilo i pešačenja predstavlja problem klasifikacije gde je na osnovu podataka o aktivnostima potrebno izvršiti njihovu klasifikaciju i odrediti aktivnost korisnika.

Rešenje ovog problema potrebno je za rad servisa za detekciju vožnje čiji je cilj skupljanje podataka na osnovu kojih će se vršiti predikcija vremena putovanja. Za detekciju ove dve aktivnosti se koristi Android *ActivityRecognition API* koji koristi senzore telefona i generiše podatke u obliku verovatnoća određene vrste aktivnosti. Kreiranje modela za klasifikovanje aktivnosti izvršeno je u dva koraka. U prvom koraku je korišćeno nenadgledano učenje kako bi se klasterovalo merenje aktivnosti u 4 klastera, koji su zatim obeleženi aktivnostima kojim odgovaraju. Kreirani su klasteri *in_vehicle*, *on_foot*, *still* i *tilting*. Za klasterizaciju je korišćen algoritam k-srednjih vrednosti. Uzimajući u obzir potencijalnu nepreciznost merenja i klasterovanja, primenjen je drugi korak mašinskog učenja koji obuhvata korišćenje kreiranog modela za klasterovanje i tri uzastopna merenja u kreiranju klasifikacionog modela. Uzimajući u obzir da klaster *tilting* ne nosi nikakvu informaciju o kretanju korisnika, za kreiranje klasifikacionog modela i njegovu kasniju primenu korišćene su samo vrednosti koje pripadaju klasterima *in_vehicle*, *on_foot* i *still*. Grafik 1. prikazuje preciznosti testiranih modela za klasifikaciju aktivnosti.



Grafik 1. Preciznost modela za klasifikaciju aktivnosti

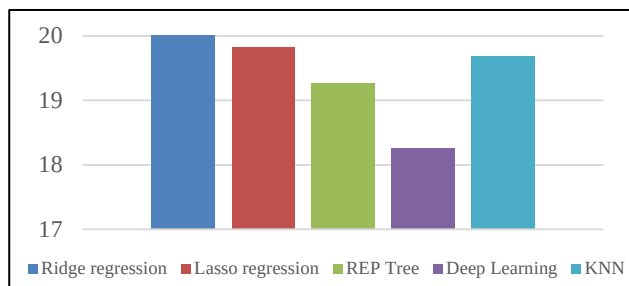
Višeslojni perceptron odabran je kao model koji se koristi u aplikaciji s obzirom na najvišu preciznost.

4.3. Predikcija trajanja vožnje

Problem predikcije trajanja vožnje predstavlja problem regresije gde je potrebno proceniti trajanje vožnje na osnovu ulaznih podataka. Za rešavanje ovog problema korišćeni su podaci o segmentima zajedno sa podacima o vremenskim uslovima. Predikcija trajanja vožnje neke linije između određenog niza stanica dekomponovana je na predikciju trajanja vožnje između parova stanica (segmenata) gde linija ne igra nikakvu ulogu. S obzirom da postoji relativno mali broj segmenata sa većim brojem podataka, razvijena su dva modela – prvi za segmente koji imaju minimalan broj podataka u kome se koriste podaci za te segmente, i drugi za segmente koji nemaju minimalan broj podataka u kome se koriste svi podaci o segmentima uz metod za izjednačavanje razlika između segmenata. Problem predikcije trajanja vožnje predstavlja problem regresije. U ovom slučaju zavisni atribut predstavlja trajanje vožnje, dok su nezavisni atributi ostali podaci o segmentu i podaci o vremenskim uslovima.

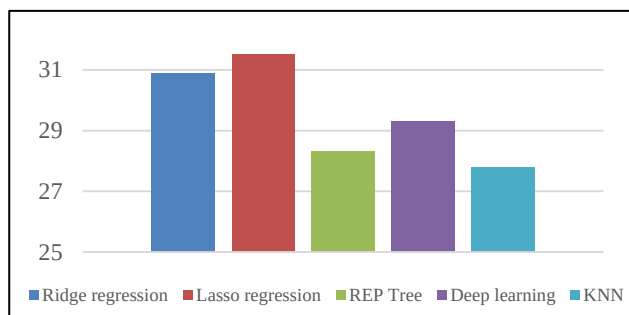
Regresioni model je odabran analizom različitih modela nad podacima grupisanim po segmentima. Grafik 2. prikazuje prosečnu grešku svih modela obučanih nad pojedinačnim segmentima koji zadovoljavaju minimalan broj podataka.

Algoritam dubokog učenja kao jedan od najkompleksnijih algoritama, uz primenjenu regularizaciju koja smanjuje preterano prilagođavanje obučavajućem skupu, pokazao se kao najbolji model.



Grafik 2. Prikaz grešaka modela za predikciju vremena trajanja vožnje

Za razliku od metoda kreiranja zasebnih regresionih modela za svaki segment opisanog u prethodnom odeljku, za segmente koji nemaju minimalan broj podataka kreiran je jedan regresivni model na osnovu svih podataka o segmentima uz dodate nove atribute koji izjednačavaju razlike između njih. Rezultati različitih modela prikazani su na grafiku 3. REP stablo odlučivanja i algoritam k najbližih suseda pokazali su se kao modeli sa najvećom preciznošću.



Grafik 3. Prikaz grešaka modela za predikciju vremena trajanja vožnje

5. SPECIFIKACIJA ZAHTEVA

Funkcionalni zahtevi su predstavljani dijagramom slučajeva korišćenja, prikazanom na slici 1., koji opisuje sve slučajeve u kojima korisnik koristi funkcije definisane u sistemu.

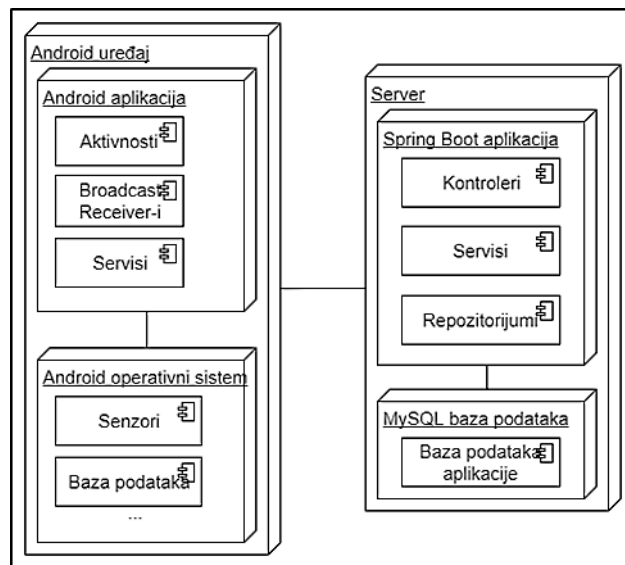


Slika 1. Dijagram rasporeda sistema

Nefunkcionalni zahtevi postavljeni nad sistemom su enkriptovana razmena podataka između mobilne i serverske aplikacije kako bi se zaštitila privatnost korisnika, optimizacija potrošnje baterije mobilnih uređaja i potrošnje interneta na mobilnim uređajima.

6. DIZAJN

Arhitektura sistema se bazira na klijent-server modelu. Ovaj model se sastoji od serverske i klijentske aplikacije, gde klijentska aplikacija inicira komunikaciju slanjem zahteva, na koje server odgovara. U implementiranom sistemu mobilna aplikacija predstavlja klijenta, dok server predstavlja aplikacioni i server baze podataka. Slika 2. prikazuje dijagram rasporeda implementiranog sistema. Dijagram se sastoji od dva elementa koji opisuju Android uređaj i server i koji su povezani vezom koja opisuje njihovu međusobnu komunikaciju. Android uređaj je opisan sa dve komponente – operativni sistem, koji omogućava dobijanje podataka od senzora, rad sa SQLite bazom podataka [2] i druge funkcionalnosti, i Android aplikacija, čija je funkcionalnost opisana koristeći Android gradivne elemente – aktivnosti, prijemnike poruka (eng. *broadcast receiver*) i servise. Server je opisan sa dve komponente – MySQL sistemom za upravljanje bazom podataka [3] koji sadrži bazu podataka serverske aplikacije i Spring Boot aplikacijom [4] koja predstavlja serversku aplikaciju čija je troslojna arhitektura opisana kontrolerima, servisima i repozitorijumima.



Slika 2. Dijagram rasporeda sistema

6.1. Model podataka

Model podataka na serveru omogućava skladištenje podataka o javnom prevozu u proizvoljnom broju gradova uz podatke o merenjima poslatih od strane korisničkih uređaja i podataka dobijenih njihovom obradom. Model podataka omogućava sinhronizaciju sa podacima na korisničkim uređajima nakon njihove izmene na serveru.

Model podataka na korisničkim uređajima predstavlja podskup modela podataka sa serveru koji nakon inicijalnog preuzimanja omogućava izvršavanje dela funkcionalnosti bez potrebe za konstantnim preuzimanjem podataka sa servera.

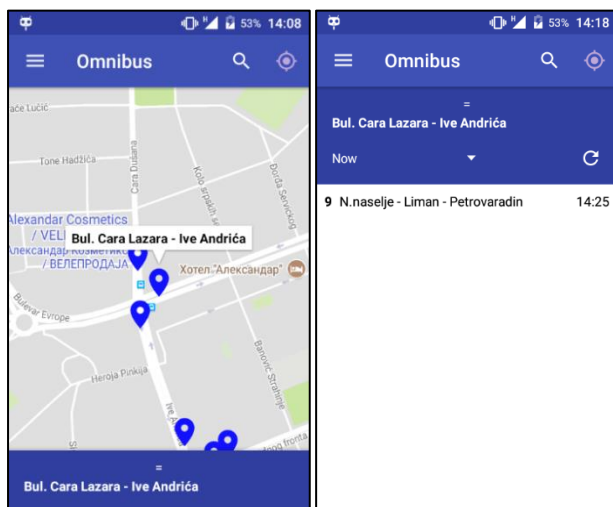
7. IMPLEMENTACIJA

Serverska aplikacija je implementirana u *Spring Framework*-u uz *MySQL* bazu podataka. Za mapiranje između *Java* objekata i podataka u bazi podataka korišćen je *Hibernate*. Za implementaciju je korišćeno nekoliko dodatnih biblioteka pored onih koje uključuje korišćenje *Spring Boot Framework*-a. Korišćen je *Spatial4j* [5], biblioteka za geoprostorne kalkulacije, i *Weka* [6], biblioteka algoritama mašinskog učenja. Korišćeno je nekoliko *API*-ja koji su doprineli nekim od bitnih funkcionalnosti aplikacije - *OSRM* [7], koji omogućava funkcionalnosti rutiranja, *OpenWeatherMap* [8], preko kojeg su dobavljani podaci o vremenu, i *Nominatim* [9], koji omogućava određivanje naziva lokacija na osnovu geografske tačke (eng. *reverse geocoding*).

Korisnička, tj. mobilna aplikacija implementirana je na *Android* platformi. Pri implementaciji aplikacije korišćene su dodatne *Android* biblioteke za podršku za starije uređaje i korišćene funkcionalnosti za mape i određivanje lokacije, kao i *Ormlite* [10], *ORM* biblioteka za *Android* uređaje, nekoliko biblioteka za elemente grafičkog interfejsa i deo *Weka* biblioteke. Sinhronizacija podataka je implementirana koristeći *SyncAdapter* komponentu *Android* platforme, koja omogućava prenos podataka između aplikacije i servera. Algoritam za detekciju i prikupljanje podataka implementiran je kao *started* servis, tj. servis koji nakon što je pokrenut može da se izvršava neodređeno vreme, čak iako se komponenta koja ga je pokrenula uništi. Ovaj servis koristi se za detekciju vožnje. Model mašinskog učenja koji omogućava detekciju vožnje se na serveru, prosleđuje mobilnoj aplikaciji prilikom sinhronizacije podataka i čuva u fajl sistemu mobilnog uređaja.

8. DEMONSTRACIJA

Glavna aktivnost omogućava pregled stanica na mapi, polazaka sa trenutne lokacije, selektovanih lokacija i stanica. Mapa prikazuje stanice javnog prevoza, koje su označene plavim markerima. Selekcijom markera stanice, mapa se centrira nad njim i aplikacija prikazuje njen naziv i podatke o sledećim polascima (videti sliku 3.).



Slika 3. Skica arhitekture sistema

Selekcijom lokacije na mapi dodaje se marker na mapu i prikazuje se naziv lokacije i sledeći polasci u njenoj blizini. Moguće je koristiti i druge funkcionalnosti koje uključuju pretraživanje stanica, linija i lokacija, pregled linija i linije.

9. ZAKLJUČAK

Kreiran je skup podataka o vožnjama javnim prevozom koji obuhvata podatke o lokacijama, aktivnostima i vremenskim uslovima. Koristeći kreirani skup podataka testirani su različiti modeli mašinskog učenja za klasifikaciju aktivnosti koja je potrebna za detekciju vožnje na mobilnom uređaju i predikciju vremena trajanja vožnje. Modeli koji su pokazali najbolje rezultate korišćeni su u implementaciji sistema koji se sastoji od mobilne i serverske aplikacije. Mobilna aplikacija je implementirana na *Android* platformi i omogućava prikupljanje podataka o vožnjama i prikaz podataka o javnom prevozu. Serverska aplikacija implementirana je na *Spring* platformi i omogućava sinhronizaciju podataka sa mobilnim aplikacijama i periodičnu obradu podataka o vožnjama javnim prevozom.

S obzirom da se predikcija trajanja vožnji javnim prevozom zasniva na podacima skupljenim koristeći korisničke mobilne uređaje i javno dostupne podatke, moguće je lako proširivanje aplikacije na nove linije i gradove.

Jedna mana aplikacije jeste da trenutno ne postoji dovoljna količina podataka da bi se pokrile različite vrednosti koje ulazni podaci predikcionog modela mogu imati. To negativno utiče na preciznost predikcionih modela. Korišćenjem aplikacije od strane korisnika skup podataka će se proširivati, što će omogućiti kreiranje preciznijeg modela za predikciju vremena trajanja vožnji.

Drugom manu predstavlja nedostatak određenih funkcionalnosti koje su prisutne u najpopularnijim aplikacijama ovog tipa kao što je kreiranje ruta za korišćenje javnog prevoza u cilju navigacije između lokacija ili praćenje vožnje uživo. Jedna od potencijalnih funkcionalnosti može biti i autentifikacija korisnika, što bi omogućilo personalizaciju aplikacije i automatizovan prenos korisnikovih podataka na više uređaja.

10. LITERATURA

- [1] V. Stjernborg, O. Mattisson, „The Role of Public Transport in Society – A Case Study of General Policy Documents in Sweden“, <http://www.mdpi.com/2071-1050/8/11/1120/pdf>, November 2016
- [2] Sqlite, <https://www.sqlite.org/>
- [3] Mysql, <https://www.mysql.com/>
- [4] Spring, „Spring Boot“, <https://projects.spring.io/spring-boot/>
- [5] Spatial4j, <https://github.com/locationtech/spatial4j>
- [6] Weka, <http://www.cs.waikato.ac.nz/ml/weka/>
- [7] Open Street Routing Machine, <http://project-osrm.org/>
- [8] OpenWeatherMap, <https://openweathermap.org/>
- [9] OpenStreetMap Vikipedija, „Nominatim“, <http://wiki.openstreetmap.org/wiki/Nominatim>
- [10] Ormlite, <http://ormlite.com/>

Kratka biografija:



Nikola Vukadinov je rođen 10.12.1992. godine u Novom Sadu. 2011. godine upisao je Fakultet tehničkih nauka, odsek Računarstvo i automatika. Nakon završenih osnovnih studija 2015. godine, upisao je master studije na kojima je položio sve ispite predviđene planom i programom.

RAZVOJ VOĐEN TESTOVIMA TEST-DRIVEN DEVELOPMENT

Miloš Marić, Branko Milosavljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast –ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj članak treba da prikaže testiranje i razvoj vođen testiranjem kroz njegove teorijske osnove, i da prikaže trenutno stanje tog pristupa razvoju kroz studiju slučaja.*

Abstract – *This article should demonstrate testing and test-driven development through their theoretical basis, and also the current state of test-driven development through a study case.*

Ključne reči: *Test, razvoj, studija, funkcionalnost, ponašanje, modul*

1. UVOD

Članak će se baviti razvojem vođenim testovima iz teoretske, kao i iz praktične perspektive u vidu studije slučaja. Razlikuju se četiri veće celine koje će približiti proces testiranja, razvoj vođen testovima kao i samu implementaciju studije slučaja koristeći ga.

Prva celina će se baviti testiranjem softvera uopšteno kroz teorijske osnove i razlike između raznih pristupa. Druga celina detaljno objašnjava razvoj vođen testovima, glavne pristupe i napredne prakse. Treća celina će se baviti specifikacijom i implementacijom studije slučaja. Četvrta celina pokriva analizu rezultata dobijenih studijom.

2. TESTIRANJE SOFTVERA

Kvalitet softverskog rešenja predstavlja trenutno stanje funkcionalnosti u odnosu na neki ideal u oblasti čije funkcionisanje rešenje potpomaže. Na karakteristike ideala kojem softverska rešenja teže utiču mnogi faktori, ali najviše sama specifikacija zahteva koja se postavlja pred razvojni tim softvera. Pored specifikacije, na sliku ideala, i samim time na kvalitet utiču i konkurentski proizvodi posmatrane oblasti, ranije verzije samog proizvoda, očekivanja korisnika, kao i standardi i zakoni koji se odnose na posmatranu oblast [1].

Primarna svrha testiranja softvera [2] je da se sistem dovede u stanje neuspeha, čime se mogu otkriti i ispraviti greške. Konačan broj testova ne može da pokrije sva moguća stanja u koja sistem može doći, ali može ustanoviti da sistem ne radi kako je očekivano u specifičnim uslovima. Testiranje često uključuje pregledanje programskog koda kao i njegovo izvršavanje u raznim okruženjima i uslovima.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red.prof.

Testiranje softvera se takođe može definisati kao proces kojim organizacija koja razvija ili na bilo koji drugi način učestvuje u razvoju proizvoda pokušava proceniti da li će on biti prihvatljiv krajnjem korisniku, kao i ljudima koji ulažu u njegov razvoj.

2.1. Metode testiranja softvera

Metode testiranja softvera se mogu podeliti na dva različita načina. Prvi način je podela u odnosu na to kada se dešavaju testovi posmatrajući od početka izvršavanja programa:

- Statičko testiranje – pre nego što se kod pokrene
- Dinamičko testiranje – tokom i posle pokretanja

Drugi način je podela u odnosu na to kako se posmatra izvorni kod:

- *White-box* testiranje – testira se unutrašnja struktura koda poznavajući sam kod.
- *Black-box* testiranje - testira se funkcionalnost bez ikakvog znanja o njoj implementaciji
- *Grey-box* testiranje – testira se kao crna kutija ali su u cilju boljeg dizajniranja testova poznati struktura podataka i korišćeni algoritmi.

2.2. Nivoi testiranja softvera

Testovi se svrstavaju u različite nivoe po različitim kriterijumima, među kojima su trenutak kad su dodati u toku razvojnog procesa kao i nivo specifičnosti samog objekta testiranja. Glavni nivoi testova su definisani u SWEBOK standardu [3] i tu spadaju jedinični, integracioni i sistemski testovi [4][5].

Jedinični testovi (*unit tests*) proveravaju funkcionalnost malog dela koda, najčešće na nivou funkcija. U slučaju objektno-orijentisanog okruženja, jedinični testovi se odnose na klase, pri čemu najmanji testovi testiraju konstruktore i destruktore. Ove testove najčešće pišu programeri koji pišu i kod (*white-box* metod) čime se osigurava da funkcija radi kako je očekivano. Jedna funkcija može imati više testova kako bi se obuhvatili granični slučajevi kao i svi tokovi unutar funkcije.

Integracioni testovi (*integration tests*) su bilo koji tip testova koji proverava interfejse komponenti u odnosu na predviđeni dizajn softvera. Ovi testovi nastoje da razotkriju defekte u interfejsima i prilikom saradnje integrisanih komponenti, odnosno modula. Progresivno se testiraju sve veći delovi softvera, sve dok softver ne proradi kao celina.

Sistemski testovi (*system tests*) proveravaju funkcionalnosti sistema kao celine ocenjujući da li zadovoljava zahteve. Ovakvi testovi se realizuju *black-box* metodom. Sistemski testovi se fokusiraju kako na interakciju unutar sistema kao celine tako i sistema sa okolinom.

2.3. pristupi testiranju softvera

Pristupi testiranju se ogledaju kroz razvojne modele koji se međusobno razlikuju po tome kada, ko i kako testira softver. Neki od pristupa su tradicionalni, “od gore ka dole”, “od dole ka gore” i agilni.

Tradicionalni “vodopad” (*waterfall*) pristup [6] nalaže da testiranje vrši nezavisna grupa testera pošto je funkcionalnost implementirana, a pre nego što je došla do korisnika. Ovakva praksa često rezultuje time da deo faze testiranja bude iskorišćen kao nastavak faze razvoja, što smanjuje vreme posvećeno testiranju.

“Od gore ka dole” (*top-down*) predstavlja pristup integracionom testiranju gde se prvo testiraju integrisani moduli sa vrha, a zatim je svaki deo tog modula testira, korak po korak, sve dok se svi moduli ne istestiraju.

“Od dole ka gore” (*bottom-up*) predstavlja pristup integracionom testiranju komponenti najnižeg nivoa, zatim pošto se iste integrišu, olakšava se testiranje modula višeg nivoa. Ovaj pristup je najbolji kad su svi ili većina modula istog nivoa spremni u isto vreme. Ovaj pristup je takođe zgodan jer omogućava da se napredak testiranja kvantifikuje, najčešće u procentima.

Agilni ili ekstremni pristup nalaže da se testovi pišu od strane programera koji će razvijati kod, pre nego što se sam kod napiše. Pošto testovi za nepostojeći kod “puknu”, inkrementalno se piše kod koji zadovoljava sve veće testove. Cilj ovog pristupa je da se dostigne neprekidna integracija gde se nove verzije softvera mogu često plasirati [7]. Ovakav model razvoja softvera se naziva “Razvoj vođen testovima” ili “*Test driven development*” (TDD).

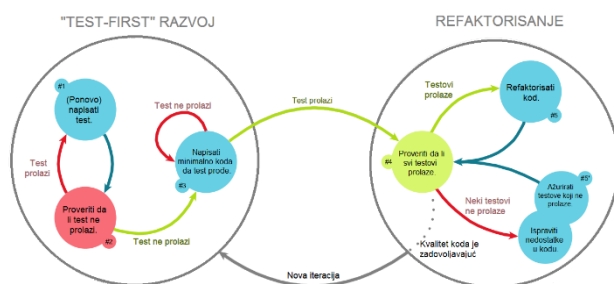
3. RAZVOJ VOĐEN TESTOVIMA

Razvoj vođen testovima (*Test-driven development* - TDD) u svojoj osnovi predstavlja inkrementalnu razvojnu praksu koja se sastoji od:

- određivanja i razumevanja zahteva koji se implementira,
- specifikovanja funkcionalnosti u vidu testa,
- osiguravanja mogućnosti da test ne uspe (ne prođe),
- pisanja implementacije funkcionalnosti u cilju zadovoljavanja testa
- refaktorisanja koda (u slučaju da je to potrebno) u cilju poboljšanja unutrašnje strukture i
- osiguravanja da svi testovi prolaze.

3.1. Tok razvoja vođenog testovima

Tok razvoja vođenog testovima [8] se može ilustrovati sa dve veće celine prikazane na slici 1.



Slika 1. Tok razvoja vođenog testovima

Kao što se vidi na slici 1, dve celine toka razvoja vođenog testovima daju ukupno pet koraka kroz koje se prolazi:

1. Pisanje testa,
2. Pokretanje novog testa,
3. Pisanje koda,
4. Ponovno pokretanje testova,
5. Refaktorisanje koda.

Prvi korak toka razvoja vođenog testovima počinje odabirom funkcionalnosti iz specifikacije zahteva koja treba da bude implementirana. Ideja je da to bude mala funkcionalnost koja se kasnije može integrisati u implementaciju neke veće. U odnosu na odabranu implementaciju se piše test koji je definiše ili poboljšava. Da bi napisao test, programer mora u potpunosti da razume odabranu funkcionalnost kroz njenu specifikaciju zahteva. Ovakav pristup zahteva fokus programera prema samom zahtevu pre nego što počne da ga implementira.

Pošto je napisan test, prelazi se na drugi korak u kom se na početku proverava da li novo napisani test prolazi ili ne. Ideja je da test ne prođe, u slučaju nove funkcionalnosti zato što ona još uvek nije implementirana, dok u slučaju proširenja već implementirane funkcionalnosti zato što trenutna implementacija ne podržava proširenje koje test proverava. Činjenicom da novi test ne prolazi sam test zadobija poverenje programera, što odbacuje mogućnost da je test faličan i da će uvek proći.

Na treći korak se prelazi pošto je utvrđeno da novi test ne prolazi. Programer implementira funkcionalnost koju zahteva napisani test iz prvog koraka. Piše se minimalna količina koda koja je potrebna da bi test prošao. Implementacija u ovom koraku mora biti najjednostavnija moguća implementacija posmatrane funkcionalnosti jer je njena jedina svrha da test prođe. Pošto se traži minimalna i najjednostavnija implementacija, očekivano je da ona neće biti elegantna. To je apsolutno prihvatljivo, jer se time očuvava jednostavnost, a i napisani kod će biti ulepšan u korak u pet.

Četvrti korak počinje u trenutku kada je funkcionalnost implementirana tako da test prolazi. U ovom koraku se pokretanjem svih testova proverava da li je izmena u kodu koja je rezultovala prolaskom testa iz koraka jedan loše uticala na ostatak, do sada implementiranih funkcionalnosti. U slučaju da zbog novog koda barem jedan test ne prolazi, potrebno je podešavati novi kod dok svi testovi ne prođu.

Poslednji korak predstavlja prečišćavanje (refaktorisanje) koda [9][10]. Jedan od načina prečišćavanja predstavlja premeštanje implementacije posmatrane funkcionalnosti sa mesta gde je postavljen da bi test prošao, na mesto gde logički pripada. Potom se kod čisti od koda koji se ponavlja, kao što su “magični brojevi” i “magični string-ovi”. Pošto ne postoji ponavljanje u kodu, prelazi se na menjanje imena gradivnih elemenata koda kao što su objekti, klase, moduli, promenljive i metode, da bi njihov naziv predstavljao njihovu svrhu. Proširenjem ili dodavanjem neke funkcionalnosti klase i tela metoda postaju sve veća, tako da se vrši razdvajanje na manje celine, da se ne bi izgubilo na čitljivosti, održivosti i jednostavnosti. Neprekidnim ponovnim pokretanjem testova tokom svih faza prečišćavanja koda se postiže sigurnost da se nijedna funkcionalnost nije “pokvarila”.

Pošto je poslednji korak završen, ciklus se završava i počinje novi odabirom nove funkcionalnosti i pisanjem testa za nju. Ciklus treba da bude mali, sa do deset izmenjenih linija koda za celokupan ciklus. U slučaju da napisan kod ne zadovoljava test brzo, ili drugi testovi neočekivano počnu “pucati”, programer treba da poništi izmene i da se vrati na prvi korak gde će ponovo razmotriti odabranu funkcionalnost.

Ovakva implementacija obezbeđuje mnogo tačaka sa malo promena koje su lako poništive. Prilikom korišćenja eksternih biblioteka, implementacija odabrane funkcionalnosti treba da bude veća od samog korišćenja biblioteke, jer se u tom slučaju testira sama biblioteka.

3.2. Pristupi TDD-u

TDD je vremenom podlegao raznim pristupima, od kojih se izdvajaju dva glavna [11] zbog značajnih razlika među njima, uglavnom kada se posmatra dizajn:

- *Classicist* pristup i
- *Outside-in* pristup.

Klasičan (*Classicist*) pristup predstavlja originalan pristup TDD-u, poznat kao “Detroitska škola” TDD-a. Ovaj pristup odlikuju sledeće karakteristike:

- Dizajn se dešava tokom faze prečišćavanja (refaktorisanja).
- Testovi su obično bazirani na stanju.
- Tokom faze refaktorisanja, testirana jedinica može da poraste u više delova / klasa.
- *Mock*-ovi (simulirane vrednosti) se retko koriste, sem ako je potrebno izolovati spoljne sisteme.
- Nema planiranja dizajna unapred. Dizajn proizilazi iz koda.
- Jednostavan je za razumeti i lako je prilagoditi mu se.
- Često je korišćen zajedno sa četiri osnovna pravila jednostavnog dizajna softvera.
- Pogodan za *black-box* testiranje.
- Odličan kada se ne možemo osloniti na eksperte.

Outside-in pristup, poznat još i kao “Londonska škola” TDD-a je razvijen od strane jednih od prvih praktikanata ekstremnog programiranja i odlikuju ga sledeće karakteristike:

- Određen je pravac razvoja testova, počevši od spolja (elementa koji prima spoljni zahtev) ka unutra (elementa koje sadrže jednostavne delove ponašanja koji zajedno zadovoljavaju funkcionalnost koju je potrebno implementirati).
- Počinje se sa testom prihvatanja koji proverava da li funkcionalnost radi kao celina. Test prihvatljivosti služi kao vodič za implementaciju.
- “Pucanje” testa prihvatljivosti obaveštava zašto funkcionalnost nije još uvek implementirana do kraja (šta još nedostaje).
- Prave se pretpostavke o tipu saradnje elementa koji se testira sa spoljnim elementima i pišu se testovi koji proveravaju proces saradnje.
- Pošto je element istestiran, prelazi se na prvi element sa kojim istestiran element sarađuje a nije implementiran, i njegovo ponašanje se implementira na isti način kao i ponašanje prethodnog elementa.

- Dizajn počinje na samom početku, dok se pišu testovi.
- Testovi nisu bazirani na stanju nego na saradnji i ponašanju elemenata.
- Dizajn se poboljšava tokom faze refaktorisanja.
- Svaki element sa kojim se sarađuje i njegov javni interfejs je napravljen da koristi postojećem elementu koji se testira, što čini kod vrlo čitljivim.
- Faza refaktorisanja je mnogo kraća u odnosu na klasični pristup.
- Omogućava bolju “enkapsulaciju” stanja.

Odluka o tome koji od ova dva pristupa treba koristiti zavisi od konkretnih potreba. Oba pristupa se mogu koristiti u isto vreme, ali je iskustvo presudan faktor prilikom prebacivanja sa jednog na drugi pristup.

3.3. Napredne prakse TDD-a

U odnosu na to od čega počinje implementacija funkcionalnosti, na šta se unutar testa stavlja akcenat i koliko je ovakav razvoj zapravo razumljiv ljudima koji nisu programeri a učestvuju u razvoju softvera, izdvojile su se tri unapređene prakse TDD-a:

- ATDD – *Acceptance test-driven development*,
- BDD – *Behavior-driven development* i
- STDD – *Storytest-driven development*.

ATDD proširuje osnovnu ideju TDD-a da je to ustvari način razvoja softvera time što postaje način komunikacije između klijenta, programera koji razvija softver i testera, čime se osigurava da su zahtevi dobro definisani. Još jedna novina koju ATDD uvodi jeste da su testovi čitljivi od strane klijenta, što za običan TDD ne mora da bude slučaj. Sa strane implementacije, proširenje u odnosu na TDD jeste pisanje i prolazak testova prihvatljivosti u obliku “omotača” u odnosu na već poznat tok TDD-a

BDD i STDD predstavljaju semantičke varijacije ATDD-a. Implementaciono gledano, oba koriste TDD i njihov cilj je da se u proces razvoja, već od samog početka uključe i korisnici, i menadžeri, i tester. Oba unapređenja koriste jezike specifične za domen, koji su sačinjeni od prirodnog jezika. Razlika među njima jeste način posmatranja na zahtev koji je postavljen pred sistem. U slučaju BDD, zahtev se definiše kao ponašanje sistema, dok se u slučaju STDD-a zahtev definiše kao “priča”, odnosno kratka rečenica koju prati detaljniji opis napisan prirodnim jezikom bez posebne strukture.

4. STUDIJA SLUČAJA

Studija slučaja TDD-a predstavlja veb aplikaciju za podršku unutrašnje organizacije kulturno umetničkog društva (KUD-a). Unutrašnja organizacija KUD-a razlikuje tri uloge korisnika:

- Anonimni (neprijavljen) korisnik – bilo ko ko pristupi aplikaciji bez prijave,
- Umetnički rukovodilac – prijavljeni korisnik koji predstavlja rukovodioca neke grupe, i
- Administrator sistema.

Za implementaciju su korišćene sledeće tehnologije: MySQL, SpringBoot, JUnit, Mockito, Joda-time, AJAX, JWT (*JSON Web Token*), JSON, NPM (*Node package*

manager), *ReactJS*, *JQuery*, *MobX* i *Jest*. Za rad sa pomenutim tehnologijama su korišćeni sledeći alati: *MySQL Workbench*, *Eclipse* i *Atom*.

Implementacija studije je realizovana kroz nekoliko većih koraka:

- Model podataka
- Serverska strana (klasični pristup TDD-u – JUnit i Mockito)
- Komunikacija između serverske i klijentske strane (razmena podataka u JSON formatu)
- Klijentska strana (klasični pristup TDD-u - Jest)

5. ANALIZA REZULTATA

Rezultati koje nam je studija slučaja pružila oslikavaju uticaj faktora na sam razvoj TDD-om.

Iskustvo u dizajniranju softvera - ljudi sa manje iskustva u dizajnu pribegavaju korišćenju tradicionalnog pristupa, dok oni malo iskusniji najčešće koriste “*Outside-in*” pristup ili mešaju oba.

Sadržina dela softverskog rešenja koje se implementira – vrlo bitna stavka koja odlučuje da li se uopšte treba koristiti TDD za implementaciju nekog dela softvera. Najčešći razlozi koji dovode do razvoja bez TDD pristupa jesu ako deo koji se implementira ne sadrži nikakvu logiku u sebi, odnosno više je deklarativnog tipa. Drugi razlog zbog kog se neki modul ne bi implementirao TDD-om je ako se u njemu koriste isključivo funkcionalnosti neke uvežene biblioteke. Razvoj takvog modula TDD-om bi efektivno značio testiranje uvežene biblioteke, za čime nema potrebe.

Korišćenje raznih tehnologija – možda i najbitniji faktor kada se gleda na TDD. Današnje platforme za razvoj nude sve više gotovih alata za postizanje često korišćenih funkcionalnosti. Isto tako, sve više se i neke ređe korišćene, parametrizovane funkcionalnosti postižu na deklarativan način, što iziskuje izostavljanje TDD-a.

Programski jezik – faktor od kog zavisi količina koda potrebna da se napiše u cilju implementacije funkcionalnosti, kao i same količine vremena da se isti razvije TDD-om. Faktori koji figurišu najviše su razlike između jezika, od čega su se u studiji najviše pokazale razlike u bibliotekama i platformama za razvoj koje jezik pruža, način modularizacije softvera kao i tipiziranost jezika.

Alati korišćeni za razvoj – pomoćni faktor koji može u mnogome unaprediti sam proces razvoja TDD-om. Razna razvojna okruženja pružaju alate za automatsko pokretanje testova u pozadini, što štedi vreme razvoja.

6. ZAKLJUČAK

Testiranje softvera uopšte nailazi na problem rokova koji se postavljaju pred programere koji implementiraju softversko rešenje. Važnost pisanja testova se ogleda u dugoročnom razvoju i održavanju proizvoda, što rezultuje čestim preskakanjem faze testiranja u cilju ispunjenja rokova. Ovaj problem delimično rešava uvođenje testiranja kao neprekidni proces koji počinje sa početkom razvoja proizvoda i obavlja se paralelno. Posvećeni timovi ljudi koji se bave osiguravanjem kvaliteta softvera u

mnogome pomažu u otklanjanju nedostataka na vreme, dok ih je relativno lako otkloniti.

Sam dizajn arhitekture koji se definiše tokom korišćenja TDD-a je u proseku značajno bolji, ali to ipak zavisi od iskustva onog ko razvija softver. Još jedna u nizu dobrih posledica koje iziskuje korišćenje TDD-a jeste jednostavnost koda. TDD u svakom svom koraku zahteva najjednostavnije moguće rešenje, što dugotrajno pomaže razumljivosti koda. Kao jedna od bitnih etapa u ciklusu implementacije funkcionalnosti TDD-om se izdvaja refaktorisanje, što se generalno jako retko radi. Ova etapa čini kod još čitljivijim i urednijim, do te mere da je samoobjašnjiv.

7. LITERATURA

- [1] <https://codurance.com/2014/12/14/quality-cannot-be-measured/>, pristupljeno u junu 2017.
- [2] Jiantao Pan: “Software testing”, 1999.
- [3] <https://www.computer.org/web/swebok>, pristupljeno u junu 2017.
- [4] Beizer Boris: “Software Testing Techniques”, 1990.
- [5] Mathur Aditya: “Foundations of Software Testing”, 2008.
- [6] Dustin Elfriede: “Effective Software Testing”, 2002.
- [7] Marchenko Artem: “XP Practice: Continuous Integration”, 2007.
- [8] K. Beck: “Test-Driven Development by Example”, 2003.
- [9] Ottinger Tim, Langr Jeff: “Simple Design”, 2013.
- [10] Martin Robert: “Clean Code, A Handbook of Agile Software Craftsmanship”, 2008.
- [11] <https://codurance.com/2015/05/12/does-tdd-lead-to-good-design/>, prisupljeno u junu 2017.

Kratka biografija:



Miloš Marić je rođen 24.3.1994. godine u Novom Sadu, Republika Srbija. Nakon završene osnovne škole “Ivan Gundulić” 2008. godine je upisao gimnaziju “Jovar Jovanović Zmaj” u Novom Sadu. Fakultet tehničkih nauka u Novom Sadu je upisao 2012. godine. Stekao je zvanje diplomiranog inženjera elektrotehnike i računarstva 2016 godine. Ispunio je sve obaveze i položio je sve ispite predviđene studentskim programom.

SISTEM ZA PRIKAZIVANJE PODATAKA O MUZIČKIM FESTIVALIMA

MUSIC FESTIVAL DATA PRESENTATION SYSTEM

Pilip Pilipović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu opisani su funkcionalni zahtevi, dizajn, implementacija i demonstracija klijent-server rešenja koje pruža informacije o rasporedu performansa za događaje festivalskog tipa. Rešenje je implementirano pomoću Android okvira za razvoj mobilnih aplikacija i Parse okvira za razvoj MBaaS servisa. Analizirane su njegove prednosti i mane u odnosu na slične aplikacije i predloženi pravci daljeg razvoja.

Abstract – This paper describes the functional requirements, the design, the implementation and the demonstration of a client-server software solution that displays timeline of festival-like events. The solution is implemented using Android mobile app development framework and Parse MBaaS cloud framework. Strengths and weaknesses of the proposed solution are analysed and direction of future development are given.

Ključne reči: Android, Parse, Mosby, MVP, festival, Google maps.

1. UVOD

Kako vreme odmiče, čini se da događaji festivalskog i edukativnog karaktera poprimaju sve šire razmere. Naime, povećava se sam broj događaja, ali i izvođača ili predavača, broj lokacija na kojima se održavaju, kao i posećenost. Problemi na koji posetioci nailaze uglavnom su u vezi sa istovremenim održavanjem performansa na više različitih lokacija. Autor rada, višegodišnji posetilac pomenutih događaja, pozeleo je više puta da napravi izmene u već gotovom rasporedu, no za to nije našao podjednako brz i efikasan način. Mana štampane verzije rasporeda ogleda se u tome što, u slučaju izmene, ne može da se ažurira. Dakle, nije moguće izostaviti performanse koji korisniku nisu od interesa i pronaći one druge. Dalje, ukoliko se događaj održava noću, neophodno je osvetljenje kako bi štampana verzija rasporeda mogla da se pročita.

Informatička pismenost i dostupnost pametnih telefona odlično su polazište za rešenje ovog problema. Implementacijom mobilne aplikacije za prikaz podataka o rasporedu performansa problem može biti rešen.

Na tržištu već postoje određene mobilne aplikacije. Android aplikacije koje se bave sličnom tematikom su: "FestiApp" [1], "Appic" [2] i "FestivAll" [3]. Aplikacije pri pokretanju prikazuju listu dostupnih događaja, informacije o izvođačima, kao i raspored performansa.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Stevan Gostojić.

Aplikacije nije moguće koristiti ukoliko uređaj nema aktivnu Internet konekciju. U slučaju "FestiApp" aplikacije, korisnik mora da ima Facebook nalog. Dakle, nedostaci se, pored njihove složenosti, ogledaju i u zavisnosti od Internet konekcije i obimnog broja komponenti korisničkog interfejsa.

Glavni zadatak aplikacije opisane u ovom radu je da bude pomoćno sredstvo koje će omogućiti korisnicima dostupnost informacija odabranog događaja u odsustvu Internet konekcije. Internet konekcija potrebna je samo inicijalno, kada se pronalazi željeni događaj i kada se preuzimaju podaci o rasporedu performansa. Od tog momenta, za primarnu namenu aplikacije, pregled i obeležavanje željenih performansa, Internet konekcija više nije neophodna.

U drugom odeljku opisane su tehnologije koje su korišćene za razvoj aplikacije. Specifikacija zahteva i dizajn sistema (kako njegovih statičkih aspekata, tako i njegovih dinamičkih aspekata) izložena je u trećem odeljku. U četvrtom odeljku su opisani bitni elementi implementacije. Demonstracija aplikacije data je u petom odeljku.

U zaključku su analizirane prednosti i mane implementirane aplikacije u odnosu na slične aplikacije, te predloženi pravci daljeg razvoja.

2. PREGLED KORIŠĆENIH TEHNOLOGIJA

Za izradu ove aplikacije korišćene su softverske platforme, biblioteke i veb servisi koje nude kompanije Google i Parse. Klijentski deo rešenja je realizovan na Android platformi, a za serverski deo korišćen je Parse kao mobilna serverska platforma u oblaku (mobile backend as a service – MBaaS).

Parse platforma nudi objektno-relaciono mapiranje i sistem za upravljanje bazom podataka, koji omogućava da se podaci preuzeti sa servera smeštaju u bazu podataka na uređaju. Pored ovih tehnologija, u izradi klijentskog dela upotrebljene su sledeće softverske biblioteke:

- Picasso (biblioteka za olakšano učitavanje slika sa Interneta i njihovo privremeno smeštanje u memoriju zarad ponovnog korišćenja);
- Google Maps Android (sastavni deo biblioteke Google Play Services koji se koristi za prikaz mapa);
- Mosby (biblioteka korišćena za implementaciju Model-View-Presenter obrasca);
- Parse Android SDK (Parse razvojno okruženje za Android).

2.1. Android

Android [4] je mobilni operativni sistem razvijen od strane Google-a, zasnovan na Linux [5] jezgri, razvijen prvenstveno za mobilne uređaje sa ekranom osetljivim na

dodir. Pod uređajima osjetljivim na dodir najčešće se smatraju pametni telefoni i tablet računari. Od datuma pojavljivanja (2007), pa do danas, objavljeno je 26 izdanja Android platforme, operativnog sistema i pratećeg API (Application programming interface), a od toga 8 glavnih izdanja operativnog sistema.

Razvoj aplikacije za Android platformu uglavnom je na Java programskom jeziku. Programski kôd u Java klasama kompajlira se u .class datoteke, koje se zajedno sa kodom biblioteka prevode u .dex kôd za izvršavanje na Dalvik virtualnoj mašini (VM) na starijim verzijama Androida ili ART (Android Run Time) na novijim. Dalvik VM je u stvari Java VM optimizovana za uređaje sa ograničenim resursima. Pametni telefoni i tablet računari, sa svojim ograničenim memorijskim i procesorskim resursima, tipični su predstavnici takve klase uređaja.

Android platforma podržava i JNI (Java Native interface) tehnologiju [6] kojom je omogućeno pisanje delova aplikacije na programskom jeziku C/C++. Najčešće se koristi kod pisanja delova koda koji su kritični po pitanju brzine (obrada slike, zvuka itd.).

Android aplikacije sačinjene su od četiri vrste komponente [7]:

- aktivnosti (jedan ekran korisničkog interfejsa),
- servisa (komponente koje obavljaju dugotrajne obrade podataka u pozadini i uglavnom nemaju korisnički interfejs),
- dobavljača sadržaja (upravljaju podacima koji mogu da se skladište u SQLite bazi podataka, u datotekama, na Internetu itd);
- prijemnika poruka (reaguju na poruke emitovane objave širom sistema; postoji širok spektar objava, od korisničkih do sistemskih).

2.2. Parse

Parse [8] je mobilna serverska platforma za izvršavanje veb aplikacija u oblaku u vlasništvu Facebook-a. Nudi podršku za aplikacije koje su razvijane na raznim platformama, između ostalih i na Android platformi. Sva potrebna hardversko-softverska infrastruktura za razvoj, otpremanje i izvršavanje aplikacija besplatno je dostupna korisnicima. Takođe se nude postavke i alati za praćenje upotrebe resursa i beleženje rada aplikacija. Mobilna aplikacija integrisana sa Parse API-jem može lako da pohranjuje podatke u oblaku, šalje i prima PUSH notifikacije, upravlja korisnicima i geoprostornim podacima i integriše se sa platformama kao što su Twitter i Facebook. Platforma je dizajnirana tako da po potrebi može lako da skalira, u zavisnosti od opterećenja i broja korisnika.

2.3. Mosby MVP (Model-View-Presenter) biblioteka

Mosby je Android biblioteka čiji je cilj da pomogne razvoj dobro dizajnirane arhitekture Android aplikacije. Problem koji ona rešava tiče se pojednostavljenja razdvajanja nadležnosti između komponenti aplikacije. Sadrži klase koje olakšavaju implementaciju MVP šablona na Android platformi.

Razdvajanje nadležnosti u Android aplikacijama korišćenjem MVC šablona zamišljeno je na sledeći način:

- Model može biti bilo koja POJO klasa (tj. običan Java objekat);

- View se definiše XML fajlom u kojem su opisane karakteristike grafičkih elemenata aplikacije,
- Controller predstavljaju Activity i Fragment klase.

U praksi se pokazalo da je ovaj princip dovoljan kada aplikacije nisu previše kompleksne. Povećavanjem kompleksnosti aplikacije, dolazi do nagomilavanja koda u Controller komponentama u kojima se manipuliše View-om. Jedan od razloga za to je i što XML definicija View-a ne podržava složenije manipulacije komponentama, animacijama, ulaznim podacima, itd. Rezultat toga je da se sva logika implementira u Activity ili Fragment klasama.

MVP princip u View komponentu odvaja propratni programski kod koji je potreban za manipulaciju ulaznim i izlaznim podacima, kao i interakciju sa elementima GUI, a u Presenter komponenti je programski kod vezan za logiku višeg nivoa apstrakcije (poslovnu logiku). Model predstavlja podatke koje View treba da prikaže. View implementira interfejs koji prikazuje podatke i na osnovu tih podataka rutira korisnike komande Presenter-u. Presenter predstavlja sponu između Model i View komponenti i ima referencu na obe komponente. Termin "model" je u ovom slučaju malo nejasan, a u suštini, predstavlja skup logike koja dobavlja ili manipuliše modelom.

3. SPECIFIKACIJA SISTEMA

U ovom odeljku specificirani su funkcionalni zahtevi aplikacije. Pored toga, specificiran je i statički model sistema. Statički model sistema čine njegova arhitektura (specificirana pomoću dijagrama rasporeda i komponenti) i model podataka (specificiran kao dijagram klase).

3.1. Funkcionalni zahtevi

Specifikacija sistema uslovljena je korisničkim zahtevima koji su dobijeni intervjuisanjem potencijalnih korisnika aplikacije. Tu grupu čine osobe koje često posećuju festivale. Pored zahteva dobijenih iz intervjua, deo zahteva je prikupljen i analizom postojećih softverskih rešenja.

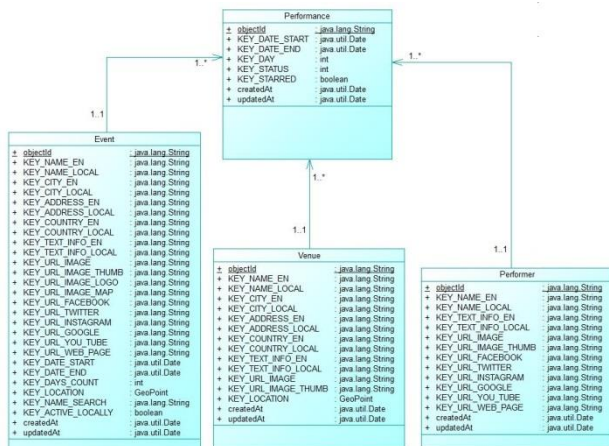
Funkcionalni zahtevi aplikacije su sledeći:

- aplikacija pri pokretanju prikazuje sve raspoložive događaje;
- korisnik može da pretražuje raspoložive događaje na osnovu željenog pojma;
- korisnik može da ručno pokreće proces osvežavanja liste raspoloživih događaja;
- korisnik može da označi željeni događaj kao aktivan;
- korisnik može da pregleda sve performanse aktivnog događaja sortirane po vremenu i lokaciji;
- korisnik može da označi željeni performans, korisnik može da označen performans vrati u neoznačeno stanje;
- korisnik može da pregleda sve označene performanse sortirane po vremenu i lokaciji; i
- korisnik može da na mapi vidi lokacije označenih performansa.

Nakon odabira događaja, korisniku ne treba Internet konekcija za dalji rad.

3.2. Model podataka

Dijagram klasa čine četiri klase: Event (događaj), Performance (performans), Performer (izvođač) i Venue (lokacija). Dijagram klasa prikazana je na slici 1.

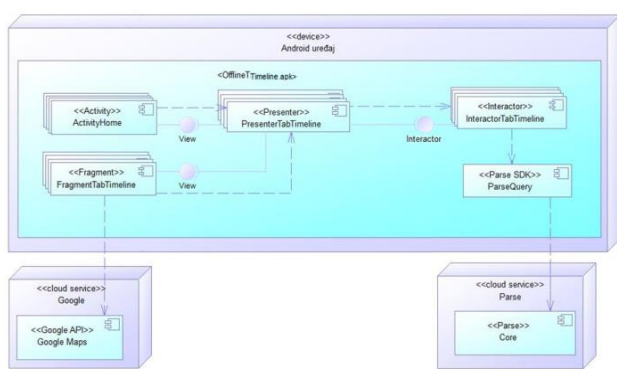


Slika 1. Dijagram klasa modela podataka

Klase sa slike 1. modelovane su po sledećim principima: Svaki performans ima tačno jednog izvođača, izvođen je na tačno jednoj lokaciji i pripada tačno jednom događaju. Događaj pripada u više performanasa. Izvođač pripada u više performanasa. Lokacija pripada u više performanasa.

3.3. Arhitektura sistema

Arhitektura aplikacije zasnovana je na dvoslojnoj klijent-server arhitekturi i prikazana na slici 2. Klijentski deo je Android aplikacija, a serverski deo Parse servis u oblaku. Aplikacija u sebi vrši obradu podataka, pa se kategorise kao "debeli" klijent. Svi podaci se perzistiraju u lokalnu bazu podataka uređaja kako bi se ispoštovao zahtev za korišćenje u uslovima bez pristupa Internetu. Klijent takođe komunicira i sa Google Maps servisima u toku iscrtaavanja mape na jednom od ekrana.



Slika 2. Dijagram rasporeda sistema

4. IMPLEMENTACIJA SISTEMA

U ovom odeljku opisana je implementacija sistema specificiranog u trećem odeljku pomoću tehnologija opisanih u drugom odeljku.

4.1. Mosby MVP

Tokom razvoja aplikacije primenjen je MVP obrazac. *Activity* i *Fragment* klase nasleđuju osnovne klase *ActivityDefault* i *FragmentDefault*, koje nasleđuju

MvpActivity i *MvpFragment* klase Mosby biblioteke. Default klase u sebi implementiraju uslužne metode kao što su: prikaz dijaloga, dobavljanje resursa, provere Internet konekcije, itd.

Klase koje nasleđuju Mosby klase imaju u sebi mehanizam koji vodi računa o životnom ciklusu tih Android komponenti i shodno tome poziva odgovarajuće metode. Jedna od njih je za kreiranje Presenter-a i pozivanje njegovih metoda.

4.2. Parse Android SDK

Parse Android SDK (Software Development Kit) omogućava da se aplikacije razvijene na Android platformi brzo i lako povežu sa Parse servisima u oblaku.

Razvojni tim u tom slučaju ne mora da se bavi implementiranjem logike za povezivanje, prenos i dobavljanje podataka sa servera zato što je ta logika dostupna unutar razvojnog okvira. Time je vreme potrebno za razvoj aplikacije značajno umanjeno.

4.3. Picasso

Picasso [9] predstavlja Android biblioteku koja olakšava proces učitavanja slika. Ona vodi računa o svim koracima potrebnim za učitavanje slika sa spoljnih lokacija. Funkcionalnost ove biblioteke može se iskoristiti u nekoliko linija koda. Posebnu upotrebljivost ova biblioteka pokazuje kod učitavanja slika sa Interneta (od inicijalnog kreiranja HTTP zahteva do smeštanja slike u lokalnoj memoriji).

4.4. Google mape za Android API

Google Maps API [10] predstavljala skup interfejsa i klase koji su u sastavu Google Play Services biblioteke. API omogućava dodavanje elemenata grafičkog korisničkog interfejsa za prikaz Google mapa u Android aplikaciji. API automatski rukuje konekcijom sa Google servisom, pozadinskim preuzimanjem podataka, prikazivanjem mapa i reagovanjem na dodire prstima.

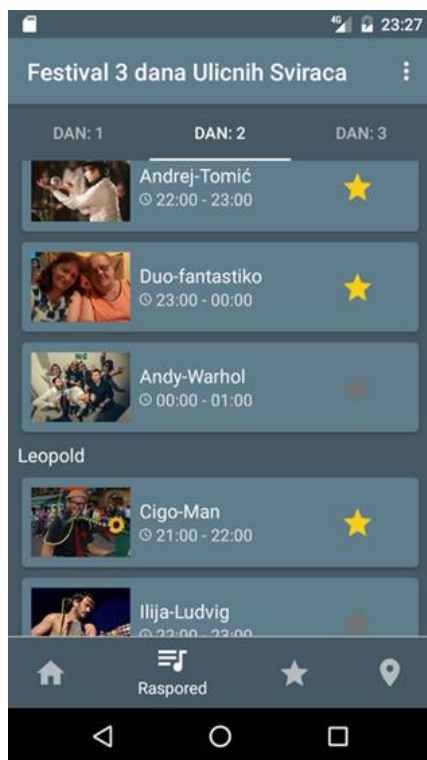
Takođe je omogućeno postavljanje obeleživača, prekrivajućih slojeva preko osnovne mape, iscrtaavanje poligona i kružnica i promena kadra kamere. Ovi elementi vrše interakciju sa korisnikom preko registrovanih osluškivača događaja, pozivanjem callback metoda.

5. DEMONSTRACIJA

Prilikom pokretanja aplikacije i odabira željenog aktivnog događaja, pojaviće se osnovni ekran sa performansima koji se odigravaju u tom trenutku.

Pritiskom na navigaciono dugme "Raspored", dobija se prikaz na kojem su po karticama označeni dani događaja; svaki u sebi ima pojedinačan prikaz performanasa za taj dan (slika 3).

Lista performanasa sortira se po mestu odigravanja i vremenu početka. Svaki označeni performans sa desne strane ima žutu zvezdicu kao indikator da je označen. Pritiskom na sivu zvezdicu, performans postaje označen, a pritiskom na žutu, performans prestaje da biva označen.



Slika 3. Prikaz rasporeda performansa

6. ZAKLJUČAK

Korisnici festivalskih aplikacija najčešće se susreću sa problemima kao što su spor odziv, prenatrpanost funkcijama i reklamama. Zbog raznovrsnosti u ponudi, autori aplikacija često dodaju funkcionalnosti koje korisnicima nisu primarne. Time se unosi bespotrebna kompleksnost koja dovodi do lošeg korisničkog iskustva.

U ovom radu opisani su Android okvir za razvoj mobilnih aplikacija i Parse okvir za razvoj MBaaS servisa i specifikirani su funkcionalni zahtevi i dizajn sistema za prikazivanje podataka o muzičkim festivalima. Sistem je implementiran pomoću Android i Parse okvira i demonstrirano je njegovo korišćenje.

Rešenje predstavljeno u ovom radu nudi osnovni skup funkcija i ima minimalnu potrebu za povezivanjem sa internetom. Brzi odziv i intuitivnost korisničkog interfejsa omogućava korisniku lako snalaženje pri promeni ekrana. Nakon inicijalnog preuzimanja sadržaja izabranog događaja, aplikaciji više nije potrebna Internet konekcija. Od tog momenta sav sadržaj se učitava iz lokalne baze podataka. Tu su izuzete slike, za čije inicijalno učitavanje je potrebna Internet konekcija.

Funkcionalni dizajn je uslovio da se naprave određeni kompromisi. Na primer, korisnik je zadužen da sam osveži rasporede događaja. Za razliku od sličnih aplikacija, ovo rešenje ne nudi pretragu performansa ili izvođača unutar izabranog događaja. Izostavljena je i integracija sa društvenim mrežama, kao i pregled video materijala. Prikaz mape događaja zavisi od Google Maps komponente. Mapa se prikazuje ukoliko uređaj već ima preuzetu mapu u trenutku prikaza. Ukoliko uređaj nema preuzetu mapu oblasti oko događaja, tada je potrebna internet konekcija da bi se ona preuzela. Ipak, najveći nedostatak ovog rešenja je način na koji se prikupljaju

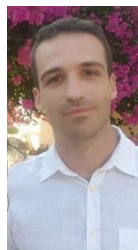
podaci o događajima. Idealno bi bilo da se podaci automatski preuzimaju sa veb strana događaja, bez intervencije organizatora. Postojao je plan da se napravi parser veb stranica koja bi dobavljala podatke o događajima sa zvanične stranice. Međutim, problem nastaje kada se napravi i najmanja izmena u strukturi veb stranice. Na primer, struktura stranica Exit festivala se menjala i po nekoliko puta dnevno u periodu neposredno pred početak festivala.

Postoji i ideja da se pronalaženje prijatelja implementira na principima izmenjene stvarnosti (korisnik bi kroz kameru telefona mogao da vidi strelicu iznad mesta na kojem se nalazi tražena osoba). Za razvoj izmenjene stvarnosti koristio bi se Vuforia [11] razvojni okvir za Android platformu, zajedno sa geoprostornim podacima korisnika.

7. LITERATURA

- [1] FestiApp, "Google Play Store," <https://goo.gl/0hu8hN>. [Poslednji pristup 1. 10. 2016.].
- [2] Appic, "Google Play Store," <https://goo.gl/xEm27F>. [Poslednji pristup 1 10 2016.].
- [3] FestivAll, "Google Play Store," <https://goo.gl/DyO3aZ>. [Poslednji pristup 1 10 2016.].
- [4] Google, "Android Guide," <https://goo.gl/qp12xm>. [Poslednji pristup 2016.].
- [5] Wikipedia, "Linux Kernel," <https://goo.gl/H8nSH9>. [Poslednji pristup 1 10 2016.].
- [6] Google, "Android NDK," <https://goo.gl/ekxfCZ>. [Poslednji pristup 1 10 2016.].
- [7] Google, "Platform Architecture," <https://goo.gl/ZIJloa>. [Poslednji pristup 1 10 2016.].
- [8] Parse, "Android Guide," <https://goo.gl/QcTPss>. [Poslednji pristup 1 10 2016.].
- [9] J. Wharton, "Google Picasso," <https://goo.gl/pbgHKO>. [Poslednji pristup 1 10 2016.].
- [10] Google, "Google Maps API," <https://goo.gl/Wg4yrj>. [Poslednji pristup 2016.].
- [11] Vuforia, "Vuforia AR Platform," <https://www.vuforia.com/>. [Poslednji pristup 1 10 2016.].

Kratka biografija:



Pilip Pilipovic rođen je u Novom Sadu 1984. godine. Upisao je master studije na smeru Računarstvo i automatika. Položio je sve ispite propisane planom i programom.

UPRAVLJANJE U SCADA SISTEMU POMOĆU DNP3 PROTOKOLA COMMANDING IN SCADA SYSTEM USING DNP3 PROTOCOL

Stefan Karać, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu predstavljeno je jedno rešenje programske podrške za upravljanje SCADA sistemom upotrebom DNP3 protokola. Pored opisanog koncepta rešenja, u radu su opisane osnove SCADA sistema, komunikacionih i industrijskih protokola kao i samog DNP3 protokola.

Abstract – This paper presents a software solution for commanding in SCADA system using DNP3 protocol. In addition to concept of the solution, there are described theoretical concepts of SCADA system, communications and industrial protocols and the DNP3 protocol.

Ključne reči: SCADA, RTU, DNP3

1. UVOD

Svakodnevni razvoj proizvodnih procesa u industriji zahteva pouzdaniji nadzor tih procesa. To podrazumeva da se akvizicija podataka i upravljanje sistemom odigrava u realnom vremenu. Glavni nosilac ove osobine jeste komunikacija između uređaja računarskog sistema, koji upravlja posmatranim industrijskim postrojenjem i on se naziva akviziciono-upravljački ili SCADA (*Supervisory Control And Data Acquisition*) sistem. Automatizacija proizvodnih procesa, uključujući akviziciju i upravljanje sistemom, u današnje vreme standardizovana je upotrebom računara a njena realizacija upotrebom adekvatne systemske i aplikativne programske podrške. Ukupna funkcionalnost industrijskog postrojenja određena je programskom implementacijom upravljačkog algoritma, nad SCADA modelom koji to postrojenje opisuje kao fizičku i logičku celinu. [1]

Osnovna ideja ovog rada je predlog rešenja za podršku DNP3 (*Distributed Network Protocol*) protokola u SCADA sistemu za komunikaciju sa RTU (*Remote Terminal Unit*) uređajima. Oslonac u ovom radu stavljen je na komandovanje nad posmatranim tačkama. Pored opisanog koncepta rešenja, u radu su opisane teorijske osnove SCADA sistema, komunikacionih i industrijskih protokola, DNP3 protokola kao i iskorišćene tehnologije prilikom implementacije rešenja. Takođe, opisan je način na koji su SCADA sistemi prvih generacija realizovani, kao i tehnologije koje se danas koriste s ciljem poboljšanja performansi SCADA sistema.

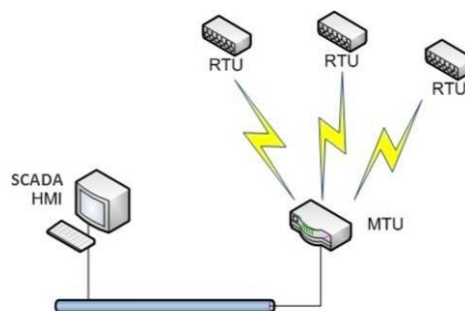
NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Pavlica, docent.

2. OSNOVE SCADA SISTEMA I INDUSTRIJSKIH PROTOKOLA

Osnovni cilj SCADA sistema je obezbeđenje efikasnog nadzora i upravljanja nad proizvoljnim fizičkim procesom, korišćenjem digitalnih računarskih komponenti. SCADA sistem obuhvata širok spektar opreme, podsistema i tehničkih rešenja koji omogućavaju prikupljanje i obradu podataka o realnim fizičkim procesima (industrijskim pre svega), i reagovanje na adekvatan način. [1]

Veliki broj merno-upravljačkih uređaja je tipična osobina većeg SCADA sistema, te je važna računarska mreža koja ih povezuje sa centralnom stanicom ili centralno postavljenim SCADA serverima. Takva mreža (Slika 2-1) se sastoji od jedne ili više nadzornih jedinica MTU (*Master Terminal Unit*) koje su zapravo računarske stanice, opremljene odgovarajućim softverom i koriste se za praćenje i kontrolu više RTU, PLC (*Programmable Logic Controller*) uređaja i senzora. RTU je takođe računarski uređaj koji je obično namenjen za rad u industrijskim uslovima, sa zadatkom da prikuplja informacije sa raznih digitalnih i analognih senzora i da prosleđuje komande uređajima koji na neki način menjaju stanje upravljanog sistema. PLC za razliku od RTU-a može da realizuje složenije upravljačke procese, preuzimajući na sebe značajan deo aplikativne programske podrške SCADA sistema. PLC omogućava automatsko vođenje procesa.



Slika 2-1 – Arhitektura SCADA sistema

Centralna stanica predstavlja mesto u SCADA sistemu gde se odvija nadzor i upravljanje posmatranog procesa u realnom vremenu. Obično je opremljena PC računarom ili nekim snažnim računarskim sistemom na kome se realizuje SCADA softver. Softver savremenih SCADA sistema, koji podrazumevaju telemetrijske komunikacione karakteristike, funkcionalnost komuniciranja sa RTU uređajima po određenom protokolu enkapsulira u posebnim softverskim komponentama. Te komponente se zovu *protocol driver*-i.

Unutar centralne stanice, svi podaci se čuvaju u bazi podataka odakle se vrši njihova prezentacija i generisanje upravljačkih akcija. Zahtev realnog vremena podrazumeva

pravovremenu reakciju na promene stanja kontrolisanog objekta, kao i adekvatno reagovanje na poremećaje u fizičkom procesu kojim se upravlja. Adekvatno reagovanje podrazumeva prihvatanje spoljnog događaja koji može izazvati i operater sistema, i generisanje upravljačke akcije u vremenskom periodu određenom dinamikom fizičkog procesa [1]. Pravovremenost reakcije postiže se dovoljno kratkim vremenskim periodom promene na ulazu i izlazu upravljačkog sistema. Pravovremenost je predefinisana osobina, pa može biti u opsegu od recimo pet milisekundi ali isto tako i pet sekundi, u zavisnosti od namene i složenosti sistema.

Funkcionalnost SCADA sistema zavisi od komunikacije između njegovih komponenti, putem koje se prenose podaci vezani za nadzor i upravljanje posmatranim fizičkim sistemom. Tok komunikacije kontrolisan je odgovarajućim komunikacionim protokolom koji podržava posmatrani sistem. Komunikacioni protokol podrazumeva skup pravila i procedura koji kontrolišu tok komunikacije između dve strane. Na taj način obezbeđuje se uspešna interakcija između udaljenih procesa. Industrijski protokoli predstavljaju klasu protokola specifičnih za SCADA sisteme. Namenjeni su za primenu u industrijskim postrojenjima i ne spadaju u grupu sa protokolima iz oblasti računarstva i telekomunikacija. Prema tipu učesnika u komunikaciji i njihovim ulogama u akviziciono upravljačkom sistemu, industrijski protokoli se mogu podeliti u sledeće kategorije: senzorski, *fieldbus*, telemetrijski i međusistemske protokoli. DNP3 spada u grupu telemetrijskih protokola, čija je primena u sistemima daljinskog nadzora i upravljanja gde je uticaj komunikacionih kanala odlučujući.

3. DNP3 Protokol

DNP3 (*Distributed Network Protocol*) predstavlja skup protokola koji se koriste za SCADA sisteme. Prvobitno je kreiran 1990. godine od strane *Westronic* kompanije (koju je danas preuzela kompanija *GE Harris*), da bi 1993. god. bila objavljena „DNP 3.0 Basic 4“ kompletna dokumentacija specifikacije protokola. Vlasništvo nad protokolom je predato novoformiranoj *DNP Users* grupi u oktobru iste godine. Od tada je protokol prihvaćen širom sveta uz formiranje korisničkih grupa u Kini, Južnoj Americi i Australiji. DNP3 je *event-oriented* protokol, što znači da *master* strana ne mora konstantno slati zahtev za trenutnu vrednost tačaka, kao Modbus recimo. Umesto toga, promene na tačkama prikuplja na dva načina: zahtevom za prikupljanje događaja (*event polling*) i *unsolicited* porukama [3].

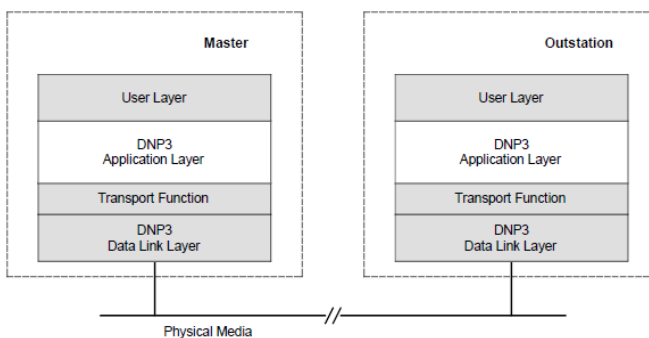
Osobine koje karakterišu DNP3 protokol su: slanje poruke na više uređaja (*broadcast*), vremensko označavanje podataka (*timestamp*), zauzeće entiteta pre upravljanja (*select before operate*), prenos datoteka, akvizicione grupe, itd.

3.1. Slojevitost DNP3 protokola

DNP3 je slojevit protokol. Međutim, umesto uklapanja u OSI arhitekturu od sedam slojeva, DNP3 podržava pojednostavljeni EPA model (Slika 3.1) sa dodatim četvrtim, pseudotransportnim slojem (*Transport Function*), koji omogućuje segmentaciju poruke.

Fizički sloj se prvenstveno odnosi na fizički medijum putem koga se vrši komunikacija između komponenti sistema. Ima informaciju o stanju medijuma, da li je slobodan ili zauzet, te vrši sinhronizaciju unutar medijuma. DNP obično za fizički sloj podrazumeva RS-232 ili RS-485 tehnologije

uz korišćenje fizičkih medijuma kao što su bakar, optičko vlakno, radio ili satelitski prenos. Noviji sistemi komuniciraju preko *Ethernet* veze.



Slika 3.1: Slojevitost arhitektura DNP3 protokola [2]

Sloj veze podataka upravlja logičkom vezom između pošiljaoca i primaoca. Za ovaj protokol važi da svaki frejm sloja veze podataka počinje zaglavljem i dodaje se kod za detekciju grešaka CRC (*Cyclic Redundancy Check*) od 16 bitova na svakih 16 bajtova frejma.

Pseudo-transportni sloj vrši segmentaciju poruka aplikacionog sloja u višestruke frejmove veze podataka. Za svaki frejm, vrši se umetanje funkcionalnog koda veličine jednog bajta koji pokazuje da li frejm veze podataka predstavlja prvi ili poslednji frejm poruke ili se poruka sastoji samo od jednog frejma.

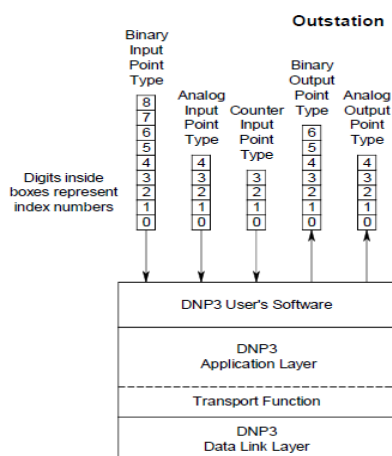
Aplikacioni sloj odgovara na kompletne poruke koje su primljene (nalazi se iznad transportnog sloja), i formira poruke na osnovu potrebe ili dostupnosti korisničkih podataka. Kada se poruka formira, ona silazi do pseudo-transportnog sloja gde se vrši segmentacija i njeno prosleđivanje do sloja veze podataka i eventualna komunikacija sa fizičkim slojem. Sve poruke aplikacionog sloja su međusobno nezavisne, s tim da jedino postoji indikacija koja je poruka poslednja i da nakon nje neće biti slanja novih poruka.

3.2 DNP3 objekti i tačke

Termin „tačka“ se odnosi na ulazne veličine kao što su analogne i diskretne vrednosti i brojači, i izlazne veličine koje podrazumevaju samo analogne i diskretne vrednosti. Jedan analogni ulaz je primer jedne tačke, povezan sa specifičnim merenim signalom ili izračunatom analognom vrednošću. Pored vrednosti, analogni ulaz može imati ime, faktor skale, *deadband* vrednost za detekciju događaja i ostale atribute [2].

Svi podaci na mernim tačkama se prezentuju pomoću DNP3 objekata. Važno je napomenuti da su DNP3 objekti samo način reprezentacije podataka na nekom entitetu, te ne predstavljaju niti fizički niti logički entitet. Objekti su grupisani u grupe. Za svaki tip merne tačke postoji jedna grupa, pa se može reći da je grupa, suštinski gledano, način prezentovanja tipa merne tačke [2].

Tačka je jedinstveno određena indeksom, a njegova vrednost ekvivalentna je rednim brojem elementa u nizu (Slika 3.2-1). Tačka obično ima statičku vrednost (*static*), koja se odnosi na trenutnu vrednost tačke. Ona je najčešće izmerena, izračunata ili dobijena vrednost. Statička vrednost za binarne ulaze odnosi se na trenutno, *on* ili *off*, stanje tačke. DNP3 dozvoljava zahtevanje vrednosti jedne ili više tačaka sa RTU stanice.



Slika 3.2-1 Tačke jedne stanice [2]

Događaj se odnosi na nešto što se desilo na tački, a to podrazumeva: promenu stanja, merenje čija vrednost prelazi određeni prag, ulazna vrednost analogne tačke je veća od *deadband*-a, trenutno stanje (*snapshot*) podataka u nekom trenutku i neke nove dostupne informacije. DNP3 koristi koncept klasa za organizaciju DNP3 objekata u nekoliko kategorija:

- Klasa 0 – statički podaci (mogu biti podskup ukupnih statičkih podataka jedne stanice)
- Klase 1, 2, 3 – klasifikacija po događajima

Tačka može biti dodeljena jednoj od četiri klase. Tačke su raspoređene po klasama tako što prioritet događaja Klase 1 je veći od prioriteta događaja Klase 2, kao što i prioritet događaja Klase 2 je veći od prioriteta događaja Klase 3.

3.3 Format poruke

Master formuliše i šalje poruke RTU stanici sa zahtevom za čitanje vrednosti, izvršavanje komande ili obavljanje neke posebne akcije. Poruke imaju definisanu strukturu po slojevima protokola. U ovom poglavlju biće opisana struktura fragmenta, poruke aplikacionog sloja.

Fragment je niz okteta koji sadrži zahtev ili odgovor na zahtev, koji se prenosi između *master* i *outstation* strane. Svaki fragment sadrži funkcionalni kod koji specificira kako će prijemna strana da procesira fragment, nula ili više zaglavlja objekata i *data-filled* objekata. Male poruke podrazumevaju svega par okteta, smeštenih u jedan fragment, dok duže poruke mogu biti raspoređene u više fragmenata. Fragmenti su nekad razbijeni u manje blokove, kada se prosleđuju nižim slojevima, a onda se ponovo sklapaju na prijemu. Svaki fragment (Slika 3.3-1) počinje aplikacionim zaglavljem (*Application Request Header*) koje sadrži informaciju o kontroli poruke.

← Start of fragment

Request fragment					
Application Request Header	First Object Header	DNP3 Objects	...	Last Object Header	DNP3 Objects

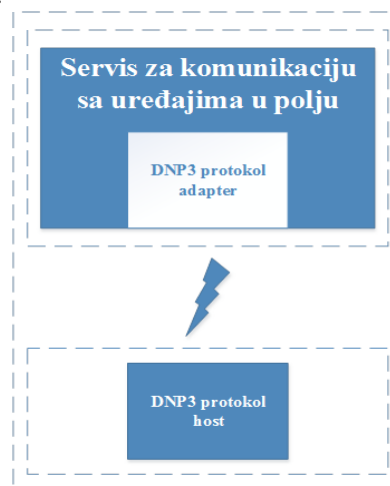
Slika 3.3-1 Sadržaj fragmenta [2]

Često, samo aplikaciono zaglavlje nije dovoljno za prenos kompletne informacije pa je praćeno jednim ili više DNP3 objekata i njihovim zaglavlja. Oni pružaju dodatnu informaciju potrebnu za formiranje kompletne poruke.

Ukratko, zaglavlje objekta definiše tip, format i identifikaciju DNP3 objekata koji slede. DNP3 poruka može da sadrži jedan ili više fragmenata. Svaki fragment počinje aplikacionim zaglavljem. Transportne funkcije dele fragmente na manje delove, segmente, od kojih svaki počinje zaglavljem transportnog sloja. Sloj veze svaki segment proširuje svojim zaglavljem.

4. PREDLOG REŠENJA

DNP3 protokol adapter je programsko rešenje opisano u ovom radu i predstavlja *driver* komponentu SCADA softvera, i kako sam naziv kaže namenjen je za upravljanje RTU uređajima koji komuniciraju pomoću DNP3 protokola.



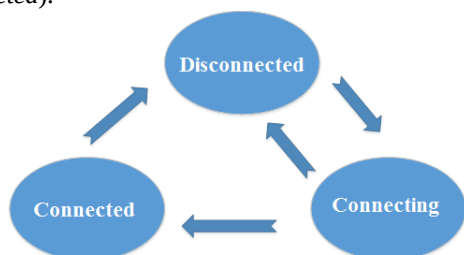
Slika 4-1 Arhitektura rešenja

Programsko rešenje podrazumeva dva modula koji su oslonac komandne linije SCADA sistema (Slika 4-1). Ti moduli su servis za komunikaciju sa RTU uređajima, u okviru koga se nalazi DNP3 protokol adapter, i DNP3 protokol host.

Servis za komunikaciju zamišljen je kao servis u okviru SCADA softvera, podignut na posebnom hostu, što bi sistemu dalo osobinu distribuiranosti. Funkcionalnost servisa za komunikaciju sa uređajima u polju realizovana je u nizu softverskih komponenti namenjenih za učitavanje konfiguracionih podataka, učitavanje dinamičkih biblioteka protokola adaptera sa predefinisane memorijske lokacije, te komponenti namenjenih za prosleđivanje komandi, odnosno konfiguracionog modela, na adekvatan protokol adapter. DNP3 protokol adapter, kao softverska komponenta, zamišljen je da služi kao komunikaciona sprega između servisa za komunikaciju i protokola hosta, a svu funkcionalnost upravljanja RTU uređajima pomoću DNP3 protokola, izvršavao bi DNP3 protokol host.

Funkcionalnost DNP3 protokol hosta podrazumeva generisanje modela sistema na osnovu konfiguracionih podataka i slanje komandi na RTU uređaj. Komandna linija DNP3 protokol hosta, pored servisa za komandovanje, podrazumeva niz komponenti zaduženih za propagiranje komande na odgovarajući RTU uređaj. Komandni dispečer je komponenta ovog modula, zadužena za propagiranje komandi na odgovarajućeg izvršitelja, a koje se odnose na RTU uređaj, odnosno DNP3 komunikacioni kanal. Zamišljeno je da ovaj modul sadrži dva interna modela podataka, tako da jedan predstavlja konfiguracioni model a drugi predstavlja dinamički model stanja.

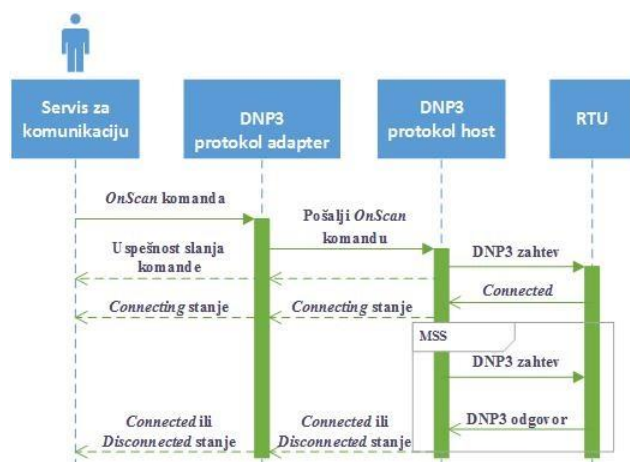
Konfiguracioni model podrazumeva kolekciju modela DNP3 komunikacionih kanala i RTU uređaja zajedno sa svojim tačkama. Za generisanje i rukovanje dinamičkim modelom zadužena je mašina stanja. Mašina stanja (Slika 4-2) predstavlja konačni automat koji definiše prelaze između stanja RTU uređaja (*Disconnected*, *Connecting* i *Connected*).



Slika 4-2: Dijagram stanja RTU uređaja

Connecting stanje podrazumeva proces inicijalnog podešavanja uređaja, zvanog *Master Startup Sequence* (Slika 4-3), neophodnog pre upravljanja RTU uređajem i obuhvata redom komande:

1. Onemogućavanje *unsolicited* poruka za tačke koje pripadaju DNP3 klasama 1, 2 i 3
2. Prikupljanje vrednosti svih njegovih tačaka – *Integrity update*
3. Podešavanje *deadband* vrednosti svih tačaka koje to zahtevaju
4. Ponovno omogućavanje *unsolicited* poruka



Slika 4-3: Master Startup sekvenca

Uspješno izvršavanje svih komandi rezultuje prelaskom uređaja u stanje *Connected*, u suprotnom uređaj prelazi u *Disconnected* stanje.

5. IMPLEMENTACIJA REŠENJA

Programsko rešenje implementirano je u programskom jeziku C#, korišćenjem .NET 4.6.2 radnog okruženja u Microsoft Visual Studio 2015 razvojnom okruženju.

Servis za komunikaciju sa uređajima u polju realizovan je kao konzolna aplikacija koja, kao korisnička programska sprega, korisniku pruža mogućnost zadavanja komandi, učitavanja i postavljanja konfiguracionog modela. DNP3 protokol host realizovan je, kao konzolna aplikacija koja vrši interakciju sa RTU uređajima. Za potrebe testiranja, iskorišten je *opendnp3* simulator.

Komunikacija između ova dva modula realizovana je WCF (*Windows Communication Foundation*) komunikacijom. Zbog postizanja boljih performansi, kao i robusnosti sistema, komandna i konfiguraciona linija odvijaju se zasebnim WCF komunikacionim kanalima. Za upisivanje i uvid u trenutno stanje sistema, koje podrazumeva stanja konekcija uređaja i trenutne vrednosti tačaka, iskorištena je *Redis*, *in-memory* baza podataka.

Opendnp3 je skalabilna, testirana implementacija DNP3 protokol komunikacije, pisana u C++ programskom jeziku. Biblioteka je dizajnirana za aplikacije velikih performansi, koje podrazumevaju istovremeni veliki broj sesija ili simulacija uređaja. [3]

Redis je *in-memory* baza podataka, mrežno implementirana, koja vrši zapisivanje u *key-value* formatu. *Redis* podržava različite vrste apstraktnih struktura podataka kao što su stringovi, mape, liste, kolekcije itd. [4] Za pristup *Redis* bazi potrebna je URL adresa na kojoj je *Redis* server pokrenut.

6. ZAKLJUČAK

U ovom radu predstavljeno je programsko rešenje za upravljanje SCADA sistemom pomoću DNP3 protokola. Cilj je bio realizovati generično i modularno rešenje, za slučaj komuniciranja sa RTU uređajima pomoću nekog drugog protokola, odnosno postizanja osobine distribuiranosti i robusnosti sistema. Trenutna verzija rešenja podrazumeva postavljanje konfiguracije sistema i slanje komandi na DNP3 RTU uređaj. Dalji pravci razvoja podrazumevali bi uvođenje akvizicione linije, komunikaciju sa uređajima pomoću nekog drugog protokola, kao i izmene na postojećoj arhitekturi s ciljem poboljšanja performansi.

7. LITERATURA

- [1] Branislav Atlagić, Fakultet tehničkih nauka, Novi Sad, *Softver sa kritičnim odzivom*, 2015
- [2] DNP3 Specification Volume 2, Version 2.11, 15 March 2009
- [3] <https://github.com/automatak/dnp3>
- [4] <https://en.wikipedia.org/wiki/Redis>

Kratka biografija:

Stefan Karać rođen je 2.3.1993. godine u Ilidži. Osnovnu i srednju tehničku školu završio je u Bijeljini, a 2012. godine upisao je osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu. Diplomski rad iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke i informatika odbranio je 2016. godine. Iste godine upisao je master studije Primenjenog softverskog inženjersva na Fakultetu tehničkih nauka. Položio je sve ispite predviđene planom i programom.

RAZVOJ ALATA ZA VERIFIKACIJU SCADA KONFIGURACIJE**DEVELOPMENT OF A TOOL FOR VERIFICATION OF SCADA CONFIGURATION**

Nemanja Popović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu predstavljen je alat za proveru konfiguracije SCADA sistema. Ukratko su opisani SCADA sistemi i IEC 60870-5-104 komunikacioni protokol. Za potrebe simulacije realnog sistema razvijen je RTU Simulator koji komunicira sa alatom preko IEC-60870-5-104 protokola.

Abstract – This paper presents a tool for configuration verification of the SCADA system. There is a brief description of SCADA systems and the IEC 60870-5-104 communication protocol. For the need of real system simulation a RTU Simulator has been developed that communicates with tool using the IEC-60870-5-104 protocol.

Ključne reči: SCADA, IEC-60870-5-104, Konfiguracija, Verifikacija

1. UVOD

Konfiguracija SCADA sistema podložna je greškama, naročito kod velikih sistema sa velikim brojem ulaznih i izlaznih tačaka. Popunjavanje tabela sa signalima vrši se uglavnom ručno. Zbog toga je važno imati alat za verifikaciju unete konfiguracije.

Zadatak alata je da komunikacijom sa uređajima u polju, koristeći IEC 60870-5-104 protokol, izvrši proveru da li su se svi uređaji koji su definisani u SCADA konfiguraciji „javili“ i da li se tipovi podataka i adrese uređaja poklapaju.

2. SCADA SISTEMI

SCADA (*Supervisory Control and Data Acquisition*) obuhvata prikupljanje podataka, prenošenje informacija na centralni sistem, izvršavanje potrebnih analiza i kontrole, kao i prikaz rezultata na operatorskim ekranima.

SCADA sistem se sastoji od udaljenih telemetrijskih stanica (*RTU - Remote Terminal Unit*) koje sakupljaju podatke sa uređaja u polju i šalju ih master stanici preko nekog komunikacionog sistema. Master strana je u mogućnosti da prikaže podatke i omogućava operaterima da zadaju komande sistemu [1].

3. KOMUNIKACIONI PROTOKOLI

Industrijski protokoli i industrijske mreže (*industrial protocols and industrial networks*) je uobičajeno ime za klasu protokola specifičnih za SCADA sisteme, ono naglašava njihovu primenu u industrijskim postrojenjima

i koje ih odvaja od protokola prisutnih u drugim oblastima računarstva i telekomunikacija.

U elektroenergetskim SCADA aplikacijama, u Evropi su dominantni protokoli IEC 60870-5-101/104, a u Americi protokol DNP3.

Oba protokola imaju zajedničko poreklo, i razvijeni su baš za primenu u sistemima za prenos i distribuciju električne energije. Inicijalni standardi 101 i DNP3 razvijeni za modemske veze i analogni prenosni sistem, kasnije su prilagođeni za prenos preko TCP/IP transportnog sloja (standardi 104 i DNP3/TCP).

4. IEC 60870-5-104 PROTOKOL

IEC 60870-5 se odnosi na kolekciju standarda izrađenih od strane IEC (*International Electrotechnical Commission*), da bi pružili otvoren standard za prenos SCADA telemetrije i informacionih podataka [2]. Standard obezbeđuje detaljne funkcionalne opise opreme za daljinsko upravljanje i sistema za upravljanje geografski rasprostranjenim procesima, drugim rečima SCADA sistema.

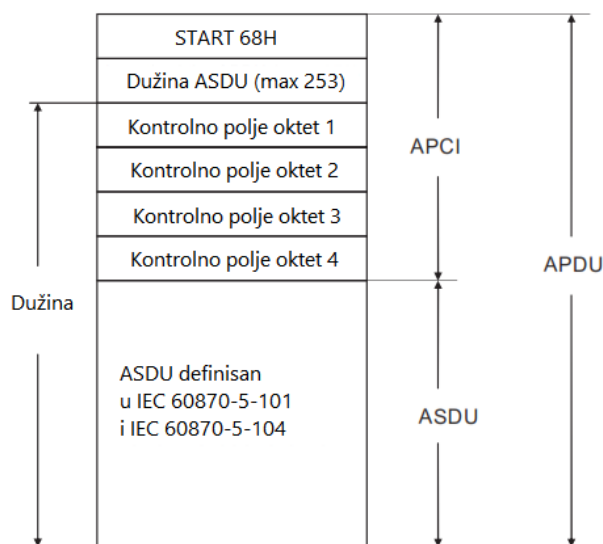
Standard je predviđen za aplikacije u elektro industriji i sadrži podatke koji su posebno namenjeni za takve aplikacije, međutim nije ograničen na te aplikacije i može se primeniti na SCADA aplikacije u bilo kojoj industriji. IEC 60870-5 se primarno koristi u elektro industriji evropskih zemalja.

Četvrti pridruženi standard, IEC 60870-5-104 [3], definiše transport aplikativnih poruka preko mreže. Njegov pun naziv je „*Network Access using Standard Transport Profiles*“ što se odnosi na upotrebu TCP/IP protokola na transportnom i mrežnom nivou. Ovaj standard je objavljen u decembru 2000, šest godina posle objavljivanja IEC 60870-5-101. Pruža dosta različite mehanizme fizičkog sloja i prenosa podataka ali ostavlja većinu funkcija viših nivoa neizmenjene.

APDU (*Application Protocol Data Unit*) predstavlja osnovnu informacionu jedinicu prenosa (Slika 4.1 Format APDU poruke). Može se sastojati iz dva dela, prvi deo je obavezan APCI (*Application Protocol Control Information*), zaglavlje sa kontrolnim informacijama, drugi deo je ASDU (*Application Service Data Unit*), servisne informacije koje sa sobom nose aplikativne podatke, odnosno rezultate merenja ili nadgledanja.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branislav Atlagić, docent.



Slika 4.1 Format APDU poruke

Aplikativni podaci se grupišu po smeru i po tipu informacije koju nose (Slika 4.2 Grupe tipova informacionih objekata).

Group	Example in Monitor Direction	Example in Control Direction
Process information	A measured value, e.g. a bit, or an analog	A command, e.g. to set a bit, or a setpoint value
System information	End of initiation flag	Interrogation command, reset process command
Parameter		Set a filter time
File transfer	Read data file, write a configuration file.	

Slika 4.2 Grupe tipova informacionih objekata

Poruka koja se razmenjuje preko IEC 60870-5-104 protokola definisana je kao ASDU (*Application Service Data Unit*). ASDU se deli na *Data Unit Identifier* i jedan ili više informacionih objekata (Slika 4.3 Format ASDU-a). Informacioni objekti su uvek iste strukture i tipa koji su definisani u *Type Identification* polju *Data Unit Identifier*-a.

Data Unit Identifier čine:

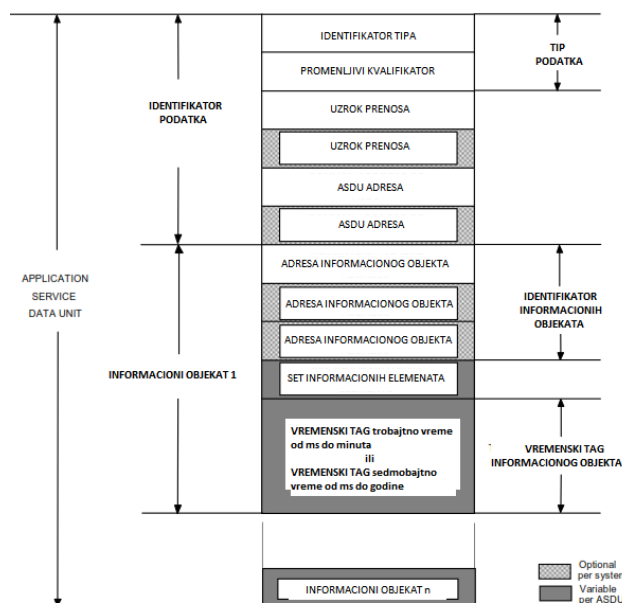
- 1 bajt TYPE IDENTIFICATION
- 1 bajt VARIABLE STRUCTURE QUALIFIER
- 1 ili 2 bajta CAUSE OF TRANSMISSION
- 1 ili 2 bajta COMMON ADDRESS OF ASDU

Type Identification poljem je prikazan tip podataka koji se šalje. Ono definiše strukturu, tip i format informacionog objekta. Svi informacioni objekti jednog ASDU-a su iste strukture, tipa i formata.

Variable Structure Qualifier (SQ) specificira broj informacionih objekata ili informacionih elemenata i kako su adresirani.

Cause Of Transmission (COT) polje se koristi za rutiranje poruka na komunikacionoj mreži i unutar stanice

upućivanjem poruke na odgovarajući program ili task za obrađivanje.



Slika 4.3 Format ASDU-a

Common Address of ASDU može biti veličine jedan ili dva bajta, obično se interpretira kao adresa stanice ali može biti strukturirana tako da formira adresu sektora/stanice gde su individualne stanice podeljene u više logičkih jedinica. Adresa 0 se ne koristi, adrese od 1 do 254 ili od 1 do 65534 su za stanice a 255 ili 65535 su globalne adrese, što znači da će ASDU poruka sa tom adresom biti obrađena od strane svih stanica.

Upotreba globalne adrese je ograničena i koriste je samo sledeće ASDU poruke:

- <100> *Interrogation command*
- <101> *Counter interrogation command*
- <103> *Clock synchronization command*
- <105> *Reset process command*

Informacioni objekti se sastoje iz *Information Object Identifier*-a, seta informacionih elemenata i ako postoji vremenskog tag-a.

Information Object Identifier – sadrži samo adresu informacionog objekta (IOA). IOA može biti veličine jedanog dva ili tri bajta, međutim slučaj od tri bajta je pružen samo da obezbedi strukturiran adresni sistem, jedna stanica može da ima najviše 65536 različitih IOA. IOA nula je rezervisana za objekat čija adresa nije bitna. IOA zajedno sa *Common Address Of ASDU* jednoznačno određuje informacione elemente u sistemu.

Informacioni elementi predstavljaju osnovnu komponentu prenosa informacija putem protokola.

5. ALAT ZA VERIFIKACIJU SCADA KONFIGURACIJE

SCADA konfiguracija predstavlja informacije o uređajima u polju, broju procesnih kontrolera, njihovim ulazima, izlazima, adresama, tipovima podataka kojima je

svaki signal predstavljen. Da bi sistem nesmetano funkcionisao, neophodno je da se ručno uneti konfiguracioni podaci u potpunosti slažu za situacijom na polju.

Provera i verifikacija SCADA konfiguracije je zadatak programskog alata SCVT (SCADA Configuration Verification Tool), razvijenog u toku izrade master rada. Alat je razvijen upotrebom *Triangle MicroWorks(TMW)* biblioteke za implementaciju IEC 60870-5-104 protokola na Microsoft .NET framework-u.

TMW biblioteku čini hijerarhija struktura koje su definisane kao kanali, sesije, sektori i tačke. Kontrolna (master) stanica može da komunicira preko više komunikacionih kanala sa više kontrolisanih (slave) uređaja. Svaki kontrolisani uređaj može da podeli svoje podatke u sektore, od kojih svaki sadrži određene tačke. Kanal predstavlja enkapsulaciju fizičkog sloja, sloja veze i transportnog sloja. Svaki kanal može da ima jednu ili više sesija. Sesija predstavlja komunikacionu vezu između master uređaja i jednog slave uređaja (remote). Ukupan broj sesija predstavlja ukupan broj uređaja na svim komunikacionim kanalima sa kojima master komunicira. Kada više uređaja koristi jedan komunikacioni kanal link adrese sesije moraju da se razlikuju za svaku sesiju koja se nalazi na tom kanalu. Sektor predstavlja kolekciju podataka u IEC 60870-5-104 protokolu. Primer nekog slave uređaja sa više sektora bi bio koncentrator podataka koji sektore koristi da prikaže niže uređaje.

Kao primer SCADA konfiguracije korišćena je OASyS SCADA konfiguracija. Konfiguracioni podaci su predstavljeni u obliku tekstualnih fajlova sa .l ekstenzijom. RTU-ovi, konekcije i svaki tip signala (*analog, status, multistate* i *rate*) su opisani u zasebnom .l fajlu. SCVT parsiranjem konfiguracionih fajlova stvara model realnog sistema.

Da bi se proverila ispravnost učitane konfiguracije SCVT komunicira sa uređajima u polju preko IEC 60870-5-104 protokola. Uspostavom komunikacije šalje se *General Interrogation* zahtev, usled čega će svi RTU-ovi poslati svoje trenutno stanje.

General interrogation počinje slanjem ASDU poruke tipa 100 od strane master stanice tj. SCVT-a. COT sadrži kod <6> što predstavlja aktivaciju (ACT). Pri prijemu poruke slave stanica šalje potvrdu aktivacije, ova poruka ima isti tip 100 ASDU ali je COT <7>. Slave tada prikuplja tražene podatke. Poruke sa traženim tipovima podataka biće slane sa COT <20> što znači da se šalju usled *genneral interrogation* zahteva. Kada su svi podaci poslani slave strana mora poslati terminaciju aktivacije ACT TERM ASDU. Ovo je tip 100 ASDU sa COT <10>.

Tipovi predstavljeni u *Type identification* polju u smeru nadgledanja koja su podržana alatom su:

- *Single-point information with time tag (M_SP_TB_1)*
- *Double-point information with time tag (M_DP_TB_1)*
- *Measured value, normalized value with time tag (M_ME_TD_1)*

- *Measured value, short floating point number with time tag (M_ME_TF_1)*

Single-point information predstavlja jednobitnu binarnu vrednost. Informacioni element čine 8 bita i predstavljen je simbolom SIQ. Nulti bit predstavlja vrednost a 4 najviša bita čine *quality information*.

Double-point information ima iste bitove kvaliteta kao i *single-point* razlika je u tome što vrednost čine najniža dva bita.

Measured value, short floating point number je četvorobajtni broj definisan IEEE 754 standardom. Čine ga frakcija, eksponent i bit za znak.

U SCADA konfiguraciji definisani su različiti tipovi tačaka:

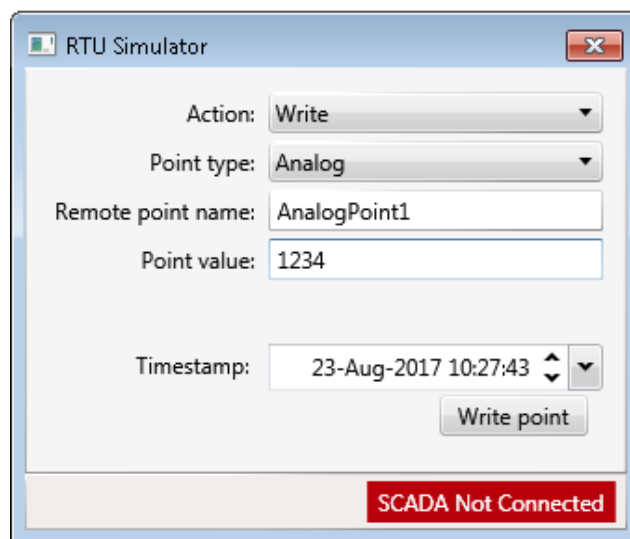
- Analogne tačke:
 - *Float* – mapira se na M_ME_TF_1
 - *Int16* – mapira se na M_ME_TD_1
- Statusne tačke:
 - Jednobitna – mapira se na M_SP_TB_1
 - Dvobitna – mapira se na M_DP_TB_1

Na osnovu adrese primljenog informacionog objekta se proverava da li je tačka definisana u SCADA konfiguraciji tj. da li se nalazi u modelu, a potom se proverava da li se tipovi poklapaju.

Alat je razvijen u *Microsoft Visal Studio 2013* razvojnom okruženju.

6.TESTIRANJE

Za proveru rada alata razvijen je RTU simulator (Slika 6.1 RTU Simulator) koji zamenjuje fizičke procesne kontrolere.



Slika 6.1 RTU Simulator

Simulator ima sopstvenu konfiguraciju RTU-ova i njihovih signala. Tipovi tačaka koje podržava su analogne i statusne. Simulator takođe ima mogućnost ručnog postavljanja *timestamp*-a.

Podatke RTU šalje pri promeni neke vrednosti na uređajima. SCVT inicira uspostavu komunikacije

Klikom na dugme „Write point“, menja se stanje signala i promena se šalje na master stranu.

Problemi detektovani u realnim sistemima koji su obrađeni u ovom radu:

- Nije stigla promena za RTU koji je konfigurisan na SCADA-i nakon GI zahteva.
- Stigla je promena za tačku koja nije konfigurisana na SCADA-i. Promena je došla ili spontano ili nakon GI zahteva.
- Nepodudaranje tipova tačaka podešenih na SCADA-i i podešenih na RTU strani.

SCVT je odrađen kao konzolna aplikacija. Na slici (Slika 6.2 SCVT) se vidi razmena poruka između alata i RTU Simulatora. Zbog bolje preglednosti, detektovane greške se zapisuju u log fajl (Slika 6.3 SCVT Log).

```

file:///C:/Users/Admin/documents/visual studio 2013/Projects/ConfigurationVerificatorSln/ScadaSim...
>>>Protocol: ### conn_Ironwood - 127.0.0.1:30002 - TCP transmit 22 bytes
>>>Protocol:
>>>Protocol: ... conn_Ironwood 68 14 00 00 02 00 67 01 06 04 b6 15 00 00 00 7e
>>>Protocol:
>>>Protocol: 9e 2d 92 dd 07 11
>>>Protocol:
>>>Protocol: <+++ Ironwood Insert request in queue: General Interrogation caused by End of Initialization
>>>Protocol:
>>>Protocol: <=== Ironwood Addr(0) Sector(5558) Application Header, Interrogation Command
>>>Protocol:
>>>Protocol: Quantity(1) SQ(0) COT(6, activation)
>>>Protocol: 64 01 06 04 b6 15 00 00 00 14
>>>Protocol:
>>>Protocol: <--- conn_Ironwood Information
>>>Protocol: N(S) 1 N(R) 1
>>>Protocol: 68 0e 02 00 02 00
>>>Protocol: 64 01 06 04 b6 15 00 00 00 14
>>>Protocol:
>>>Protocol: ### conn_Ironwood - 127.0.0.1:30002 - TCP transmit 16 bytes
>>>Protocol:
>>>Protocol: ... conn_Ironwood 68 0e 02 00 02 00 64 01 06 04 b6 15 00 00 00 14
>>>Protocol:
>>>Protocol: <+++ Ironwood Insert request in queue: Counter Interrogation caused by End of Initialization
>>>Protocol:
>>>Protocol: <=== Ironwood Addr(0) Sector(5558) Application Header, Counter Interrogation command
>>>Protocol:
>>>Protocol: Quantity(1) SQ(0) COT(6, activation)
>>>Protocol: 65 01 06 04 b6 15 00 00 00 05
>>>Protocol:
>>>Protocol: <--- conn_Ironwood Information
>>>Protocol: N(S) 2 N(R) 1
>>>Protocol: 68 0e 04 00 02 00
>>>Protocol: 65 01 06 04 b6 15 00 00 00 05
>>>Protocol:
>>>Protocol: ### conn_Ironwood - 127.0.0.1:30002 - TCP transmit 16 bytes
>>>Protocol:
>>>Protocol: ... conn_Ironwood 68 0e 04 00 02 00 65 01 06 04 b6 15 00 00 00 05
>>>Protocol:
RockyPT status point IOA mismatch, status point with IOA:7, not found in SCADA configuration
RockyPT analog point type mismatch on point named: RP144396834857806224 IOA:4, Expected short float, point type on SCADA is: IEC_16_INI

```

Slika 6.2 SCVT

Za proveru prvog slučaja na SCADA-i je konfigurisan jedan RTU kojeg nema u konfiguraciji RTU Simulatora. SCVT nakon poslatog GI zahteva očekuje prijem trenutnog stanja na RTU-u ali usled nepostojanja tog RTU na simulatoru desiće se *timeout* i greška će biti detektovana.

Za proveru drugog slučaja na RTU simulatoru je konfigurisana jedna analogna tačka koja ne postoji u SCADA konfiguraciji.

Za proveru trećeg slučaja u SCADA konfiguraciji je definisana tačka sa tipom *int16*, a na RTU je definisana sa tipom *float*.

```

SCVTLog.txt - Notepad
File Edit Format View Help
RockyPT Status point:TI_RockyPT_001 IOA:1 THERE
RockyPT Status point:RockyPoint1233 IOA:3 THERE
RockyPT Status point:DISCP3 IOA:5 THERE
RockyPT Status point IOA:7 NOT FOUND
RockyPT Analog point FLOAT: RP144396834857806223 IOA:2 THERE
RockyPT Analog point:RP144396834857806224 IOA:4 TYPE
MISSMATCH
RockyPT Analog point IOA:33 NOT FOUND
RTU with name: RockyPT OK
RTU with name: Shands_Central FAIL

```

Slika 6.3 SCVT Log

7. ZAKLJUČAK

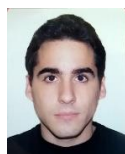
U ovom radu predstavljeni su alat za verifikaciju SCADA konfiguracije, primena RTU Simulatora kao zamene za realne procesne kontrolere i koncept IEC 60870-5-104 protokola namenjenog komunikaciji između alata i simulatora.

Dalje unapređenje alata bi bilo fokusirano na bolji vizuelni prikaz, razvoj korisničkog interfejsa koji bi omogućio i detektovanje grešaka komandovanjem uređajima.

8. LITERATURA

- [1] Branislav Atlagić: *Softver sa kritičnim odzivom u elektroenergetskim sistemima*, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2014
- [2] Gordon Clarke, Deon Reynders: *Practical modern SCADA protocols: DNP3, 60870-5 and Related Systems*, Elsevier, 2004
- [3] IEC 60870-5-104:2006, *Telecontrol equipment and systems- Part 5-104: Transmission protocols - Network access for IEC 60870-5-101 using standard transport profiles*

Kratka biografija:



Nemanja Popović rođen je u Novom Sadu 1989. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2015. godine. Master rad odbranio je 2017. godine na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjeno softversko inženjerstvo

РЕАЛИЗАЦИЈА КОМУНИКАЦИОНОГ ПРОТОКОЛА ICCP У SCADA ОКРУЖЕЊУ**IMPLEMENTATION OF COMMUNICATION ICCP PROTOCOL IN SCADA ENVIRONMENT**Јелена Карага, *Факултет техничких наука, Нови Сад***Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО**

Кратак садржај – У овом раду је описана реализација симулатора индустријског постројења, са комуникацијом преко **ICCP** протокола, развијеног ради тестирања **SCADA** система. Симулатор врши симулацију динамичких промена мерних величина, ишчитавање њихових вредности, али пре свега успоставу конекције између **SCADA** система и самог **ICCP** симулатора.

Abstract – This paper describes development of industrial plant simulator using **ICCP** protocol for purpose of testing of **SCADA** system. Simulator allows simulated dynamic changes of measured values, reading of their values, but first of all, establishing a connection between **SCADA** and **ICCP** simulator.

Кључне речи: **SCADA**, **ICCP**, симулатор

Keywords: **SCADA**, **ICCP**, simulator

1. УВОД

Врло значајан аспект свих аквизиционо управљачких система намењених управљању индустријским постројењима у реалном времену јесте међусобна комуникација њихових компоненти [1].

Управљање уз помоћ рачунара је стандардна особина савремених индустријских постројења. Аутоматизација производних функција постројења реализује се програмски, уз помоћ системске и апликативне програмске подршке.

Основни проблем при развоју апликативне програмске подршке аквизиционо управљачког система јесте немогућност њеног испитивања на реалном постројењу. Стога је апсолутно неопходно обезбедити што реалније симулационо окружење, које се извршава над програмским моделом физичког постројења. У току развоја софтвера треба извршити све потребне провере, чиме се предупређују или минимизују проблеми при инсталацији, односно при пуштању **SCADA** система у рад. Сам развој системског дела **SCADA** софтвера захтева адекватно симулационо окружење које на погодан начин омогућује контролисано креирање и репродукцију различитих тестних сценарија. Теоријски, овај део може се радити и помоћу реалне опреме, али је то у пракси отежано из низа разлога који нису само финансијски.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Бранислав Атлагић, доцент.

Због тога је расположивост различитих **SCADA** симулационих окружења кључна како за развој **SCADA** софтвера, системског и апликативног, тако и **SCADA** система као целине. Њихова функционалност и квалитет директно одређују свих активности везаних за развој **SCADA** система [1].

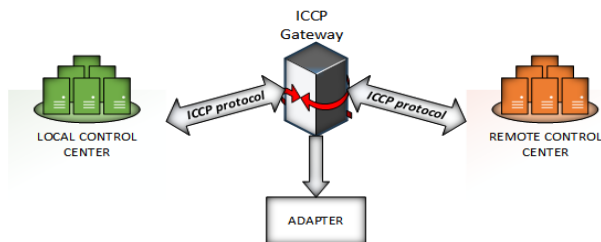
Суштински задатак **SCADA** симулатора (симулационог окружења) је динамичка симулација окружења **SCADA** система, тачније процесног окружења које одређује понашање **SCADA** станице **NUS** (Надзорно Управљачке Станице). Подаци који се преносе преко **RTU** комуникационе спреге обухватају промене које настају у постројењу, и управљачке команде **SCADA** система. Све промене које **SCADA** систем прима посредством **RTU** уређаја и комуникационог протокола, дефинишу понашање и динамику индустријског постројења. За развој апликативног кода потребна је што потпунија и детаљнија симулација постројења, што је лакше урадити на нивоу **SCADA** модела података и радне конфигурације [1].

Идеја овог решења је да се развије симулатор **ICCP** протокола, односно његов **core** део, тако што ће методе из интерфејса другог симулационог окружења бити имплементирани, и мапиране на методе **ICCP** протокола. Као језгро симулатора **ICCP** протокола, предвиђено је да сам тај део буде представљен као **plugin** који ће бити увучен у симулациони процес. У њему су пружене услуге за омогућавање симулирања реалних физичких процеса, и подржавање следеће функционалности: иницијализацију, и покретање комуникације са **SCADA** системом у складу са **ICCP** протоколом, подршку за руковање командама добијених од стране **SCADA** система и подршка за слање промена вредности тачака на удаљене уређаје. Промену вредности тачака може да иницира **SCADA** систем (преко одговарајуће команде) или сам симулатор, у облику спонтане промене. **Plugin**/симулатор протокола као симулационо окружење, користи динамичку **.NET** библиотеку која пружа функционалности **ICCP** протокола, као и **MEF** (*Managed Extensibility Framework*). **MEF** се састоји из **import** и **export** дела, где је сам **plugin export** део, а **plugin** или његове делове импортујемо у одговарајућу класу другог симулационог окружења, односно у овом случају, то је симулатор процеса.

Промену вредности тачака може да иницира **SCADA** систем (преко одговарајуће команде) или сам симулатор, у облику спонтане промене. **Plugin**/симулатор протокола као симулационо окружење, користи динамичку **.NET** библиотеку која пружа функционалности **ICCP** протокола.

2. ICCP ПРОТОКОЛ

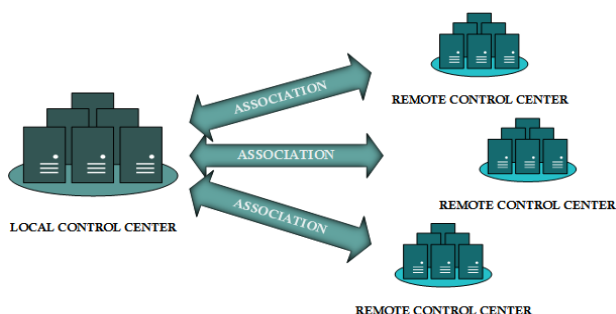
ICCP (*Inter-Control Center Communications Protocol*) је протокол који се може наћи и под другим називима као што су **IEC 60870-6** и **TASE.2**. Један је од интернационалних стандарда - *International Electrotechnical Commission (IEC) Telecontrol Application Service Element 2 (TASE.2)*. **ICCP** спада у међусистемски протокол, који ради на повезивању **SCADA** система – контролних центара. [1] Спрега између аутономних **SCADA** система, стандардизована је углавном кроз **ICCP**. **ICCP** стандард, је једини релевантан стандард, јер је независан од програмске платформе и једини нуди адекватне перформансе [1]. Размена података се може обављати између више контролних центара. **ICCP** везе између **SCADA** система су приказани на слици 2.1.



Слика 2.1 **ICCP** везе између **SCADA** система

2.1 Клијент – Сервер модел

ICCP је базиран на архитектури **Client- Server**. Уколико имамо два контролна центра која комуницирају, у сваком моменту, и један и други центар могу се понашати као клијент и сервер. Сав пренос података потиче од захтева једног контролног центра који је у улози клијента, другом контролном центру који је у улози сервера, који поседује и управља подацима. На пример, уколико апликација у **LCC**-у (*Local Control Center*) треба податке из **SCADA** базе, који представља **RCC** (*Remote Control Center*), **LCC** ће деловати као клијент и послаће захтев **RCC**-у, и тада ће **RCC** понашати се као сервер, послаће податке датом клијенту.



Слика 2.2 Асоцијација између **LCC**-а и **RCC**-а

Интерфејс за конекцију **ICCP**-а, може користити било које транспортне и мрежне сервисе које одговарају **OSI** моделу, али се најчешће користи **TCP/IP**. Користе се логичке конекције познате као асоцијације између контролних центара. Клијенти могу да оснивају асоцијације са више сервера истовремено, а такође могу да оснивају асоцијацију са истим сервером, што је приказано на слици 2.2 [2].

2.2 ICCP сервер објекти

У **ICCP** сервер објекте спадају следећи елементи:

- **Association** – асоцијација или логичка веза
- **DataValue** - податак
- **DataSet** – скуп података
- **TransferSet** – објекат који се користи за контролу размене података
- **Device** – уређај

Association – користи се за оснивање асоцијације или логичке везе између две **ICCP** инстанце. **DataValue** – репрезентује вредности елемената података из контролног центра, укључујући **SCADA** тачке или структуре података. Садржи четири операције које се користе искључиво од стране клијента и то су:

- 1) **Get Data Value** – ова операција се може користити приликом захтева за вредност једне **SCADA** тачке
- 2) **Set Data Value** – ова операција за циљ има да омогући да се вредност податка запише или сетује на **LCC** преко **RCC**-а.
- 3) **Get Data Value Names** – ова операција омогућава клијенту да добије списак са именима од свих **Data value** објеката од **RCC**-а. [2]

DataSet - представља листу објеката која је одржавана од стране сервера, логична група тачака. Омогућава да клијент удаљено дефинише **Data Sets** преко **ICCP** везе [2].

2.3 ICCP типови података

ICCP садржи три основа податка:

- **real** – реални тип података
- **descrete** – целобројни тип података
- **state** – тип који означава стање

Сваки тип садржи следеће атрибуте:

- назив променљиве – јединствени назив који тачно одређује објекат
- тип податка – приказује коме припада од основних типова података, и наспрам тога ће садржати различите атрибуте – реални, статусни или целобројни

2.4 ICCP блокови

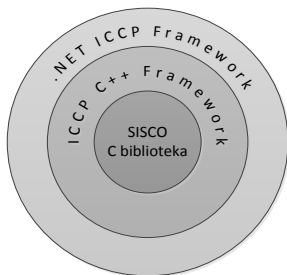
ICCP блокови представљају специфичну функционалност или скуп функционалности. Састоји се из девет блокова:

- Блок 1 – Периодична размена података
- Блок 2 – Проширена подешавања за праћење скупа података
- Блок 3 - Компресија података за пренос
- Блок 4 – Пренос порука
- Блок 5 – Управљање уређајима
- Блок 6 – Контрола апликације
- Блок 7 – Извештавање о догађајима
- Блок 8 – Додатни кориснички објекти
- Блок 9 – Временски подаци

Програмска библиотека **ICCP Framework**, која пружа **ICCP** функционалности, подржава блокове 1, 2, 4 и 5, и сам симулатор протокола подржава идентичне блокове.

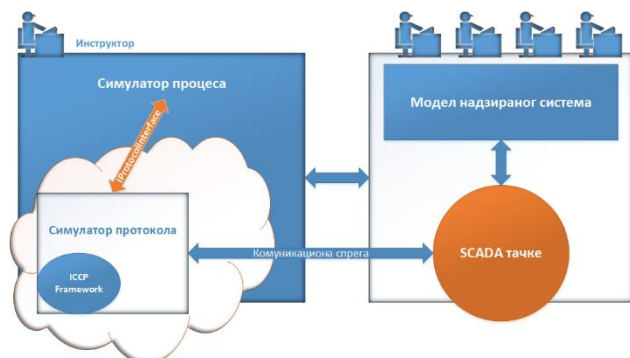
2.5 ICCP Framework

Рад се заснива на **ICCP** функционалности коју пружа **SISCO** библиотека написана у **C** програмском језику. Зарад коришћења **ICCP** функционалности у **C++** окружењу, имплементирано је објектно окружење (**ICCP C++ Framework**) које представља омотач око **SISCO**-ове библиотеке. Такође постоји и **.NET** окружење намењено програмској подршци у **C#** програмском језику. Коришћењем **.NET** библиотеке олакшана је имплементација протокола јер је пружен објектни приступ **SISCO**-вој апликативној програмској спрези. **SISCO** библиотека подржава 1, 2, 4, и 5 блок **ICCP** протокола. Библиотека поседује могућност конфигурисања из **XML** датотека. На слици 2.3 се налази **ICCP Framework** и **SISCO** библиотека.



Слика 2.3 **ICCP Framework** и **SISCO** библиотека

3. КОНЦЕПТ РЕШЕЊА



Слика 3.1 Приказ архитектуре решења

На слици 3.1 је приказана архитектура решења овог задатка. Иницијално, симулатор протокола је креиран као **ICCP** симулатор, где су имплементирани методе интерфејса **IProtocolInterface**. Овај интерфејс се налази у склопу другог симулационог окружења, симулатор процеса, где се његове методе мапирају на методе **ICCP Framework**-а. **IProtocolInterface** представљен је као интерфејс јер се учитава динамички коришћењем **MEF**-а. Направљена је конзолна апликација, да би се тестирале саме методе које обухватају првенствено иницијализацију, и на тај начин, била омогућена комуникација са **SCADA** системом. Кад је само тестирање успешно прошло, сама апликација је преточена у **plugin** који је убачен у симулатор процеса, где је даље тестирање базирано на интерактивној комуникацији између инструктора и учесника.

Симулатор процеса има улогу симулирања понашања система који се надзире. Аутоматски рачуна симулиране вредности на основу улаза које поставља инструктор. Све акције које инструктор задаје се чувају

као сценарији тренинга, да би се накнадно могле поновити. Симулатор процеса има стање и врши прорачуне на основу одговарајућег симулационог модела које то стање одржавају. Симулирајући одговарајуће дискретне вредности као што је стање прекидача, као вредности физичких величина, он генерише промене које се шаљу са мерних уређаја. **SCADA** систем је задужен за читавање тих промена и њихов пренос до Модела надзираног система. Модел надзираног система рачуна ново стање посматраног процеса и саветује учеснике обуке како да реагују, узимајући у обзир телеметрисане и нетелеметрисане тачке. Телеметрисане тачке су тачке чије вредности су добијене приликом читавања са терена а у овом случају симулатора, док су нетелеметрисане тачке оне тачке које су ручно унете, то је оно што инструктор задаје. Сама улога **ICCP** као симулатор протокола је да омогући симулатор процесу да пренесе вредности телеметрисаних тачака преко **ICCP** протокола до **SCADA** система. Основна идеја овог рада је да се направи део **ICCP** симулатор протокола који ће пружити подршку за базичне функционалности, на основу **ICCP** протокола, где би симулатор процеса могао да симулира одговарајуће физичке процесе које покривају базичне функционалности.

Спецификација симулатора протокола:

- обезбедити конфигурисање симулатора из две конфигурационе **XML** датотеке. У једној датотеци се налази целокупна конфигурација **ICCP** док је у другој, конфигурација мрежног **OSI** слоја – најчешће користи **TCP**.
- креирати конзолну апликацију која учитава **ICCP** функционалности из **ICCP** симулатора за потребе тестирања
- успоставити конекцију конзолне апликације са **ICCP** сервисом ради тестирања функционалности
- иницијализација **ICCP Framework**-а, покретање и заустављање **ICCP** услуге
- ручна промена локалних тачака које су послужене, промена подразумева мењање самих вредности тачака од стране корисника
- читање локалних тачака
- приказ информација о локалним уређајима
- промена локалне вредности ради верификације успешног командовања над локалним уређајем
- приказ информација о клијентским тачкама
- управљање клијентским тачкама

4. ИМПЛЕМЕНТАЦИЈА РЕШЕЊА

Програмско решење симулатора за **IEC 60870-6** протокол, и одговарајуће подршке, имплементирано је коришћењем **C#** програмског језика у **Microsoft Visual Studio 2013 – Professional Edition** развојном окружењу.

Реализација је започета анализом **ICCP** симулатора и **ICCP Framework**-а. Намапиране су методе из интерфејса **IProtocolInterface** на методе **ICCP** симулатора. Имплементирана је подршка за логику симулатор протокола у смислу потребних функционалности да би могао да успешно комуницира са другом страном. Након тога је направљена конзолна апликација да би се тестирале све имплементирани функционалности.

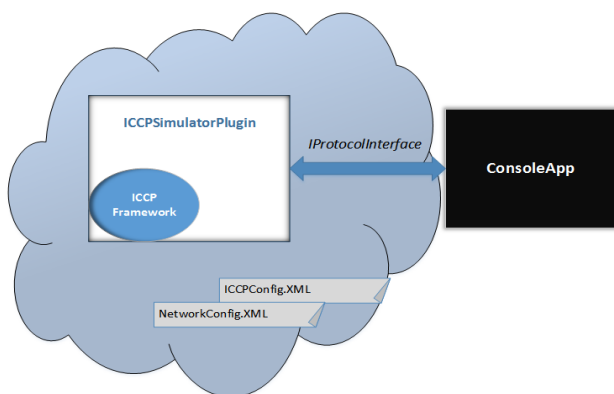
Процес иницијализације симулатор протокол дела, започиње учитавањем конфигурације из две .xml датотеке. Када симулатор затражи иницијализацију, позива се метода *InitializeSim*. У једној датотеци налази се *ICCP* конфигурација (*ICCPConfig.xml*), док је у другој конфигурација мрежног слоја (*NetworkConfig.xml*).

Уколико операција конфигурисања није успешно извршена, корисник ће бити о томе обавештен. Информације које садрже конфигурационе датотеке су назив локалног контролног центра и бројеви дефинисаних локалних тачака, и уређаја.

С обзиром да је симулатор независан од *SCADA* система, мора бити свестан тачака са којима треба да врши симулирања.

5. ТЕСТИРАЊЕ РЕШЕЊА

Тестирање функционалности симулатор протокола, обављено је повезивањем са *SISCO* тестном апликацијом. Симулатор представља локални контролни центар, а тестна апликација удаљени и обрнуто.



Слика 5.1 Архитектура тестне апликације и симулатора

Тестна апликација (слика 5.1) је конзолног типа, подржава *ICCP* функционалности, тако да у зависности од конфигурације, може бити послужеоц, или клијент. Конфигурише се путем *XML* датотека као и симулатор, датотеке се морају налазити у истом директоријуму, заједно са тестном апликацијом. Тестира функционалности датог решења. У виду логова, као *XML* датотека *"iccpSimPlugin.xml"* исказане су успешност и неуспешност позива функција. Успостављањем асоцијације, остварена је комуникација преко *ICCP* протокола, између тестне апликације и симулатора. Тестна апликација може да генерише промене вредности локалних тачака, уређаја, мења стања асоцијација, покреће и зауставља, као и да ишчитава вредности локалних и клијентских тачака и уређаја из симулатора, што је одличан модел за тестирање.

6. ЗАКЉУЧАК

У овом раду је представљен концепт и реализација *ICCP* протокола у циљу комуникације са *SCADA* системом. Објашњени су основни концепти самог протокола, као и појмови који су везани за овај рад. Самим тим што је *ICCP* протокол реализован као симулатор у виду *plugin*-а, односно као *plugin* оставља могућност проширења или смањења функционалности

које нуди, за потребе других, те је независан од саме апликације, говори колико погодности нуди за било које симулационо окружење, као средство за комуникацију са *SCADA* системом.

Може да се користи за тестирање самог надзорно управљачког система, да провери све подржане функционалности, да ли раде како би иначе требало. *ICCP* као симулатор протокола – *plugin*, може oponaшати уређаје, односно стање на терену и да, користећи симулатор, комуницира са надзорно управљачким системом. Кроз то ће указати да ли постоје евентуалне грешке у истом. Може користити за обуку корисника.

Корисно би било да се од стране корисника додели могућност коришћења произвољних симулација, због oponaшања стварног понашања једног реалног технолошког процеса.

На основу концепта самог протокола је успешно реализовано програмско решење, које у потпуности испуњава тражене функционалности.

7. ЛИТЕРАТУРА

- [1] Бранислав Атлагић, Софтвер са критичним одзивом – Пројектовање *SCADA* система, ФТН издаваштво, Нови Сад, 2015.
- [2] Utility Communications Specification Working Group: IEC 870-6-503 TASE.2 Services And Protocol Version 1996-08, 1996.
- [3] Wikipedia, *SCADA*, 2014, <https://sr.wikipedia.org/sr/SCADA>
- [4] Research Gate, *ICCP*, https://www.researchgate.net/figure/224598619_fig12_Fig-13-ICCP-mapping-onto-the-OSI-stack
- [5] Wikipedia, .Net Framework, 2016, https://sr.wikipedia.org/sr/.NET_Framework
- [6] Andrew Troelsen, PRO C# 5.0 and the .NET 4.5 Framework, 2012, APRESS, Sixth edition
- [7] Microsoft, Managed Extensibility Framework (MEF), 2016, [https://msdn.microsoft.com/en-us/library/dd460648\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/dd460648(v=vs.110).aspx)
- [8] AMA-SYSTEMS GMBH D 75179 Pforzheim April 2011, *ICCP Basics + ICCP Lite* Ivan Enchev
- [9] Utility Communications Specification Working Group: IEC 870-6-802 Object Models Version 1996-08, 1996

Кратка биографија:



Јелена Карага рођена је 23. маја 1992. у Ђаковици. Завршила средњу школу, ЈУ Гимназија „Васо Пелагић“ у Брчко Дистрикту, 2011. године. Факултет Техничких Наука у Новом Саду је уписала 2011. године, смер Рачунарство и аутоматика. Дипломирала је 29.09.2016. године. У октобру 2016. године, уписала је Мастер академске студије, смер Примењено софтверско инжењерство.

КРЕИРАЊЕ ГЕНЕРИЧКИХ ИЗВЕШТАЈА О КОМПОНЕНТАМА ЕЛЕКТРО-ЕНЕРГЕТСКОГ СИСТЕМА**CREATING GENERIC REPORTS ABOUT COMPONENTS OF THE ELECTRIC POWER SYSTEM**

Бојан Вујиновић, Факултет техничких наука, Нови Сад

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – Овај рад приказује генерички приступ креирању извештаја о компонентама електро-енергетског система. Омогућено је селектовање компоненте одабраног типа, приказ њених података, као и руковање њеним подацима. Развијена је подршка за избор дела електро-енергетског система из кога се селекују компоненте, лоцирање компоненти у шеми електро-енергетског система, штампање извештаја, експорт података у csv формат, као и задавање конфигурационих параметара за селекцију компоненти. Апликација је развијена помоћу WPF (Windows Presentation foundation) технологије, као и помоћу Model-View-ViewModel (MVVM) шаблона.

Abstract – This paper presents a generic approach to creating reports of components of the power system. It is possible to select component of some type, and handle its data. Support was developed for selection of a part of the electric power system from which components are selected, locating the components in the scheme of the power system, printing reports, exporting data into the CSV format, and setting configuration parameters for component selection. The application was developed using WPF (Windows Presentation Foundation) technology, as well as using the Model-View-ViewModel (MVVM) template.

Кључне речи: генерички извештај, XML, WPF, MVVM

1. УВОД

У овом раду је описана апликација намењена да олакша рад оператерима електро-енергетске мреже, тако што брзо и ефикасно приказује елементе мреже. Програмско решење апликације је генеричко како би се обезбедио рад са различитим моделима података. У 2. глави изложени су захтеви за генерички приказ елемената електро-енергетске мреже. Глава 3. садржи преглед алата и технологија, коришћених за имплементацију апликације.

Имплементација апликације је приказана у 4. глави. Рекапитулација урађеног и смернице за даљи рад садржи 5. глава. На крају овог рада налази се списак коришћене литературе, као и кратка биографија аутора овог рада.

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је био др Мирослав Хајдуковић, р. проф.

2. ОПИС РЕШАВАНОГ ПРОБЛЕМА

Разматрана апликација се бави креирањем генеричких извештаја за компоненте електро-енергетске мреже. Она треба да омогући приказ тачних података за сваки елемент мреже, који је одабран по задатом критеријуму. Задатак ове апликације је да се олакша рад оператерима електро-енергетске мреже тако што ће им понудити детаљан увид у комплетне податке одабраних компоненти мреже, као и увид у позицију компоненте у мрежи. Кориснички интерфејс ове апликације треба да омогући не само природан приступ њеној функционалности, него и њено конфигурирање. Свака компонента електро-енергетске мреже има свој тип. У табели 1 је приказана структура типа компоненте.

ComponentType		
Ознака	Тип податка	Опис
ComponentID	long	Јединстваени идентификатор компоненте
ComponentName	string	Назив компоненте

Табела 1 Структура типа компоненте

Селекција и филтрирање компоненти се обавља помоћу њиховог ComponentID и може да обухвати целу електро-енергетску мрежу или само један њен део. Пошто свака компонента има више врста података, за селектоване компоненте је потребно навести који од података ће бити приказани. Избор приказиваних података представља опцију за конфигурирање апликације. Избор врсте приказиваних података се заснива на увођењу њихових атрибута, чија структура је приказана у табели 2.

Attribute		
Ознака	Тип податка	Опис
AttributeID	long	Јединстваени идентификатор атрибута
AttributeName	string	Назив атрибута

Табела 2 Структура атрибута

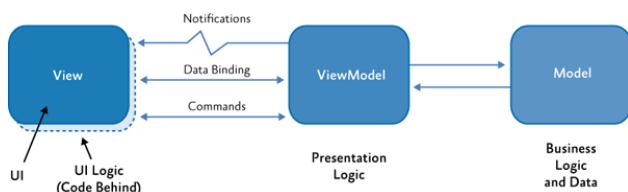
AttributeName се користи као наслов приказиване врсте података. AttributeID представља јединствени идентификатор атрибута и садржи информацију ком

типу података припада изабрани атрибут. Извештај са приказаним подацима одабраних компоненти електро-енергетске мреже треба да има форму табеле, чији сваки ред садржи податке једне компоненте. Пошто су компоненте различитих типова, апликација треба да има својство генеричности. Више колона ове табеле представља једну селекцију. Приказивани подаци се налазе на више различитих сервиса и пре приказивања се морају додати са њих.

3. КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ

Апликација је развијена помоћу *MVVM* шаблона [1] коришћењем *Windows Presentation Foundation (WPF)* технологије [2]. Опције су дефинисане коришћењем *XML*-а [3].

MVVM шаблон је замишљен да се користи у склопу *WPF*-а како би се олакшало креирање апликација. Главна сврха овог шаблона је да раздвоји *UI* део од саме апликативне логике. На слици 1 је приказан *MVVM* шаблон.



Слика 1 *MVVM* шаблон

MVVM шаблон се састоји од три дела:

- *View*,
- *Model*,
- *ViewModel*.

View је задужен за дизајн интерфејса апликације. Представља скуп прозора или *UserControl*-а дефинисаних помоћу *Extensible Application Markup Language* [2]. *Model* представља део где се дефинише модел података.

ViewModel представља посредника између предходна два дела *MVVM* шаблона и садржи скуп правила за дефинисање формата података.

Windows Presentation Foundation је један од релативно нових *framework*-а, чија је сврха креирање конзистентног интерфејса *Windows* и *Web* апликација. *WPF* се први пут уводи од *.NET framework 3.0* верзије и представља графички подсистем овог *framework*-а. Концепт презентационог дела је дефинисан кроз *XAML*. Обезбеђује конзистентан програмски модел као и раздвајање корисничког интерфејса (*UI*) и бизнис логике.

4. ОПИС РЕШЕЊА ПРОБЛЕМА

Сврха описивања апликације је да олакшава рад оператерима електро-енергетске мреже, тако што нуди ефикасан генерички приказ података за све елементе изабраног типа компоненте, који припадају жељеном делу мреже.

4.1. Опис програмског решења

У *View* се налазе *XAML* фајлови у којима је дефинисан дизајн приказа. У *Model*-у су дефинисани модели података, док *ViewModel* обухвата класе које дефинишу већи део апликативне логике. Овај шаблон

програмеру омогућава једноставније руковање кодом, као и лакше сналажење у комплексним решењима.

Креирање генеричког модела је постигнуто коришћењем *DynamicObject*-а.

Апликација подржава добављање и приказ података, креирање филтера, као и приказ тренутне странице у табели.

Прикупљање и добављање података захтева сарадњу са више различитих сервера. На овим серверима се налазе подаци за све елементе у мрежи и потребно их је након добављања спојити. Спајање елемената се обавља уз помоћ јединственог идентификатора елемента у мрежи.

Формат добављених података је приказан на слици 2.

`Dictionary<long, List<object>> result`

Слика 1 Формат података добављених са сервиса

Кључ колекције са слике 2 садржи податке типа *long* и представља *gid*(јединствени идентификатор) елемента у мрежи. Вредност колекције садржи податке типа *List<object>* и представља листу добављених података.

Након што се добава подаци, потребно их је сместити у универзалну колекцију у којој кључ представља идентификатор елемента, а вредност представља податке, који су преузети са одговарајућих сервиса. Спајање података се обавља помоћу *gid*-а. Спајањем података ће се објединити сви подаци за приказ у табели. Сваки ред у овој табели представља податке за по један елемент у мрежи.

Да би се могло радити са *DynamicObject*-ом потребно је укључити библиотеку *System.Dynamic*. *DynamicObject* представља основну класу за динамичко понашање у току извршавања. Ову класу није могуће директно инстанцирати, већ ју је потребно директно наследити из класе која представља модел податка.

Пре приказа података они се складиште у колекцију чији је елемент класа која наслеђује *DynamicObject*. У истој класи је потребно проширити и методу *TryGetMember* приказану на слици 3. Ова метода служи за дефинисање *Binding*-а. *Binding* по *MVVM* шаблону омогућава спајање података са колоном у табели.

```
public virtual bool TryGetMember(
    GetMemberBinder binder,
    out object result
)
```

Слика 3 *TryGetMember* метода

Након додавања података у колекцију и дефинисања *Binding*-а, потребно је податке приказати у табели.

Да би се приказали подаци, потребно је дефинисати *ItemsSource* у оквиру *DataGrid* компоненте. У *ItemsSource* се дефинише назив колекције у којој се налазе подаци (Слика 4).

`<DataGrid ItemsSource="{Binding MyList}" AutoGenerateColumns="False">`

Слика 4 Дефинисање колекције у *ItemsSource*

Овим приступом је обезбеђен приказ података кроз генерички модел података у оквиру табеле.

Након додавања података у колекцију и прављења *Binding*-а, креирају се и подаци у филтеру.

Филтер обухвата листу колона из табеле апликације. Дизајн филтера зависи од типа податка ком колона припада. Овај тип податка се добија на основу јединственог идентификатора атрибута.

Приликом приказа података креира се компонента за приказ елемената по страницама. Ова компонента приказује само 30 елемената по страници. У *ItemsSource* се додаје 30 елемената из помоћне колекције, па се на сваку промену странице *ItemsSource* ажурира.

Додати подаци се смештају у табелу која се налази у *ContentControl*-и и прослеђују одговарајућем прозору. Овим решењем се избегава брисање података приликом новог извршавања опције, што се показало као брзо решење.

Постоје два типа опција:

- статичке и
- динамичке.

Статичке опције су унапред задате у посебном *XML* фајлу и представљају почетне опције, којима ће бити попуњен прозор за додавање опција.

Динамичке опције представљају скуп атрибута накнадно додатих кроз прозор за додавање опција.

4.2. Опис корисничког интерфејса

Кориснички интерфејс апликације се састоји из два дела. Први део представља главни прозор апликације, док други део представља прозор за рад са опцијама.

4.2.1 Опис прозора за рад са опцијама

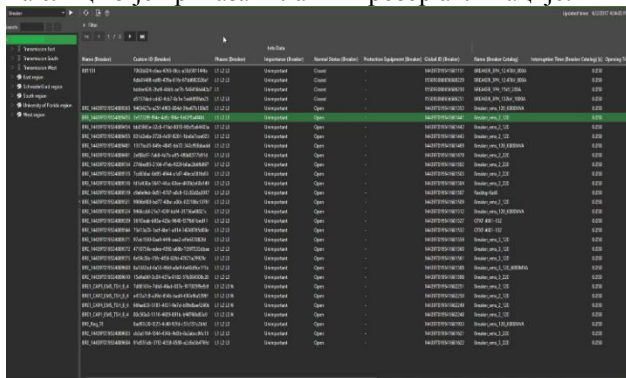
Прозор за рад са опцијама омогућава избор података који ће бити приказани у оквиру табеле. Бирањем опција корисник одређује атрибуте које жели да приказује, као и назив секције којој припадају одређени атрибути.

4.2.2 Опис главног прозора апликације

Главни прозор апликације чине:

- *toolbar*,
- *network tree*,
- филтер,
- компонента за приказ података по страницама,
- табела за приказ података.

На слици 5 је приказан главни прозор апликације.



Слика 5 Главни прозор апликације

4.2.2.1 Toolbar

Toolbar омогућује:

- избор типа компоненте,
- *Execute*,
- *Refresh*,
- *Export to CSV*,
- *Print*,
- *Updated time*.

4.2.2.2 Network tree

Network tree представља хијарархијску структуру која се састоји од:

- *feeder-a*,
- *substation-a*,
- *region-a*,
- трансформаторских области.

4.2.2.3 Филтрирање података

Да би се олакшао рад са великим бројем података, додата је и могућност филтрирања. Филтрирање података је могуће само по колонама у табели.

4.2.2.4 Табела за приказ података

Називи секција, као и колона су преходно дефинисани кроз прозор за додавање опција. У табели се приказују сви подаци који су дефинисани за изабрани тип компоненте кроз прозор за дефинисање опција. Приказују се само оне компоненте које припадају изабраном делу мреже кроз *Network tree*.

На самом врху табеле главног прозора апликације се налазе називи секција, док се испод њих налази група колона додељена приказаној секцији.

Сваки ред у табели садржи податке за једну компоненту припадајуће мреже. У табели се искључиво приказују подаци компоненти које су директно везане за селектовани део електро-енергетске мреже. Понекад је податке са сервиса потребно конвертовати у други тип података. У том случају се користе конвертери. Обично су то подаци типа енумерације, *datetime* или броја.

Максималан број елемената који могу да се прикажу у табели је ограничен на 30 елемената. У случају да постоји више елемената за приказ од 30, прелази се на приказ елемената по страницама. У склопу табеле се налазе и опције контексног менија. Након извршавања опције *Execute*, промене селекције у компоненти *Network tree* или након филтрирања мењају се и подаци у табели.

4.2.2.5 Приказ података по страницама

Поље за приказ података по страницама ради у синхронизацији са табелом. Намењено је да кориснику олакша рад кроз ефикасније и једноставније сналажење међу подацима.

4.2.2.6 Опције контексног менија

Кориснику се додатно олакшава рад коришћењем контексног менија, који се састоји од поља за: лоцирање, за копирање ћелије и реда у табели, за конвертовање у *csv* формат, за отварање селектоване компоненте у шеми, као и за штампање извештаја и отварање детаљнијег приказа података.

5. ЗАКЉУЧАК

У овом раду представљен је опис корисничког интерфејса и програмског решења апликације. Ова апликација је намењена да олакша рад оператерима електро-енергетског система. Помоћу ове апликације је решен проблем приказа различитих модела података. Развијен је посебан табеларни приказ генеричког модела података.

Објашњене су могућности ове апликације, које олакшавају једноставно и ефикасно сналажење у електро-енергетском систему.

ЛИТЕРАТУРА

- [1] XML Introduction, https://www.w3schools.com/xml/xml_what.asp
- [2] WPF Tutorial, <https://www.tutorialspoint.com/wpf/>
- [3] MVVM Pattern, <https://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93ViewModel>
- [4] Erik T. Ray, Learning XML, 2nd Edition, 2003

[5] Chris Sells, Ian Griffiths, Programming WPF, 2nd Edition, December 2008

[6] Arnaud Weil, Learn WPF MVVM – XAML, C# and the MVVM pattern, 2017

Кратка биографија:



Бојан Вујиновић је рођен 1. октобра 1992. године у Новом Саду. Завршио је средњу електротехничку школу Михајло Пупин у Новом Саду. Школске 2011/2012. године уписао је Факултет Техничких Наука у Новом Саду, одсек Електротехника и Рачунарство, смер Рачунарство и аутоматика, усмерење Примењене рачунарске науке и информатика. Основне академске студије завршио је 2016. године. Одмах након тога уписао је мастер академске студије Примењено софтверско инжењерство.

KONDOMAT SA UGRAĐENIM FUNKCIJAMA ZA DALJINSKI NADZOR CONDOM MACHINE WITH BUILT-IN FUNCTIONS FOR REMOTE CONTROL

Goran Ćutić, Darko Marčetić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu predstavljene su dve bitne celine. Prvu celinu čini projektovanje i razvoj hardvera za prototip Kondomata, mašine za distribuciju kondoma i razvoj softvera za upravljanje Kondomatom. Drugu celinu čini daljinski nadzor Kondomata putem GSM/GPRS modula i Internet baze podataka (Firebase – baza), dodatak je Online registracija korisnika za generisanje šifre za besplatno podizanje kondoma.

Abstract – In this paper two important parts are presented. The first part makes designment and development of hardware for prototype of the condom distribution machine and software development for condom machine management. The second part makes remote monitoring of the condom machine via GSM/GPRS module and Internet database (Firebase database), the plugin is an online user registration to generate codes for free condomes.

Кljučне речи: Kondomat, GSM/GPRS, IoT, Firebase, softver i hardver.

1. UVOD

Vending mašine služe za prodaju raznih proizvoda. Jedna od mašina čiji će nastanak, razrada ideje, principi rada, izbor komponenti, razvoj hardvera, razvoj softvera, testiranje GSM/GPRS komunikacije sa Internet bazom podataka i pravljenje sajta za registraciju korisnika biti opisani u nastavku je KONDOMAT, mašina za prodaju kondoma. Ideja o izradi kondomata je nastala u okviru Studentskog preduzeća inženjera Novog Sada – „SPINS“. Prototip i koncept kondomata se mogu videti na slici 1.



Slika 1. Prototip i koncept kondomata

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Marčetić, red.prof.

2. PROJEKTOVANJE I RAZVOJ HARDVERA ZA PROTOTIP KONDOMATA

Kondomat se sastoji od više hardverskih modula koji zajedno sa mikroprocesorom čine jedinstvenu celinu. U nastavku se može videti blok dijagram kondomata, slika 2.

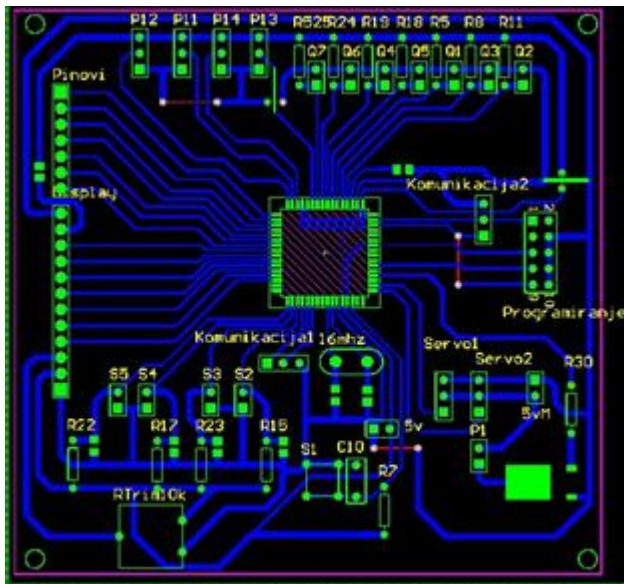


Slika 2. Opšti blok dijagram kondomata

Kondomat sadrži sledeće osnovne komponente:

1. Digitalni upravljački deo – mozak celog sistema. Korišćen je mikrokontroler ATMEGA128, proizvođača ATMEL;
2. Programator čija je osnovna uloga učitavanje HEX fajla u FLASH memoriju mikrokontrolera. Korišćen je USBASP programator;
3. Modul za prihvatanje novca, čija je uloga prijem novčanica i naplata. Korišćen je modul NV10-USB, firme Innovative Technology;
4. Motori koji su zaduženi za izbacivanje kondoma;
5. Optički senzori imaju ulogu očitavanja brojnog stanja prezervativa u uređaju i detekciju punog okretaja motora za izbacivanje kondoma.;
6. Napajanje koje obezbeđuje adekvatan vid napona za određene periferije. Glavni izvor napajanja je ispravljač i stabilizator napona (230VAC/12VDC) tipa RS-150-12, proizvođača Mean Well. Kao sekundarni izvor jednosmernog napona se koristi Step down pretvarač LM2596 (12VDC/5VDC).
7. Tablet, veličine 10.1“ koji je namenjen kao prostor za reklamiranje i distribuciju video snimaka;
8. LCD displej, čija je uloga da informiše kupca o koracima koji se odvijaju tokom procesa kupovine i korišćenja uređaja, korišćen je displej tipa 2X16LCD;
9. Tastatura ima ulogu da obezbedi korisniku unos šifre za besplatan kondom ili jednostavno unos količine kondoma koju korisnik želi da kupi;

Kada se definišu komponente i izradi električna šema pristupa se izradi rasporeda komponenti na štampanoj pločici. Na slici 3 se može videti projektovana štampana pločica (donja strana) mikroprocesorskog modula.



Pri raspoređivanju opreme važno je voditi računa da se vodovi što manje seku (po mogućnosti da se ne seku uopšte), da se ne bi prelazilo u drugi sloj bez potrebe. Komponente treba da budu grupisane tako da se što jednostavnije i sa što kraćim vodovima povežu. Važno je napomenuti da oscilator sa svojim kondenzatorima mora biti fizički što bliže mikrokontroleru zbog svoje velike frekvencije rada.

Takođe, gde god postoje kondenzatori, koji služe za hardversko otklanjanje smetnji pri pritiskanju tastera, moraju biti što bliže odgovarajućem tasteru. Kada se komponente optimalno rasporede prelazi se na iscrta-vanje vodova. Na samom početku važno je znati u kojoj se tehnologiji izrađuje pločica, tj. koliki je najmanji mogući razmak između vodova i koliki je najtanji dozvoljeni vod (u navedenom slučaju je 0.20mm). Po podešavanju ovih parametara pristupa se povezivanju vodova. U nedostatku prostora je moguće provlačenje vodova ispod mikrokon-trolera i prelaženje preko pinova koji nisu u upotrebi.

Osnovni kriterijum pri izboru mikrokontrolera jeste da on zadovoljava hardverske potrebe samog automata, odnosno da ima dovoljan broj određenih ulaza i izlaza, komunikacijskih portova, zadovoljavajuću brzinu, dovoljno memorije kako EEPROM tako i FLASH, pristupačan na tržištu i da postoji dobra podrška za rad sa kontrolerom. Po definisanju I/O liste i konsultacijom sa mentorima na projektu izabran je ATMEGA128 mikrokontroler proizviđača ATAMEL [3].

Paralelno sa razvojem hardvera razvijao se i softver. Na početku je pisan softver za svaki modul hardvera posebno, da bi na kraju sklopili sve u jedinstveni sistem. Blok dijagram algoritma glavnog programa je prikazan na slici 5.

- Potrebni periferiji (UART, PWM, timer),
- Programskih konstanti (cena kondoma, parametri za "debouncing" itd.),
- Programskih globalnih promenljivih (postavljanje flegova na nulu itd.).

- Ispisuje se potreban tekst na displej,
- Proverava se da li je primljena poruka od strane serviser a stanju kondoma,
- Provera tastera za unos količine i provera tastera za potvrdu kupovine.

- Ispisuje se nova poruka na displej sa sadržajem cene željenog broja kondoma,
- Setuje se tajmer na vreme od 30 sekundi (kada istekne ovo vreme program se vraća u inicijalni korak 0),
- Uključuju se predajničke diode za foto senzore.

- Uključuje se modul za prihvrat novčanica,
- Definisane promenljive koje označavaju trenutnu i očekivanu količinu unetog novca.

- Proverava se da li je ubačena novčanica.

- Proverava se da li je uneta novčanica validna.

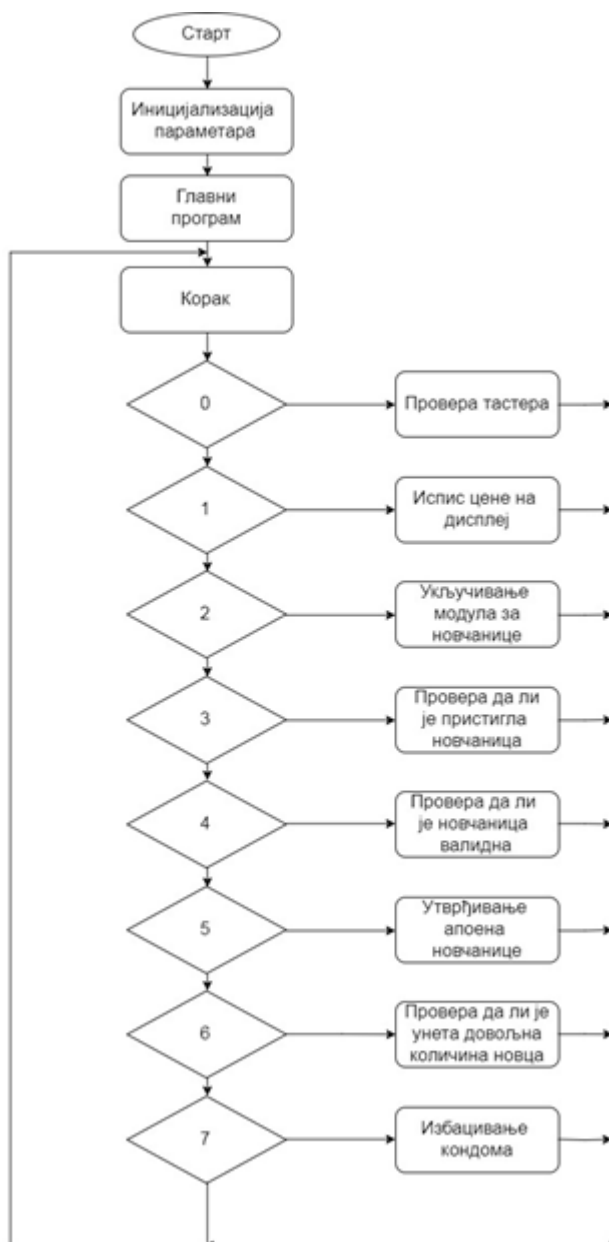
- Proverava se koja novčanica je uneta,
- Trenutna vrednost unetog novca se povećava za vrednost ubačene novčanice,
- Na displeju se ispisuje koliko je trenutno uneto novca,
- Ukoliko novčanica prelazi 200 RSD, ona se vraća korisniku.

- Proverava se da li je uneta količina novca jednaka očekivanoj,
- Ako je uneta količina manja od očekivane program se vraća u korak 3,
- Ukoliko je uneta količina veća ili jednaka očekivanoj isključuje se modul za novčanice i prelazi se u korak7.

U koraku 7:

- Setuju se pomoćni flegovi na nulu,
- Uključivanje motora koji izbacuje kondome,
- Nakon izbacivanja kondoma proverava se stanje u šaržeru,
- Isključuju se predajne diode fotosenzora,
- Čisti se ekran.

Nakon ovog stanja vraćamo se u inicijalni korak 0.



Slika 5. Blok dijagram algoritma glavnog programa

4. DALJINSKI NADZOR KONDOMATA

Internet stvari (IoT- *The Internet of Things*) je novi koncept pametne automatizacije i pametnog nadgledanja uz pomoć Interneta kao medija komunikacije. „Stvari“ u IoT se obično odnose na uređaje koji imaju jedinstvene identifikatore povezane sa Internetom za međusobnu razmenu informacija. Takvi uređaji imaju senzore i/ili aktuatora koji se mogu koristiti za prikupljanje podataka o njihovim okruženjima, kao i za nadgledanje i kontrolisanje istih. Prikupljeni podaci se mogu obraditi lokalno i mogu biti poslani na centralizovane servere ili oblak radi

čuvanja i obrade na daljinu. Kondomat zahvaljujući GSM/GPRS SIM800L modulu ostvaruje komunikaciju sa oblakom i šalje SMS poruke serviseru o stanju kondoma. Prikupljeni podaci se upisuju na oblak i posle čitaju sa oblaka radi obrade potrebnih informacija. Na slici 6 se može videti ilustracija komunikacije kondomata preko GSM/GPRS modula sa oblakom [1], [2].



Slika 6. Ilustracija komunikacije kondomata preko GSM/GPRS modula sa oblakom

Osnovni način upisivanja i čitanja u bazu podataka je putem URL-a (engl. Uniform Resource Locator). URL za čitanje iz baze podataka se sastoji iz tri segmenta:

- Prvi segment sadrži osnovni URL baze podataka. U navedenom projektu to je:

<https://kondomat-23ddd.firebaseio.com/>

- Drugi segment se gradi tako što se putuje kroz stablo do željene lokacije. Na primer ukoliko je potrebno očitati podatke šifre 2330 na prvi segment se dodaje:

[users/2330/2330.json?](https://kondomat-23ddd.firebaseio.com/users/2330/2330.json?)

odnosno ukoliko je potrebno da se proverí samo status šifre dodaje se:

[users/2330/2330/userStatus.json?](https://kondomat-23ddd.firebaseio.com/users/2330/2330/userStatus.json?)

- Treći segment nosi podatke o autentifikaciji. Na drugi segment se dodaje autentifikacioni kod u sledećem zapisu:

[auth=SttUUMtCeM2B1HH5LMXBIXww08kM5krok0e7BzRp](https://kondomat-23ddd.firebaseio.com/users/2330/2330/userStatus.json?auth=SttUUMtCeM2B1HH5LMXBIXww08kM5krok0e7BzRp)

Konačni izgled celog URL-a za čitanje stanja šifre 2330 iz baze podataka se može videti u nastavku:

<https://kondomat-23ddd.firebaseio.com/users/2330/2330/userStatus.json?auth=SttUUMtCeM2B1HH5LMXBIXww08kM5krok0e7BzRp>

Kao što je već napomenuto Firebase se koristi za skladištenje šifri za besplatno podizanje kondoma. Osnovni razlog za opredeljenje cloud baze podataka je to što jedna jedinstvena šifra može da se proverava sa bilo kog kondomata.

Takođe ovim je izbegnuta potreba za računarnom na kome bi bila baza podataka, ne postoje troškovi održavanja i nema potrebe za brigom da li će podaci nestati sa računara.

Na sledećem linku može se otvoriti veb stranica Firebase platforme <https://firebase.google.com/>

GSM/GPRS SIM800L modulu se pristupa putem AT komandi. Pomoću funkcije HTTPS_GET() se čitaju podaci sa Firebase baze podataka. C programski kod ove funkcije se može videti u nastavku:

```
void HTTPS_GET()
{
    SendString1("AT+HTTPINIT\r");
    delay_ms(1000);
    SendString1("AT+HTTTPARA=\"CID\",1\r");
    delay_ms(1000);
    SendString1("AT+HTTTPARA=\"URL\", \"https://kondomat-7038d.firebaseio.com/users/8745.json?auth=cb5d00e0AgpZzIT4gbkf6bkS6DW2u9QD57WwGg7a\" \r");
    delay_ms(1000);
    SendString1("AT+HTTTPARA=\"REDIR\",1\r");
    delay_ms(1000);
    SendString1("AT+HTTTPARA=\"CONTENT\", \"application/json\" \r");
    delay_ms(1000);
    SendString1("AT+HTTTPARA? \r");
    delay_ms(1000);
    SendString1("AT+HTTTPSSL=1\r");
    delay_ms(1000);
    SendString1("AT+HTTTPACTION=0\r");
    delay_ms(2000);
    SendString1("AT+HTTTPREAD=0,100\r");
    delay_ms(5000);
    SendString1("AT+HTTPTERM\r");
    delay_ms(1000);
}
```

5. ONLINE REGISTRACIJA KORISNIKA

Online registracija korisnika za besplatan kondom je prois-tekla iz ideje kako bi se promovisao kondomat u prvim mesecima njegovog plasiranja na tržište. U tu svrhu napravljen je sajt na kome korisnik može da se registruje unošenjem imena, prezimena i email-a. Sajt sve te podatke upisuje na Firebase bazu podataka i automatski generiše šifru za besplatno podizanje kondoma, koju putem email adrese prosleđuje korisniku.

Sajt predstavlja veb aplikaciju koja je razvijena korišćenjem Spring boot framework-a. Korišćeno razvojno okruženje je IntelliJ. Izgled veb str-a-nice za registraciju korisnika se može videti na slici 7.

Slika 7. Izgled veb stranice za registraciju korisnika

Aplikacija je realizovana upotrebom troslojne arhitekture. Troslojna arhitektura predstavlja sistem sa tri sloja koji su u velikoj meri nezavisni:

1. Prezantacioni sloj – implementira funkcije korisničkog interfejsa,
2. Sloj poslovne logike – iplementira poslovnu logiku aplikacije,
3. Sloj podataka – iplementira pukovanje podacima

6. ZAKLJUČAK

Projekat Kondomat predstavlja jedinstvenu strukturu primene teorije u praktičnom radu, istraživanja, razvoja i multidisciplinarnosti. Sve to se ogleda kroz projektovanje, programiranje u C programskom jeziku za rad mikrokot-rolera, veb programiranje stranice za registraciju i istraži-vanje mogućnosti GSM/GPRS komunikacije kroz IoT (*Internet Of Things*).

IoT se trenutno nalazi u velikom usponu, što čini ovaj rad dosta značajnim. Najznačajnija znanja su ona stečena razmenom iskustava tokom same izrade kondomata.

7. LITERATURA

- [1] Upoznavanje sa PIC mikrokontrolerima – INTERNET STVARI, Dogan Ibrahim – ELEKTOR, INFOelektronika
- [2] GSM/GPRS Projekti – Dogan Ibrahim i Ahmet Ibrahim – ELEKTOR i INFOelektronika
- [3] Programiranje mikrokontrolera PICBASIC-om, Vojo Milanović, Niš, 2011

Kratka biografija:



Goran Ćutić rođen je u Bosanskoj Gradišci, BiH, 1992. god, sa mestom prebivališta u Melencima, opština Zrenjanin. Odmah nakon diplomiranja oktobra 2015. se zaposlio u INSIST elektrik doo., Novi Sad gde se zadržao do februara 2016. U martu 2017 prelazi u Energotehniku Južna Bačka a.d. Novi Sad, da bi u septembru iste godine prešao u INDAS doo Novi Sad gde se trenutno nalazi. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska elektro–nika i električne mašine odbranio je 2017.god.

BLOCKCHAIN DISTRIBUIRANI SISTEMI I NJIHOVA PRIMENA U FARMACEUTSKOJ INDUSTRIJI**BLOCKCHAIN DISTRIBUTED SYSTEMS AND THEIR APPLICATION IN THE PHARMACEUTICAL INDUSTRY**

Aleksa Mirković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu opisane su osnove blockchain tehnologija i softverski alat koji bi trebalo da podrži primenu blockchain-a u lancu nabavke u farmaceutskoj industriji.

Abstract - This paper describes the fundamentals of blockchain technologies and the software tool that should support the use of blockchain for the supply chain in the pharmaceutical industry.

Ključne reči: *blockchain, Hyperledger Fabric, farmaceutska industrija, distribuirani sistemi.*

1. UVOD

Razmena dobara i usluga, kao aktivnost, prisutna je još od pojave društva. Svaka konkretna razmena predstavlja transakciju. Pri poslovanju, transakcija je jedinica kojom se ispunjavaju ugovorne obaveze. Povećanje obima poslovanja nameće potrebu beleženja i čuvanja izvršenih transakcija kako bi se na adekvatan način upravljalo poslovnim procesima.

Prvobitno su se računarski informacioni sistemi oslanjali na sistem datoteka za skladištenje podataka. Kod sistema datoteka bilo je komplikovano implementirati poslovne transakcije. Poslovnu transakciju je bilo potrebno realizovati kao niz operacija upisa i čitanja tako da ili sve operacije uspeju ili ne uspe ni jedna. Krajem dvadesetog veka relacije baze podataka postaju nezaobilazni činilac informacionih sistema. Relaciona baza podataka oslanjajući se na sistem za upravljanje bazom podataka podržava izvršavanje poslovnih transakcija kao nedeljivi niz operacija nad bazom podataka. Sa stanovišta načina skladištenja podataka delimo baze podataka na centralizovane i distribuirane.

Centralizovane baze podrazumevaju smeštanje podataka na jednu lokaciju. Kod njih je redundansa podataka minimalna.

Kod distribuiranih baza može se iskoristiti snaga svih računara na kojima je baza distribuirana, a podaci mogu biti dostupni na više lokacija, pa gubitak podataka sa jedne lokacije nije fatalan.

Distribuirane baze pružaju veću stabilnost i dostupnost sistema, ali ne rešavaju neke probleme načina poslovanja među partnerima. Prilikom saradnje svaki učesnik beleži sopstvene podatke i može doći do razlika u podacima koji se tiču međusobnog poslovanja. Banka se smatra nezaobilaznim posrednikom prilikom poslovanja dva entiteta.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, docent.

Bankarske usluge često mogu biti netolerantno spore i skupe. Blockchain sistemi omogućavaju da se poslovanje odvija bez posrednika, a da se zadrže prednosti koje pružaju distribuirane baze. Primena tehnologija zasnovanih na blockchain-u (eng. *blockchain*- lanac blokova) u poslovnim sistemima bi trebalo da doprinese većoj transparentnosti, većoj sigurnosti, većem otporu na otkaze, većoj brzini izvršavanja transakcija i smanjenoj ceni troškova transakcija usled uklanjanja posrednika [1]. Korišćenjem blockchain-a svi učesnici zajedno upravljaju podacima relevantnim za njihovo poslovanje, nasuprot tradicionalnim sistemima gde svaka strana održava sopstvene podatke.

Cilj ovog rada je da predloži način korišćenja blockchain tehnologije u farmaceutskoj oblasti. Obrađivaće se proces dostavljanja farmaceutskih proizvoda od proizvođača do krajnjih korisnika. Blockchain sisteme je pogodno primeniti u ovom domenu jer postoji veliki broj učesnika u lancu i veliki broj transakcija.

Na osnovu postavljenog cilja definisana su tri zadatka. Prvi zadatak podrazumeva izučavanje osnova blockchain tehnologije. Drugi zadatak se tiče istraživanja razlika između Bitcoin [1] javne mreže i Hyperledger Fabric [3] privatne mreže. Treći zadatak obuhvata realizaciju jednostavnog softverskog rešenja koje bi trebalo da podrži proces nabavke u farmaceutskoj industriji.

Osim uvoda i zaključka rad sadrži tri sekcije. U sekciji 2 opisane su osnove blockchain-a, a u sekciji 3 se porede Hyperledger Fabric i Bitcoin blockchain implementacije. U sekciji 4 je opisan primer upotrebe blockchain-a u farmaceutskoj industriji. U poslednjoj sekciji izneseni su najvažniji zaključci.

2. OSNOVE BLOCKCHAIN DISTRIBUIRANIH SISTEMA

Struktura koju danas nazivamo blockchain prvi put je uvedena u [1]. U najjednostavnijoj formulaciji se kaže da je blockchain deljeni, distribuirani, replicirani dnevnik transakcija. Pomenuti dnevnik transakcija se može posmatrati kao baza podataka. Za razliku od tradicionalnih baza gde svaka strana koja učestvuje u poslovanju ima svoje podatke, blockchain omogućava da svi učesnici poseduju iste podatke, kojima svi učesnici veruju. Najvažniji činilci blockchain sistema su distribuirani dnevnik transakcija, pametni ugovori (engl. *smart contracts*) [9] i pravila konsenzusa.

2.1. Distribuirana mreža

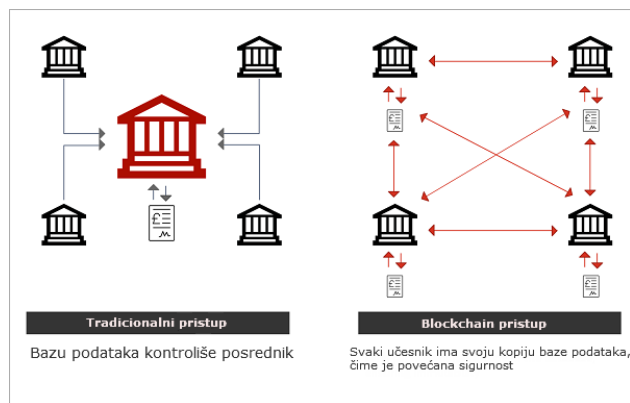
Komunikacija između aktera se odvija kroz *peer-to-peer* mrežu [1]. Razlikujemo privatne i javne blockchain mreže.

Kod javnih mreža svaki član ima prava upisa u bazu podataka, kao i čitanja iz nje. Potvrda transakcija se obavlja najčešće kroz dokaz posla (eng. *Proof of Work -PoW*) [2] ili dokaz posedovanja (eng. *Proof of Stake -PoS*) [3]. Privatne mreže podrazumevaju različita prava pristupa za čvorove unutar mreže. Identiteti učesnika u privatnoj mreži su poznati. Razlike između privatnih i javnih mreža su date u tabeli 1.

Tabela 1: Poređenje javnih i privatnih blockchain mreža

	Javna mreža	Privatna mreža
Pristup	Otvoreni pristup za čitanje i pisanje u bazu	Čitanje i pisanje u bazu zahteva dozvolu
Potvrda transakcije	Sporije	Brže
Održavanje ispravnosti mreže	Dokaz posla/dokaz posedovanja	Unapred određeni učesnici
Identitet učesnika	Anonimni	Identifikovani
Održavanje	Skupo	Jeftino
Broj čvorova	Milioni	Do nekoliko stotina
Primer	Bitcoin	Hyperledger Fabric

Blockchain tehnologija je karakteristična po tome što svi učesnici mreže posmatraju iste podatke i stvaraju se uslovi za nestajanjem posrednika (slika 1).

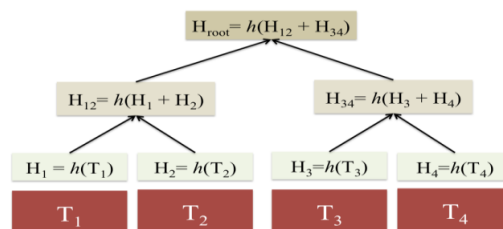


Slika 1: Tradicionalni i blockchain pristup

2.2. Struktura dnevnika transakcija

Svaka validna transakcija se upisuje u dnevnik transakcija i ne može se izbrisati ili promeniti. Transakcije se pre upisivanja u dnevnik smeštaju u blokove. Blokovi se uvezuju u linearnom poretku. Zaglavlje bloka referencira zaglavlje prethodnog bloka i sadrži koren Merkleovog stabla [5]. Osim zaglavlja postoji i sekcija podataka bloka gde se nalaze transakcije. Heš bloka se računa na osnovu transakcija unutar bloka i heša prethodnog bloka. U zavisnosti od implementacije ulazni podaci za heš funkciju mogu biti prošireni. Lanac blokova je struktura otporna na izmene. Promena transakcije u jednom bloku bi rezultovala drugačijim hešom tog bloka. Posledično, svi heševi blokova u lancu nakon njega bi bili promenjeni, a takvu aktivnost bi ostatak mreže prepoznao kao malicioznu i odbacio. Merkle stablo je struktura koja se koristi za heširanje podataka unutar bloka. Da bi se formiralo Merkle stablo, svaka transakcija se hešira čime nastaju listovi stabla. Na sledećem nivou se heširaju heševi koji se nalaze u listovima.

Postupak se nastavlja sve dok se ne dođe do korenskog heša (slika 2).



Slika 2: Merkle stablo

2.3. Konsenzus

Proces održavanja dnevnika transakcija sinhronizovanog preko mreže, odnosno osiguravanje da se dnevnik ažurira samo kada su transakcije odobrene od strane odgovarajućih učesnika, naziva se konsenzusom [7].

Kod tradicionalnog poslovanja načina poslovanja postoji problem poverenja i utvrđivanja koji podaci su validni. Ovakvi problemi su se rešavali uvođenjem posrednika, a to je često skupo i vremenski zahtevno. U ovu svrhu se kod blockchain-a implementiraju pravila za prihvatanje transakcija u dnevnik transakcija. Dokaz posla, dokaz posedovanja i praktična vizantijska tolerancija greške samo su neki od sistema za postizanje konsenzusa u blockchain mrežama.

2.3.1. Dokaz posla

Dokaz posla je sistem koji se koristi kako bi se garantovala ispravnost funkcionisanja javne blockchain mreže. Ideja je da čvor unutar mreže mora uložiti procesorsko vreme da bi poslao podatke ostatku mreže na verifikaciju. Odlika sistema je da čvor koji šalje podatke mora da reši kompleksan računarski problem, dok je za ostatak mreže veoma jednostavno da proveri da li je problem rešen ispravno. Prvi čvor koji pruži „dokaz posla” naziva se rudar i uvezuje novi blok na kraj lanca.

2.3.2. Dokaz posedovanja

Dokaz posedovanja predstavlja alternativu dokazu posla, ali ne zahteva kompjutersko računanje. Za razliku od dokaza posla sistema, dokaz posedovanja ne nameće takmičenje među čvorovima rudarima. Svaki čvor koji želi da validira blok, mora da uloži neku količinu kriptovalute¹. Mreža na slučajan način bira validatora uzimajući u obzir količinu kriptovalute koju čvorovi poseduju.

2.3.3. Praktična vizantijska tolerancija greške

Praktična vizantijska tolerancija greške (engl. *Practical Byzantine Fault Tolerance*) [6] je jedan od algoritama kojim se rešava problem „Vizantijskih generala” (engl. *Byzantine Generals Problem*) [8]. Pretpostavka je da neki čvorovi mogu biti nedostupni ili slati pogrešne podatke i poruke ne moraju stizati redosledom kojim su poslate. Algoritam zahteva da unutar mreže bude bar $3f+1$ čvorova, gde je f maksimalni broj neispravnih čvorova. Svaki čvor će sam za sebe odlučiti da li je poruka koju je primio korektna. Nakon toga, da bi poruka bila označena kao validna, potrebno je da $2f+1$ čvorova to potvrdi.

2.4. Pametni ugovori

Pametni ugovori se koriste da bi se podržala poslovna saradnja među entitetima koja je često kompleksna. Ovakvi ugovori su poslovni ugovori implementirani u programskom kodu i na taj način automatski osiguravaju ispunjenje ugovornih obaveza.

¹ Digitalno sredstvo za razmenu

3. HYPERLEDGER FABRIC I BITCOIN

Kada se zahteva veći nivo privatnosti i sigurnosti potrebno je koristiti privatne blockchain mreže. Zbog potreba da se blockchain prilagodi industriji kreiran je Hyperledger projekat pod pokroviteljstvom Linux fondacije. Unutar Hyperledger-a postoji Fabric platforma [7] koja podržava različite implementacije komponenata, da bi se prilagodila raznim potrebama poslovanja.

3.1. Hyperledger fabric

Prema [3], unutar Fabric blockchain mreže postoje 3 vrste čvorova: uređivači (eng. *orderers*), ravnopravni članovi (eng. *peers*) i klijenti. Ravnopravni članovi izvršavaju transakcije i održavaju dnevnik transakcija. Ravnopravni čvor može biti prihvatač (eng. *endorser*) ili potvrđivač (eng. *commiter*). Potvrđivači dobijaju validirane transakcije od strane uređivača i upisuju ih u dnevnik transakcija. Prihvatač proverava validnost transakcije, na osnovu unapred definisanih pravila. Uređivači pakuju transakcije u blokove, dok klijenti pozivaju transakcije.

Fabric pored dnevnika transakcija ima i bazu stanja (eng. *world state*) [7]. Baza stanja sadrži trenutne vrednosti unutar blockchain-a. Organizovana je kao kolekcija parova ključ-vrednost.

Koncept kanala (engl. *channels*) [7] koji Fabric uvodi služi kao privatna podmreža za komunikaciju dva ili više učesnika blockchain mreže. Svaki kanal ima svoj dnevnik transakcija i bazu stanja.

Pametni ugovori se u Fabric-u implementiraju programima koji se nazivaju kodovi lanca (engl. *chaincodes*) [7]. Ovi programi se pišu u jeziku Go.

Digitalna dobra (engl. *assets*) [7] su digitalne predstave materijalnih ili nematerijalnih svojina. Digitalna dobra se u Fabric-u čuvaju kao kolekcija parova ključ-vrednost.

Kada je u pitanju konsenzus, postoje tri faze, a to su saglasnost, uređivanje i validacija.

Pružalac servisa članstva (engl. *Membership Services Provider – MSP*) [7] je komponenta koja ima za cilj da pojednostavi upravljanje mehanizmima za identifikaciju i prijavljivanje korisnika.

3.2. Razlike HyperLedger Fabric-a i BitCoin-a

Pre svega Hyperledger Fabric i Bitcoin se razlikuju po nameni. Bitcoin mreža služi za razmenu kriptovalute, sa druge strane, Fabric teži da podrži zahteve industrije.

Kod Bitcoin-a svi čvorovi imaju istu ulogu u održavanju mreže, sa tim da su neki posvećeni rudarenju. Prema [10] postoje „debeli“ klijenti (engl. *full nodes*) koji čuvaju sve transakcije unutar Bitcoin blockchain-a, i „tanki“ klijenti (engl. *lightweight nodes*) koji čuvaju samo njima relevantne transakcije. U tabeli 2. prikazane su neke razlike između Bitcoin-a i Fabric-a.

Tabela 2: Razlike Hyperledger Fabric-a i Bitcoin-a

	Fabric	Bitcoin
Opis platforme	Blockchain opšte namene	Blockchain za transfer kriptovalute
Razvoj	Linux Foundation	Programeri Bitcoin-a
Valuta	/	Bitcoin (BTC)
Nagrade za rudarenje	Nema rudarenja	Bitcoin

Skladište podataka	Baza stanja i dnevnik transakcija (Kafka, PBFT) Promenljiv po potrebi	Dnevnik transakcija Dokaz posla
Konsenzus	Privatna	Javna
Otvorenost mreže	Korišćenjem kanala	/
Privatnost	Da, jezik Go	Rudimentarni, skript jezik
Pametni ugovori	Poznat	Anoniman

4. PRIMENA BLOCKCHAIN DISTRIBUIRANIH SISTEMA U OBLASTI FARMACIJE

U procesu dobavljanja lekova postoji mnogo aktera od proizvođača do krajnjih korisnika. Po nekim procenama 10% svih lekova na tržištu je lažno, a na internetu kupljenih čak 50%². Sastojci lekova se često nabavljaju iz više zemalja i proizvodi se mogu transportovati na razne načine od strane više kompanija. Ovo ostavlja mnogo prostora za ilegalne radnje.

Sa druge strane, veleprodajne i maloprodajne kompanije imaju česta neslaganja u podacima i proces njihovog usaglašavanja je vremenski veoma zahtevan.

Blockchain sistemi bi mogli da reše ili umanje pomenute probleme. Svaka transakcija bi morala biti odobrena od strane relevantnih aktera unutar mreže koji bi imali uvid u transakcije, a poslovne obaveze bi se automatski izvršavale primenom pametnih ugovora. Za realizaciju idejnog rešenja u vezi nabavke farmaceutskih proizvoda korišćen je alat Fabric Composer [4].

4.1. Korišćeni alati

Fabric Composer je radni okvir (engl. *framework*) koji olakšava razvoj i testiranje Hyperledger Fabric aplikacija. Composer je fokusiran na implementaciju poslovne logike i pruža jednostavan način da se modeluju učesnici unutar mreže sa pravima pristupa i transakcije među njima.

4.2. Opis sistema

Proizvodi se uvoze ili proizvode, a svaki lek mora dobiti dozvolu od strane regulatornog tela. U slučaju uvoza, uvoznik predstavlja primarnog distributera, koji proizvode distribuira veleprodajama ili maloprodajama. Maloprodajna kompanija može i direktno da poručuje od proizvođača, iako češće sarađuje sa veleprodajnim kompanijama. Transport rade kompanije od kojih se roba dobavlja ili kurirske službe, ali tako da se lekovi prevoze u posebnim uslovima. Krajnji korisnici proizvode nabavljaju od maloprodajnih kompanija.

U sistemu su identifikovani učesnici: krajnji korisnici, maloprodajne, veleprodajne, transportne kompanije i kontrolni akteri (agencija za lekove i medicinska sredstva i republički fond za zdravstveno osiguranje).

Učesnik u mreži je modelovan kao apstraktan tip koji se identifikuje preko adrese elektronske pošte i ima stanje na računu. Ostale grupe korisnika nasleđuju ovaj apstraktni tip. U slučaju korisnika koji predstavlja maloprodajnu ili veleprodajnu kompaniju dodaju se obeležja kompanije. Ova grupa korisnika u modelu ima i listu dolazećih narudžbenica kako bi pratili pristigle porudžbine. Pored toga oni poseduju i listu konkretnih proizvoda koji bi trebalo da pređu u njihov posed (naručeni proizvodi).

² Izvor: <http://www.pharmexec.com/fake-medications-real-solutions>

Što se tiče digitalnih dobara identifikovani su klase proizvoda, instance proizvoda i narudžbenice sa svojim stavkama. Model digitalnih dobara u Composer-u je dat na slici 3.

```
asset Product identified by productName{
  o String productName
  o String productBarcode
  o String productDescription
  o ProductType productType
  o Double productPrice optional
}
asset ProductInstance identified by productInstanceId {
  -->Product product
  o String productInstanceId
  -->NetworkParticipant owner
  o DateTime producedDateTime optional
  o Double productInstancePrice
  o ProductStatus status optional
}
asset OrderItem identified by orderItemId{
  o String desc optional
  o String orderItemId
  -->Product product
  -->ProductInstance [] instances
  o Double amount
}
asset Order identified by orderId{
  o String orderId
  o Double totalPrice optional
  o DateTime createTime optional
  o OrderStatus orderStatus default = "CREATED" optional
  --> OrderItem[] items
  -->NetworkParticipant sender
  -->NetworkParticipant receiver
}
```

Slika 3: Model digitalnih dobara u Composer-u

Kako je vrlo često u interesu da se zna odakle je svako pakovanje proizvoda došlo, potrebno je da se nivo identifikacije spusti u odnosu na tradicionalne poslovne sisteme. Svaki proizvod predstavlja klasu konkretnih proizvoda, gde bi konkretan proizvod predstavljao samo pakovanje. Prema tome, konkretan proizvod referencira klasu proizvoda kojoj pripada i pripada akteru unutar mreže. Narudžbenica referencira pošiljaoca, primaoca i sadrži listu stavki. Postoji mnoštvo operacija unutar sistema, a ovde će biti opisane one transakcije koje omogućuju transfer proizvoda među akterima.

SendOrder transakcija podrazumeva slanje narudžbenice akteru kojem je namenjena. Ako pošiljalac nije već slao tu narudžbenicu i ako ima dovoljno sredstava, narudžbenica se smešta u dolazeće narudžbenice primaoca.

ApproveAndPrepareOrder operacija proverava da li svaki traženi proizvod postoji u dovoljnoj meri kod primaoca narudžbenice. Ukoliko se ova provera prođe uspešno onda se svakoj stavki narudžbenice pridružuju instance proizvoda koje će biti dostavljene. Broj pridruženih instanci jednak je traženoj količini klase proizvoda kojoj pripada instanca.

ShipOrder operacija menja status instance proizvoda koji će se slati i smešta ih u dolazeće instance proizvoda naručioca.

DeliverOrder bi izvršavala posebna kurirska služba ili zaposleni u kompaniji od koje su i naručeni proizvodi. U ovom koraku se stanje računa naručioca umanjuje za iznos narudžbenice, a za isti taj iznos se uvećava stanje na računu pošiljaoca. Pored toga, menja se vlasnik instanci proizvoda.

Prava pristupa su definisana u posebnom fajlu. Svaki akter mreže ima uvid u klase proizvoda, a veleprodajne kompanije mogu davati posebne cene za određene partnere. Proizvođači ili uvoznici mogu da stave novi proizvod u promet, a regulatorna tela mogu da ga menjaju, odnosno da daju ili povuku dozvolu za proizvod. Svaka kompanija može da kreira narudžbenicu. Uvid u konkretnu narudžbenicu imaju samo pošiljalac i primalac.

5. ZAKLJUČAK

U ovom radu opisane su osnove blockchain distribuiranih sistema i predlog njihove primene za podršku lancu nabavke u farmaceutskoj industriji koji uključuje mnoge aktere. Proces praćenja lekova u lancu nabavke je složen, jer se neki lekovi uvoze, a sastojci lekova se mogu nabavljati iz više izvora. Saradnja maloprodajnih, veleprodajnih kompanija i proizvođača je jako intenzivna i sklona greškama zbog načina komunikacije. Prikazano idejno rešenje zasnovano na blockchain-u efikasno rešava pomenute probleme. Blockchain tehnologija je mlada i još uvek ne postoji dovoljno razvijena Internet zajednica gde ljudi razmenjuju iskustva. Iskustvo stečeno u ovom radu pokazuje da je Fabric platforma pogodna za primenu u ovoj oblasti. Nije jednostavno pokrenuti i testirati Fabric mrežu, što je otežavalo razvoj praktičnog rešenja. Kao budući rad, planirano je proširivanje funkcionalnosti i razvoj čitave aplikacije direktno nad Fabric platformom bez posredstva Fabric Composer-a.

6. LITERATURA

- [1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, October, 2008.
- [2] Proof of work, dostupno na: https://en.bitcoin.it/wiki/Proof_of_work, poslednji pristup 30.8.2017.
- [3] Proof of stake, dostupno na: https://en.bitcoin.it/wiki/Proof_of_Stake, poslednji pristup 30.8.2017.
- [4] Fabric Composer Documentation, dostupno na: <https://hyperledger.github.io/composer/>, poslednji pristup 20.9.2017.
- [5] R. C. Merkle, Protocols for public key cryptosystems, in *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pp. 122-133, April, 1980.
- [6] M. Castro and B. Liskov, Practical Byzantine fault tolerance, in *Proc. 3rd OSDI*, pp. 173-186, February, 1999.
- [7] Hyperledger Fabric Documentation, dostupno na: <https://hyperledger-fabric.readthedocs.io/en/latest/>, poslednji pristup 24.9.2017.
- [8] L. Lamport, R. Shostak, M. Pease, The Byzantine Generals Problem, *ACM Transactions on Programming Languages and Systems*. 4(3): pp. 382-401, July, 1982.
- [9] M. Swan, *Blockchain: Blueprint for a New Economy*, O'Reilly Media, Sebastopol, CA, USA, 2015.
- [10] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*, O'Reilly Media, Sebastopol, CA, USA, 2014.

Kratka biografija:



Aleksa Mirković rođen je u Šapcu 1993. god. Fakultet tehničkih nauka upisao je 2012. god. Diplomski rad iz oblasti Elektrotehnike i računarstva – računarske nauke i informatika, odbranio je 2016. god

BLOCKCHAIN DISTRIBUIRANI SISTEMI I NJIHOVA PRIMJENA U INDUSTRIJI VIDEO IGARA**BLOCKCHAIN DISTRIBUTED SYSTEMS AND THEIR APPLICATION IN VIDEO GAMES INDUSTRY**

Marina Nenić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu opisani su osnovni koncepti blockchain tehnologije i jedne od njenih implementacija, Hyperledger Fabric-a. Takođe, opisana je i aplikacija razvijena upotrebom Hyperledger Fabric platforme, sa ciljem upravljanja resursima unutar video igara.

Abstract – In this work we present the fundamentals of the blockchain technology and one of its popular implementations, Hyperledger Fabric. This work also describes an application implemented using the Hyperledger Fabric platform, which is developed for asset management in video games.

Ključne reči: Baze podataka, distribuirani sistemi, block-chain, Hyperledger Fabric.

1. UVOD

Osnovni mehanizam razmjene dobara kao i samog poslovanja čine transakcije. Čitavo savremeno poslovanje zasniva se na tom konceptu. Informativna vrijednost poslovnih transakcija je izuzetno velika jer transakcije posjeduju informacije potrebne za utvrđivanje vlasništva nad dobrima, odgovornosti pojedinaca, uvid u stanje finansijskih sredstava, a takođe pružaju i uvid u istoriju aktivnosti poslovanja te je njihovo zapisivanje i čuvanje od ključnog značaja. Sa napretkom tehnologije sve veći dio ljudskih aktivnosti počinje da se zasniva na upotrebi računara koji postaju nezamjenljivi u poslovanju i svakodnevnom životu pa tako i transakcije dobijaju svoj elektronski oblik. Tu na scenu stupaju baze podataka kao centralno mjesto za skladištenje transakcija.

Dugi niz godina u informacionim sistemima dominirale su centralizovane baze podataka, kod kojih se svi podaci čuvaju na jednom centralnom mjestu. Prednost ovakvog vida organizacije podataka ogleda se u tome što su takve baze lakše za upravljanje i održavanje, ali to sa sobom nosi i određene nedostatke. Jedan od glavnih nedostataka predstavlja opasnost od pada sistema. Nemogućnost pristupa podacima u centralizovanoj bazi značio bi prestanak funkcionisanja čitavog informacionog sistema. Rješenje ovog problema pronađeno je u decentralizaciji. Međutim, jedan bitan problem koji decentralizacija ne može da riješi je posredništvo. U većini slučajeva poslovne transakcije treba da prođu niz koraka da bi se utvrdila tačnost informacija i da bi bile potvrđene kao validne.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, docent.

Potrebno je pronaći organizaciju od povjerenja koja bi bila zadužena za takve provjere, što znači da se javlja potreba za posrednikom. Odličan primjer posrednika jesu banke. Posredništvo uvodi dodatne korake u procese poslovanja koji zahtijevaju dodatni utrošak vremena i sredstava. Zbog pomenutih problema koji značajno otežavaju proces poslovanja javlja se potreba za novim načinom organizacije i čuvanja podataka i transakcija. Javlja se potreba za tehnologijom koja bi obezbijedila veću dostupnost podataka i ukinula posredništvo, ali istovremeno zadržala neophodne funkcionalne karakteristike kao što su sigurnost, brzina i otpornost na zloupotrebe. Tehnologija blockchain-a (engl. *Blockchain* – lanac blokova) upravo ispunjava sve prethodno navedene zahtjeve [1]. Ova tehnologija unosi značajne promjene u dosadašnji pristup realizaciji transakcija, čini ih pristupačijim i javno provjerljivim. Distribuirana priroda omogućava lako kreiranje efikasnih i ekonomičnih poslovnih mreža koje ne zahtijevaju centralno kontrolno mjesto (banke, države) kao što je slučaj sa dosadašnjim načinom poslovanja.

Cilj ovog rada jeste upoznavanje sa blockchain tehnologijom i njenim najpoznatijim implementacijama, kao i primjena iste u industriji video igara. Za ispunjenje pomenutog cilja potrebno je realizovati nekoliko koraka. Najprije je potrebno izvršiti analizu teorijskih osnova blockchain tehnologije. Tu se prije svega misli na koncepte i mehanizme koji omogućavaju blockchain tehnologiji da ispuni prethodno navedene zahtjeve pristupačnosti, sigurnosti i otpornosti. Sledeći korak je proučavanje implementacija blockchain-a i izbor jedne od implementacija koja će biti osnov za izgradnju softverskog rješenja. U ovom radu izabran je *Hyperledger Fabric* [2]. Nakon toga potrebno je specificirati model koji će opisivati učesnike koji će koristiti softversko rješenje i resurse kojima će moći da rukuju. Model će biti specificiran upotrebom *Hyperledger Composer* alata, koji će takođe izvršiti automatsko mapiranje modela na generičke pametne ugovore koji se izvršavaju na blockchain mreži. Na kraju je potrebno implementirati korisničku aplikaciju koja će pozivati i koristiti funkcionalnosti pomenutih pametnih ugovora.

2. OSNOVNI POJMOVI

Blockchain je vrsta replicirane distribuirane baze podataka koja čuva digitalne transakcije. Organizovana je u formi jednostruko spregnute liste (lanac), gdje ulogu čvorova imaju blokovi koji sadrže podatke o transakcijama zaštićenim kriptografskim metodama. Iz takve strukture potiče naziv blockchain. Baza je replicirana na mreži koji je zasnovana na *peer-to-peer* arhitekturi [1].

2.1 Peer-to-peer arhitektura

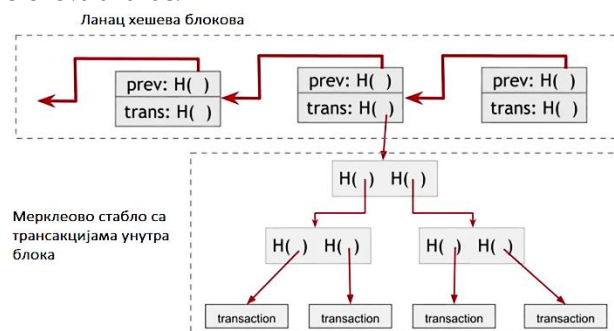
Čvorovi koji učestvuju u realizaciji transakcija posjeduju kopije svih zapisa, te stalno međusobno komuniciraju i sinhronizuju zapise što omogućava realizaciju digitalnih transakcija bez posrednika. Da bi se transakcija upisala u *blockchain* dnevnik transakcija (engl. *ledger*) potrebno je objaviti transakciju ostalim članovima. Svaki od članova po prijemu transakcije izvršava niz provjera da bi utvrdio da je ista validna. U *Bitcoin* implementaciji, član mreže koji prvi riješi određeni kompjuterski problem dobija pravo dodavanja novog bloka u lanac [1].

U mreži mogu postojati čvorovi sa različitim ulogama. Puni čvorovi (engl. *full nodes*) su čvorovi koji održavaju kompletnu kopiju dnevnika transakcija u svojoj memoriji i mogu da provjeravaju transakcije bez spoljne pomoći. Zadatak punih čvorova jeste da održavaju mrežu i vode računa da u lanac dospiju samo validne transakcije. Sa druge strane postoje tanki čvorovi (engl. *lightweight nodes*) koji prate samo one blokove koji se odnose na njima bitne transakcije. Tanki čvorovi se oslanjaju na pune kada je u pitanju sigurnost i validnost ostalih transakcija u lancu [3]. Pored navedenih čvorova postoje i čvorovi rudari (engl. *miners*) koji vrše dodavanje novih blokova u lanac. Pored različitih tipova čvorova postoje i različiti tipovi mreža. Neke mreže su dizajnirane tako da budu potpuno otvorene i anonimne, gdje bilo ko može da se priključi na mrežu i odlučuje o tome koje transakcije će ući u *block-chain* bazu. Takve mreže se nazivaju javne (engl. *permissionless*). Sa druge strane postoje i privatne mreže sa ograničenim pravima pristupa (engl. *permissioned*). Pošto je za učešće u odlukama u takvim mrežama potreban utvrđen identitet nema potrebe za utroškom resursa te su zahvaljujući tome ove mreže dosta pogodnije za industrijsku primjenu.

2.2 Struktura otporna na izmjene

Blockchain lanac predstavlja jednostruko spregnutu listu u kojoj je svaki blok sa prethodnim povezan preko heš pokazivača. Zahvaljujući heš pokazivačima ne postoji mogućnost brisanja ili izmjene podataka. Na primjer, neka je izmijenjena transakcija u bloku k . Pošto je sadržaj bloka k izmijenjen onda će i heš vrijednost čitavog bloka da se promijeni. U tom slučaju heš pokazivač iz bloka $k+1$ neće više pokazivati na k -ti blok i na taj način smo detektovali izmjenu. Da bi se uklonilo neslaganje između pokazivača iz bloka $k+1$ i sadržaja bloka k može se zamijeniti dati pokazivač ali ta izmjena predstavlja izmjenu sadržaja bloka $k+1$, što opet dovodi do neslaganja. Dakle, čuvanjem na sigurnom mjestu heš vrijednosti sa kraja lanca zapravo se garantuje da će čitava struktura *blockchain* baze biti otporna na izmjene. Heš pokazivači mogu se iskoristiti i u binarnim stablima i takva struktura naziva se Merkleovo stablo (eng. *Merkle tree*) [3]. Listovi, koji predstavljaju podatke, grupišu se u parove i za svaki par formira se struktura koja sadrži dva heš pokazivača, jedan za svaki od listova. Isti postupak se ponavlja za svaki sledeći nivo sve dok se ne stigne do jednog čvora koji predstavlja korijen stabla. Merkleova stabla iskorišćena su za organizovanje transakcija unutar bloka. Na slici 1. prikazana je struktura *blockchain* lanca [3]. Kao i kod lanca dovoljno je da zapamtimo samo heš pokazivač sa vrha stabla i na osnovu njega možemo da

garantujemo da se podaci unutar stabla nisu mijenjali. Pokazivač na koren smešta se u zaglavlje bloka zajedno sa heš pokazivačem na prethodni blok. U tom slučaju da bi se očuvala otpornost lanca na izmjene dovoljno je uzeti samo heš vrednost zaglavlja prilikom povezivanja blokova u lanac.



Slika 1. Struktura *blockchain* lanca

2.3 Konsenzus

Konsenzus predstavlja mehanizam pomoću kojeg se svi članovi dogovaraju o opšte prihvaćenom redoslijedu transakcija i blokova u lancu. Kod javnih mreža svaki član ima pravo učestvovanja u konsenzusu, dok je kod mreža sa pravima pristupa učestvovanje ograničeno na određen skup članova sa utvrđenim identitetima. Pošto se u javnim mrežama očekuje veliki broj anonimnih i nepouzdanih članova, mehanizam za postizanje konsenzusa treba da bude otporan na pokušaje pojedinaca da generišu više lažnih identiteta i na taj način većinskim prisustvom kontrolišu proces koncenzusa. Kod većine javnih mreža, taj problem je rešen zahtevanjem rješavanja zadatka zahtjevnog za izvršavanje, kao načina da se ostvari pravo dodavanja bloka. Taj pristup se naziva dokaz posla (eng. *proof-of-work*). Glavni cilj zadatka jeste da se pronađe slučajni broj (eng. *nonce*), koji zajedno sa zaglavljem bloka daje heš vrijednost određenih osobina. Ako heš vrijednost ne zadovoljava zadate osobine, bira se novi broj i ponavlja postupak. Osobine heš funkcija garantuju da ne postoji prečica koja bi omogućila lako nalaženje traženog broja. Član koji prvi pronađe odgovarajući broj, dobija pravo dodavanja narednog bloka u lanac. Blokovi dodati u *blockchain* mogu biti isključeni iz glavnog (najdužeg) lanca. Prilikom dodavanja bloka provjerava se validnost poslednjih blokova koji se nalaze na glavnom lancu i novi blokovi se po pravilu dodaju na poslednji validan blok [1]. Ako neki od članova uspije da doda blok u kom se nalaze zlonamjerne transakcije, taj blok će biti zanemaren od strane poštenih članova koji se neće nastaviti nadovezivati na njega. Osnovni nedostatak ovog pristupa jeste velika potrošnja energije, kao i brzina izvršavanja.

Mreže sa pravima pristupa mogu primijeniti drugačije mehanizme. Praktična vizantijska tolerancija greške je algoritam koji je nastao kao rješenje problema poznatog pod nazivom „Problem Vizantijskih Generala“ (*Byzantine Generals Problem*). Problem je dat u vidu priče u kojoj više divizija vizantijske vojske treba da napadne isti grad. Divizijama upravljaju generali od kojih neki mogu biti izdajnici. Da bi napad bio uspješan većina generala treba da donese odluku da napadne grad. Cilj ovog problema jeste da se osigura da će se odani generali donijeti istu odluku i da mali broj izdajnika neće biti u stanju da ih

dovede do donošenja drugačije odluke. Dokazano je da je to moguće samo ukoliko su jedna trećina generala, ili manje, izdajnici [4]. U slučaju *blockchain* tehnologije, članovi mreže predstavljaju generale a odluka o napadu je odluka o tome da li je neki podatak validan ili ne. Algoritam funkcioniše tako što svaki član nakon prijema poruke, bez ičijeg uticaja, provjerava validnost. Ako poruka dobije dovoljan broj potvrda validnosti biće upisana u *blockchain*. Ovaj način postizanja konsenzusa zahtijeva manji utrošak resursa i energije.

2.4 Pametni ugovori

Pametni ugovori (engl. *smart contracts*) predstavljaju dogovor, ili skup pravila koji usmjeravaju poslovne transakcije, kodiranih u nekom programskoj jeziku, koji se sami izvršavaju. U njima se definišu pravila ponašanja i odgovornosti stranaka, načini na koji se određene akcije moraju sprovesti kao i posljedice do kojih može doći u koliko nisu zadovoljeni uslovi definisani ugovorom. U slučaju ispunjenja dogovorenih uslova, ugovor se sam aktivira i izvršava operacije dogovorene prilikom kreiranja ugovora. Jedna od najvećih prednosti pametnih ugovora jeste da se samostalno izvršavaju i obezbjeđuju finalan skup ishoda koji garantuju da nikada neće doći do zabuna i neće postojiti potreba za sudskim postupcima.

3. HYPERLEDGER FABRIC

Problem skalabilnosti, nedostatak podrške za privatne i povjerljive transakcije, anonimnost učesnika u mreži i još mnogo toga činilo je prve implementacije *blockchain* tehnologije neupotrebljive u razvoju poslovnih aplikacija.

Da bi riješili pomenute probleme i približili *blockchain* tehnologiju industrijskoj upotrebi *Hyperledger* je razvio platformu pod nazivom *Fabric* [2]. *Hyperledger Fabric* uvodi inovaciju u vidu servisa članstva pomoću kojih članovi dobijaju provjerene identitete i pristup mreži. Uloga servisa članstva (engl. *membership services*) jeste da vrše autentifikaciju i vode računa o identitetima, privatnosti i pravima pristupa na mreži. Svaki član mreže mora da se registruje pomoću servisa članstva da bi dobio identitet i pravo izvršavanja transakcija. Servisi članstva nude tri osnovne funkcionalnosti: registraciju članova (engl. *Registration*), učlanjivanje (engl. *enrollment*) i izdavanje sertifikata za izvršavanje transakcija.

Pametni ugovori u *Fabric*-u nazivaju se *chaincode*-ovi. Svaki *chaincode* implementira interfejs koji ima tri metode koje se pozivaju po prijemu transakcija. Te metode su *Init*, *Query* i *Invoke* [5]. *Init* se poziva kada se *chaincode* prvi put smesti na *blockchain*. Prilikom smještanja, *chaincode* se šalje u vidu transakcije i proslijeđuje svim članovima mreže koji kreiraju nove *Docker* kontejnere u koje se ugrađuju *chaincode*-ovi. Kao što samo ime nagovještava, *Init* metoda se koristi sa sve inicijalizacije koje su potrebne *chaincode*-u. *Query* se poziva svaki put kada se pretražuju podaci. Transakcije koje samo pretražuju podatke ne pamte se u dnevniku transakcija. *Invoke* se poziva kada je potrebno izvršavanje konkretne poslovne logike. Pozivi *Invoke* metode čuvaju se kao transakcije koje se organizuju u blokove i smještaju na *blockchain*. Konsenzus je u *Fabric*-u je podijeljen u tri faze: odobravanje, uređivanje i validacija [6]. Odobravanje transakcije je određeno politikom odobravanja koja se definiše posebno za svaki *chaincode*. Klijent kreira transakciju i

šalje je određenom skupu čvorova koji koji zatim vrše simulaciju transakcije tako što je lokalno izvršavaju. Kao rezultat simulacije kreira se skup pročitanih (engl. *readset*) i izmenjenih resursa (eng. *writeset*). Za odobrene transakcije čvor šalje potvrdu da je transakcija odobrena, odnosno *transaction endorsed* poruku, u suprotnom, šalje se *transaction invalid* poruka. Transakcije sa dovoljnim brojem potvrda šalju se uređivačima putem *broadcast* metode. Faza uređivanja prihvata odobrene transakcije i uređuje ih u blokove nakon čega su spremne da se dodaju u dnevnik transakcija. Kada čvor primi blok od uređivača počinje proces validacije. Validacija se vrši da bi se osiguralo da od momenta odobravanja do momenta dodavanja u dnevnik transakcija nije došlo do promjene stanja korišćenih resursa. U koliko prođe sve provjere, transakcija se dodaje u dnevnik transakcije i vrijednosti iz skupa izmijenjenih resursa se primjenjuju na trenutno stanje resursa. U suprotnom transakcija se označava kao nevalidna, ipak upisuje u dnevnik transakcija ali se nastale izmjene ne primjenjuju [6].

4. OPIS SOFTVERSKOG RJEŠENJA

Za definisanje specifikacije *blockchain* mreže i generisanje klijentske aplikacije korišćen je *Hyperledger Composer* alat. Specifikacija *blockchain* mreže je proces koji se sastoji iz definisanja modela koji opisuje resurse unutar mreže, učesnike i operacije koje učesnici mogu izvršavati nad resursima, predstavljene u vidu transakcija. Pored modela potrebno je opisati logiku svake od transakcija koja je data u modelu i definisati prava pristupa nad resursima. Opis logike transakcija zadaje se putem *JavaScript* jezika i zapravo predstavlja poslovnu logiku sistema. Na slici 2. prikazan je model *blockchain* mreže koja služi za upravljanje resursima u video igrama.

<pre> * Game assets management network definition. namespace com.myapp enum ItemListingState { o IN_AUCTION o OWNED } participant Player identified by email { o String email o String firstName o String lastName o Double balance } asset Game identified by gameId { o String gameId o String name o String publisher } asset ItemType identified by typeId { o String typeId o String name o Double price --> Game game } asset Item identified by itemId { o String itemId o ItemListingState status --> ItemType itemType --> Player owner } asset ItemListing identified by listingId { o String listingId o Double minPrice o ItemListingState status --> Item item o Offer[] offers </pre>	<pre> transaction BuyTokens { --> Player buyer o Double quantity } transaction BuyItem { --> ItemType itemType --> Player buyer } transaction TradeItems { --> Player trader1 --> Player trader2 --> Item[] trader1items --> Item[] trader2items o Double trader1tokens o Double trader2tokens } transaction ListItem { --> Item item o Double bidPrice } transaction Offer { o Double bidPrice --> ItemListing listing --> Player player } transaction CloseBidding { --> ItemListing listing } </pre>
---	---

Slika 2. Model *blockchain* mreže za upravljanje resursima u video igrama

Definisani model sadrži jedan tip učesnika, četiri vrste resursa i šest različitih transakcija. Igrači su modelovani kao tip učesnika *Player* i oni predstavljaju jedine učesnike u mreži. Svaki igrač identifikovan je preko *email* adrese, ima ime, prezime i trenutno stanje računa, odnosno količinu tokena koju posjeduje. Tokeni predstavljaju

sredstvo plaćanja i mogu se koristiti prilikom kupovine raznih dobara (*items*) ili prilikom razmjene sa drugim igračima. Aplikacija podržava razmjenu dobara između različitih video igara te je iz tog razloga u modelu je definisan resurs *Game*. Svaka igra ima skup predefinisanih dobara koje igrači mogu da kupe. Ta predefina dobra modelovana su putem *ItemType* resursa. Pored *ItemType* definisan je i resurs *Item* koji predstavlja konkretnu instancu određenog *ItemType*-a. Drugim rečima, *ItemType* predstavlja apstraktan opis određenog *Item*-a u kom su opisani svi njegovi atributi, kao što su cena i video igra kojoj pripada to dobro. Svaka instanca resursa *Item* posjeduje atribut koji ukazuje na koji *ItemType* se odnosi i svog vlasnika koji predstavlja konkretnog igrača. Igrači mogu da vrše kupovinu novih dobara, direktnu razmjenu sa drugim igračima i da stavljaju dobra koje posjeduju na aukciju. Kupovina novih dobara vrši se isključivo u zamjenu za tokene i predstavljena je *BuyItem* transakcijama. Transakcija ima dva parametra, igrača tipa *Player* koji vrši kupovinu i klasu dobra, tipa *ItemType*, koji igrač želi da kupi. Sistem provjerava stanje računa igrača i ukoliko igrač posjeduje dovoljnu količinu tokena kreira se nova *Item* instanca kojoj se kao tip i vlasnik atributi dodijele prosljeđeni parametri transakcije. Igrači tokene mogu nabaviti na dva načina. Kupovina tokena novcem predstavljena je *BuyToken* transakcijom koja kao parametre prima igrača koji kupuje tokene i količinu tokena koju želi da kupi. Drugi način jeste prodaja dobara drugim igračima. To se takođe može postići na dva načina. Pri način je putem *TradeItems* transakcija. *TradeItems* transakcija predstavlja direktnu razmjenu između dva igrača. Transakcija kao parametre prima igrača koji učestvuju u razmjeni i za svakog igrača listu dobara i količinu tokena koje nudi u zamjenu za listu dobara i količinu tokena drugog igrača. Svi su parametri, osim igrača, opcioni. To znači da *TradeItems* transakcija podržava bilo koji od sledećih slučajeva: zamjena jednog ili više dobara jednog igrača za jedno ili više dobara drugog igrača, zamjena jednog ili više dobara jednog igrača za određenu količinu tokena drugog igrača, zamjena jednog ili više dobara jednog igrača za jedno ili više dobara i određenu količinu tokena drugog igrača, kao i opciju poklanjanja jednog ili više dobara ili tokena jednog igrača drugom igraču. Drugi način na koji igrači mogu nabaviti tokene jeste postavljanje dobara na aukciju. To se vrši putem *ListItem* transakcija, koje kreiraju nove instance *ItemListing* resursa. *ItemListing* kao atribut sadrži jedinstveni identifikator, minimalnu cijenu za koju dobro može da se proda, trenutni status aukcije, dobro koji se prodaje i niz ponuda za datu aukciju. Pravo izvršavanja *ListItem* transakcije ima samo onaj igrač koji je vlasnik dobra koji se stavlja na aukciju. Kupovina dobara koja se nalaze na aukciji vrši se isključivo putem ponuda, koje su predstavljene *Offer* transakcijama. *Offer* transakcija prima sledeće parametre: igrača, tipa *Player*, koji daje ponudu, aukciju za ono dobro koje taj igrač želi da kupi, tipa *ItemListing*, i količinu tokena koja predstavlja ponudu. Zatvaranje aukcije i prodaja dobara vrše se pozivom *CloseBidding* transakcije. Pravo izvršavanja *CloseBidding* transakcije, kao kod *ListItem* transakcije, ima samo igrač koji je vlasnik dobra koji je na aukciji. Prilikom *CloseBidding* transakcije sistem uzima najveću ponudu iz niza ponuda i igraču koji

je ponudio datu ponudu dodjeljuje dobro koji se nalazi na aukciji. Na osnovu specifikacije mreže, *Hyperledger Composer* mapira opisane transakcije na generičke pametne ugovore koji se nalaze na *Hyperledger Fabric blockchain* mreži i generiše *REST* servise i *Angular 2* korisničku aplikaciju koji poziva date pametne ugovore.

5. ZAKLJUČAK

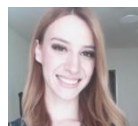
U ovom radu opisane su osnove *blockchain* distribuiranih sistema i predloženo je jedno softversko rješenje za njihovu primjenu u industriji video igara. *Hyperledger Fabric*, koji je izabran za razvoj softverskog rješenja, je implementacija *blockchain*-a koja je se fokusirala na zadovoljavanje velikog broja industrijskih zahtjeva i nudi jako širok spektar primjene. To sa sobom nosi određene nedostatke kao što je težina upotrebe. Pored toga što je i sam *blockchain* relativno nova i neistražena tehnologija, upotrebu tehnologije dodatno otežava i mnoštvo opcija koje *Hyperledger* nudi. Proces podešavanja i pokretanja mreže je jako mukotrpan i slabo dokumentovan, te predstavlja veliku prepreku u razvoju aplikacija. Tu se *Composer* pokazao kao jako koristan alat jer proces razvoja aplikacija i definisanja mreže diže na veći nivo apstrakcije. Umjesto da se definišu koncepti na nivou *blockchain*-a, kao što su čvorovi mreže i pametni ugovori, *Composer* pruža mogućnost definisanja mreže na nivou učesnika i transakcija što je i korisnicima i programerima dosta poznatije i prirodnije. Međutim ni *Composer* nije idealno rješenje jer predstavlja alat koji je još uvijek u procesu razvoja i ne pokriva u potpunosti sve funkcionalnosti koje *Hyperledger Fabric* nudi.

Iz tog razloga, planovi za dalji rad uključuju razvoj određenih funkcionalnosti softverskog rješenja direktno na *Fabric*-u, kao što su definisanje pametnih ugovora i politika odobravanja, kao i povezivanje implementiranog softverskog rješenja sa aplikativnim programskim interfejsima (engl. *Application Programming Interface - API*) konkretnih video igara.

6. LITERATURA

- [1] Nakamoto S., *Bitcoin: A Peer-to-Peer Electronic Cash System*, dostupno na: <https://bitcoin.org/bitcoin.pdf> poslednji pristup 25.09.2017.
- [2] *Hyperledger Fabric documentation*, dostupno na: <https://hyperledger-fabric.readthedocs.io/en/latest/>, poslednji pristup 25.09.2017.
- [3] Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S., *Bitcoin and Cryptocurrency Technologies*, Princeton University Press, 2016.
- [4] Baliga A., *Understanding Blockchain Consensus Model*, <https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf>, poslednji pristup 25.09.2017.
- [5] *Learn Chaincode*, <https://github.com/IBM-Blockchain-Archive/learn-chaincode>, poslednji pristup 25.09.2017.
- [6] *Next Consensus Architecture Proposal*, <https://github.com/hyperledger/fabric/blob/master/proposals/r1/Next-Consensus-Architecture-Proposal.md#32-transaction-evaluation-against-endorsement-policy>, poslednji pristup 25.09.2017.

Kratka biografija:



Marina Nenić rođena je u Brčkom, Bosna i Hercegovina, 1994. god. Fakultet tehničkih nauka upisala je 2012. god. Osnovne akademske studije završila je 2016. god.

**ДИГИТАЛНО УПРАВЉАЊЕ СИНХРОНИМ РЕЛУКТАНТНИМ МОТОРОМ СА
ДАВАЧЕМ ПОЛОЖАЈА НА ВРАТИЛУ****DIGITAL CONTROL OF SYNCHRONOUS RELUCTANCE MOTOR WITH POSITION
SENSOR ON THE SHAFT**Стефан Немет, Дарко Марчетић, *Факултет техничких наука, Нови Сад***Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО**

Кратак садржај – Због повећаног интересовања за синхроне релуктантне моторе, у овом мастер раду су анализирани карактеристике перформанси овог мотора као и различити типови контроле. Показан је утицај засићења на перформансе мотора и како се засићење одражава на примењене типове контроле. У раду је размотрена примена синхроних релуктантних мотора, његове предности и мане у односу на друге моторе, математички модел, основне карактеристике ових мотора и упоређени су различити типови контроле. Реализовано је дигитално управљање са максималним моментом по амперу са и без уважавања засићења.

Abstract – Due to increased interest in synchronous reluctance motors, in this master thesis were analyzed performance characteristics of this motor as well as different types of controls. The effect of saturation on the performance of the motor is shown and how the saturation reflects on the applied control types. The paper examines the application of synchronous reluctance motors, its advantages and disadvantages to other motors, the mathematical model, the basic characteristics of these motors, and different types of controls are compared. Digital control with maximum torque per amper was realized with and without respect for saturation.

Кључне речи: Синхрони релуктантни мотор, максимални момент по амперу, Дигитално управљање, SynRM, МТПА, Digital Control

1. УВОД

Синхрони мотори данас се широко користе у индустријским апликацијама и апаратима. Обе, синхрона машина са површински постављеним сталним магнетима (ен. *Surface Permanent Magnet Synchronous Machine* – SPMSM) и синхрона машина са унутрашње постављеним сталним магнетима (ен. *Interior Permanent Magnet Synchronous Machine* – IPMSM) због своје високе ефикасности замењују асинхроне машине (енгл. *Induction machine* – IM). Овај тренд је најочигледнији за моторе релативно мале снаге мање од 1 kW. Последњих година синхрони релуктантни мотори (енгл. *Synchronous Reluctance Motor* – SynRM) су привукли пажњу.

НАПОМЕНА:

Овај рад је проистекао из мастер рада чији ментор је био др Дарко Марчетић, ред. проф.

SynRM захтева синусоидну побуду и ствара скоро константан моменат, стога немају проблеме везане за прекидачки релуктантни мотор (без буке, мање пулзације момента, итд.). Такође, скорашњи трендови у дизајну SynRM мотора, алгоритам оптималног управљања, доводе до смањења трошкова SynRM, али и повећања његове ефикасности. Као резултат, SynRM добија предност у трошковима, издржљивости и излазној снази у поређењу са IM и PMSM [1].

SynRM постаје важна алтернатива у векторски контролисаним погонима. Поред своје једноставне и робусне структуре, предности модерних SynRM у поређењу са другим наизменичним моторима су:

- Одсуство роторског намотаја даје већу ефикасност и већу температуру (тј. већи номинални моменат за дати ред величине) него код IM.
- Нижи производни трошкови у поређењу са PMSM. Брз одзив и контрола слабљења поља могу се лако постићи без магнета у ротору.
- Уобичајено је мања пулзација момента, вибрација и шум у поређењу са прекидачким релуктантним мотором.

Све ове предности SynRM привлаче индустрију и академце, али још увек постоје проблеми који треба размотрити у SynRM, а један од њих је моделирање магнетске засићености. SynRM често раде са екстремном засићености како би се постигла велика густина момента (момент по амперу), а индуктивности варирају као функција струја (или флуксева). Такође је значајна унакрсна засићеност индуктивности. Дакле, у многим апликацијама је потребан модел са уваженом сатурацијом, као што је контрола покрета без сензора и оптимизација ефикасности [1].

Векторско управљање струјом у роторском референтном систему може се посматрати као зрела техника у контроли ротационих наизменичних машина. Међутим, иако једноставна и интуитивна, контрола са повратном спрегом (енгл. *Feedback control*) струје d и q осе на основу идеалног модела има одређене недостатке, а који су посебно везани за велике брзине, велика оптерећења или област слабљења поља, због губитака у статору/ротору и нелинеарности као што је засићење флукса [3].

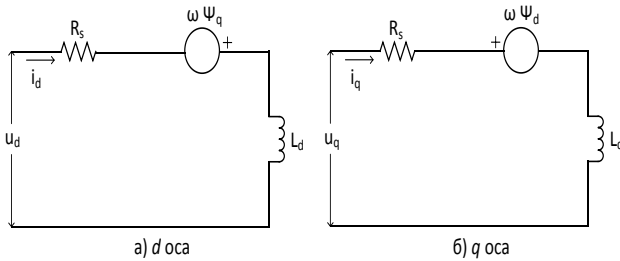
Прецизно процењене/измерене вредности обе индуктивности су битне како због моделирања карактеристике засићења тако и због фактора истурености ξ , чија нетачност (непрецизност) има утицај на кључне критеријуме перформанси алгоритма контроле који се заснивају на контроли струјног угла [4].

Због велике промене параметара са повећањем оптерећења, поготово индуктивности подужне осе, параметри регулатора се такође мењају у великом распону вредности, а сами регулатори су нелинеарни. Зато су параметри пропорционално-интегралних регулатора струје и брзине прорачунати са променом индуктивности, а на графицима је показано да се добијају аperiодични одзиви на одскочну промену референце. Индуктивност је представљена у функцији струје, а вредности се ажурирају или преко look-up табеле или се у програму аналитички израчунавају на основу апроксимираних функција.

2. МАТЕМАТИЧКИ МОДЕЛ У DQ КООРДИНАТНОМ СИСТЕМУ

Трансформацијом у ротирајући координатни систем везан за ротор добија се модел статора у коме ће релевантне променљиве у устаљеном стању имати константне вредности.

Еквивалентна шема синхроног релуктантног мотора приказана је у dq координатном систему на слици 1. На слици R_s представља отпорност статорског намотаја, док L_d и L_q представљају индуктивности у подужној d и попречној q оси.



Слика 1. Еквивалентна шема SynRM у dq координатном систему

Напонске једначине за SynRM у dq домену исте су као и за све наизменичне машине са роторским референтним координатним системом:

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = R_s \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \Psi_d \\ \Psi_q \end{bmatrix} + \omega \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \Psi_d \\ \Psi_q \end{bmatrix}, \quad (1)$$

где су u_d и u_q dq компоненте вектора статорског напона, i_d и i_q dq компоненте вектора статорске струје, ω је електрична угаона брзина, а Ψ_d и Ψ_q су компоненте вектора флуksа које се рачунају на основу следећих једначина:

$$\Psi_d = L_d i_d, \quad (2)$$

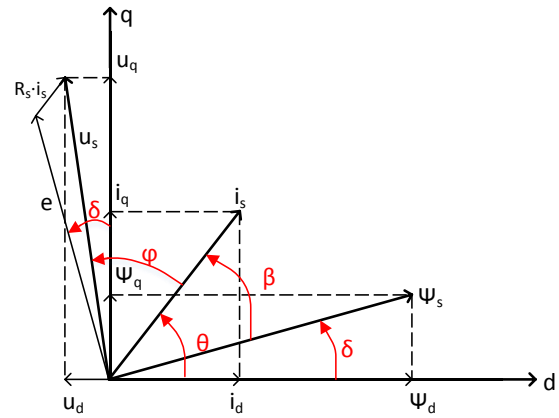
$$\Psi_q = L_q i_q. \quad (3)$$

Ако се у напонским једначинама узме да су индуктивности константне, добијају се напонске једначине са статичким индуктивностима:

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = R_s \begin{bmatrix} R_s & -\omega L_q \\ \omega L_d & R_s \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} L_d & 0 \\ 0 & L_q \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} i_d \\ i_q \end{bmatrix} \quad (4)$$

Векторски дијаграм, који описује математички модел у dq домену, је приказан на слици 2.

Ако се напонске једначине распишу и уважи да се и индуктивности мењају у времену, добијају се напонске једначине са динамичким индуктивностима:



Слика 2. Векторски дијаграм SynRM у dq координатном систему

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} R_s & -\omega L_q \\ \omega L_d & R_s \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} L'_d & 0 \\ 0 & L'_q \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} i_d \\ i_q \end{bmatrix}, \quad (5)$$

где су динамичке индуктивности дате изразима:

$$L'_d = L_d + i_d \frac{dL_d}{di_d}, \quad L'_q = L_q + i_q \frac{dL_q}{di_q} \quad (6)$$

Електромагнетни моменат зависи од истурености полова ротора, тј. од разлике индуктивности или фактора истурености $\zeta = L_d/L_q$, и производа dq компоненти вектора струје статора. На основу слике 1 моменат се може изразити на следећи начин:

$$T_e = \frac{3}{2} p (L_d - L_q) i_d i_q. \quad (7)$$

Механичке једначине кретања су:

$$J \frac{d\omega_m}{dt} = T_e - T_L \quad (8)$$

$$\frac{d\theta_m}{dt} = \omega_m \quad (9)$$

где су T_e и T_L електромагнетни момент и момент оптерећења, респективно, а ω_m и θ_m су механичка угаона брзина и механички угао, респективно.

Електричне вредности угаоне брзине и угла се добијају на следећи начин:

$$\omega = p \cdot \omega_m, \quad \theta = p \cdot \theta_m \quad (10)$$

па се онда механичка једначина кретања може изразити и на следећи начин:

$$\frac{d\omega}{dt} = \frac{p}{J} (T_e - T_L) \quad (11)$$

3. РЕАЛИЗАЦИЈА ДИГИТАЛНО УПРАВЉАНОГ SYNRM ПОГОНА

Треба имати на уму да, иако је контрола често дефинисана као векторска контрола, то заправо није векторска контрола јер нема фиксног угла између вектора статорске струје и вектора флуksа статора.

Алгоритам максималног момента по амперу, MTPA (Maximum Torque Per Ampere) или MTC (Maximum Torque Control), омогућава максимални моменат за дату амплитуду струје. Помоћу MTPA стратегије могуће је смањити губитке у баку SynRM. MTPA повећава ефикасност у поређењу са стандардном шемом управљања, али ефикасност не може бити максимална са MTPA јер се не прави баланс између губитака у баку и губитака у гвожђу, него се само

умањују губици у бакру. [1] *MTPA* је прихватљив и зато што може издржати високе температуре као и високе центрифугалне силе [5].

Оптималан струјни угао θ за *MTPA* зависи од момента машине али је скоро независан од брзине машине. Ово може бити главни разлог зашто је *MTPA* контрола најзанимљивија врста контроле у литератури, поред тога што је *MTPA* лакше имплементирати у контролној шеми. Код нижих брзина, губици у гвожђу су занемарљиви у поређењу са губицима у бакру. Према томе, тип контроле *MTPA*, који смањују губитке у бакру, је популаран. Ако је магнетна zasiћеност занемарена, струјни угао θ од 45 степени (у односу на d -осу) даје *MTPA* радну тачку.

Међутим, магнетна zasiћеност и варијације параметара обично доводе до оптималног угла струје θ већег од 45° степени. *MTPA* струјни угао θ за ниски моменат 25% од номиналног, је око 50° што је врло близу идеалном *MTPA* струјном углу од 45°. Међутим, оптимална вредност се повећава и на 65° за номиналне вредности момента [6].

У наставку је дато извођење *MTPA* струјног угла θ , при чему је занемарено zasiћење. При одређивању компоненти вектора струје статора за тип контроле *MTPA* полази се од израза за електромагнетни моменат.

Трансформацијом вектора струје статора из роторском референтног система у статорски и уврштавањем у израз за моменат добија се израз за моменат у функцији амплитуде статорске струје:

$$i'_s = i_s \cdot e^{-j\theta_r} = |i_s| \cdot e^{j(\theta+\theta_r)} \cdot e^{-j\theta_r} = |i_s| \cdot e^{j\theta} \quad (12)$$

$$T_e = \frac{3}{2} p \cdot (L_{sd} - L_{sq}) \cdot i_{sd} \cdot i_{sq} \quad (13)$$

$$T_e = \frac{3}{4} p \cdot (L_{sd} - L_{sq}) \cdot |i_s|^2 \cdot \sin(2\theta) \quad (14)$$

где је i_s амплитуда статорске струје.

При прорачуну максималног момента за дату струју, треба извод момента по количнику I_q/I_d или по струјном углу θ изједначити са нулом:

$$\frac{dT_e}{d\theta} = \frac{3}{4} p \cdot (L_{sd} - L_{sq}) \cdot |i_s|^2 \cdot 2 \cdot \cos 2\theta = 0 \quad (15)$$

$$\rightarrow \cos 2\theta = 0 \rightarrow \theta = 45^\circ.$$

Уврштавањем услова за *MTPA* у следеће једначине добијају се компоненте вектора струје статора за тип контроле *MTPA*:

$$i_{sd_ref} = \sqrt{\frac{|T_{e_ref}| \cdot \cot \theta}{\frac{3}{2} p \cdot (L_{sd} - L_{sq})}} = \sqrt{\frac{|T_{e_ref}|}{\frac{3}{2} p \cdot (L_{sd} - L_{sq})}} \quad (16)$$

$$i_{sq_ref} = \text{sign}(T_{e_ref}) \cdot i_{sd_ref} \cdot \tan \theta = \text{sign}(T_{e_ref}) \cdot i_{sd_ref} \quad (17)$$

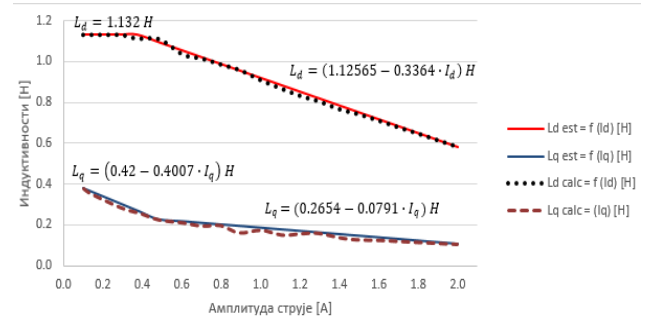
Ово је пожељна радна тачка ако се желе минимизирати губици у бакру машине или ако је ограничена излазна струја спрам снаге претварача [3]. Оваквим избором једнаких референтних вредности струја минимизира се максимална струја.

Да би се уважило zasiћење при прорачуну максималног момента за дату струју, треба извод момента по количнику I_q/I_d или по струјном углу θ изједначити са нулом:

$$\frac{dT_e}{d\theta} = \frac{3}{2} (L_d - L_q) I_s^2 \cdot 2 \cdot \cos 2\theta + \frac{3}{2} \left(\frac{dL_d}{d\theta} - \frac{dL_q}{d\theta} \right) I_s^2 \sin 2\theta \quad (18)$$

$$\frac{dT_e}{d\theta} = \frac{3}{2} I_s^2 \left[(L_d - L_q) \cdot 2 \cdot \cos 2\theta + \left(\frac{dL_d}{d\theta} - \frac{dL_q}{d\theta} \right) \sin 2\theta \right] \quad (19)$$

Да би се нашли први изводи индуктивности по струјном углу θ треба индуктивности представити у функцији dq компоненти струја тј. у функцији амплитуде статорске струје и струјног угла θ . На основу прорачунатих индуктивности (L_{d_calc} , L_{q_calc}), добијених на основу напонских једначина мерењем напона и струја, ове криве се могу апроксимирати са правима, слика 3.



Слика 3. Криве индуктивности апроксимирани са по две праве

На основу права, са којима се апроксимирају индуктивности (L_{d_est} , L_{q_est}), могу се израчунати први изводи ових индуктивности по струјном углу θ , јер ове индуктивности зависе од dq компоненти струја, тј. од амплитуде статорске струје I_s и струјног угла θ :

$$L_d = 1.132 H \quad \text{за } I_d \leq 0.37131 A,$$

$$\frac{dL_d}{d\theta} = 0 \quad \text{за } I_d \leq 0.37131 A,$$

$$L_d = (1.12565 - 0.3364 \cdot I_s \cos \theta) H \quad \text{за } I_d > 0.37131 A,$$

$$\frac{dL_d}{d\theta} = 0.3364 \cdot I_s \sin \theta \quad \text{за } I_d > 0.37131 A,$$

$$L_q = (0.42 - 0.4007 \cdot I_s \sin \theta) H \quad \text{за } I_q \leq 0.48116 A, \quad (20)$$

$$\frac{dL_q}{d\theta} = -0.4007 \cdot I_s \cos \theta \quad \text{за } I_q \leq 0.48116 A,$$

$$L_q = (0.2654 - 0.0791 \cdot I_s \sin \theta) H \quad \text{за } I_q > 0.48116 A,$$

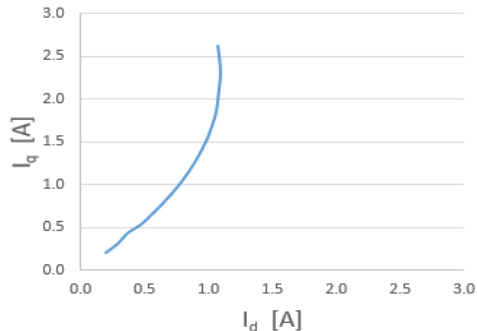
$$\frac{dL_q}{d\theta} = -0.0791 \cdot I_s \cos \theta \quad \text{за } I_q > 0.48116 A.$$

Када се ови изрази уврсте у једначину (19) и када се ова једначина за први извод момента по струјном углу изједначи са нулом добијају се струјни углови θ који задовољавају ову једначину, дати у табели 1.

Табела 1. Прорачун МТРА струјног угла θ са уваженим засићењем

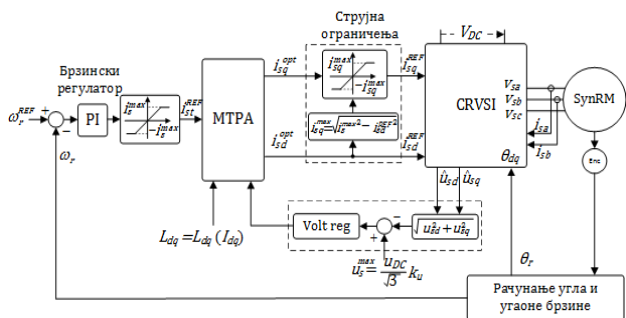
$L_d - L_q$ [H]	I_{s_ref} [A]	I_{s_amp} [A]	$dT_e/d\theta$	θ [rad]	θ [°]	I_d [A]	I_q [A]	T_{e_sat} [Nm]
0.766	0.2	0.283	0	0.811	46.46	0.19	0.21	0.09
0.829	0.3	0.424	0	0.820	47.00	0.29	0.31	0.22
0.867	0.4	0.566	0	0.869	49.77	0.37	0.43	0.41
0.866	0.5	0.707	0	0.847	48.54	0.47	0.53	0.64
0.835	0.6	0.849	0	0.863	49.43	0.55	0.64	0.89
0.810	0.7	0.990	0	0.879	50.35	0.63	0.76	1.17
0.785	0.8	1.131	0	0.896	51.34	0.71	0.88	1.47
0.762	0.9	1.273	0	0.914	52.36	0.78	1.01	1.79
0.732	1	1.414	0	0.934	53.52	0.84	1.14	2.10
0.705	1.1	1.556	0	0.955	54.73	0.90	1.27	2.41
0.680	1.2	1.697	0	0.977	55.98	0.95	1.41	2.72
0.658	1.3	1.838	0	0.999	57.23	0.99	1.55	3.04
0.630	1.4	1.980	0	1.024	58.65	1.03	1.69	3.29
0.607	1.5	2.121	0	1.048	60.02	1.06	1.84	3.55
0.575	1.6	2.263	0	1.076	61.64	1.07	1.99	3.69
0.550	1.7	2.404	0	1.102	63.14	1.09	2.14	3.85
0.530	1.8	2.546	0	1.126	64.52	1.10	2.30	4.00
0.502	1.9	2.687	0	1.154	66.13	1.09	2.46	4.03
0.477	2	2.828	0	1.181	67.68	1.07	2.62	4.02

На основу таб. 1 нацртан је график, приказан на слици 4, на ком су приказане dq компоненте статорске струје када је код МТРА алгоритма управљања уважено засићење. Подсећања ради, када се занемари утицај засићења код МТРА су струје d и q компоненти једнаке, тј. струјни угао је 45° . На основу графика може се закључити да са повећањем амплитуде статорске струје (оптерећења), вредност струјног угла са 45° расте према 70° .



Слика 4. dq компоненте статорске струје

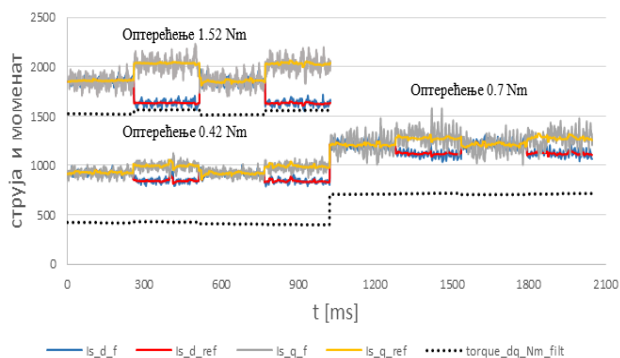
На слици 5 је приказан дигитално управљан SynRM погон, са енкодером на вратилу, код ког је примењен овај МТРА алгоритам контроле.



Слика 5. Дигитално управљан SynRM применом МТРА алгоритма [1]

4. ЕКСПЕРИМЕНТАЛНИ РЕЗУЛТАТИ

МТРА алгоритам описан у претходном поглављу је експериментално проверен на расположивом прото-типу векторског погона са SynRM. Резултати су приказани за МТРА алгоритам који не уважава засићење магнетних кола и за МТРА алгоритам са уваженим засићењем, слика 6, а на слици 7 су упоређени. Оба алгоритма су тестирана на две исте брзине (500 о/мин при оптерећењу 0.42 Nm и 1.52 Nm и 1000 о/мин при оптерећењу 0.7 Nm.



Слика 6. МТРА без и са уваженим засићењем на 500 и 1000 о/мин

500 о/мин са оптерећењем 0.42 Nm				
Брзински регулатор	Линеарни $P_{I_\omega} - I_\omega^2$	Нелинеарни $P_{I_\omega} - I_\omega$		
Струјни регулатор	Линеарни $P_{I_d} - I_d = I_q$	нелинеарни $P_{I_d} - I_d, I_q = f(I_d, L_d)$	Линеарни $P_{I_q} - I_d = I_q$	нелинеарни $P_{I_q} - I_d, I_q = f(I_d, L_q)$
$I_{m_average}$ [fasc]	925.729	923.225	919.484	917.050
$I_{m_average}$ [A]	0.399	0.398	0.397	0.396

500 о/мин са оптерећењем 1.52 Nm				
Брзински регулатор	Линеарни $P_{I_\omega} - I_\omega^2$	Нелинеарни $P_{I_\omega} - I_\omega$		
Струјни регулатор	Линеарни $P_{I_d} - I_d = I_q$	нелинеарни $P_{I_d} - I_d, I_q = f(I_d, L_d)$	Линеарни $P_{I_q} - I_d = I_q$	нелинеарни $P_{I_q} - I_d, I_q = f(I_d, L_q)$
$I_{m_average}$ [fasc]	1856.838	1843.155	1855.500	1840.643
$I_{m_average}$ [A]	0.801	0.795	0.801	0.794

1000 о/мин са оптерећењем 0.7 Nm				
Брзински регулатор	Линеарни $P_{I_\omega} - I_\omega^2$	Нелинеарни $P_{I_\omega} - I_\omega$		
Струјни регулатор	Линеарни $P_{I_d} - I_d = I_q$	нелинеарни $P_{I_d} - I_d, I_q = f(I_d, L_d)$	Линеарни $P_{I_q} - I_d = I_q$	нелинеарни $P_{I_q} - I_d, I_q = f(I_d, L_q)$
$I_{m_average}$ [fasc]	1213.501	1211.761	1207.305	1204.544
$I_{m_average}$ [A]	0.524	0.523	0.521	0.520

Слика 7. Поређење МТРА алгоритама

5. LITERATURA

- [1] Darko P. Marčetić, „Mikroprocesorsko upravljanje energetskim pretvaračima“, Fakultet tehničkih nauka, Novi Sad, 2014.
- [2] Zengcai Qu, Toni Tuovinen, Marko Hinkkanen, „Inclusion of Magnetic Saturation in Dynamic Models of Synchronous Reluctance Motors“, Aalto University, Finland, IEEE, 20th International Conference on Electrical Machines (ICEM), September 2012.
- [3] Jae Do Park, „Modeling And Control Of A High-Speed Solid-Rotor Synchronous Reluctance Flywheel Motor/Generator“, doctoral thesis, The Pennsylvania State University, USA, 2007.
- [4] R.E. Betz, „Teoretical aspects of control of synchronous reluctance machines“, IEE PROCEEDINGS-B, Vol. 139, No. 4, pp. 355-364, July 1992.
- [5] M.N. Ibrahim, E.M. Rashad, P. Sergeant, „Steady-State Analysis and Stability of Synchronous Reluctance Motors Considering Saturation Effects“, IEEE 10th International Symposium On Diagnostics For Electric Machines, Power Electronics And Drives (SDMPED), pp.345-350, 2015.
- [6] Reza Rajabi Moghaddam, „Synchronous Reluctance Machine (SynRM) in Variable Speed Drives (VSD) Applications“, doctoral thesis, Royal Institute of Technology, Stockholm, Sweden, 2011.

Кратка биографија:



Стефан Немец рођен је у Сремској Митровици 1993. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Енергетска електроника и електричне машине одбранио је септембра 2017. год.

MODIFIKACIJA PBRC ALGORITMA SA PRIMENAMA U PROBLEMIMA ADAPTIVNE REGRESIJE**MODIFICATION OF THE PBRC ALGORITHM WITH APPLICATIONS IN ADAPTIVE REGRESSION**Vukan Turkulov, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – Predviđanje izlaza dinamičkih sistema se često realizuje pomoću algoritama obučениh na osnovu podataka prikupljenih iz prošlosti. Međutim, prikupljanje ovakvih podataka za obuku je nekada skupo u pogledu utrošenog vremena i drugih resursa, zbog čega su podaci za obuku u takvim sistemima sporadično dostupni. Takođe, mnogi sistemi su vremenski promenljivi, te se zahteva neki vid adaptacije u algoritmu. U ovom radu predložena je jedna adaptivna metoda vršenja estimacije izlaza procesa u realnom vremenu na osnovu retkih periodičnih podataka za obuku.

Abstract – The output of a dynamic system is frequently estimated using learning-based algorithms which are trained on some previous data. However, acquiring such training data is sometimes costly in terms of time and other resources, making the training data periodically available. Moreover, many systems are time-variant and thus some form of adaptation is required. This paper presents an adaptive real-time estimation method based on sparse periodic training data.

Ključne reči: Regresija, veštačka neuronska mreža, adaptacija, estimacija

Key words: Regression, artificial neural network, adaptation, estimation

1. UVOD

Nekada je bitno pratiti režime rada procesa i predviđati vrednosti nekih veličina koje ne možemo uvek direktno meriti. Veštačke neuronske mreže (VNM) su čest izbor za estimaciju na osnovu već postojećih podataka, ali mogu biti problematične kada ti podaci redovno pristižu i zahtevaju da se mreža iznova obučava. Osim toga, VNM su dobre za previđanje veličina u okviru jednog režima rada procesa, ali imaju lošije performanse kada procesi menjaju režime rada. Takođe, u pojedinim procesima su očekivane blage promene nekih veličina u okviru jednog režima rada. Ukoliko je neophodno pratiti takve promene, algoritmi mogu sadržati neki vid adaptacije ulaznih podataka pre nego ih proslede samom estimatoru. Problem ćemo formalnije definisati kao estimaciju neke veličine y dinamičkog sistema čija dinamika nam nije direktno poznata. Periodično dobijamo vrednosti pojedinih veličina iz sistema. Taj skup ćemo označiti

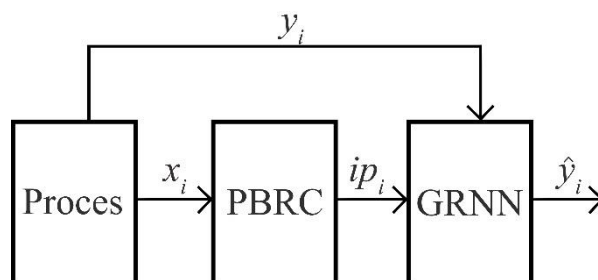
vektorom x_i , a odgovarajuću periodu T_x . Periodom T_y dobijamo vrednosti tražene veličine y , gde je $T_y > T_x$. Na sličan način možemo definisati i problem u kom su periode T_x i T_y jednake, ali dobijene vrednosti tražene veličine y kasne u odnosu na dobijanje odgovarajućih veličina x . Drugim rečima, u trenutku i dobijamo vrednost y_{i-k} , gde je $k \geq 1$.

2. ALGORITAM

Predloženo rešenje se sastoji iz dve celine. Prvu celinu čini algoritam koji detektuje i prati režime rada procesa, te u svakom trenutku daje informacije o tome koliko trenutno ponašanje procesa liči na sve do sada viđene režime rada. Te informacije zovemo informacionim potencijalima, kao što je predloženo u [1]. Osim toga, prva celina vrši adaptaciju viđenih režima rada ukoliko se sami režimi promene u dozvoljenim granicama. Ovaj algoritam je realizovan kao modifikovana verzija *Potential-Based Recursive Clustering* (PBRC) algoritma opisanog u [2] i [3].

Zadatak druge celine je da na osnovu do sada viđenih vrednosti y iz sistema i trenutnih informacionih potencijala iz PBRC-a izračuna estimaciju veličine y . Ovu estimaciju obeležavamo sa $\hat{y}$. Ovaj postupak bi mogao biti implementiran veštačkom neuronskom mrežom, međutim ukoliko mrežu iznova obučavamo svaki put kada pristigne nova vrednost y iz procesa, obuka će se vršiti sa periodom T_y .

Da bismo izbegli čestu obuku, možemo primeniti dva pristupa - da ponovno obučavanje vršimo tek kada prikupimo dovoljno velik broj novih odbiraka y , ili da mrežu uopšte ne obučavamo nego da druga celina vrši regresiju između dosada viđenih odbiraka. U ovom radu korišćen je drugi. Druga celina je implementirana kao modifikovana verzija *General Regression Neural Network* (GRNN) algoritma opisanog u [4]. Opšti pregled celog sistema prikazan je na slici 1:



Slika 1. Opšti pregled sistema

NAPOMENA:

Ovaj rad je proistekao iz master rada čiji mentor je bio dr Milan Rapaić, vanred.prof.

2.1. PBRC

Angelov i Filev su u [1] uveli pojam informacionog potencijala u cilju realizovanja metoda za klasifikaciju i identifikaciju. Rapaić i dr. su u [3] opisali otežani informacioni potencijal kao pogodniju veličinu za adaptivne metode koje treba da rade u realnom-vremenu. Sada ćemo ukratko izložiti metode i pojmove opisane u radovima [2] i [3].

Cilj algoritma je pronalazak potencijala vektora z u odnosu na sve prethodno viđene vektore koje obeležavamo skupom Z_k . Prvo se uvodi pojam otežane srednje kvadratne udaljenosti vektora z u odnosu na skup Z_k :

$$S_\lambda(z, Z_k) = (1 - \lambda) \sum_{i=0}^k \lambda^{k-i} \|z - z_k\|^2 \quad (1)$$

Parametar λ predstavlja faktor zaboravljanja. Što je ovaj parametar veći, PBRC prirođaje više značaja starijim odbircima. Potom se informacioni potencijal računa na sledeći način:

$$P_\lambda(z, Z_k) = \frac{1}{1 + S_\lambda(z, Z_k)} \quad (2)$$

Vrednost informacionog potencijala je u intervalu (0,1]. Što novi odbirak više liči na skorašnje ponašanje procesa, njegov informacioni potencijal je veći. Na sličan način se može odabrati fiksni vektor, i tokom rada procesa se računati koliko odabrani vektor odgovara skorašnjem ponašanju procesa. Bitno je napomenuti da se račun za informacioni potencijal može vršiti rekurzivno, što je neophodno za efikasnu implementaciju u realnom vremenu. Dokazi za tu tvrdnju su izloženi u [2] i [3].

U PBRC-u postoji skup vektora koji su reprezentativni za viđenje režime rada procesa. Ove vektore nazivamo fokusima. Koristeći rekurzivne formule, prilikom svakog novopristiglog odbirka računamo trenutne informacione potencijale svih fokusa i prosleđujemo ih GRNN-u. Takođe uvek računamo i informacioni potencijal novog odbirka.

Ukoliko je taj potencijal dovoljno velik, pravi se izmena u skupu fokusa. Ukoliko je novi odbirak dovoljno udaljen od svih fokusa u sistemu, novi odbirak postaje novi fokus sistema i ubacuje se u skup fokusa PBRC-a. Ukoliko postoji obližnji fokus, on se ažurira umesto pravljenja novog fokusa. Ovakvo ažuriranje omogućuje adaptaciju na blage promene procesa, jer fokusi mogu da se kreću unutar dozvoljene udaljenosti u odnosu na njihovu početnu poziciju. Međutim za razliku od predloženog algoritma u [2] i [3], fokusi imaju ograničenu mogućnost kretanja.

Ovo ograničenje mora biti uvedeno jer informacioni potencijali fokusa učestvuju u regresiji GRNN-a. Ukoliko bi se fokus previše udaljio od svoje početne pozicije, njemu pridružene vrednosti y_i u GRNN-u bi postale neažurne. Kada fokus dođe do novog položaja koji je izvan granice dozvoljenog kretanja vraća se u prvobitni položaj. Potom se na položaj novog odbirka postavi obližnji fokus ukoliko takav postoji, ili napravi novi ukoliko ne postoji odgovarajući fokus. Najveća dozvoljena udaljenost od početne pozicije fokusa obeležava se sa d i predstavlja jedan od meta-parametara algoritma.

2.2. GRNN

D. F. Specht je u [4] predložio GRNN kao rešenje za regresiju kontinualnih promenljivih. GRNN, za razliku od uobičajenih VNM, ne zahteva obuku. Za potrebe ovog rada GRNN kao ulazni vektor dobija trenutne vrednosti informacionih potencijala svih fokusa na osnovu čega treba odrediti estimaciju veličine $\hat{y}_i$. Periodično se iz procesa dobija i prava vrednost y_i , što predstavlja podatke na osnovu kojih GRNN vrši regresiju.

Vektor informacionih potencijala i odgovarajućih izlaza iz sistema koje je GRNN dobio za „obuku“ u prošlosti obeležavamo sa P_i i y_i . Ove vektore ćemo u nastavku rada zvati podacima za obuku iako se zapravo ne vrši nikakva obuka u uobičajenom smislu te reči. Na osnovu tih podataka, regresija novog vektora P se računa na sledeći način:

$$\hat{y}(P) = \frac{\sum_{i=1}^n y_i \cdot \exp\left(-\frac{\sum_{j=1}^m (P_j - P_j^i)}{2\sigma^2}\right)}{\sum_{i=1}^n \exp\left(-\frac{\sum_{j=1}^m (P_j - P_j^i)}{2\sigma^2}\right)} \quad (3)$$

n je broj dobijenih ulazno-izlaznih parova za obuku, m je broj fokusa u PBRC-u odnosno dimenzionalnost vektora P . σ je meta-parametar algoritma koji utiče na strminu regresorske površi. U matematičkom smislu σ je širina kernela za estimaciju koji je u ovom slučaju Gausov. P_j^i je j -ti informacioni potencijal iz i -tog podatka za obuku.

Suštinski, estimacija $\hat{y}(P)$ je parametrizovana srednja vrednost svih do sada dobijenih vrednosti y_j , gde težina svake pojedinačne vrednosti eksponencijalno opada sa razdaljinom $P - P_j$. Parametar σ utiče na oblik površi estimacije, kao i na otpornost na šum. Ukoliko je σ veoma veliko, $\hat{y}$ će dobiti vrednost najbližeg odbirka čime se smanjuje otpornost na šum, a ukoliko je σ blisko nuli, $\hat{y}$ će biti srednja vrednost svih do sada viđenih odbiraka y .

Za razliku od sistema opisanog u [4], ulazni vektor u GRNN u našem algoritmu menja dimenzionalnost tokom vremena. Svaka pojava novog fokusa u PBRC prouzrokuje povećanje broja dimenzija ulaznog vektora GRNN-a. Da bi GRNN i dalje mogao da računa udaljenost novih vektora u odnosu na stare, prilikom stvaranja novog fokusa u PBRC-u se odmah proširuju i svi vektori informacionih potencijala u memoriji GRNN-a na odgovarajući broj dimenzija. Nove dimenzije se popunjavaju nulama.

Naš sistem je namenjen radu u realnom vremenu, što znači da nije praktično čuvati sve prethodno dobijene odbirke za obuku. Ovaj problem je rešen grupisanjem (klasterovanjem) ulaznih odbiraka na način predložen u [4]. Svaka grupa ima svoj centar i dva pridružena parametra, A i B . Parametar A_i predstavlja zbir izlaza y svih odbiraka koji pripadaju i -toj grupi, a parametar B_i predstavlja broj odbiraka u i -toj grupi. Prilikom dobijanja novog odbirka za obuku, proverava se udaljenost tog odbirka od centra najbliže grupe. Ukoliko je udaljenost manja od unapred definisane granice r , ažuriraju se

parametri A i B za tu grupu. Ukoliko je udaljenost veća od r , novi odbirak postaje centar nove grupe. Estimacija se onda vrši izmenjenom formulom:

$$\hat{y}(P) = \frac{\sum_{i=1}^n A_i \cdot \exp\left(-\frac{\sum_{j=1}^m (P_j - P_j^i)}{2\sigma^2}\right)}{\sum_{i=1}^n B_i \exp\left(-\frac{\sum_{j=1}^m (P_j - P_j^i)}{2\sigma^2}\right)} \quad (4)$$

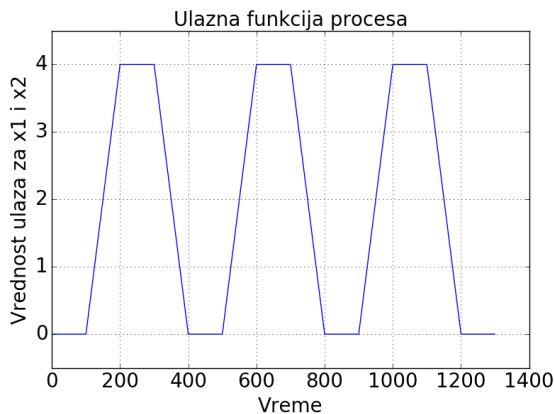
Najveći deo vremena izvršavanja algoritma predstavlja račun GRNN-a. Taj račun se može znatno ubrzati ukoliko arhitektura računara dozvoljava paralelizam, jer su formule (3) i (4) vrlo pogodne za paralelizaciju.

3. NUMERIČKI PRIMERI

Na jednostavnom primeru ćemo pokazati ponašanje algoritma. Sistem koji pokušavamo da estimiramo je linearan, dvodimenzionalan i ima sledeću formu:

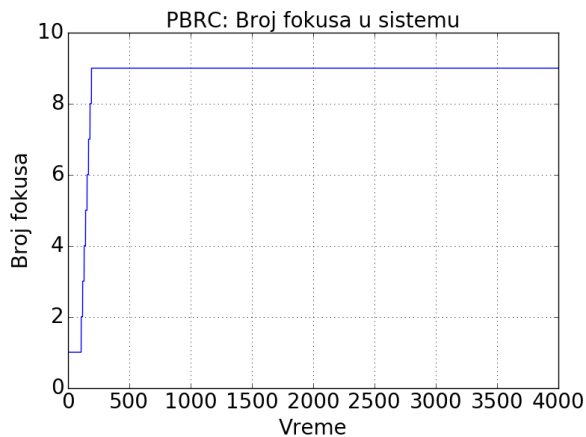
$$y = \alpha_1 \cdot x_1 + \alpha_2 \cdot x_2 \quad (5)$$

U nastavku primera parametri sistema su postavljeni na $\alpha_1 = 10$ i $\alpha_2 = 5$. Obe ulazne veličine x_1 i x_2 će biti predstavljene istom periodičnom funkcijom prikazanom na slici 2.



Slika 2. Ulazni signal za proces

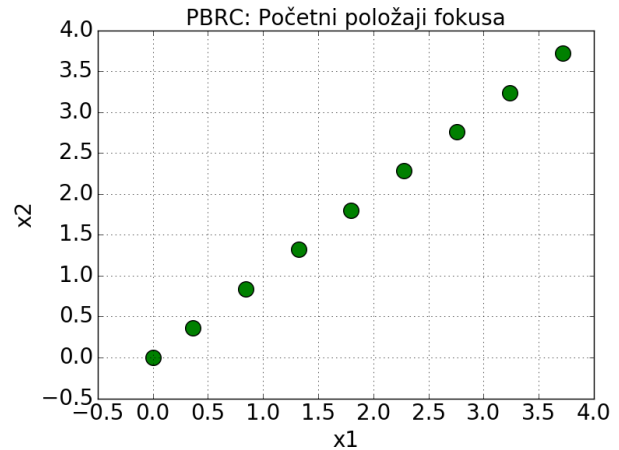
Prvo ćemo prikazati ponašanje PBRC-a. Na slici 3 prikazan je broj fokusa u sistemu u zavisnosti od vremena:



Slika 3. Broj fokusa u PBRC-u

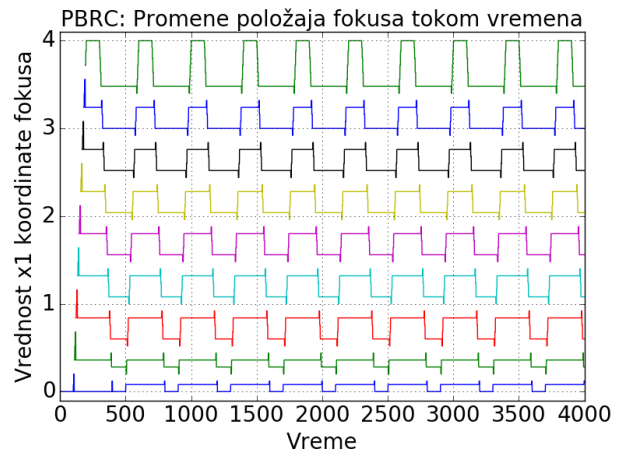
Pošto je ulazna funkcija periodična, novi fokusi se stvaraju jedino tokom prve periode. Posle toga ulazi se

kreću po već postojećim fokusima, te broj fokusa ostaje isti. Na slici 4 su prikazani početni položaji svih 9 fokusa u dvodimenzionalnom prostoru:



Slika 4. Početni položaji fokusa

Fokusi su stvarani na položajima koji odgovaraju kretanju ulaznog vektora, što je i očekivano. Na slici 4 se vide samo početne pozicije fokusa, ali fokusi zapravo menjaju svoje pozicije u dozvoljenim granicama zbog adaptacije. Slika 5 prikazuje promene položaja fokusa tokom vremena. Na slici su zbog preglednosti prikazana samo kretanja koordinate x_1 . Kretanja druge koordinate su slična.



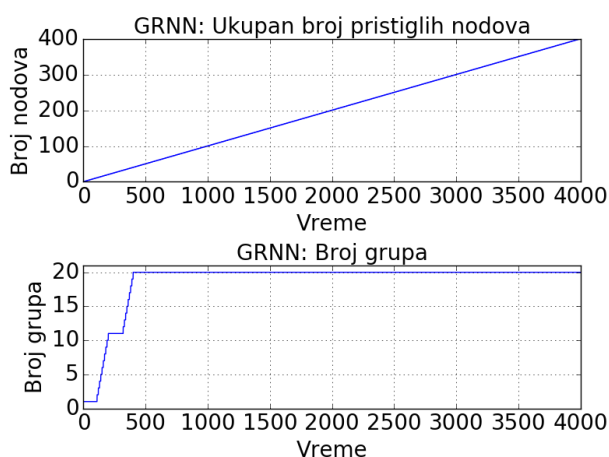
Slika 5. Položaji fokusa tokom vremena

Interesantno je primetiti da se u svakom trenutku može promeniti položaj najviše dva fokusa - jedan koji se vraća na svoj prvobitni položaj i drugi koji zauzima njegov prethodni položaj.

Sada ćemo prikazati ponašanje GRNN-a. Od interesa nam je da proučimo da li je grupisanje nodova doprinosi održivosti sistema koji treba dugo da radi u realnom vremenu.

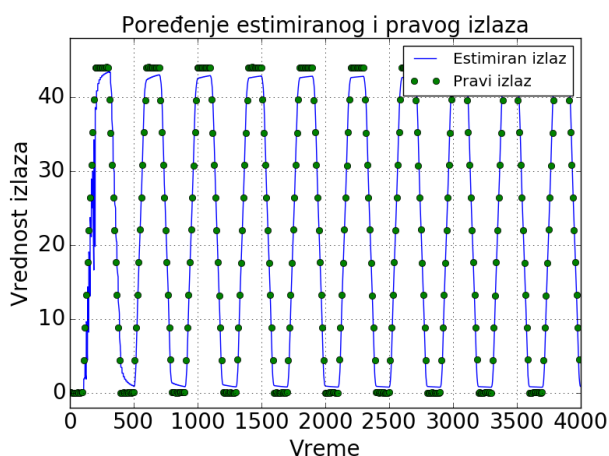
Slika 6 uporedo prikazuje ukupan broj nodova koji bi postojao u GRNN-u bez grupisanja, i broj grupa u GRNN-u koji zamenjuje pojedinačne nodove.

Kao što se vidi, sistem bez grupisanja bi bio neodrživ jer broj nodova linearno raste vremenom. Koristeći grupisanje, u našem primeru se efektivno koristi 20 nodova.



Slika 6. Broj nodova u GRNN-u

Preostalo je još uporediti izlaz GRNN-a (a ujedno i celog sistema) sa podacima koji se dobijaju iz samog procesa. Pošto se ti podaci dobijaju samo periodično, na slici 7 su predstavljeni tačkama, dok je izlaz GRNN-a prikazan linijom:



Slika 7. Poređenje pravog i estimiranog izlaza

Rezultati pokazuju dobru estimaciju sistema. Ukoliko bi se parametar σ povećao, greška algoritma bi se smanjila, ali bi porasla osetljivost na šum.

4. ZAKLJUČAK

Algoritam je već tokom prve periode procesa vršio dobru estimaciju. Broj fokusa u PBRC-u i grupa u GRNN-u nije bio velik, što omogućava brzo izvršavanje programa na računaru. Količina podataka koji se čuvaju u memoriji raste tokom pojavljivanja novih stanja procesa, posle čega je konstantna. Takođe, sav račun u PBRC-u se može vršiti rekurzivno. Te dve činjenice omogućuju implementaciju algoritma u integrisanim sistemima koji rade u realnom vremenu. Takvi sistemi često imaju ograničene resurse, što zahteva adekvatne algoritme.

Postoje 4 meta-parametra u algoritmu koje treba podesiti - faktor zaboravljanja λ , maksimalno dozvoljeno kretanje fokusa d , širina regresorskog kernela σ i poluprečnik grupa GRNN-a r .

Da bi se ovi parametri ispravno podesili, potrebno je proučiti proces koji se estimira, vršiti simulacije i analizirati rezultate. Ukoliko je jednostavno testirati algoritam na pravom procesu, moguće je algoritamski pronaći meta-parametre koristeći neku metodu optimizacije poput direktne pretrage.

5. LITERATURA

- [1] Angelov, P. & Filev, D. P. (2004). „An approach to on-line identification of Takagi–Sugeno fuzzy models“, *IEEE Transactions on Systems Man and Cybernetic, Part B –Cybernetics*, 34 (1), 484–498.
- [2] Milena Petković, Milan R. Rapaić, Zoran D. Jeličić, Alessandro Pisano, „On-line adaptive clustering for process monitoring and fault detection“, *Expert Systems with Applications*, Volume 39, Issue 11, 2012, Pages 10226-10235, ISSN 0957-4174
- [3] Milan R. Rapaić, Milena Petković, Zoran D. Jeličić, Alessandro Pisano, „An Adaptive Clustering Procedure with Applications to Fault Detection“, *Electronics*, Vol. 15, No. 2, December 2011
- [4] D. F. Specht, „A general regression neural network“, *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568-576, Nov 1991.

Kratka biografija:



Vukan Turkulov je rođen u Novom Sadu 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – adaptivno i napredno upravljanje je odbranio 2017. god.

DECENTRALIZOVAN SISTEM ZA DETEKCIJU KVAROVA ASINHRONIH MOTORA DECENTRALIZED SYSTEM FOR FAULT DETECTION OF INDUCTION MOTORS

Stefana Jocić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu prikazan je sistem za detekciju slomljene šipke rotora asinhronih motora, upotrebom cloud platforme, Microsoft Azure. Sistem koristi RBF neuronsku mrežu da klasifikuje stanja motora. Upotrebom softverskog paketa za analizu podataka Microsoft Power BI prikazani su rezultati klasifikacije podataka.

Abstract – In this paper, a decentralized system is presented for detection of broken rotor bar of induction motors, using Microsoft Azure, a cloud computing service. This system uses RBF neural network to classify motors by its condition. Results are presented using a suite of business analytics tools, Microsoft Power BI.

Ključne reči: detekcija kvarova, RBF neuronska mreža, Microsoft Azure, Power BI

1. UVOD

Najveći deo električne energije koja se koristi u industriji troše asinhroni motori. Zbog svoje robusne konstrukcije, pouzdanosti, niske cene i široke mogućnosti primene, predstavljaju najzastupljeniju komponentu u industrijskim sistemima. To je upravo razlog postojanja velikog broja postupaka za ranu detekciju kvarova asinhronih motora, kako bi se smanjili ekonomski gubici i omogućio bezbedan rad u industrijskim postrojenjima.

Detekcija kvarova motora može se postići analizom mnoštva signala sa motora, kao na primer struje, napona, vibracije, magnetnog polja itd. U ovom radu za detekciju kvarova koristiće se analiza signala vibracija, u cilju otkrivanja slomljenje šipke rotora i klasifikaciju motora prema svom stanju.

U poglavlju 2 predstavljen je postupak za detekciju i klasifikaciju kvara baziranog na analizi signala vibracija. U trećem poglavlju biće pokazana softverska arhitektura rešenja, dok će rezultati biti prikazani u poglavlju broj 4.

2. POSTUPAK REŠAVANJA PROBLEMA

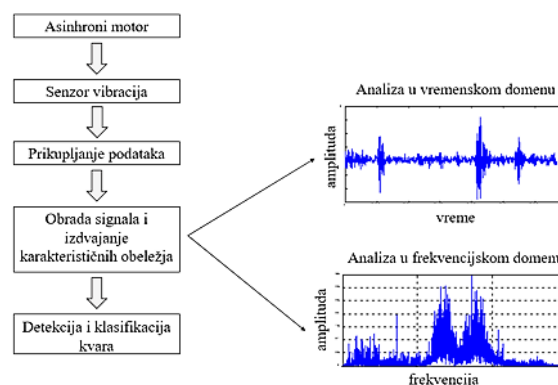
Raznovrsne analize signala vibracija se koriste kako bi se detektovao kvar i pratilo stanje motora. Motori, čak i kada veoma dobro rade, imaju neke nesavršenosti, pa samim tim poseduju neki nivo vibracija koji se smatra normalnim. Međutim, problem nastaje onda kad se taj nivo vibracija poveća, a razlog je obično neko mehaničko oštećenje. Za merenje nivoa vibracija mogu se koristiti različiti senzori, a jedni od najčešćih su akcelometri [1].

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Rapaić, vanr. prof.

Akcelometri registruju ubrzanje komponente za koju su pričvršćeni i pretvaraju je u električni signal, pa se pomoću transmitera taj signal standardizuje i koristi za dalju obradu [3]. Detekcija kvarova motora obično se odvija u tri koraka, koji su prikazani na slici ispod [1]:

- Prikupljanje podataka;
- Obrada signala i izdvajanje obeležja;
- Detekcija i klasifikacija kvarova.



Slika 1. Šema postupaka detekcije kvarova analizom signala vibracija

Signali korišćeni u ovom radu snimljeni su u TE – TO Zrenjanin, u cilju analize rada viskonaponskih motora napojnih pumpi dva kotla, K1 i K2. Snimljeni signali predstavljaju izmerene vrednosti vibracija dva elektromotora, od kojih je jedan u potpunosti ispravan, a za drugi je ustanovljeno da poseduje dve slomljene šipke rotora. Za merenje su korišćena dva akcelometra, postavljeni tako da prikupljaju podatke radijalnih vibracija u vertikalnoj ravni i aksijalnih vibracija u horizontalnoj ravni [2]. Pozicija senzora se može videti na slici 2.



Slika 2. Način montaže senzora vibracija

2.1. Prikupljanje podataka

U prvoj fazi vrši se prikupljanje izmerenih podataka sa senzora vibracija. Signali su snimani sa frekvencijom odabiranja 10 kHz, a svaki snimak sadrži 100000 odbiraka [2].

2.2. Obrada signala i izdvajanje obeležja

U drugoj fazi vrši se obrada signala vibracija koji su prikupljeni sa senzora. Signali predstavljaju niz numeričkih vrednosti (pomeraja, brzine ili ubrzanja) $x(i)$ u vremenskom domenu [3]. Koristeći takav signal moguće je izračunati u vremenskom domenu obeležja koja mogu ukazati na prisustvo pojedinih kvarova i pokazuju opšte stanje elektromotora. Efektivna vrednost signala (eng. *root mean square, RMS*) predstavlja meru „snage“ u signalu vibracija i koristi se kao pokazatelj neizbalansiranosti komponenta motora, a definiše se izrazom [1, 4]:

$$x_{rms} = \left(\frac{1}{N} \sum_{i=1}^N x^2(i) \right)^{1/2} \quad (2.1)$$

Efektivna vrednost signala se radi preciznije detekcije kvarova kombinuje sa tzv. „Crest“ faktorom, koji se definiše izrazom [3]:

$$C = \frac{\max(x(i)) - \min(x(i))}{x_{rms}} \quad (2.2)$$

Moguće je detektovati pojedine kvarove i analizom funkcije gustine verovatnoće signala vibracija, stoga se mogu izračunati sledeća obeležja [1, 3]:

- Srednja vrednost (eng. *mean*):

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x(i) \quad (2.3)$$

- Kvadrat srednje vrednosti korena odbiraka (eng. *square mean root*):

$$x_r = \left(\frac{1}{N} \sum_{i=1}^N |x(i)|^{1/2} \right)^2 \quad (2.4)$$

- Faktor iskrivljenja (eng. *skewness*):

$$x_{ske} = \frac{\frac{1}{N} \sum_{i=1}^N (x(i) - \bar{x})^3}{x_v^{3/2}}, \quad (2.5)$$

gde je x_v varijansa x : $x_v = \frac{1}{N} \sum_{i=1}^N (x(i) - \bar{x})^2$.

- Faktor spljoštenosti (eng. *kurtosis*):

$$x_{kur} = \frac{1}{x_v^2} \left(\frac{1}{N} \sum_{i=1}^N (x(i) - \bar{x})^4 \right) \quad (2.6)$$

- Faktor oblika (eng. *Shape factor*):

$$S = \frac{x_{rms}}{\bar{x}} \quad (2.7)$$

- L faktor:

$$L = \frac{\max(x(i)) - \min(x(i))}{x_r} \quad (2.8)$$

- I faktor:

$$I = \frac{\max(x(i)) - \min(x(i))}{\bar{x}} \quad (2.9)$$

Kako bi se sa sigurnošću utvrdilo poreklo i vrsta kvara, navedenim obeležjima pridružuju se i obeležja iz frekventijskog domena. Potrebno je na signal iz vremenskog domena primeniti brzu Furijeovu transformaciju (FFT), da bi se izračunao spektar signala vibracija. Obeležja koja karakterišu najznačajnije kvarove kod elektromotora su [3]:

1. Karakteristična frekvencija spoljne staze ležaja:

$$f_{rpfo} = f_r \times \frac{N}{2} \left(1 - \frac{d}{D} \cos \phi \right) \quad (2.10)$$

2. Karakteristična frekvencija unutrašnje staze ležaja:

$$f_{bpfi} = f_r \times \frac{N}{2} \left(1 + \frac{d}{D} \cos \phi \right) \quad (2.11)$$

3. Frekvencija rotacije kotrljajućeg elementa (kuglice):

$$f_{bsf} = f_r \times \frac{d}{2D} \left(1 - \left(\frac{d}{D} \right)^2 \cos^2 \phi \right) \quad (2.12)$$

4. Karakteristična frekvencija za oštećenje kotrljajućeg elementa:

$$f_{bff} = 2 \times f_{bsf} \quad (2.13)$$

5. Karakteristična frekvencija kaveza:

$$f_{ftf} = f_r \times \frac{1}{2} \left(1 - \frac{d}{D} \cos \phi \right) \quad (2.14)$$

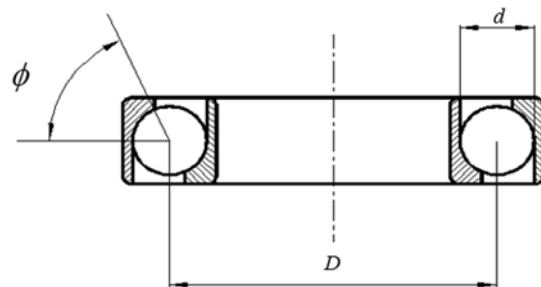
6. Karakteristična frekvencija oštećenja šipke rotora:

$$f_{brb} = (1 \pm 2ks)f_r, k = 1, 2, \dots \quad (2.15)$$

7. Karakteristična frekvencija ekscentriciteta:

$$f_{ecc} = 2f_s \pm f_r \quad (2.16)$$

U jednačinama (2.10) – (2.14), f_r predstavlja frekvenciju obrtanja rotora asinhronog motora, N je broj kotrljajućih elemenata u ležaju, ϕ i d su ugao kontakta i prečnik kotrljajućeg elementa, a D je srednji prečnik ležaja, kao što je pokazano na slici ispod.

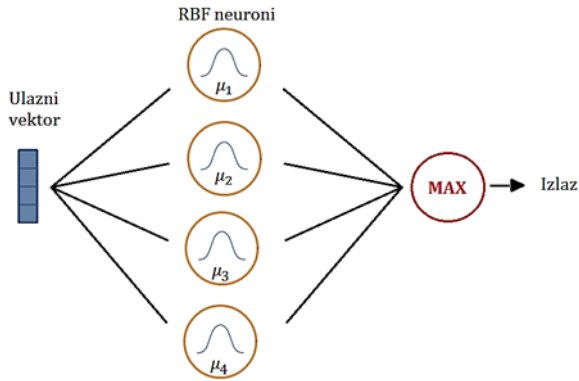


Slika 3. Geometrija tipičnog kotrljajućeg ležaja

Oštećenje šipke rotora u spektru signala vibracija ogleda se u prisustvu povećane amplitude spektra na frekvencijama bočnih opsega, koje se mogu izračunati uz pomoć izraza (2.15), gde s predstavlja klizanje, a f_r označava frekvenciju obrtanja rotora [3]. Za detekciju ekscentriciteta koristi se izraz (2.16), gde f_s predstavlja frekvenciju naponskog izvora.

2.3. Detekcija i klasifikacija kvarova

U ovoj fazi se na osnovu skupa obeležja dobijenih na gore opisan način, određuje da li se kod posmatranog motora nalazi neki od kvarova. Obzirom da je vrsta kvara unapred poznata za svaki signal, potrebno je samo izvršiti klasifikaciju stanja motora, odnosno da li je motor ispravan ili ne. Za klasifikaciju stanja motora korišćena je RBF neuronska mreža. Struktura neuronske mreže prikazana je na slici 4.



Slika 4. Struktura RBF neuronske mreže

Mreža se sastoji od tri sloja: ulaznog, skrivenog i izlaznog sloja. Ulaz u neuronsku mrežu je vektor od 4 elementa. Prvi element predstavlja opterećenje motora (može biti 60 ili 80 t/h, dok je nominalno opterećenje 300 t/h).

Drugi i treći element predstavljaju količnik amplitude spektra sig-nala vibracija na frekvenciji bočnih opsega karakterističnih za kvar tipa slomljene šipke rotora i amplitude spektra signala vibracija na frekvenciji naponskog izvora. Poslednji element je frekvencija obrtanja rotora asinhronog motora.

U skrivenom sloju nalaze se 4 neurona redom za sva 4 karakteristična slučaja koja se javljaju:

1. Ispavan elektromotor sa opterećenjem od 100 t/h;
2. Neispravan elektromotor sa opterećenjem od 80 t/h;
3. Neispravan elektromotor sa opterećenjem od 100 t/h;
4. Ispravan elektromotor sa opterećenjem od 80 t/h.

Svaki od neurona sadrži centorid μ_j , koji predstavlja vektor, čija je struktura ista kao i kod ulaznog vektora. Izlaz iz neurona definiše se izrazom:

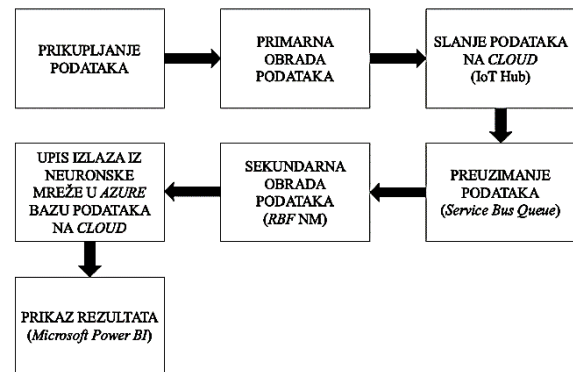
$$\phi_j = e^{-\frac{\|x - \mu_j\|^2}{2\sigma_j^2}}, \quad (2.17)$$

gde x predstavlja ulazni vektor, a σ promenljiv parametar koji u ovom slučaju uzima vrednost 1. Kada se izračunaju izlazi iz neurona, svi izlazi čija je vrednost manja od praga $\varepsilon = 0.01$ postavljaju se na 0. Nakon toga potrebno je podeliti ϕ_j sa sumom svih vrednosti za ϕ_j , ukoliko je suma pozitivna.

Na kraju, za svaki snimljeni signal od dobijene 4 vrednosti bira se maksimalna vrednost, koja predstavlja izlaz iz neuronske mreže. Na osnovu indeksa tog maksimalnog elementa motor se može klasifikovati kao ispravan ili neispravan i moguće je utvrditi opterećenje.

3. SOFTVERSKA ARHITEKTURA REŠENJA

Za izradu aplikacije korišćen je Microsoft Azure, Microsoft-ova platforma za cloud. Ova platforma omogućava razvoj aplikacija uz resurse koji se mogu „iznajmiti“ od servis provajdera po određenim cenama [5]. Cilj softverskog rešenja je omogućiti slanje sirovih i primarno obrađenih merenja (različitih veličina od interesa) na cloud-servis radi čuvanja i sekundarne obrade. Na taj način, formira se jedinstveni servis koji potencijalno omogućava praćenje stanja asinhronih mašina koje su međusobno udaljene. Takođe, ovakav servis omogućava da se postupak prikupljanja podataka i primarne obrade potpuno odvoji od finalnog postupka detekcije kvarova samih motora, koji može biti složen i zasnovan na naprednim algoritmima. Šta više, softverske i hardverske komponente, pomoću kojih se vrši akvizicija, mogu biti projektovane nezavisno od tih naknadnih i naprednijih postupaka koji se mogu dodavati i kasnije, ukoliko se sa njima pomaže potreba u eksploataciji rešenja. Struktura softverskog rešenja prikazana je na slici ispod.



Slika 5. Struktura softverskog rešenja

Prikupljanje podataka predstavlja čitanje iz datoteka, odnosno učitavanje već snimljenih signala vibracija sa elektromotora. Kao deo Microsoft Azure-a korišćen je IoT Hub, servis koji omogućava dvosmernu komunikaciju između pametnih IoT uređaja i aplikacije [6]. Na portalu Microsoft Azure-a moguće je kreirati IoT Hub, koji će služiti za primanje podataka sa uređaja.

Da bi uređaj mogao da šalje podatke na IoT Hub potrebno je da bude registrovan, odnosno da poseduje ključ preko koga će pristupati IoT Hub-u.

Kreiranjem jednostavne konzolne aplikacije dobija se ključ za određeni uređaj. Podaci koji se šalju na cloud predstavljaju već obrađene podatke, odnosno ulaze u neuronsku mrežu, koji su detaljnije objašnjeni u prethodnom poglavlju.

Slanje podataka vrši se u okviru druge konzolne aplikacije, gde se podaci šalju u tačno određeni endpoint, koji ovde predstavlja Service Bus Queue. Kreiranje dodatnih endpoint-a moguće je preko Microsoft Azure portala.

Za preuzimanje poslatih podataka upotrebljen je Service Bus Queue, koji predstavlja deo Azure messaging infrastrukture [7].

Svaka poslata poruka je preuzeta iz reda, obrađena i potom označena kao obrađena, kako se ne bi desilo da ista poruka bude obrađena dva puta.

Kada je poruka obrađena, izlazi iz skrivenog sloja neuronske mreže upisuju se u bazu podataka, koja se može kreirati na pomenutom portalu. Zabeležene vrednosti u bazi su potom iskorišćene za prikaz rezultata, što će biti pokazano u narednom poglavlju.

4. REZULTATI

Eksperimentalni uzorak na kome su vršena istraživanja sastoji se od dva identična visokonaponska elektromotora napojnih pumpu dva kotla. Podaci o elektromotorima su sledeći:

- Nazivna snaga: 3150 kW;
- Nazivni napon: 6000 V;
- Nazivna struja: 373 A;
- Nazivna frekvencija: 50 Hz;

Snimani su signali na oba motora, ispravan motor sa opterećenjima 80 i 100 t/h, dok je neispravan motor bio optečen sa 60, 80 i 100 t/h. Jedan motor je bio ispravan u potpunosti, dok je drugi imao dve slomljene šipke rotora [2].

Tabela 1. Broj tačnih klasifikacija elektromotora

Stanje motora	Opterećenje motora [t/h]	Broj signala	Broj tačnih klasifikacija
Ispravan	80	5	2
Ispravan	100	5	5
Neispravan	80	5	8
Neispravan	100	5	5

U tabeli 1 prikazan je broj signala koji su korišćeni za testiranje. Za svaki primer motora upotrebljeno je po 5 signala. Dobijeni rezultati su očekivani, jer prilikom manjeg opterećenja motora teško je detektovati kvar tipa slomljene šipke rotora.

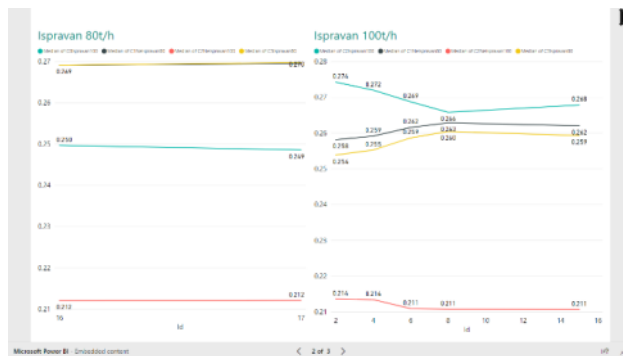
Zbog toga su od ukupno 5 signala snimljenih na ispravnom motoru opterećenim sa 80 t/h, samo 2 klasifikovano kao takvi, a ostali kao neispravni.

Za prezentaciju podataka iskorišćen je softverski paket za analizu podataka *Microsoft PowerBI*. Na narednim slikama mogu se videti grafici, koji predstavljaju izlaz iz skrivenog sloja neuronske mreže korišćene za klasifikaciju motora.

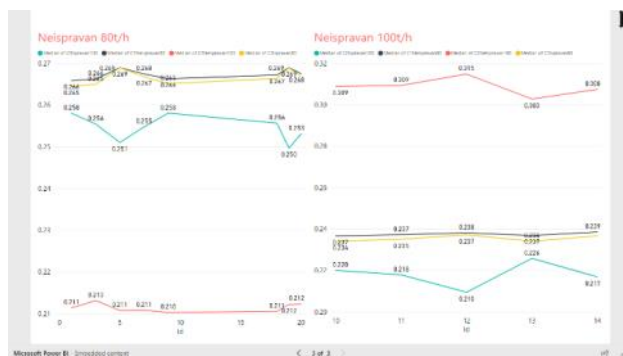
Moguće je izabrati pojedinačno merenje ili sva merenja odjednom. Za svako merenje postoje 4 tačke, a ona koja ima maksimalnu vrednost pokazuje stanje motora i njegovo opterećenje.



Slika 5. Prikaz izlaza iz skrivenog sloja neuronske mreže za sva merenja



Slika 6. Pregled elektromotora klasifikovani kao ispravni



Slika 7. Pregled elektromotora klasifikovani kao neispravni

6. LITERATURA

- [1] M. Vishwakarma, R. Purohit, V. Harshlata, P. Rajput, "Vibration analysis and Condition Monitoring for Rotating Machines: A Review", *Materials Today: Proceedings* 4 (2017), pp. 2659–2664
- [2] Ž. Kanović, D. Reljić, Đ. Oros, „Izveštaj o izvršenim merenjima na visokonaponskim elektromotorima napojnih pumpi kotlova K1 i K2 u TE-TO Zrenjanin“, 2012.
- [3] Ž. Kanović, „Modifikacije algoritma optimizacije rojem čestica sa primenom u detekciji kvarova na objektima automatskog upravljanja sa kontinualnom dinamikom“, doktorska disertacija, Fakultet tehničkih nauka, Univerzitet u Novom Sadu, Novi Sad, Srbija, 2012.
- [4] P. Stepanić, I. Latinović, Ž. Đurović, „A new approach to detection of defects in rolling element bearings based on statistical pattern recognition“, *Int J Adv Manuf Technol* (2009) 45:91-100
- [5] <https://azure.microsoft.com/en-us/overview/what-is-paas/>, pristupljeno u julu 2017.
- [6] <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-iot-hub>, pristupljeno u julu 2017.
- [7] <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions>, pristupljeno u julu 2017.

Kratka biografija:



Stefana Jocić rođena je u Prizrenu 1993. godine. Osnovne studije Računarstva i automatike na Fakultetu tehničkih nauka upisala je 2012. god., i diplomski rad na temu Sistem za koordinaciju autonomnih vozila odbranila je 07.07.2016. Istog godine upisala je master studije Računarstvo automatika na Fakultetu tehničkih u Novom Sadu.

OPTIMIZACIONA ANALIZA KRITERIJUMA INFRASTRUKTURE EV STANICA OPTIMIZATION ANALYSIS CRITERIA INFRASTRUCTURE EV STATIONS

Milica Repac, Vladimir Katić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu je dat predlog rasporeda stanica za punjenje električnog vozila duž pojedinih deonica koridora 10 i 11 na osnovu postavljenih kriterijuma. Pored toga je predstavljen i ekonomski model stanice uz pomoć SWOT analize.

Abstract – This paper describes plan for allocation EV stations along certain parts of corridors 10 and 11, based on set criteria. Here is also represented economic model of EV station with SWOT analysis.

Ključne reči: el. punjači, el. vozila, priključci, standardi, kriterijumi

1. UVOD

Srbija je 4. decembra 2015. godine usvojila Strategiju razvoja energetike do 2025. god. sa projekcijama do 2030. god., kojom je predložen put ka tehnološkoj modernizaciji energetike. Do 2020. godine Srbija u planu ima znatnu eks-panziju po pitanju EV stanica u gradovima i na koridorima.

Prvi brzi punjač za električna vozila u Srbiji je postavljen 25. aprila ove godine ispred Mašinskog fakulteta u Beogradu, kao deo evropskog projekta *Zelena Balkanika*. Ovaj punjač može da napuni svako vozilo za samo 15 minuta. Reč je o punjaču od 50 kW, koji dnevno može da napuni i do 70 vozila.

2. KONCEPT E-MOBILNOSTI

Tehnološki koncept se sastoji od sledećih komponenti:

- ❖ stanice za punjenje (eng. *charging station*)
- ❖ električnog vozila (eng. *Electric vehicle, EV*)
- ❖ upravljačkog (informativnog) centra (eng. *Charging Station Control Centar, CSCC*)
- ❖ opreme za napajanje električnog vozila (eng. *Electric Vehicle Supply Equipment, EVSE*)

3. MODELI IZGRADNJE JAVNE INFRASTRUKTURE ZA PUNJENJE EL. VOZILA

Prema EURELECTRIC-u su identifikovana 4 modela za izgradnju infrastrukture na javnim površinama i u javnom vlasništvu. Modeli se biraju u skladu sa uslovima tržišta, novčanim potsticajima datim od strane Vlade...

Svaki od navedenih modela obuhvata Operatora Distribucijskog Sistema (ODS) i snabdevača električnom energijom (stanice za punjenje). Modeli ne obuhvataju: kupovinu baterije, održavanje i upravljanje vozilima, informisanje vozača u realnom vremenu.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Vladimir Katić.

Izbor modela zavisi i od Regulatorne agencije o stepenu regulacije područja na kom bi pomenuti model trebalo primeniti.

3.1. Integrisani model

U ovom modelu je vlasnik stanice za punjenje Operator Distribucijskog Sistema (ODS). Snabdevač i kupac sklapaju ugovor za uslugu e-mobilnosti (može biti pretplatna ili u skladu sa potrošnjom). Pri tom ODS dodeljuje kupcu identifikacioni broj ID u skladu sa ugovorom koji kupac potpisao sa snabdevačem za uslugu e-mobilnosti, odnosno za snabdevanje električnom energijom. S tim da je kupac u mogućnosti da sklopi više ugovora sa više snabdevača, pa da tako ima više identifikacionih brojeva ID-eva. Svi snabdevači mogu slobodno nuditi svoje usluge i proizvode, a na kupcu je da odluči koji mu snabdevač najviše odgovara na osnovu nekih svojih kriterijuma koje postavlja.

3.2. Model odvojene infrastrukture

U ovom modelu je vlasnik stanice za punjenje Operator Infrastrukture, a ne Operator Distribucijskog Sistema (ODS). Jedini uslov je da svi snabdevači imaju jednake uslove pristupa. Operator infrastrukture je odvojen od snabdevača i on naplaćuje snabdevačima naknadu za priključenje (eng. *access fee*), a snabdevači će uvećati svoje cene tako da ne budu na gubitku i naplatiti ih krajnjem korisniku. Tako da se slobodno može reći da krajnji korisnici finansiraju izgradnju infrastrukture. Operatora infrastrukture može biti više, ali zato moraju posedovati potrebne dozvole od lokalne samouprave za postavljanje pomenute infrastrukture. Svi moraju imati sistem za autentifikaciju i obračun ili mogu imati zajednički IT sistem.

3.3. Model nezavisnog Operatera e-mobilnosti

Odve pružatelj usluga e-mobilnosti pored postavljanja sopstvene infrastrukture mora da ima dozvolu i za snabdevanje električnom energijom. Izgradnju infrastrukture finansiraju krajnji korisnici, tako što im se naplaćuje u vidu mesečne naknade. U početku će snabdevač sa monopolnim položajem imati većinu infrastrukture u svom posedu.

3.4. Model Spot Operatora e-mobilnosti

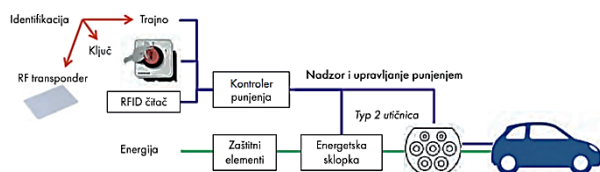
U ovom modelu Operator poseduje stanice za punjenje, odnosno vlasnik parkinga je istovremeno i vlasnik infrastrukture i prodaje električnu energiju, dok krajnji korisnici finansiraju održavanje izgrađene infrastrukture. Gradske uprave su vlasnici javnih parkinga, ali i one mogu uvesti sistem licenciranja preduzeća kako bi ona mogla ravno-pravno da se nadmeću oko najzanimljivijih lokacija. Odve

na vlasniku ostaje izgradnja infrastrukture i njeno upravljanje i prodaja električne energije.

Dodatak ovom modelu je opcija „roaming“ koja predstavlja način da jedan Operator prepozna ID kupca drugog Operatora i da mu dozvolu da se priključi na njegovu infrastrukturu. Operatori slobodno postavljaju cene i nadmeću se oko najtraženijih lokacija.

4. MODEL PRIVATNE INFRASTRUKTURE ZA PUNJENJE ELEKTRIČNIH VOZILA

Suštinska razlika u verzijama stanica koje se mogu naći na tržištu jeste u zahtevu za merenje električne energije. Neke primene zahtevaju, a neke ne zahtevaju merenje električne energije, odnosno postojanje brojala u stanici za punjenje. Za dalje razlikovanje u verzijama stanica je bitan način identifikacije korisnika, odnosno kome je dopušten pristup energiji iz stanice za punjenje električnog vozila.



Slika 1. Prikaz stanice za kućnu upotrebu i prenosne stanice

U stanicama za kućnu upotrebu ili za upotrebu manjeg broja korisnika identifikacija se uopšte ne koristi ili se koristi ključ, RFID kartica (eng. *Radio Frequency Identification*). Kada kontroler punjenja prihvati identifikaciju korisnika i kada obradi podatke dobijene od utičnice da je vozilo priključeno tada se uključuje energetska sklopka i napon mreže se prosleđuje vozilu.

5. DEFINISANJE KRITERIJUMA ZA IZBOR STANICE ZA PUNJENJE ELEKTRIČNIH VOZILA

Izbor kriterijuma procene je važan zbog optimalnog izbora lokacije stanica za punjenje električnih vozila. Potrebno je ustanoviti odgovarajući sistem ocenjivanja koji će se ogledati u karakteristikama bitnim za odabir lokacije. Međutim, industrija električnih vozila je i dalje u početnoj fazi tako da još uvek nije ustanovljena lista kriterijuma. Električna vozila teže ka ekološkoj održivosti, a ta održivost se bazira na sledećim kriterijumima:

1. profitabilnosti (SWOT analiza),
2. dostupnosti stanice korisniku,
3. uticaju na distributivnu mrežu
4. mogućnosti priključenja i
5. algoritmu rutiranja.

5.1. Profitabilnost (SWOT ANALIZA)

SWOT (*Strength Weakness Opportunity Threat*) analiza se primenjuje kako bi se ispitala isplativost planirane investicije, odnosno da bi se ispitala njene mane i prednosti na osnovu kojih bi se dalje odredio period otplate (*amortisation period*). Sa ovom analizom se kreira finansijski plan, koji predstavlja ključan faktor za realizaciju planiranog projekta.

Prva stavka od koje polazi svaki projekat jeste finansijsko planiranje: da li je kratkoročno ili dugoročno (strategijsko planiranje). Ono direktno zavisi od životnog, radnog veka same investicije. Potrebno je definisati sam izvor sredstava izdvojenih u ulaganje investicije: da li su to vlastita ili tuđa sredstva. Na osnovu toga se dalje definiše stopa povraćaja investicije (1), prikazana jednačinom:

$$\text{stopa povraćaja investicije} = \frac{\text{dobit (profit)}}{\text{investirana imovina}} \quad (1)$$

Da bi se odredila stopa povraćaja investicije potrebno je prvo proceniti neto novčani tok (godišnji profit). Za to treba definisati funkciju iskorišćenosti stanice za punjenje električnih vozila na dnevnom nivou, a potom dobijenu vrednost preračunati na iskorišćenost stanice na godišnjem nivou. Funkcija iskorišćenja na dnevnom nivou glasi:

$$F_i(x) = \frac{n \cdot d \cdot e}{o \cdot p} \quad (2)$$

gde je,

- n – broj vozila koja koriste EV stanicu
- d – pređeno dnevno rastojanje [km]
- e – potrošnja el. energije po EV vozilu [kWh/km]
- o – izlazna snaga po priključku [kW]
- p – broj priključaka

Kao pretpostavka je uzeto da je radno vreme stanice za punjenje električnog vozila 8h svaki dan. Za ovaj proračun je odabran i-CHARGE brzi punjač (CHADEMO priključak) od 20 kW proizvođača Schrack, čija cena iznosi 17 265,53 (eur). Prosečno punjenje vozila traje 20 min. Uzimajući u obzir trajanje radnog vremena od 8h i trajanje punjenja sa brzim punjačem od 20 min, može se zaključiti da bi se u idealnom slučaju sa 80% aktivnog radnog vremena moglo napuniti do 19 vozila u jednom danu. Za pređeno rastojanje se računa deonica Horgoš – Novi Sad, koja iznosi 108 km.

$$F_i(x) = \frac{19 \cdot 108 \cdot 0,15}{20} = 15,4 \text{ eur} \quad (3)$$

Na osnovu urađenog proračuna (3) na dnevnom nivou se može dobiti novčani tok na godišnjem nivou koji iznosi 5621 eur. Kao rezultat toga vrednost stope povraćaja investicije (1) iznosi 32,56%.

Da bi odredili period povraćaja investicije važi sledeće,

$$\text{per. povraćaja investicije} = \frac{\text{investiciona imovina}}{\text{neto novčani tok}} \quad (4)$$

$$\text{neto novčani tok} = \text{godišnja ušteda (profit)} \quad (5)$$

Tako da za zadati primer modela period povraćaja (4) investicije iznosi:

$$\text{period povraćaja investicije} = \frac{17265,53}{5621} = 3 \text{ god.} \quad (6)$$

Po dobijenom proračunu (3) nakon isteka prve tri godine investicija je vraćena i Investitor ima čist profit.

5.2. Dostupnost stanice korisniku

Ovim kriterijumom se definiše stepen korisnosti stanice. Zavisi od snabdevenosti i potražnje, pri čemu se visoka vrednost ovog kriterijuma dobija ako je vrednost snabdevenosti velika, a potražnja je mala. Snabdevenost je određena sa izlaznom snagom punjača i efektivnim vremenom pri kom stanica radi maksimalnim kapacitetom. Potražnja

je određena prosečnom potrošnjom električnog vozila, kao i brojem vozila koja se nalaze u okruženju stanice.

Ukoliko bi se desilo da potražnja prevazilazi snabdevenost, onda bi stanici pretila preopterećenost. Drugim rečima, kada vrednost potražnje prevazilazi vrednost snabdevenosti onda ta stanica nije dostupna korisniku kojem je u tom trenutku potrebna ta usluga.

To će dalje uticati na njegovu rutu vožnje (zavisno od kapaciteta baterije). Može se reći da ovaj kriterijum utiče na navike korisnika, te da se smatra i kriterijumom društva. Funkcija za određivanje dostupnosti glasi:

$$F_d(x) = \text{snabdevanje} - \text{potražnja} \quad (7)$$

$$F_d(x) = (o \cdot t \cdot p) - (n \cdot d \cdot e) \quad (8)$$

gde je,

o – izlazna snaga po priključku [kW]

t – efektivno vreme pri kom stanica radi maksimalnim kapacitetom [t]

n – broj električnih vozila koja koriste stanicu

p – broj priključaka

d – dnevno pređeno rastojanje [km]

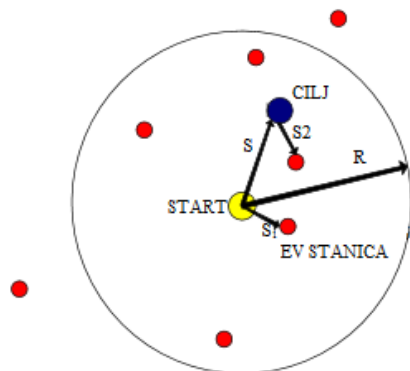
e – prosečna potrošnja po vozilu [kWh/km]

6. ALGORITAM RUTIRANJA

Za raspodelu stanica za punjenje električnih vozila je bitno imati algoritam rute vozila, kojim se određuje, odnosno procenjuje radijus kretanja. Algoritam rute pokriva tri situacije kretanja vozila ka svojoj ciljnoj destinaciji [18]:

- direktan dolazak vozila do cilja,
- dopuna baterije vozila pre dolaska do cilja i
- vozilo ne može da se pokrene.

Ukoliko vozilo može da dođe do cilja direktno (bez dodatne dopune), onda mora da zadovoljava sledeće uslove [18]:



Slika 4. Uslov $R \geq S + S_2$

- 1) kapacitet baterije mora biti veći od udaljenosti između starta i cilja, odnosno

$$R > S \quad (9)$$

R – domet vozila zahvaljujući trenutnom kapacitetu baterije [km]

S – rastojanje starta od cilja [km]

- 2) kada vozilo dođe do cilja može se desiti da mu se baterija isprazni upravo tad, pa pod tim uslovima mora se naći najbliža EV stanica:

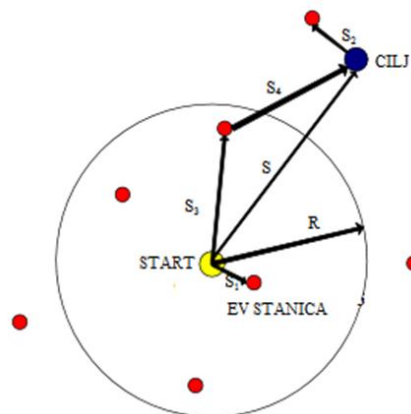
$$R \geq S + S_2 \quad (10)$$

S_2 – rastojanje između cilja i najbliže EV stanice

Ukoliko je situacija takva da vozilo ne može da dođe do cilja, odnosno da ne zadovoljava gore navedene uslove, već mora pronaći stanicu pre dolaska na cilj što dovodi do formiranja novih uslova za novonastalu situaciju. Uslovi koji ovde važe glase [18]:

$$R < S + S_2 \text{ i } R \geq S_1 \quad (11)$$

S_1 – rastojanje između starta i najbliže EV stanice



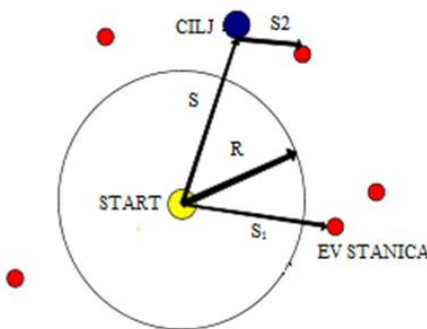
Slika 5. Uslov $R < S + S_2$ i $R \geq S_1$

Ovde ukoliko bi vozilo išlo direktno do cilja, bez dopune baterije, ostalo bi bez napajanja u određenom momentu i ne bi došlo do željene destinacije.

U situaciji kad vozilo ne može da se pokrene važe dole navedeni uslovi,

$$R < S + S_2 \text{ i } R < S_1 \quad (12)$$

Ovde baterija nije dovoljno puna da bi vozilo moglo da dođe do neke EV stanice, što predstavlja dokaz da raspored EV stanica treba poboljšati.



Slika 6. Uslov $R < S + S_2$ i $R < S_1$

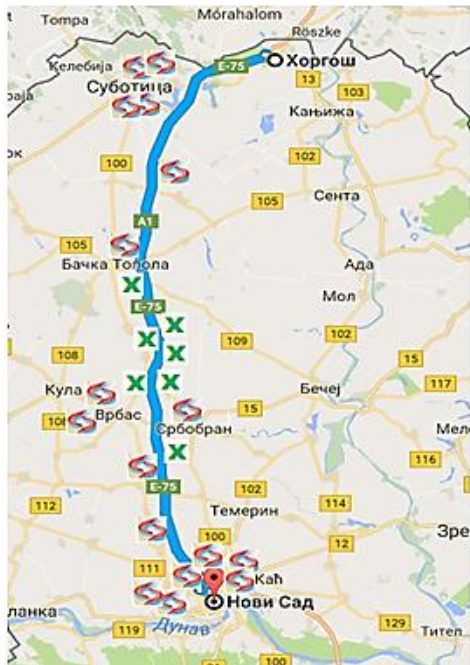
Ukoliko su zadovoljeni uslovi za sve tri situacije, odnosno njihovi uslovi tek onda se može reći da je raspodela na nekoj određenoj deonici na ispravan način učinjena.

7. PREDLOG RASPODELE STANICA PUNJENJE ELEKTRIČNIH VOZILA DUŽ KORIDORA

Podacima dobijenim iz algoritma rutiranja autoputa Horgoš – Novi Sad se definiše optimalna deonica za postavku EV stanice. Optimalna deonica za ovaj deo puta je Bačka Topola – Srbobran.

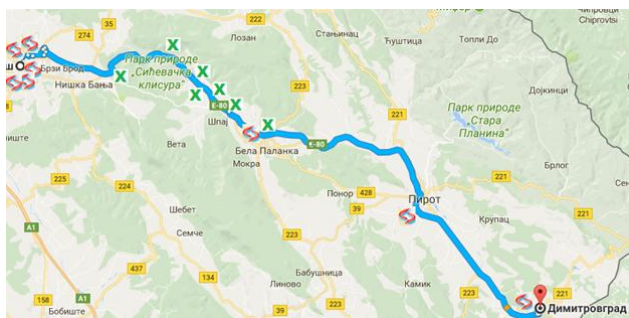
Međutim, uzimanjem dodatnog kriterijuma blizinu benzinskih stanica dobija se preklapanje sa potencijalnim mestom iz deonice.

Za deonicu Bačka Topola – Srbobran su to dve lokacije jedna je upravo kod Bačke Topole, druga je kod Novog Sada. Rastojanje između njih je 72 km.



Slika 7. Deonica Horgoš-Novi Sad – Potencijalna mesta za instaliranje EV stanica za punjenje električnih vozila (potencijalna mesta su obeležena sa X)

Autoput Niš – Dimitrovgrad (E80) je dužine 86,9 km. Iz dobijenih rezultata (Slika 8.) se vidi optimalna deonica za postavku stanica za punjenje električnih vozila kada sa starta (Dimitrovgrad) kreću sa 50 % baterije. Optimalna deonica za ovaj deo puta je od Niške banje do Bele Palanke. Sa unosom postojećih benzinskih stanica dobija se preklapanje u jednoj tački – Bela Palanka.



Slika 8. Potencijalna mesta za instaliranje EV stanica za punjenje električnih vozila na deonici Niš – Dimitrovgrad (potencijalna mesta su obeležena sa X)

Autoput E763 Beograd – Boljare (Južni Jadran) je dužine 269 km. Polazi se od toga da je u Beogradu postavljena EV stanica.

Prvo se ispituje situacija kada vozila sa starta kreću 100 % napunjena. U ovom slučaju skoro sva vozila ne mogu da

preći posmatranu trasu bez nijednog punjenja, odnosno ne mogu direktno doći do cilja.

Pri startu sa napunjenom baterijom vidi se da je optimalna deonica za postavku EV stanica Preljine – Požege za ovu trasu.

Pri startu sa 50 % napunjenosti baterije potrebno je planirati stanicu u deonici Ub - Ljig, tačnije kod Lajkovca. Pošto se obično EV stanice planirane pored puteva instaliraju kod benzinskih pumpi, uzima se i ovaj faktor u obzir. Pored navedenih lokacija potrebno je uzeti u obzir i Sjenicu, tako da je na ovoj trasi poželjno imati minimum tri potencijalna mesta za postavku stanica.

8. ZAKLJUČAK

Prema podacima iz MUP-a dostavljenih Tanjugu vidi se da je u Srbiji registrovano svega 128 električnih vozila, pri čemu su u tu brojku uključeni i električni skuteri, električni autobusi i električni trolejbusi. Pritom je procenjeno da je čak 90 vozila registrovano u Beogradu.

Stoga se može slobodno reći da je Srbija na samom početku po pitanju upotrebe ovih vozila upravo zbog nedovoljnih ulaganja i nepostojanja potrebne infrastrukture. Međutim, u narednih par godina vlada je obećala znatna ulaganja kako domaća tako i strana, kao i stvaranje odgovarajućih normi i propisa.

9. LITERATURA

- [1].Tina Jakaša, Stjepan Sučić, Nikica Mikulandra, "Infrastruktura za punjenje električnih vozila", novembar 2011. godine
- [2].Qi Han, Bauke de Vries, Geet Kanter, "An agent-based multi-objective optimization model for allocation public charging stations for electrical vehicles", 2015. godine
- [3].Schrack katalog, "Stanice za punjenje, Elektromobilnost", 2017. godine

Kratka biografija:



Milica Repac rođena je u Zrenjaninu 1992 god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva - Energetska elektronika i električne mašine odbranila je 2017. god.



Vladimir Katić rođen je 1954. godine u Novom Sadu. Doktorirao je na Univerzitetu u Beogradu 1991. godine. Od 2002. godine je redovni profesor Univerziteta u Novom Sadu. Trenutno je prodekan Fakulteta tehničkih nauka i šef katedre za Energetsku elektroniku i pretva-rače. Oblasti interesovanja su energetska elek-tronika, obnovljivi izvor električne energije, električna vozila i kvalitete električne energije.

PREGLED NEURONSKIH MREŽA SA PRIMERIMA PRIMENE

REVIEW OF NEURAL NETWORKS WITH EXAMPLES OF APPLICATION

Stanko Kuveljić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Tema ovog rada jeste pregled neuronskih mreža uz dva realna primera primene. Prvi primer predstavlja analizu slika lica, dok je drugi generisanje teksta. Ideja koja leži iza analize slika lica jeste da se koristi neuronska mreža koja će da za datu sliku lica odredi neke osnovne karakteristike kao što su pol, godine, boja kose, prisutnost kape, nakita, šminke, naočara, brade, brkova itd. Za potrebe ovog projekta korišćeni su javno dostupni skupovi podataka sa obeleženim slikama lica. Za rešavanje problema primenjena je tehnika polunadgledanog učenja. Korišćen je konvolucionni *stacked denoising autoencoder* praćen *fully connected* (potpuno spojenim) slojevima. Ovakav pristup postigao je tačnost za prepoznavanja pola od 98% na test skupu podataka i *root mean square error* od 7 godina za predikciju broja godina. Za projekat generisanja teksta primenjena je tehnika rekurentnih neuronskih mreža, tačnije jedna modifikovana verzija koja se zove *Long short term memory (LSTM)*. Ideja ovog projekta je automatsko generisanje novog karaktera na osnovu prethodno generisanih karaktera. Obučena su dva modela. Jedan model je obučen nad tekstovima pesama sastava *Ghost*, dok je drugi model obučen nad tekstovima blogova sa sajta www.smartcat.io. Rezultujući tekstovi pesama su se pokazali kao bolji za razliku od tekstova blogova, najviše zato što su tekstovi pesama mnogo manji po obimu od tekstova blogova.

Abstract – This paper presents a review of neural networks with two examples of real world applications. The first example represents an analysis of images containing faces, while the second problem is text generation. An idea behind the face image analysis is to use a neural network, which will for given image of a face return a prediction of gender, age, hair color, jewelry, makeup, eyeglasses, beard, mustaches, etc. For the purpose of this project we used a publicly available labeled data set and a semi-supervised learning technique called *convolution stacked denoising autoencoder* followed up with *fully connected* layers. The approach (evaluated on the on the test set) achieved an accuracy of 98% for gender and *root mean square error* of 7 years for the prediction of age. The *Long Short Term Memory (LSTM)* recurrent neural network was applied to the text generation problem. The aim was to automatically generate of a new character based on the previously generated characters. The models were trained on song's lyrics of band *Ghost*, and blogs retrieved from www.smartcat.io. The results show that shorter texts such as songs can be generated with higher quality than blogs.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kovačević, vanr.prof.

Ključne reči: *Neuronske mreže, duboko učenje, detekcija lica, konvolucione neuronske mreže, autoenkoderi, rekurentne neuronske mreže, generisanje teksta, Long short term memory, prepoznavanje pola i broja godina, analiza slika lica.*

1. UVOD

Ovaj rad predstavlja pregled neuronskih mreža uz dva projekta na kojima je radio autor, a to su predikcija atributa za slike lica i generisanje teksta. Oba projekta su rađena isključivo upotrebom tehnika neuronskih mreža i *deep learning*-a (dubokog učenja). Kao *deep learning framework* korišćen je *TensorFlow*¹ u programskom jeziku *Python*.

Facelyzr predstavlja projekat na kojem autor ovog rada radi u okviru firme *SmartCat* iz Novog Sada. Ideja ovog projekta je analiza lica pomoću *deep learning*-a. Tačnije, aplikacija je da na osnovu slike lica prepozna odlike kao što su broj godina i pol, ali i ostale karakteristike kao što su: dužina kose, boja kose, prisutnost naočara, brade, šminke, nakita, prepoznavanje emocija... Za potrebe ovog projekta korišćen je obeleženi skup podataka *CelebA* [1] koji sadrži lica poznatih osoba prikupljenih sa interneta. Svaka slika u skupu podataka je obeležena sa 40 atributa. Za predikciju broja godina korišćen je skup podataka *IMDB* [2] koji za svaku sliku sadrži podatak o broju godina.

Drugi projekat na kojem je autor radio je generisanje teksta pomoću rekurentnih neuronskih mreža. Ideja je da se neuronska mreža nauči da generiše karakter po karakter tako što će uzimati u obzir trenutnu generisanu istoriju. U ovom radu korišćeni su tekstovi pesama sastava *Ghost*² kao jedan skup podataka, dok su za drugi skup podataka korišćeni tekstovi blogova sa www.smartcat.io.

2. SRODNI RADOVI

Prepoznavanje i detekcija lica predstavljaju predmet mnogobrojnih istraživanja. Rad [3] se bavi problemom detekcije lica, ali i pronalaženja delova lica kao što su oči, nos, usta, kosa... Ovaj rad je koristio duboku konvolucionu mrežu i postigao tačnost u detekciji lica 90.99% na test skupu podataka i time prevazišao do tada *state of art* rešenje za 2.91%. Konvolucionna neuronska mreža je prethodno pretrenirana na slikama lica nenadgledano, zatim je za svaki atribut lica istreniran poseban klasifikator koji su izvršili fino podešavanje celokupne neuronske mreže. Ideja pretreniranja i finog podešavanja neuronske mreže potiče od jednog od radova [4] koji su uveli ideju nenadgledanog

¹ Framework za mašinsko učenje: www.tensorflow.org

² Korišćeni tekstovi pesama upotrebljeni su samo u istraživačke i edukativne svrhe. Ni na koji način nisu ugrožena autorska prava niti su generisane pesme monetizovane

učenja. U ovom radu, autori su pokazali da pretreniranje i fino podešavanje mogu da dovedu do boljih rezultata kako u rekonstrukciji slike tako i u raznim zadacima predikcije. Autori rada [2] bavili su se tematikom prepoznavanja pola i godina sa slike lica. Prvi zadatak njihovog rada bio je konstrukcija obeleženog skupa podataka. Sa interneta su preuzeli pola miliona slika poznatih osoba. Svaka slika ima podatak o broju godina i polu osobe sa slike. Korišćena je pretrenirana arhitektura *VGG16* [5] i na nju je dodat ansambl od 20 neuronskih mreža koji su istrenirani da vrše predikciju broja godina. Postignuti *mean absolute error* na test skupu iznosio je 3 godine. Rekurentne neuronske mreže su se pokazale kao odlične za obradu vremenski zavisnih podataka, kao što je slučaj sa tekstualnim podacima. U blogu [6], autor je uradio primer generisanja teksta na osnovu tekstualnog skupa podataka. Ideja je da se na osnovu prethodno unetog skupa karaktera odredi naredni karakter. Ideja generisanja teksta korišćenjem rekurentnih neuronskih mreža upotrebljena je u radu [7] za automatske mašinske prevode. Ulaz u neuronsku mrežu je tekst napisan na jednom jeziku, dok je izlaz tekst napisan na drugom jeziku. U radu je pokazano da je korišćenje *LSTM* neuronskih mreža dovelo do uspeha kako u prevodima fraza tako i u prevodima dužih rečenica. Rad [8] prikazuje upotrebu *LSTM* neuronske mreže u generisanju teksta na osnovu slike. Rezultati su postigli *state of art* u generisanju teksta na osnovu slike.

3. METODOLOGIJA

U ovoj sekciji biće prezentovane metodologije dubokog učenja korišćene u analizi slika lica, kao i za generisanje teksta. Oba projekta urađena su pomoću neuronskih mreža i tehnika dubokog učenja. Projekat za analizu lica koristi konvolucione neuronske mreže i autoenkodere, dok projekat za generisanje teksta koristi rekurentne neuronske mreže, odnosno njihovu modifikaciju *LSTM*.

3.1 Potpuno povezane neuronske mreže

Potpuno povezane (*fully connected*) neuronske mreže [9] predstavljaju tehniku mašinskog učenja koja je inspirisana biološkim sastavom neurona. Osnovna jedinica neuronske mreže je jedan neuron. Ulaz u neuron predstavljaju dendriti čiji signali se sumiraju i propuštaju kroz aktivacionu funkciju koja je zadužena za okidanje neurona. Ukoliko se neuron aktivira, njegova vrednost se šalje kroz akson u druge neurone. Mesta gde se spajaju akson i dendrit nazivaju se sinapse. Sinapse imaju svoje težine i zadužene su da propuštaju određenu količinu signala iz aksona u naredni neuron. Sa matematičke strane, neuron predstavlja funkciju [10]:

$$y_1 = f(W_1x + b_1) \quad (1)$$

gde y_1 predstavlja izlaz neurona, W_1 je matrica parametara težina gde broj vrsta predstavlja broj ulaza koje ima svaki neuron u sloju, a broj kolona predstavlja broj neurona u sloju. Parametar b_1 predstavlja *bias*. Funkcija f predstavlja funkciju aktivacije i zadužena je za unos nelinearnosti u neuronsku mrežu. Izlaz iz prvog sloja neuronske mreže y_1 predstavlja ulaz u drugi skriveni sloj neuronske mreže, a njen izlaz se izračunava kao:

$$y_2 = f(W_2y_1 + b_2) \quad (2)$$

Dodavanjem skrivenih slojeva omogućava neuronskoj mreži da nauči kompleksnije zavisnosti u podacima, ali isto tako usporava se proces obučavanja. Neuronska mreža se obučava nadgledanim učenjem, algoritmom koji se naziva *backpropagation* [11]. Ideja je da se za obučavajući par $(x1, y1)$ odredi izlaz neuronske mreže i zatim se na osnovu zadatog $y1$ izračuna koliko je neuronska mreža pogrešila. Dobijena greška se propagira u nazad i računa se koliko je svaki neuron uticao na grešku, tačnije računaju se prvi izvodi parametara po funkciji greške.

3.2 Konvolucione neuronske mreže

Problem *fully connected* neuronskih mreža je njihova kompleksnost. Ukoliko su ulazni podaci visokih dimenzija, kao što je u slučaju slika (na primer dimenzija $100 \times 100 \times 3$), gde svaki piksel predstavlja jedan ulazni neuron, mreža će imati veliki problem prilikom obučavanja. Problem visokih dimenzija rešile su konvolucione mreže [10]. Ideja konvolucionih mreža je da postoje filteri malih dimenzija (3×3 , 5×5), koji će da „klize“ po slici i primenjuju operaciju konvolucije, tako stvarajući modifikovanu izlaznu mapu. Filteri zapravo predstavljaju male neuronske mreže koje je potrebno obučiti tako da rade sa delovima ulaznih podataka. Tipična arhitektura konvolucione neuronske mreže sastoji se od konvolucionih slojeva sa filterima. Nakon svakog konvolucionog sloja nalazi se funkcija aktivacije. Za predikciju koriste se *fully connected* neuronske mreže koje se nalaze nakon poslednjeg konvolucionog sloja. Obučavanjem nad slikama, može se primetiti da su prvi slojevi konvolucione mreže zaduženi za otkrivanje ivica, boja, dok su naredni slojevi zaduženi za teksture, a poslednji slojevi su u stanju da detektuju kompleksne oblike kao što su lice, nos, oči.

3.3 Autoenkodori

Autoenkodori [9] predstavljaju tehniku nenadgledanog učenja čiji je cilj da pronade takvu unutrašnju strukturu u podacima koja ih najbolje opisuje. Može se svrstati u tehnike redukcije dimenzionalnosti. Autoenkoder se obučava tako što mu se na ulaz dovedu neki podaci (na primer slike), podaci se propuste kroz unutrašnju strukturu nazvanu enkoder. Rezultat enkodera je skriveni (latentni) vektor koji opisuje ulaz. Skriveni vektor se tada propušta kroz dekoder čiji je zadatak da dobije rekonstrukciju ulaza enkodera samo na osnovu skrivenog vektora. Ovim putem mreža će sama naučiti unutrašnju strukturu podataka.

3.4 Rekurentne neuronske mreže

Ideja rekurentnih neuronskih mreža (RNN) [6] je da obrađuju vremenski zavisne podatke (sekvence) kao što su prepoznavanje govora, mašinski prevodi sa jednog jezika na drugi, sinteza govora, teksta, itd. Vremenske zavisne podatke mogu da obrade veoma uspešno zahvaljujući svojoj unutrašnjoj strukturi koja čuva stanje i modifikuje stanje kroz vreme. Kao ulaz, RNN može da prima sekvence vektora, a kao izlaz može da daje sekvence vektora, jednu vrednost, ili jedan vektor vrednosti. Modifikacija RNN koja se više koristi u praksi je *Long short term memory (LSTM)* [14] koja uvodi *gate*-ove za manipulaciju unutrašnjim stanjem kao što su *forget*, *input* i *update*. *Forget gate* utiče na *reset* unutrašnjeg stanja na osnovu ulaza. *Input gate* određuje šta će od ulaza biti iskorišćeno za modifikaciju unutrašnjeg stanja, dok *update gate* računa nove vrednosti stanja ćelije kao i izlaz ćelije.

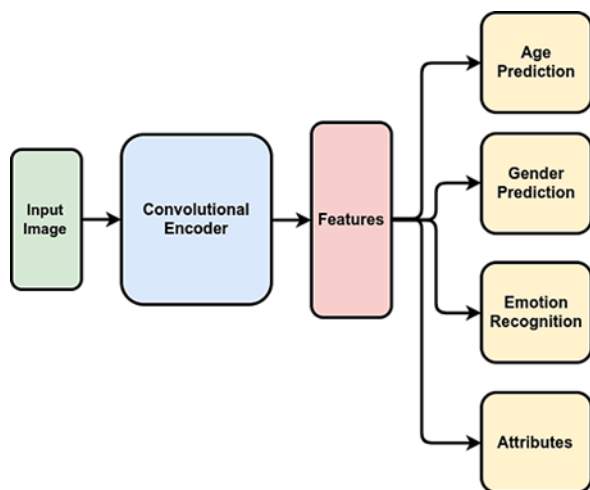
3.5. Prepoznavanje atributa lica

Za određivanje atributa sa slika lica korišćen je obeležen *CelebA* skup podataka koji sadrži 200000 slika poznatih. Svaka slika sadrži 40 atributa kao što su pol, boja kose, prisutnost brade, naočara, nakita, šminke itd. Za predikciju broja godina korišćen je skup podataka *IMDB* koji sadrži 450000 slika poznatih sa informacijama o broju godina osobe na svakoj slici.

Oba skupa su podeljena u obučavajući (90%) i test (10%) skup podataka. Pre obučavanja, sa svake slike su isečeni krajevi, tako što je uzeta sredina u frakciji 0.75. Zatim su odsečki skalirani na dimenzije 100x100x3. Nakon skaliranja, od svake vrednosti piksela oduzeta je srednja vrednost datog piksela. Takva obrađena slika predstavlja ulaz u neuronsku mrežu.

3.5.1 Arhitektura rešenja

Za obučavanje neuronske mreže korišćen je konvolucioni *stacked denoising autoencoder* [11]. Ideja ovakve vrste autoenkodera je da obučiti sloj po sloj nenadgledano, tako što će svaki sloj naučiti da rekonstruiše svoj ulaz. Kao dodatno poboljšanje kvaliteta obučavanja na svaki ulaz se u fazi obučavanje dovodi šum. Dovođenjem šuma, neuronska mreža će postati otpornija na razne anomalije nad ulaznim podacima. Nakon obučavanja potrebno je bilo obučiti prediktore za attribute lica, kao i prediktor za broj godina. Arhitektura rešenja se nalazi na slici 1.



Slika 1. Arhitektura neuronske mreže

Kao ulaz se dobija slika koja se propušta kroz enkoder deo obučenog autoenkodera, izlaz enkodera predstavljaju naučene karakteristike sa slike koji ujedno predstavljaju ulaz u *fully connected* prediktore. U tabeli 1. je data struktura konvolucione neuronske mreže (autoenkodera) koja je upotrebljena za ovaj zadatak. Kolona konv. sloj predstavlja redni broj konvolucionog sloja. Kolona Filter (dim) je dimenzija neuronske mreže koja predstavlja filter.

Ulazni broj kanala u prvi konvolucioni sloj je 3^3 . Broj filtera predstavlja broj filtera u konvolucionom sloju, takođe označava broj ulaznih kanala za sledeći sloj. Parametar *stride* označava koliko piksela se „preskače“ prilikom izvršavanja konvolucije u konvolucionom sloju.

³ Skup podataka su slike u boji, koje imaju tri kanala (crveni, zeleni i plavi) RGB (R-red, G-green, B-blue)

Tabela 1. Slojevi konvolucione neuronske mreže

Konv. Sloj	Filter (dim)	Broj filtera	Stride
1	5x5	64	2x2
2	3x3	128	2x2
3	1x1	128	1x1
4	3x3	256	2x2
5	3x3	256	1x1
6	1x1	256	1x1
7	3x3	328	2x2

Za predikciju korišćena je neuronska mreža sa dva *fully connected* sloja od 512 i 256 neurona.

3.6. Generisanje teksta

Za generisanje teksta korišćena su dva skupa podataka. Jedan predstavlja tekstove pesama sastava *Ghost*, dok drugi predstavlja tekstove blogova sa www.smartcat.io. Ideja je da se obuče dva modela koja će da automatski generišu tekst karakter po karakter. U moguće karaktere spadaju sva velika i mala slova, cifre, oznake za početak i kraj teksta, oznaka za novi red i znakovni interpunkcije. Skup podataka za tekstove pesama poseduje 34 pesme i svaka pesma sadrži oko 1000 karaktera. Skup podataka tekstova blogova sadrži 38 blogova i svaki blog sadrži oko 10000 karaktera. Svaki karakter je predstavljen u vektorskoj reprezentaciji kao *one hot encoding*.

3.6.1 Arhitektura rešenja

Za oba skupa podataka koristila se *LSTM* neuronska mreža od 2 sloja, svaka sa vektorom stanja dužine 256. Neuronskoj mreži se dovodi obučavajući par (x, y) tako da x predstavlja trenutni karakter a y predstavlja naredni karakter. Neuronska mreža će da uz pomoć unutrašnjeg stanja računa naredno stanje neuronske mreže i računa vektor za naredni karakter. Drugim rečima, ovo je problem klasifikacije gde je cilj odrediti raspodelu verovatnoće za naredni karakter na osnovu prethodno unete istorije karaktera.

Nakon obučene neuronske mreže, predikcija se radila tako što se zada oznaka za početak teksta, i neuronska mreža računa naredni karakter. Zatim na osnovu narednog karaktera dobije se sledeći karakter. Generisanje teksta se završava nakon ispisanog određenog broja karaktera, ili dok se ne dođe do oznake za kraj teksta, čime je neuronska mreža obuhvatila jednu celinu.

4. REZULTATI

Primenom nenadgledanog učenja na slikama lica neuronska mreža je uspela dobro da se inicijalizuje i postavi za naredne korake nadgledanog učenja kao što su prepoznavanje broja godina, prepoznavanje pola, prepoznavanje atributa sa lica. Tačnost za prepoznavanje pola na test skupu podataka iznosi 98%, dok je *root mean square error* 7 za predikciju broja godina na test skupu podataka. Isprobana je detekcija naočara (98% tačnost), detekcija brade (98% tačnost).

Na problemu generisanja teksta, za oba skupa podataka *LSTM* je uspeo da nauči sintaksu jezika, sa veoma malim greškama. Što se tiče smislenosti generisanog teksta bolji rezultati su postignuti na generisanju tekstova pesama, iz razloga što su pesme manjeg obima i zato što su više „umetničkog“ karaktera. Primer generisane pesme:

"Here Comes The Sun"
 Little darling, it's been a long cold lonely winter
 Little darling, it feels like years since it's been here
 Harvesting crops of damnations

Guiding my way with its light
 And in the glow of the moon
 Know my deliverance will come soon

Kao što se vidi iz primera, *LSTM* je naučio da zada naslov, da nakon svakog stiha ide novi red, prazan red posle kraja strofe, nakon otvorene zgrade ide zatvorena zagrada. Rezultati za generisanje blogova su malo lošiji u smislu dobijanja smislenih rečenica, iako je neuronska mreža uspela da se izbori sa sintaksom jezika.

5. ZAKLJUČAK

U ovom radu prikazana su dva projekta na kojima je radio autor. Jedan je određivanje broja godina, pola i atributa sa slike lica, dok je drugi primena rekurentnih mreža za generisanja teksta. Određivanja atributa sa slike lica rešen je pomoću polunadgledanog obučavanja, tako što se prvo nenadgledno obučio autoenkoder, zatim se na deo enkodera doda *fully connected* neuronska mreža za predikciju. Ovaj pristup dao je tačnost od 98% za prepoznavanje pola i *root mean square error* je 7 za predikciju broja godina.

Projekat generisanje teksta pomoću rekurentnih neuronskih mreža postigao je dobre rezultate za generisanje pesama, dok je malo lošije rezultate postigao za generisanje blogova. U oba slučaja, neuronska mreža je uspela da nauči sintaksu jezika i da proizvede osnovna pravila kao što su naslovi, novi red, veliko slovo posle tačke itd.

Pravci daljeg rada su izgradnja celokupnog prepoznavanja lica sa slike, kao što su detekcija lica na slici, njegova lokalizacija, pronalaženje koordinata očiju, isecanje lica sa slike i zatim određivanje karakteristika, godina i pola sa isečka. Ideja za nastavak projekta za generisanje teksta jeste da se prikupi više različitih tekstova pesama i zatim da se obučio kompleksniji model nad takvim većim skupom tekstualnih podataka.

6. LITERATURA

- [1] Z. Liu et al., "Deep learning face attributes in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
- [2] R. Rasmus, R. Timofte and L. V. Gool., "Dex: Deep expectation of apparent age from a single image," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015.
- [3] S. Yang et al., "From facial parts responses to face detection: A deep learning approach," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
- [4] G. Hinton and S. Ruslan R., "Reducing the dimensionality of data with neural networks.," *science* 313.5786, 2006.
- [5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

- [6] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," [Online]. Available: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>. [Accessed 9 9 2017].
- [7] I. Sutskever, O. Vinyals and Q. V. Le., "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014.
- [8] O. Vinyals et al, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.
- [9] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning (Adaptive Computation and Machine Learning series)*, MIT, 2016.
- [10] A. Karpathy, "CS231n: Convolutional Neural Networks for Visual Recognition," Stanford, [Online]. Available: <http://cs231n.github.io/linear-classify/>. [Accessed 7 9 2017].
- [11] G. E. Hinton, D. E. Rumelhart and R. J. Williams, "Learning internal representations by error propagation.," in *MIT Press*, Cambridge, 1986.
- [12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation* 9.8, 1997.
- [13] P. Vincent et al., "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, 2010.

Kratka biografija:



Stanko Kuveljić je rođen 1993. godine u Somboru, a u Sivcu je završio osnovnu školu. Srednju školu je nastavio u gimnaziji „Žarko Zrenjanin“ u Vrbasu na prirodno-matematičkom smeru. Maturirao je 2012. godine sa odličnim uspehom. Fakultet tehničkih nauka upisao je 2012. godine, smer Računarstvo i automatika. Studije je završio u roku 2016. godine. Master akademske studije je upisao na Fakultetu tehničkih nauka 2016. godine, smer Računarstvo i automatika.

**МОДЕЛ КЛИМА КОМОРЕ СА АКЦЕНТОМ НА УЛОГУ И УПРАВЉАЊЕ
ВЕНТИЛАТОРОМ У ЊИМА****AIR HANDLING UNIT MODEL WITH FOCUS ON ROLE AND MANAGEMENT OF THE
FAN IN THEM**

Душан Вукославовић, Велимир Чонградац, Веран Васић; *Факултет Техничких Наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – Рад се базира на анализирању управљања вентилатором у клима коморама употребом фреквентног претварача, које се заснива на промени фреквенције тако да се одржи константан статички притисак у потисном, односно одсисном ка-налу. Симулација је извршена употребом PID и FUZZY регулатора.

Abstract – This study is based on analysing the management of the fan in the air handling units with VRF which role is to keep static pressure in AHU ducts by changing frequency of fans. Simulation is realised with PID and FUZZY regulators.

Кључне речи: Фреквентни исправљач, BMS, статички притисак

1. УВОД

Системи климатизације, грејања и хлађења - КГХ, тј. HVAC (енг. Heating, Ventilation and Air Conditioning) играју централну улогу у регулисању унутрашњих услова комфора и здраве средине за људе у згради. Потрошња енергије у зградама широм света расте на различитим нивоима- стамбеном, комерцијалном, као и индустријском.

HVAC системи су важни и веома популарни у дизајну средњих и великих индустријских и пословних зграда јер обезбеђују повољне услове за рад. Под повољним радним условима подразумева се одржавање жељене температуре ваздуха, жељене влажности ваздуха, потребне количине кисеоника и угљен-диоксида итд. Због тога људи покушавају да креирају удобније окружење. Као резултат тога, последњих година расте интересовање за студије у вези са пројектовањем и контролом амбијенталних услова.

Оптимална конфигурација HVAC система постиже пројектоване унутрашње услове комфора са минималном потрошњом енергије и трошковима. Различити се приступи користе у истраживањима у којима се анализирају карактеристике HVAC система. Општи критеријуми и аспекти оптималне конфигурације HVAC система се идентификују коришћењем енергетског биланса. Истраживања су показала да енергије из постојећих извора има све мање, па се у последње време све више говори о енергетској ефикасности.

НАПОМЕНА:

Овај рад произтекао је из мастер рада, чији ментор је био др Веран Васић, ред. проф.

2. ПАМЕТНЕ ЗГРАДЕ

Статус паметне зграде добијају сви објекти комерцијалне или индустријске намене које имају имплементиран BMS систем, често превођен на српски као ЦСНУ (Централни Систем за Надзор и Управљање). BMS систем представља рачунарски базиран систем управљања инсталиран у зградама, који контролише и надзира механичку и електричну опрему, такође, представља управља различите подсистеме попут [1]:

- Управљање HVAC системима
- Аутоматизација расвете
- Управљање ролетнама
- Управљање базенском техником
- Интеграција са различитим системима (попут лифтова, UPS-еви, DEA, итд.)
- Сигнализација свих електропотрошача
- Мерење електричне и топлотне енергије

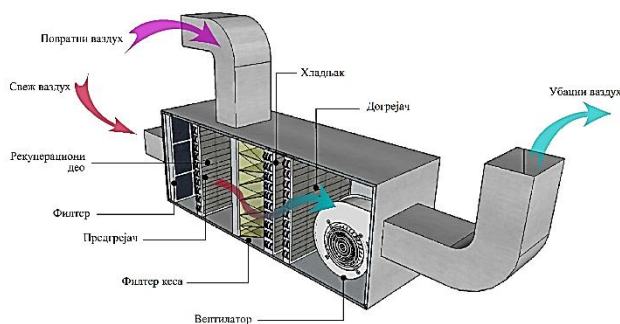
3. КЛИМА КОМОРЕ

Клима коморе су намењене за централну припрему ваздуха и омогућавају основне функције:

- Грејање
- Хлађење
- Филтрација
- Влажење
- Сушење
- Рекуперација
- Рецикулација

Делови клима коморе приказане на слици су:

- Рецикулациони део - користи се као опција ако се врши рецикулација собног ваздуха
- Филтерска секција - користи се за пречишћавање ваздуха који се уводи у систем
- Грејач - може бити подељен на предгрејач и догрејач, користи се за загревање ваздуха довођењем топле воде из подстанице на њега
- Хладњак - користи се за расхлађивање ваздуха системом довођењем хладне воде на њега
- Вентилаторска комора
- Центрифугални вентилатори, обично фреквентно вођени, који врше убацивање ваздуха у простор



Слика 1. Изглед клима коморе

Клима коморе су погодне за примену у свим великим објектима који поседују велики број одвојених и затворених просторија без могућности природне вентилације. Примењују се за вентилацију и климатизацију болница, индустријских хала, складишта, пословних и продајних простора, банака, пошта, спортских дворана, јавних простора и сл. Осим обезбеђивања жељене температуре у климатизованом простору улога система за климатизацију јесте и да обезбеди:

- Одговарајућу релативну влажност ваздуха
- Одговарајућу количину свежег ваздуха за вентилацију
- Потребан ниво чистоће, односно квалитета ваздуха

4. ПОТРЕБА ЗА ВЕНТИЛАЦИЈОМ

CO₂ је продукт дисања. Како људи издишу, CO₂ се пушта у околину. Људи стварају CO₂ у предвиђеној количини дисањем и као резултат, CO₂ је одлична замена за одређивање присуства. Константна количина CO₂ издахнутог од стране људи је разређена спољашњим ваздухом који се убацује у простор било механичком вентилацијом или инфилтрацијом. Измена ваздуха има потенцијал за велику уштеду енергије уз очување квалитета унутрашњег ваздуха. Ове уштеде се остварују аутоматским подешавањем количине спољашњег ваздуха који се уводи у систем HVAC. Минимална количина свежег ваздуха потребна за дисање је стварно мала око 0,2 l/s по особи. За одржавање угодних услова у једној средини, није довољно доводити само овако мале количине свежег ваздуха. Количине ваздуха веће од минималних морају се доводити да би се постигло:

- задовољење потреба у кисеонику за људе у просторији
- разблаживање присутних мириса до одређеног прихватљивог нивоа,
- разблаживање концентрације CO₂ до задовољавајућег нивоа.

Разблаживање мириса постиже се довођењем свежег спољашњег ваздуха, или се мириси могу одстранити употребом филтара са активним угљем. Количина ваздуха која се доводи у просторију зависи пре свега од количине произведених мириса, као и намене саме просторије. CO₂ је присутан у свежем ваздуху од 0,03% док су концентрације од 0,1% прихватљиве у климатизованим и вентилираним просторијама, које су заузеле људима. Из овога произилази да је

довођење свежег ваздуха убедљив начин за смањење концентрације CO₂ у просторији. Концентрација CO₂ се наравно не може смањити испод 0,03% и ефикасност изабраног броја измена ваздуха, пошто људи у просторији приликом дисања континуално производе CO₂. У ствари, количина CO₂ коју производи један човек износи $4,72 \cdot 10^{-3} \text{ l/s}$. Количина свежег ваздуха која је неопходна да одговори горњим захтевима зависи од густине заузетости простора и његове намене. Ова количина је изражена у три различита облика:

- као број измена ваздуха (n - измена запремине простора у једном часу)
- као количина ваздуха која се доводи у l/s по човеку, икао количина ваздуха која се доводи у l/s по m² пода

5. ЕНЕРГЕТСКА ЕФИКАСНОСТ КЛИМА КОМОРА

Енергетска ефикасност се не сме посматрати као енергетска уштеда или штедња енергије, јер штедња представља нека одређена одрицања, док ефикасна употреба енергије не нарушава никад услове рада и живљења, али енергетска ефикасност може довести до пропорционалних уштеда новца.

Мере које се најчешће подразумевају у циљу повећања енергетске ефикасности и смањењу губитака енергије су:

- Замена енергетски неефикасних потрошача ефикасним
- Изолација простора који се греје или хлади
- Замена необновљивих енергената обновљивим
- Уградња мерних и регулационих уређаја за потрошаче енергије
- Контрола уласка сунчеве светлости и топлоте у простор
- Увођење тарифних система

Енергетска ефикасност би требало да обезбеди повећање конкурентности на микро и макро плану кроз смањење трошкова енергије, развој нових услуга и производа, повећање запослености, смањење утицаја на животну средину, првенствено кроз мању емисију CO₂.

Контрола HVAC системима на оптималном енергетско-ефикасном нивоу у комерцијалном окружењу поприлично је компликован, јер да би рад HVAC система у великим зградама, у којима се уграђују BMS системи, био ефикасан мора се на јасан начин дефинисати рад свих компоненти система попут: котлова, чилера, пумпи, вентилатора, сетпоинта температура, влажности, угљен диоксида, итд.

6. ФРЕКВЕНТНИ ПРЕТВАРАЧ У КЛИМА КОМОРАМА

Фреквентни претварач су електронски уређаји који омогућавају управљање брзином асинхроних мотора претварајући улазни мрежни напон и фреквенцију, који су иначе фиксирани вредности, у променљиве величине.

Вентилатори са променљивом брзином представља вентилатор који је опремљен са фреквентним претварачем. Како њихова брзина зависи од фреквенције на-

пајања, ови вентилатори су често називани вентилаторима променљиве фреквенције- VFF (енг. *Variable Frequency Fan*). Радна карактеристика чини вентилаторе идеалним за инсталацију фреквентних претварача. Такође, утрошак енергије вентилатора сразмеран је трећем степену брзине што их чини погодним за уштеду енергије која се постиже смањивањем брзине обртања са одговарајућом контролом [2].

Проток ваздуха, притисак и снага у директној су вези са брзином обртања. Ове везе дате су преко следећих једначина:

$$\frac{Q_1}{Q_2} = \frac{N_1}{N_2} \quad (1)$$

$$\frac{P_1}{P_2} = \left(\frac{N_1}{N_2}\right)^2 \quad (2)$$

$$\frac{W_1}{W_2} = \left(\frac{N_1}{N_2}\right)^3 \quad (3)$$

6.1. Уштеда енергије употребом фреквентног претварача

Односи између притиска, протока, брзине обртања осовине, снаге могу се изразити законима афинитета датим на следећем графику који је добијен из формула (1), (2) и (3).

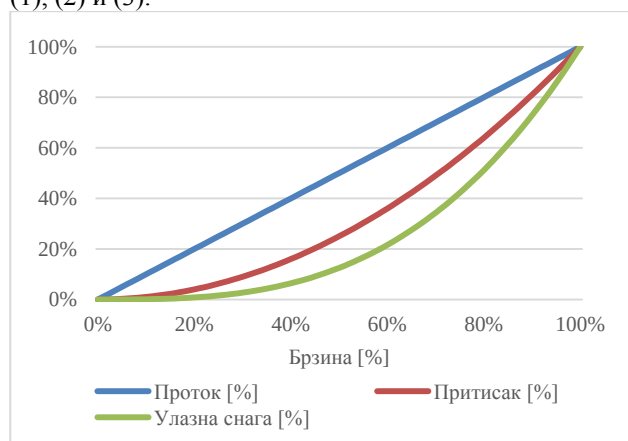


График 1. Закони афинитета

Проток је директно пропорционалан са брзином, притисак је пропорционалан квадрату брзине, док је, што је са становништа уштеде енергије најбитније, снага пропорционална трећем степену брзине [3].

Употребом фреквентних претварача постижу се велике уштеде у контроли процеса где су оптерећење или брзина променљиви. Нарочите уштеде се постижу код контроле оптерећења који имају променљиви моменат оптерећења у зависности од брзине а то су управо вентилатори.

Целокупан HVAC систем се дизајнира према највећим захтеваним вредностима целокупних процесних променљивих (проток, притисак, температура...), што значи да су вентилатори предимензионисани током највећег дела радног времена.

7. СОФТВЕРСКО РЕШЕЊЕ

7.1. Модел система

У раду је приказано пројектовање и моделовање *fuzzy* и *PID* регулатора. Моделовање је реализовано у оквиру програмског пакета *Matlab* уз помоћ *Simulink* алата. Пројектовање *PID* регулатора вршено је преко

функције *PID Tuner*, док је за пројектовање *fuzzy* регулатор је коришћен *Fuzzy Logic Toolbox* алат са графичким интерфејсом. Из разлога што подаци прикупљени са реалног система нису доступни, софтверско решење се заснива на симулацији. Резултати добијени помоћу симулација биће приказани преко алата *2-D simulink plot maker*. Модел се састоји из следећих елемената, односно блокова:

- *PID* регулатор који моделује фреквенцију вентилатора убацног ваздуха
- *PID* регулатор који моделује фреквенцију вентилатора избацног ваздуха
- *FUZZY* регулатор који моделује фреквенцију вентилатора убацног ваздуха
- *FUZZY* регулатор који моделује фреквенцију вентилатора избацног ваздуха
- Мотор вентилатора убацног ваздуха
- Мотор вентилатора избацног ваздуха
- Блок вентилатора убацног ваздуха
- Блок вентилатора избацног ваздуха
- Филтер ваздуха убацног ваздуха
- Сензор притиска убацног ваздуха
- Сензор притиска избацног ваздуха
- Блок мерења потрошње мотора вентилатора убацног ваздуха
- Блок мерења потрошње мотора вентилатора избацног ваздуха

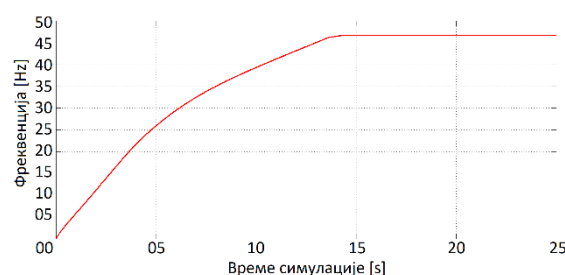
8. РЕЗУЛТАТИ И ДИСКУСИЈА

Симулација је спроведена тако што су сви улазни параметри у *Simulink* модел предефинисани и унесени у сам модел. У наредним примерима ће бити предочени резултати оба регулатора.

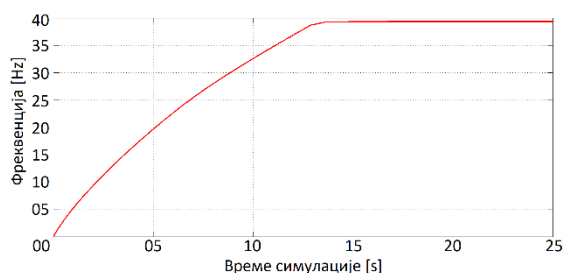
Илустровано је понашање система када су параметри за које је вршена симулација постављени на следеће вредности:

- Команда притиска у каналу убацног ваздуха: 360Pa
- Подаци вентилатора убацног ваздуха:
 - Снага 4kW
 - Клизање 0.0267
 - Број пари полова 2
- Команда притиска у каналу избацног ваздуха: 250Pa
- Подаци вентилатора избацног ваздуха:
 - Снага 1.1kW
 - Клизање 0.05
 - Број пари полова 2

8.1. Регулација *FUZZY* регулатором



Слика 2. Промена фреквенције мотора вентилатора улазног ваздуха

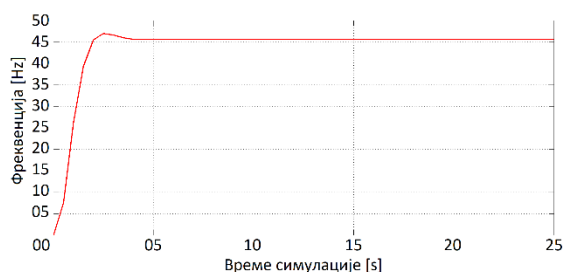


Слика 3. Промена фреквенције мотора вентилатора излазног ваздуха

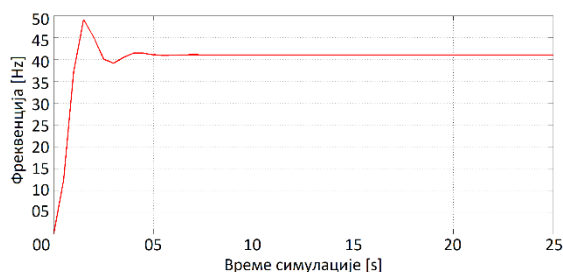
	Примена <i>Fuzzy</i> регулатора		Без регулисане фреквенције	
	Снага	Потрошња	Снага	Потрошња
Вентилатор убацног ваздуха	3.25 kW	78 kWh	4 kW	96 kWh
Вентилатор избацног ваздуха	0.55 kW	13.2 kWh	1.1 kW	26.4 kWh

Табела изнад приказује разлику у потрошњи електричне енергије када се примењује *Fuzzy* регулатор приликом управљања фреквенцијом и случаја када вентилатор нема регулисану фреквенцију. Од карактеристика вентилатора (снага, клизање, број пари полова...) и од подешене вредности притиска у каналу зависи промена снаге, самим тим и промена потрошње електричне енергије.

8.2. Регулација *PID* регулатором



Слика 4. Промена фреквенције мотора вентилатора улазног ваздуха



Слика 5. Промена фреквенције мотора вентилатора излазног ваздуха

Табела испод приказује разлику у потрошњи електричне енергије када се примењује *PID* регулатор приликом управљања фреквенцијом и случаја када вентилатор нема регулисану фреквенцију.

	Примена <i>PID</i> регулатора		Без регулисане фреквенције	
	Снага	Потрошња	Снага	Потрошња
Вентилатор убацног ваздуха	3.25 kW	78 kWh	4 kW	96 kWh
Вентилатор избацног ваздуха	0.55 kW	13.2 kWh	1.1 kW	26.4 kWh

У случају убацног ваздуха, дошло је до уштеде од 19%, док је код избацног ваздуха потреба притиска у каналу мања у односу на димензије и снагу вентилатора, па је дошло од уштеде од чак 50%. Исте резултати се добију применом оба регулатора, Осим што се време устаљеног стања (смирења фреквенције) разликује, а за регулаторе има следеће вредности:

- *Fuzzy* регулатор 13 s - 14.5 s
- *PID* регулатор 6 s

Редукација потрошње једнака.

9. ЗАКЉУЧАК

Задатак овог рада била је симулација рада клима коморе, односно вентилатора убацног и избацног ваздуха и регулација њихових брзина на основу количине притиска у потисном односно одсисном каналу. Након упознавања са појмом паметних зграда, односно BMS система и значајом и улогом клима комора, описан је начин рада фреквентног претварача и енергетским ефикасностима које његова примена доноси систему. Задатак је реализован применом два пројектована регулатора: *Fuzzy* и *PID*. Вршена је симулација са пројектованим регулаторима и посматрана је динамика система. Приказани су резултати симулације.

На основу симулације може се доћи до закључка да постоји могућност уштеде електричне енергије, поготово у односу на случај да се уопште не одабере фреквентни претварач за вођење брзине вентилатора, што је дато у табелама након приказаних одзива симулације.

10. ЛИТЕРАТУРА

- [1] BMS системи, преузето са странице: http://en.wikipedia.org/wiki/building_management_system
- [2] Yunhua Li – Variable Frequency Drive Applications in HVAC Systems, Bes-Tech Inc., Omaha, NE, USA, 2015
- [3] Фреквентни претварач, преузето са странице: <http://www.momentum-automaton.com/>

Кратка биографија:



Душан Вукославовић, рођен је 29.04.1991. године у Суботици. Факултет Техничких Наука у Новом Саду, одсек Електротехника и рачунарство уписује 2010. године. Дипломирао је у октобру 2014. године и након тога уписује мастер академске студије на Факултету Техничких Наука.

**PRIMENA GENETSKOG ALGORITMA ZA ODREĐIVANJE PARAMETARA
EKVIVALENTNE ŠEME ASINHRONOG MOTORA****APPLICATION OF GENETIC ALGORITHM FOR DETERMINING THE PARAMETERS
OF EQUIVALENT CIRCUIT OF ASYNCHRONOUS MOTOR**

Mihajlo Fejsa, Veran Vasić, Đura Oros, *Fakultet tehničkih nauka, Novi Sad*

Oblast - ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Kako bi se postiglo upravljanje asinhronim motorima neophodno je poznavanje parametara ekvivalentne šeme asinhronog motora. Postoji više načina određivanja parametara. Jedan od njih je konvencionalan način koji se zasniva na ogledu praznog hoda, ogledu kratkog spoja i merenju otpornosti statorskog namotaja. Ovaj rad se bavi određivanjem parametara pomoću genetskog algoritma. Postavka optimizacionog problema je opisana na dva različita pristupa. Prvi pristup je nazvan „offline“, gde se kao zadate veličine koriste kataloški podaci motora i to su polazni, prevalni i nominalni momenti motora. Drugi pristup je nazvan „online“ koji zahteva merenu struju motora, klizanje i faktor snage u nekoliko različitih tačaka opterećenja.

Abstract – In order to achieve the control of asynchronous motors, it is necessary to know the parameters of an equivalent circuit of asynchronous motor. There are several ways of parameters determination. One of them is a conventional way based on no load test, blocked rotor test and measurement of stator winding resistance. This paper deals with the determination of parameters using a genetic algorithm. The optimization problem setup is described in two different approaches. The first approach is called "offline", where the required values are used from manufacturer's catalog, and they are the rated, breakdown and starting torque. The second approach is called "online" which for input values requires measured motor current, slip and power factor at several different load points.

Ključne reči: Genetski algoritam, optimizacioni algoritam, asinhroni motor, parametri ekvivalentne šeme, električne mašine.

1. UVOD

Parametre ekvivalentne šeme je moguće odrediti na različite načine. Jedan od najpoznatijih načina je pomoću ogleda praznog hoda i kratkog spoja uz merenje otpornosti statorskog namotaja.

Ovaj rad se bavi određivanjem parametara ekvivalentne šeme pomoću optimizacionog algoritma. Optimizacioni algoritmi predstavljaju moćan alat, koji ima veoma široku primenu za rešavanje različitih optimizacionih problema.

U radu je detaljno opisan jedan od mnogobrojnih optimizacionih algoritama, a to je genetski algoritam čiji je cilj

da se pronađe globalni minimum funkcije. Nastojace se rešiti dve različite postavke optimizacionog problema koje su predstavljene kriterijumskim funkcijama. Cilj je da se pomoću genetskog algoritma odrede vrednosti promenljivih u kriterijumskoj funkciji tj. parametri ekvivalentne šeme asinhronog motora tako da vrednost kriterijumske funkcije za te vrednosti bude što manja. Kriterijumska funkcija je definisana da daje informaciju o grešci koja je nastala kod određenih parametara. U radu su opisana oba navedena načina za određivanje parametara ekvivalentne šeme i upoređeni su poznati i određeni parametri.

2. GENETSKI ALGORITAM

Genetski algoritam spada u tehnike pretrage koje teže da pronađu što približnije rešenje za problem optimizacije uz zadovoljenje raznih kriterijuma konvergencije. Njegova primena je široka, a neke oblasti primene su neuronske mreže, ostvarivanje veštačke inteligencije, pretraga geografskih mapa, uključujući razne kriterijume, kao i optimizacija upita nad bazama podataka.

Genetski algoritam se zasniva na upravljanju populacijom koja se sastoji od jedinki (hromozoma). Pod jedinkom se podrazumeva pojedinačno rešenje problema dok je populacija skup više jedinki koje su trenutno uključene u proces traženja pogodnog rešenja. Jedinke mogu biti kodirane na različite načine. Najčešći oblik kodiranja je binarno kodiranje gde se za predstavu jedinke koriste nizovi od nekoliko bita. Nad tako kodiranim jedinkama, primenjuju se operatori reprodukcije.

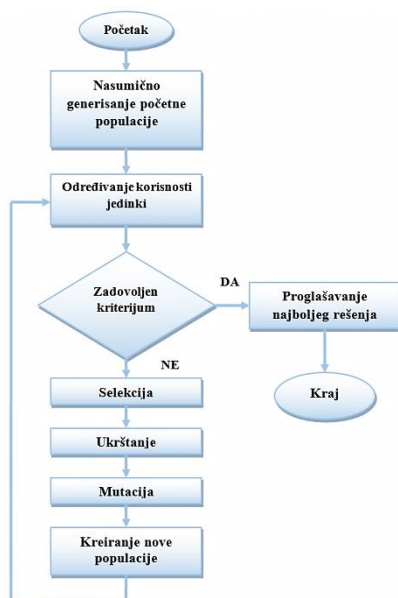
Operatori utiču direktno na jedinke kako bi se mogle ostvariti razne varijacije i rekombinacije nad mogućim rešenjima problema. Pod operatorima se smatra proces selekcije, ukrštanja i mutacije. Selekcijom se postiže upoređivanje jedinke u populaciji i kao rezultat se dobija prenošenje dobrih svojstava jedinki na nove u sledeću generaciju. Poređenje jedinki se vrši na osnovu korisnosti jedinke. Korisnost govori koliko je ta jedinka kvalitetna, a određuje se pomoću funkcije korisnosti (eng. *fitness function*), veoma čest naziv je i kriterijumska funkcija (eng. *objective function*). Prenos kod selekcije se postiže kada je korisnost jedinke tj. vrednost funkcije za određenu jedinku veća. Princip rada genetskog algoritma je prikazan kroz dijagram toka koji je dat na slici 2.1. [1]

- **Početak** – Generiše se početna populacija od N jedinki, koja sadrži moguće rešenje za posmatran optimizacioni problem. Zahteva se da populacija ima što veću raznolikost genetskog materijala, za to je dovoljno nasumično generisanje jedinki.
- **Korisnost** – Određivanje korisnosti svake jedinke u prethodno generisanoj populaciji.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Veran Vasić.

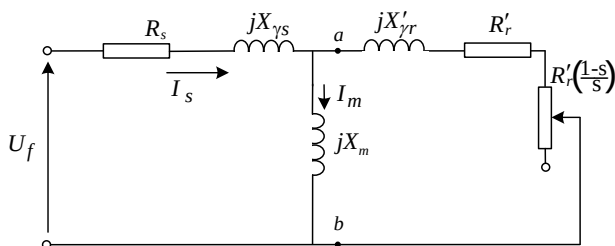
- **Kreiranje nove populacije** – Pokreće se petlja kojom će se stvoriti nova populacija, a ona se sastoji iz sledećih operacija:
 - **Selekcija** – Iz postojeće populacije biraju se jedinke (roditelji) u zavisnosti od njihove korisnosti. Što je bolja korisnost, veća je šansa da će biti odabrani selekcijom.
 - **Ukrštanje** – Ukrštaju se odabrane jedinke kako bi nastale nove jedinke (potomci).
 - **Mutacija** – Predstavlja promenu jednog ili više gena (bita) jedne jedinke, tako da kao rezultat daje izmenjenu jedinku.
 - **Prihvatanje** – Postavljanje nove jedinke u novu populaciju.
- **Zamena** – Koristi se nova populacija za sledeću iteraciju kroz petlju.
- **Ispitivanje kriterijuma** – Ispituje se kriterijum zaustavljanja algoritma. U slučaju ispunjenja, algoritam se zaustavlja i kao rezultat vraća najbolju jedinku (rešenje). U slučaju da nije zadovoljen kriterijum, kompletan ciklus se ponavlja.



Slika 2.1. Dijagram toka genetskog algoritma

3. POSTAVKA OPTIMIZACIONOG PROBLEMA

Za postavku optimizacionog problema koristi se kriterijumska funkcija. Kriterijumska funkcija se određuje polazeći od ekvivalentne šeme asinhronog motora (slika 3.1) u kojoj su rotorske veličine svedene na statorsku stranu i zanemareni su gubici u gvožđu mašine.



Slika 3.1. Ekvivalentna šema asinhronog motora

Gde je:

U_f – napon po fazi statorskog namotaja

R_s – otpornost statorskog namotaja

X_{ys} – reaktansa rasipanja statorskog namotaja

X_m – reaktansa magnećenja

X'_{yr} – svedena reaktansa rasipanja rotorskog namotaja na statorsku stranu

$R'r$ – svedena otpornost rotorskog namotaja na statorsku stranu

Prva postavka optimizacionog problema je nazvana „offline“ [2], gde se kao zadate veličine koriste kataloški podaci motora i to su polazni, prevalni i nominalni momenti motora. Naziv je dobila po tome što je za određivanje parametara dovoljno koristiti samo podatke iz kataloga proizvođača, nema potrebe za pokretanjem motora i obavljanjem raznih merenja. Druga postavka optimizacionog problema se naziva „online“ [3]. Za njen rad je potrebno imati informacije o merenoj struji motora, klizanju i faktoru snage u nekoliko različitih tačaka opterećenja.

3.1 Offline kriterijumska funkcija

Kriterijumska funkcija u offline postavci se zasniva na primeni jednačina za proračun momenta asinhronog motora. Da bi se one izvele polazi se od ekvivalentne šeme asinhronog motora, slika 3.1. Koristeći se Tevenenovom teoremom gde je dozvoljena zamena bilo kojeg dela šeme koji je sastavljen od linearnih elemenata, naponskog ili strujnog izvora i impedansi sa rednom vezom Tevenenove ekvivalentne impedanse i ekvivalentnog naponskog izvora.

Napon Tevenenovog generatora dat je jednačinom 3.1.1. On je izračunat jednostavnim naponskim razdelnikom. To je napon koji se javlja između tačaka a i b na prikazanoj ekvivalentnoj šemi tj. napon na reaktansi magnećenja X_m .

$$U_{th} = U_f \left(\frac{jX_m}{R_s + j(X_{ys} + X_m)} \right) \quad (3.1.1)$$

Ekvivalentna Tevenenova impedansa data je jednačinom 3.1.2. To je impedansa koja se vidi gledano u levo iz tačaka a i b na slici 3.1, pod uslovom da se naponski izvor kratko spoji.

$$Z_{th} = R_{th} + jX_{th} = \frac{jX_m(R_s + jX_{ys})}{R_s + j(X_{ys} + X_m)} \quad (3.1.2)$$

Iz poznatog napona i impedanse Tevenenovog ekvivalenta, može se odrediti struja u ekvivalentnoj šemi, jednačinom 3.1.3.

$$I_{th} = \frac{U_{th}}{Z_{th} + jX'_{yr} + \frac{R'_r}{s}} \quad (3.1.3)$$

Kada je struja određena, na osnovu nje može se odrediti i mehanički momenat na vratilu motora pomoću jednačine 3.1.4.

$$M_{meh} = \frac{P_{meh}}{\omega_{meh}} = \frac{P_{ob}}{\omega_s} = \frac{3I_{th}^2 \frac{R'_r}{s}}{\omega_s} \quad (3.1.4)$$

Gde je:

P_{ob} – snaga obrtnog polja

$\omega_s = \frac{2\pi f_e}{p}$ – sinhrona mehanička ugaona brzina

p – broj parova polova mašine

f_e – učestanost izvora napajanja.

Jednačinom 3.1.4. dolazi se do krajnjeg oblika jednačine za momenat motora, jednačina 3.1.5. Kako bi se odredio potreban nominalni momenat motora umesto klizanja s

treba uvrstiti nominalno klizanje s_n . Prevalni momenat se dobija uvrštavanjem s_{pr} čija se vrednost može odrediti jednačinom 3.1.6. Kako bi se proračunao polazni momenat potrebno je klizanje s zameniti sa 1.

$$M_{meh} = \frac{3U_{1f}^2}{w_s} \left[\frac{\frac{R'_r}{s}}{\left(R_{th} + \frac{R'_r}{s}\right)^2 + (X_{th} + X'_{yr})^2} \right] \quad (3.1.5)$$

$$s_{pr} = \frac{R'_r}{\sqrt{R_{th}^2 + (X_{th} + X'_{yr})^2}} \quad (3.1.6)$$

Kriterijumska funkcija F_{obj} je definisana nad nepoznatim parametrima asinhronog motora (jednačina 3.1.7). U jednačini 3.1.7, veličine sa r u indeksu predstavljaju proračunate vrednosti jednačinama za momenat, dok su veličine bez r momenti iz kataloga proizvođača za određen motor.

$$F_{obj} = \left(\frac{M_{nr}}{M_n} - 1\right)^2 + \left(\frac{M_{pr}}{M_p} - 1\right)^2 + \left(\frac{M_{polr}}{M_{pol}} - 1\right)^2 \quad (3.1.7)$$

3.1 Online kriterijumska funkcija

Ovom funkcijom se takođe određuje pet parametara ekvivalentne šeme asinhronog motora koji su nepoznati. Zahteva se da se poznaje napon napajanja motora, struja motora, faktor snage $\cos \varphi$ i klizanje s za jednu ili više tačaka opterećenja. Vrednosti struje, faktora snage i klizanja se dobijaju prethodnim merenjima iz oglada opterećenja, za nekoliko tačaka opterećenja. Pristup dobija naziv „online“, jer se mora pokrenuti motor da bi se merenjem odredile potrebne veličine za definisanje kriterijumske funkcije. Ovaj pristup ima veliku prednost u tome što prilikom remontovanja motora, npr. ako je statorski namotaj premotavan, ne može se više osloniti na podatke sa natpisne pločice niti katalogske podatke kako bi se odredili parametri.

Iz ekvivalentne šeme asinhronog motora (slika 3.1) dobijaju se potrebne jednačine za kriterijumsku funkciju. Polazi se od izraza za struju asinhronog motora po jednoj fazi, jednačina 3.2.1, gde Z_e predstavlja ekvivalentnu impedansu asinhronog motora.

$$I_f = \frac{|U_f|}{|Z_e|} = \frac{|U_f|}{|R_e + jX_e|} \quad (3.2.1)$$

Posmatrano od strane izvora napajanja ekvivalentne šeme, postoji paralelna veza grane magnećenja i rotorskog kola dok je na to redno vezana statorska impedansa. Na osnovu tog zaključka ekvivalentna otpornost kola R_e dobija se jednačinom 3.2.2, dok se ekvivalentna reaktansa X_e dobija jednačinom 3.2.3.

$$R_e = R_s + \frac{\frac{X_m^2 R'_r}{s}}{\left(\frac{R'_r}{s}\right)^2 + (X'_{yr} + X_m)^2} \quad (3.2.2)$$

$$X_e = X_{ys} + \frac{\left(\frac{R'_r}{s}\right)^2 X_m + X'_{yr} X_m (X'_{yr} + X_m)}{\left(\frac{R'_r}{s}\right)^2 + (X'_{yr} + X_m)^2} \quad (3.2.3)$$

Na osnovu ekvivalentne otpornosti i reaktanse može se napisati izraz 3.2.4. za izračunavanje faktora snage motora.

$$\cos \varphi = \cos \left(\tan^{-1} \frac{X_e}{R_e} \right) \quad (3.2.4)$$

Kriterijumska funkcija opisuje grešku koja nastaje prilikom procenjivanja parametara motora. Funkcija je data izrazom 3.2.5; teži se da njen rezultat bude što je moguće manji, da bi određeni parametri bili tačniji.

$$F_{obj} = \sum_{i=1}^n \left[\left(\frac{\cos \varphi_{r,i}}{\cos \varphi_{m,i}} - 1 \right)^2 + \left(\frac{I_{1r,i}}{I_{1m,i}} - 1 \right)^2 \right] \quad (3.2.5)$$

Veličine koje u indeksu sadrže m predstavljaju poznate, merene veličine, dok su sa r predstavljene proračunate veličine na osnovu izvedenih jednačina.

4. REZULTATI

Prikazani su rezultati dobijeni za dva asinhrona motora proizvođača Sever na osnovu „offline“ i „online“ pristupa. Podaci motora su dati u tabeli 4.1.

Tabela 4.1. *Katalogski podaci za ispitivane motore sa jednim parom polova*

TIP	1.ZK80B2	1.ZK280M2
Snaga (kW)	1,1	90
n (ob/min)	2770	2960
η (%)	74	93
$\cos \varphi$	0,84	0,89
I_n (A)	2,7	165
M_n (Nm)	3,79	290
M_p/M_n	2,2	2,3
M_{max}/M_n	2,1	2,8

4.1 Rezultati offline metode

U tabelama su prikazana poređenja poznatih parametara ekvivalentne šeme i određenih parametara dobijenih offline metodom.

Tabela 4.1.1. *Poređenje parametara ekvivalentne šeme motora 1.ZK80B2*

Tip motora 1.ZK80B2	Fabrički parametri (Ω)	Određeni parametri (Ω)	Relativna greška (%)
R_s	9,009	10,054	11,60
X_{ys}	7,152	5,032	29,64
X_m	134,812	142,81	5,93
R'_r	7,4301	7,188	3,26
X'_{yr}	3,3867	5,091	50,32

Tabela 4.1.2. Poređenje parametara ekvivalentne šeme motora 1.ZK280M2

Tip motora 1.ZK280M2	Fabrički parametri (Ω)	Određeni parametri (Ω)	Relativna greška (%)
R_s	0,038	0,284	647,37
X_{ys}	0,2245	0,215	4,23
X_m	11,208	29,787	165,77
R_r'	0,0519	0,0979	88,63
X_{yr}'	0,3141	0,108	65,62

4.2 Rezultati online metode

Prikazani su rezultati koji su određeni na osnovu poznate struje, faktora snage i klizanja za dve tačke opterećenja. Korišćene su veličine za nominalnu tačku opterećenja motora i za tačku gde je motor opterećen sa 10% većim opterećenjem od nominalnog.

Tabela 4.2.1. Poređenje parametara ekvivalentne šeme motora 1.ZK80B2

Tip motora 1.ZK80B2	Fabrički parametri (Ω)	Određeni parametri (Ω)	Relativna greška (%)
R_s	9,009	8,017	11,01
X_{ys}	7,152	6,419	10,25
X_m	134,812	142,73	5,87
R_r'	7,4301	7,533	1,38
X_{yr}'	3,3867	6,011	77,49

Tabela 4.2.2. Poređenje parametara ekvivalentne šeme motora 1.ZK280M2

Tip motora 1.ZK280M2	Fabrički parametri (Ω)	Određeni parametri (Ω)	Relativna greška (%)
R_s	0,038	0,136	257,89
X_{ys}	0,2245	0,297	32,29
X_m	11,208	12,004	7,10
R_r'	0,0519	0,05	3,66
X_{yr}'	0,3141	0,385	22,57

5. ZAKLJUČAK

Iz ukazanih rezultata i proračunatih relativnih grešaka između poznatih i određenih parametara „offline“ pristup je pokazao solidne rezultate samo u slučaju određivanja parametara kod motora malih snaga.

U slučaju gde je ispitivan motor veće snage zastupljena su veća odstupanja. Razlog za takva odstupanja vidimo u tome što se postavka problema zasniva na traženju tri momenta opterećenja za određenu mašinu, što znači da je prilikom traženja rešenja fokus bio postavljen na momente mašine dok su parametri mašine bili u drugom planu.

Dodatni razlog za takve rezultate jeste i to što je algoritam težio naći samo jedan set parametara koji bi bio pogodan za različita stanja mašine, dok se u realnim situacijama parametri mašine razlikuju u različitim stanjima kao što je npr. u kratkom spoju i kada je mašina nominalno opterećena.

Opisani „online“ pristup u slučaju kada se koriste veličine samo za jednu tačku opterećenja davao je različite rezultate pri svakom ponovnom pokretanju algoritma.

Prikazani su rezultati dobijeni samo na osnovu dve tačke opterećenja. Takođe postoje greške kod parametara za oba motora, međutim u ovom slučaju su greške mnogo manje za motor veće snage, dok greške kod određenih parametara za motor manje snage su približne.

Kako bi se dobili što bolji rezultati, ovaj metod zahteva da se koriste poznate veličine iz što više tačaka opterećenja, i potrebno je da opterećenje bude što bliže nominalnom kako bi se odredili parametri za stanje gde je mašina nominalno opterećena.

6. LITERATURA

- [1] Sivanandam S. N., Deepa S. N.: „*Introduction to Genetic Algorithms*“, Springer Berlin Heidelberg, Njujork, 2008.
- [2] Ali İhsan ÇANAKOĞLU, Asım Gökhan YETGİN, Hasan TEMURTAŞ, Mustafa TURAN: „*Induction Motor Parameter Estimation Using Metaheuristic Methods*“, Turkish Journal of Electrical Engineering & Computer Sciences, Ankara, 2014, raspoloživo na: <http://journals.tubitak.gov.tr/elektrik/issues/elk-14-22-5/elk-22-5-6-1211-171>.
- [3] Vasil Spasov, Vanya Rangelova, Ivan Kostov, Vasil Drambalov: „*An Efficient Approach for Determining Induction Motors Parameters*“, Faculty of Engineering, Hunedoara (Rumunija), 2014, raspoloživo na: <http://acta.fih.upt.ro/pdf/2014-2/ACTA-2014-2-18.pdf>

Kratka biografija:



Mihajlo Fejsa rođen u Vrbasu 1991. god. Završio je 2010. godine Srednju elektrotehničku školu Mihajlo Pupin u Kuli i nakon toga upisao osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, odsek Elektrotehnika i računarstvo i kasnije smer Energetska elektronika i električne mašine.

SISTEMI VIDEO NADZORA NOVE GENERACIJE**NEW GENERATION VIDEO SURVEILLANCE SYSTEMS**Miloš Ivković, Željens Trpovski, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – U radu su opisani sistemi video nadzora, ali je dat i opis sistema za automatsku dojavu požara i kontrole pristupa. Prikazane su komponente ovih sistema, kako se oni ponašaju u realnom vremenu i kakav uticaj imaju na nas.

Abstract – In this thesis are described the video surveillance systems, the system for automatic fire alarm and the access control. We show components of these systems, how they behave in real time and what impact they have on us.

Ključne reči – SISTEMI VIDEO NADZORA, SISTEMI AUTOMATSKE DOJAVE POŽARA, SISTEMI KONTROLE PRISTUPA.

1. UVOD

Sigurnosni sistemi poslednjih godina dobijaju na značaju i ostvaruju široku i raznovrsnu upotrebu. Cenom su poprilično pristupačni, a veoma su potrebni domu ili poslovnoj prostoriji.

U ovom radu pokazane su osnovne karakteristike sistema za video nadzor, šta su analogni i digitalni video nadzor i opisane su neke od njihovih podtipova.

Opisan je sistem za automatsku dojavu požara, prikazane su vrste ovih sistema koje se koriste i delovi ovih sistema, kao što su detektori požara, centralna jedinica sistema. Takođe, u radu se posvećuje pažnja sistemima za kontrolu pristupa.

2. SISTEMI VIDEO NADZORA

Video nadzor je sistem koji se sastoji od kamera za video nadzor kao i uređaja koji snimaju sliku sa kamera. On predstavlja elektronski sistem za nadgledanje i snimanje situacija na nekom prostoru i prenos signala sa kamera na predefinisano lokaciju. Osnovna funkcija sistema video nadzora jeste pružanje informacija o bezbednosnom statusu na osnovu koje će korisnik moći da organizuje odgovarajuće mere zaštite. Planiranje i projektovanje ovih sistema je složen proces koji mora da se uklopi u rezultate analize bezbednosne ugroženosti objekta, lica ili materijalnih dobara uz poštovanje svih postojećih specifičnosti objekta, položaja i njegove namene pa sve do uklapanja u tehnologiju rada koja se u njemu odvija. Analogne HD kamere imaju potrebe za napajanjem odvojeno od prenosa signala. Najčešće se koriste RG59 koaksijalni kablovi koji na sebi imaju i paricu za napajanje, tako da se praktično provlači jedan kabl za dve namene.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željens Trpovski, vanr. prof.

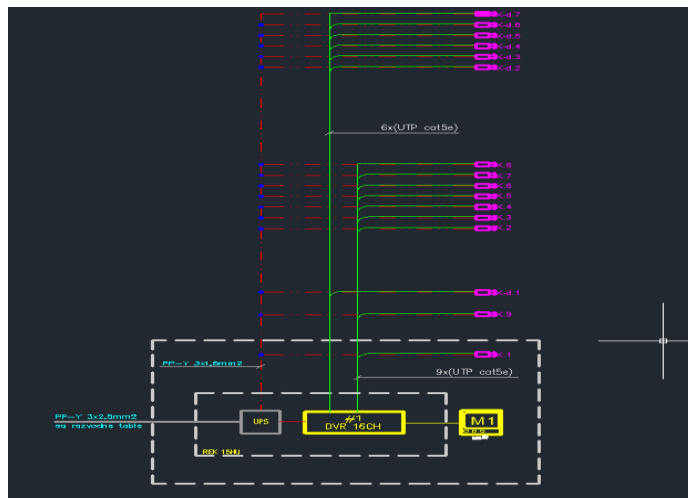
IP kamere su sa snimačem povezane preko UTP mrežnog kabla. Ovaj kabl u sebi sadrži četiri parice. Kod njih postoji mogućnost da se jedna parica iskoristi za dovođenje napona na kameru [1].

2.1. Analogni video nadzor

Kamere se direktno povezuju sa DVR-om. Sve što kamera snimi, oni snimaju na hard disku. DVR snimač je poznat i pod nazivom digitalni video rekorder. Dolazi obično u kombinaciji sa 4, 8, 16 ili 32 kamere. Kamere i DVR snimač su povezani koaksijalnim kablom. Dostupni su preko LAN ili WAN. Sva podešavanja su preko odgovarajućeg softvera [2].

Vrste sistema analognog video nadzora:

- CCTV video nadzor
- HDSDI (High Definition Serial Digital Interface) video nadzor
- HDCVI (High Definition Composite Video Interface) video nadzor



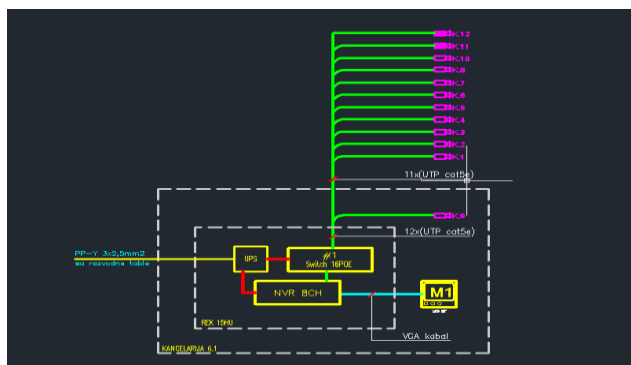
Slika 1. Blok šema analognog video nadzora

2.2. Savremeniji tip video nadzora – digitalni IP video nadzor

Glavna prednost ovog sistema za video nadzor jeste snimanje u visokim rezolucijama. IP kamere su samostalne celine, nemaju ograničenje u rezoluciju i poseduju sopstveni kompjuter. Snimljeni materijal se automatski kompresuje, pretvara u podatke i šalje kroz mrežu. Za razliku od DVR snimača ovde imamo server na mreži ili NVR sa kojim se IP kamere povezuju. Kamere se povezuju sa sistemom preko standardnog mrežnog računarskog kabla (UTP ili FTP). Pored toga što sve informacije idu preko jednog kabla (slika, zvuk, komande za pan-tilt-zoom), postoji i mogućnost napajanja kamera preko tog istog kabla (PoE – Power Over Ethernet).

Neograničen broj kamera se može priključiti na NVR. Prednost IP kamere jeste da može da prenosi i druge podatke. IP sistemu se lako može pristupiti sa bilo kog računara ili mobilnog telefona jer je već na mreži. IP sistemi su sa druge strane znatno skuplji u odnosu na analogni tip, pa su analogni i dalje zastupljeniji.

Mrežni video nadzor je idealan za nadzor jedne prostorije. Tada NVR nije ni potreban i kamere se mogu povezati direktno na već postojeći ruter kućnog internet. Rezolucija koju pružaju IP kamere su HD (High Definition) od 720p i Full HD, odnosno 1080p.



Slika 2. Blok šema digitalnog video nadzora

Savremene digitalne kamere sliku dobijaju pomoću mikročipa tipa CMOS ili tri CCD. U pitanju su tri CCD čipa od kojih svaki obrađuje jednu od tri osnovne boje. Postoje i jeftinije alternativne vrste. Mikročip digitalizuje dobijenu sliku.

Podela kamera za video nadzor :

Dome – kupolastog odnosno ovalnog oblika. U zavisnosti od IP zaštite koju poseduju mogu se montirati unutra ili napolju. Jedine su kamere koje mogu biti antivandal (sa zaštitom od jakih udaraca).

Bullet – cevastog oblika. Mogu se montirati unutra ili napolju, zavisi koliki nivo IP zaštite poseduju.

Cube – kockastog oblika. Nalaze se na nosaču i uglavnom se montiraju unutra.

PTZ – skraćena "pan, tilt, zoom". Imaju mogućnost upravljanja putem upravljačke konzole, tako da obično imaju kružni oblik i napravljene su tako da mogu da se okrenu ka svakom uglu.

Box – ove kamere predviđaju dodatno spoljašnje kućište, a montiraju se samo unutra.

2.3. Kablovi za video nadzor

Da bi sistem za video nadzor radio mora biti pravilno povezan. Kada govorimo o kablovima tu postoji takođe nekoliko bitnih stvari. Prva je udaljenost kamere od DVR-a ili NVR-a. Kablovi imaju propisano slabljenje pri određenim razdaljinama i toga se treba pridržavati inače će doći do lošeg kvaliteta video materijala ili nemogućnosti da se signal prenese do željene destinacije. FTP (Foiled Twisted Pair –folijom oklopljene upredene parice) kabl štiti protok podataka od uticaja spoljašnjeg elektromagnetnog i radiofrekvencijskog zračenja, ali i od emitovanja elektromagnetnog zračenja, koje se indukuje u kabl pri protoku električnog signala.

FTP kablovi pružaju najbolji odnos cene i kvaliteta u poređenju sa UTP i STP kablovima. Prilikom povezivanja potrebno je voditi računa o optimalnim rastojanjima i dužinama kablova. Po specifikaciji FTP kabl između dva

uređaja može biti dug najviše 100 metara. U posebnim slučajevima može biti i duži kabl, ali već preko 80m znatno opada brzina protoka podataka [3].

3. SISTEMI AUTOMATSKE DOJAVE POŽARA

Sistem automatske dojava požara sadrži veliki broj uređaja koji rade zajedno kako bi otkrili i upozorili ljude putem vizuelnih i audio aparata kada su prisutni dim, vatra, ugljen monoksid ili druge hitne situacije. Ovi alarmi se mogu aktivirati automatski, preko detektora dima i detektora toplote ili se takođe mogu aktivirati pomoću uređaja za ručno aktiviranje požarnog alarma.

Sa aspekta sistema za detekciju požara od interesa je prva faza, tj. najraniji deo - početni požar. U toj fazi posle inicijalnog paljenja u zatvorenoj prostoriji dolazi najčešće, ali ne i obavezno, do laganog početnog razvoja požara praćenog značajnom emisijom dima u početku uz neznatan porast temperature. Kako je već rečeno, od interesa je da se požar detektuje u ovoj fazi, jer samo tada detekcija ima smisla.

Sistemi dojava požara razlikuju se po veličini i tehnologiji rada, a koji će se koristiti zavisi od aplikacije, tipa objekta, ugroženosti od požara, broju ljudi u objektu, propisima i željama investitora.

Prilikom projektovanja sistema za automatsku dojavu požara, potrebno je držati se određenih pravila koja su definisana zakonom [4].

Neka od njih su:

Rastojanje optičkih javljača treba da bude sledeće:

- od zida treba da bude min 50 cm
- od grede (rebra) treba da bude min 50 cm
- od mesta ubacivanja vazduha treba da bude min 50 cm
- od uskladištene robe treba da bude min 50 cm.

Svi javljači moraju imati vidne oznake (nalepnice) sa oznakom adresne linije (zone).

Napajanje energijom mora biti iz dva izvora. Prvi izvor je električna mreža, a drugi rezervni jeste akumulatorska baterija.

3.1. Delovi sistema za automatsku dojavu požara

Sistemi za detekciju požara po pravilu se sastoje od automatskih detektora požara, ručnih javljača požara, centralne jedinice za detekciju i signalizaciju požara, uređaja za zvučno i svetlosno alarmanje, uređaja za paralelnu indikaciju, uređaja za daljinski prenos alarma i izvršnih uređaja za aktiviranje sekundarnih funkcija sistema, kao što je uključivanje stabilnih automatskih sistema za gašenje, isključenje ventilacije, aktiviranje sistema za odimljavanje, zatvaranje protivpožarnih vrata, protivpožarnih klapni na ventilacionim i/ili klimatizacionim sistemima, aktiviranje daljinskog prenosa alarma.

Centralna jedinica za detekciju i alarmiranje požara, ili kako se često pojednostavljuje naziva centrala za detekciju požara, ima zadatak da napaja detektore požara i linije detekcije stabilnim i pouzdanim naponom.

Centrala prima signal o normalnom stanju, stanju alarma, prekidu linije ili vađenju detektora, kratkom spoju linije i ona po potrebi prosleđuje signal uređajima za zvučno i svetlosnu signalizaciju [7].

Većinu savremenih detektora požara predstavljaju tačkasti detektori koji reaguju na promenu koncentracije dima ili temperature u neposrednoj blizini senzorskog elementa. Linijski detektora požara reaguje na promenu detektovanog fenomena u okolini određene detekcije

linije. Tipičan linijski detektor toplote predstavlja termički kabl koji reaguje na promene temperature u okolini kabla.

Optički detektori dima su detektori koji ne poseduju radioaktivni izvor i ova činjenica predstavlja njihovu nesumnjivu prednost u odnosu na jonizacione detektore, koji više ili manje opravdano, pobuđuju strah kod korisnika već na samu pomisao o izvoru radioaktivnosti.

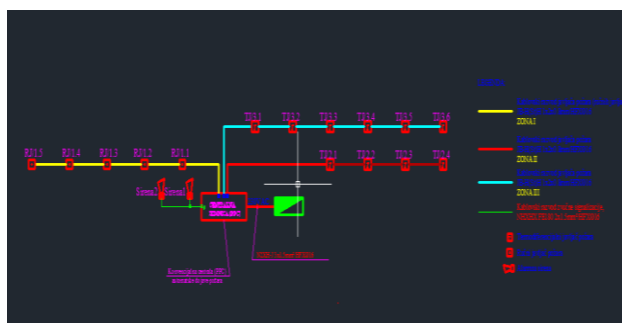
Detektori toplote, predstavljaju uređaje koji registruju porast temperature u mernom domenu detektora i koji imaju mogućnost da taj porast, iznad određene granice, transformišu u alarm za požar. Detektori mogu reagovati na porast temperature u njegovom apsolutnom iznosu i biti koncipirani tako da daju signal za požar ako temperatura okoline detektora premaši podešenu vrednost. Takvi detektori su prvi razvijani, a primenjuju se i danas i nazivaju se obično termički detektori.

Detektor sa svetlosnim snopom je detektor koji se sastoji od predajnika svetlosti i prijemnika koji se postavljaju na različitim stranama štićenog prostora.

Sisteme možemo podeliti na konvencionalne i adresabilne.

3.2. Konvencionalni sistemi

Konvencionalne centrale se sastoje od određenog broja konvencionalnih zona. Prema propisu, jedna zona sadrži do 30 javljača požara ili ručnih javljača požara. U slučaju aktiviranja javljača požara, na centrali za dojavu požara javlja se signal alarma na liniji (zoni).



Slika 3. Šema konvencionalnog sistema

Detektori požara u konvencionalnim sistemima postavljaju se u linije koje predstavljaju i fizičke celine u kojima je lako otkriti mesto sa koga je eventualni alarm prispeo. Ove linije se u tehničkoj praksi nazivaju zonama detekcije. Zone moraju biti tako izabrane da je otkrivanje lokacije alarma brzo i u što većoj meri nedvosmisleno. Lice koje proverava alarm, odnosno istražuje mesto sa koga je alarm prispeo mora jednostavno i bez dilema da otkrije mesto porekla alarma, odnosno da pronađe detektor koji je u alarmu.

Međutim, za veliki broj detektora u zoni postavlja se pitanje da li je "vreme traženja" požara dovoljno kratko i da li će otkrivanje lokacije eventualnog požara, zbog prevelikog broja detektora koji kod alarma moraju biti pregledani, biti suviše dugo, a alarm zakasnio [6].

Ručne javljače požara treba postaviti u posebne zone detekcije. Britanski standard dozvoljava da se detektori i ručni javljači postave u istu zonu, ali ukazuje da to treba izbegavati. Razlog je u tome što ručni javljač u većini organizacija alarmiranja označava siguran požar, jer ga aktivira osoba koja je požar videla. Eventualne zloupotrebe ne uzimaju se u obzir i one moraju na svaki način biti sprečene odgovarajućom disciplinom. Za

razliku od toga, automatski detektori požara imaju značajno veći broj lažnih alarma u odnosu na stvarne alarme i stoga je, generalno, pre inicijacije akcije gašenja potrebno provesti akciju "traženja" ili "izviđanja" požara koja ima za cilj da se proverda da li požar stvarno postoji i da li se može ugasiti sredstvima i potencijalima koji su na raspolaganju na licu mesta.

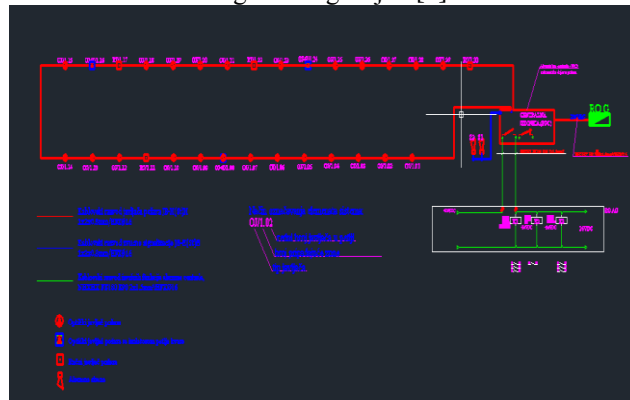
Ako bi se u istoj zoni detekcije koristili i automatski detektori i ručni javljači, onda se ne može praviti razlika između alarma od automatskih detektora kojima je neophodna provera i alarma ručnih javljača koji znače siguran požar i kojima provera nije potrebna.

3.3. Adresabilni sistemi

Adresabilni sistemi za detekciju i dojavu požara pružaju najmoderniju i najbolju zaštitu od požara. Osim olakšanog testiranja ovi sistemi se odlikuju velikom fleksibilnošću i preciznošću detekcije požara.

Kada kažemo da je sistem adresabilan to znači da svaki detektor u sistemu ima svoju individualnu adresu, tako da u slučaju požara imamo preciznu informaciju koji je detektor u kojoj prostoriji aktiviran.

Dok je kod konvencionalnih sistema identifikacija alarma na nivou grupe detektora, odnosno centralna jedinica i prisutno lice dobija informaciju o grupi (zoni) koja je u alarmu, dotle kod adresabilnih sistema svaki detektor dobija svoj kod (adresu) kojom identifikuje i "saopštava" centralnoj jedinici i prisutnom osoblju svoje stanje. Dakle, kod sistema sa grupnom identifikacijom alarma centralna jedinica dobije informaciju od grupe detektora (zone) o alarmu ili nekom drugom "događaju" [5].



Slika 4. Šema adresabilnog sistema

Kod adresabilnih sistema detektor se "proziva" u određenim vremenskim intervalima i u komunikaciji sa centralnom jedinicom je u svakom trenutku samo jedan detektor ili grupa detektora sa istom adresom.

Međutim, interval vremena između dve predaje informacija detektoru nije najčešće od posebnog značaja, jer je većina požara u početnoj fazi ima mali gradijent razvoja i ovakvi kratki vremenski intervali nisu relevantni. Postoji obično mogućnost da se na centralnoj jedinici programira da se prvi alarm detektora prihvati kao predalarm i da se detektor resetuje i čeka potvrdu, odnosno još jednu ili više proroda detektora, posle čega se aktivira puno alarmno stanje. Ako ponovljeni alarm izostane, predalarm se ignoriše.

Broj provera često se može programirati kao npr. 2, 10 i 16. Ako je požarni rizik u nekom prostoru veliki, tada će se programirati da detektori koji štite taj prostor idu u alarm već kod drugog signala alarma dobijenog od istog detektora. Pod pretpostavkom da je ukupno vreme za koje

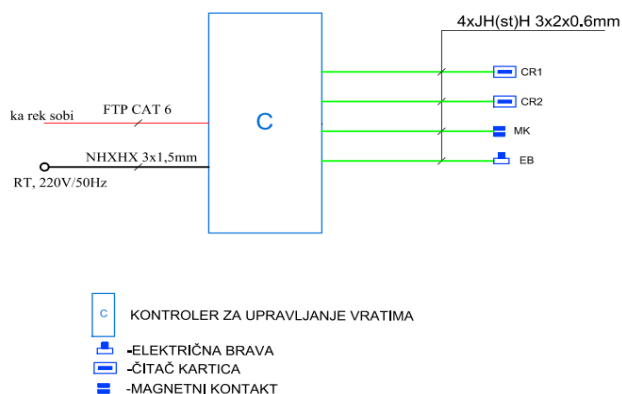
centralna jedinica uspostavi komunikaciju sa svakom adresom, dobije i obradi informaciju o stanju detektora, odnosno ručnog javljača 2,5 sekunde, onda će alarm biti dat najkasnije za 5 sekundi od nastanka požara. Adresni sklop može biti ugrađen u podnožje - bazu detektora ili biti sastavni deo samog detektora.

Slučajna zamena dva detektora može dovesti do fatalnih grešaka, jer adresa nije više povezana sa određenom fizičkom pozicijom i ovo može dovesti do intervencije na sasvim pogrešnom mestu. Stoga je bolje rešenje lociranje sklopa za adresiranje u podnožju detektora jer se ovim garantuje da će adresa detektora odgovarati određenoj poziciji.

4. SISTEMI KONTROLE PRISTUPA

Sistemi za kontrolu pristupa služe za dodeljivanje prava ko, kad i gde može da pristupi štićenom prostoru. Evidentiraju uspešne i neuspešne pristupe štićenom prostoru. Kontrolom pristupa se reguliše ulazak i/ili izlazak iz štićenog prostora. Ovaj sistem bi trebalo integrisati sa ostalim alarmnim sistemima kao što su: protivprovalni sistem, sistem video nadzora, sistem zaštite od požara.

Zahvaljujući ovoj tehnologiji, poslodavci mogu imati tačan uvid kada su im zaposleni došli na radno mesto, kao i kada su isto napustili. To su veoma korisne informacije za svakog poslodavca, odnosno kontrolom pristupa obezbediće autorizovan pristup u važne prostorije unutar objekata, kao i u kompletne objekte [8].



Slika 5. Šema sistema za kontrolu pristupa

Za kompjutersku bezbednost, kontrola pristupa uključuje ovlašćenja, autentifikaciju i prepoznavanje identiteta u pokušavaju da odobri pristup. Modeli kontrola pristupa koriste subjekat i objekat. U ovom slučaju subjekat bi bio čovek koji pokušava da dobije pristup nad nekim štićenim objektom što bi bio najčešće softver.

U računarskim sistemima koristi se spisak za kontrolu pristupa koji sadrži spisak dozvola i korisnika nad kojima ove dozvole važe. Ove podatke mogu videti samo određeni ovlašćeni ljudi. Ovo omogućava administratoru da obezbedi informacije i postavi određena ovlašćenja nad korisnicima kada i gde ko može da pristupi.

Kontrola pristupa koja se vrši uz pomoć takozvane RFID tehnologije, koristi radio frekvenciju da bi se mogle razmenjivati informacije između host računara i uređaja.

U početku za kontrolu pristupa koristili su se šifrotori, a potom magnetne kartice. U poslednje vreme sve se češće koriste kombinovani sistemi tj. protivprovalni sistemi sa kontrolom pristupa.

4.1. Komponente sistema kontrole pristupa

Mesto za kontrolu pristupa mogu biti vrata, okretno staklo, parkirna kapija, lift ili druga fizička barijera, gde se omogućavanje pristupa može elektronski kontrolisati. Obično, pristupna tačka su vrata.

Elektronska vrata za kontrolu pristupa mogu sadržati nekoliko elemenata. U svom osnovnom stanju postoji samostalna električna brava. Bravu otključava operator sa prekidačem. Da bi se to automatizovalo, operatera zamenjuje čitač. Čitač može biti tastatura u kojoj se unosi kôd, može biti čitač kartica ili biometrijski čitač.

Čitači obično ne donose odluku o pristupu, već šalju broj kartice na kontrolnoj tabli za pristup koji verifikuje broj prema pristupnoj listi.

Kada se dugme pritisne ili detektor pokreta detektuje kretanje na vratima, alarm se automatski privremeno ignoriše dok se vrata otvaraju. Izlazak sa vrata bez potrebe otključavanja električnih vrata naziva se mehanički slobodni izlaz. Ovo je važna sigurnosna karakteristika.

5. ZAKLJUČAK

Sistemi obezbeđenja postali su nezaobilazni deo zaštite kako poslovnih tako i privatnih objekata i imovine. Sistemi video nadzora, sistemi za kontrolu pristupa i sistemi automatske dojava požara pružaju brojne pogodnosti i mogućnosti i njihova primena i potražnja raste iz dana u dan, te se očekuje da život bez ovih sistema postane skoro pa nezamisliv u narednom periodu.

6. LITERATURA

- [1] <http://alfaing.co.rs/video-nadzor/>
- [2] <https://svezavideonadzor.rs>
- [3] <https://muricmilorad.files.wordpress.com/2011/11/sistemivideonadzorauskolama.pdf>
- [4] Nedžad Hadžiefendić, dipl. ing. el., DETEKCIJA POŽARA, Beograd 2006.
- [5] Projektovanje i održavanje sistema za dojavu požara, prof.dr. Milan Đ. Blagojević, Niš 2015.
- [6] www.quadel.rs/media/Uputstvo_za_konvencionalne
- [7] http://www.occush.vtsnis.edu.rs/aktivnost_8/tribina_Nis_februar_mart_2013/prilog_8_tribina_nis_2013
- [8] <http://www.navidiku.rs/firme/kontrola-pristupa>

Kratka biografija:



Miloš Ivković rođen je u Trebinju 1990. god. Master rad odbranio je 2017. godine na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Telekomunikacioni sistemi.



Željen Trpovski rođen je u Rijeci 1957. godine. Doktorirao je na Fakultetu tehničkih nauka 1998. god. Oblast interesovanja su telekomunikacije i obrada signala.

SINHRONIZACIJA MULTIPLAYER VIDEO IGARA

SYNCHRONIZATION OF MULTIPLAYER VIDEO GAMES

Marko Tot, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – Ovaj rad se bavi problemima sinhronizacije multiplayer video igara. Opisani su problemi koji nastaju prilikom prenosa podataka o stanju igre i rešenja kojima se ti problemi prevazilaze.

Abstract – This paper deals with the problems of synchronization of multiplayer video games. Problems in data transfer regarding state of the video game are described as well as the solutions to the problems that arise

Ključne reči: Video igra, sinhronizacija, server, latency, stanje igre

1. UVOD

Tokom razvoja video igara, na sceni su se pojavljivale video igre sa raznom tematikom. Od First Person Shooter (FPS) igara kod kojih je na ekranu prikazan pogled junaka iz prvog lica, preko strateških igara - gde igrač kao komandant vodi gomilu jedinica, svakoj zadaje naređenja a sve to nadgleda iz ptičije perspektive, do puzzle igara gde je na svakoj sceni prikazana slika sa određenim problemom datim na rešavanje igraču. Danas se strateške i FPS igre najčešće igraju u više igrača (multiplayer igre).

Video igre koje se igraju u više igrača u isto vreme zahtevaju posebnu pažnju pri prenosu informacija. Tokom igre osnovne akcije koje svaki igrač može da izvrši prilikom kontrolisanja svog karaktera su kretanje i pucanje. Kretanje igrača u video igri vrši se tako što se pritiskaju komande za kretanje. Igra proverava da li je na tastaturi pritisnut taster za kretanje, nakon toga se na osnovu pritisnutih tastera određuje smer kojim se karakter kreće i njegova brzina. Dobijena brzina se kao vektor kretanja dodeljuje karakteru unutar igre.

Kada bi igru igrao jedan igrač na ovaj način bi kretanje karaktera bilo potpuno rešeno. Međutim kako se u multiplayer igrama, igra istovremeno izvršava na više računara postavlja se problem kako sinhronizovati informacije o kretanju svih karaktera koji učestvuju u igri. Svaki računaru se moraju proslediti iste informacije.

2. KARAKTERISTIKE MULTIPLAYER IGRE

Multiplayer igra je karakteristična po tome što računar svakog igrača treba da pokazuje isto stanje igre.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, van.prof.

Igrači mogu igrati jedan protiv drugog ili zajedno protiv računara. Svi igrači istovremeno učestvuju u igri i bitno je obezbediti da računari svakog pojedinačnog igrača poseduju iste informacije o igri.

Kako svaki igrač direktno može da pomera samo svog karaktera, potrebno je proslediti te informacije o kretanju i drugim igračima. Te informacije se šalju preko mreže, direktno drugim igračima, ili serveru zaduženom za igru koji kasnije prosleđuje informacije ostalim igračima.

Ukoliko se informacije ne bi prosledile ostalim računarima, karakter igrača bi se pomerio samo na računaru domaćinu (host računaru) dok bi kod ostalih igrača taj karakter stajao u mestu. Ovakva situacija dovela bi do problema sa sinhronizacijom igre jer bi postojala dva različita stanja igre. Host računar bi pokazao stanje gde se karakter igrača pomerio a ostali računari bi pokazivali drugo stanje gde je karakter i dalje na originalnoj poziciji. Ovakva situacija sa dva ili više različitih stanja onemogućila bi dalje odvijanje igre. Svaka video igra mora imati jedinstveno stanje, kako bi se svakom igraču mogao pružiti isti prikaz igre.

3. PRENOS INFORMACIJA

Da bi video igra tekla glatko potrebno je obezbediti da se stanje igre može promeniti od 30 do 60 puta u sekundi [1]. Manji broj promena doveo bi do vidnog pogoršanja u kontinuitetu kretanja karaktera.

Nakon promena stanja igre od strane jednog igrača potrebno je poslati informacije o promenama svim ostalim računarima. Pokreti jednog igrača moraju se preneti ostalim računarima što znači da se za svaki registrovani pritisak na taster svim ostalim igračima moraju pružiti informacije o brzini i smeru kretanja karaktera.

3.1. Prenos informacija bez servera

Da bi se obezbedio utisak kontinuiranog kretanja jedan računar treba da pošalje preko mreže poruke o kretanju karaktera 60 puta u sekundi.

Kako se broj igrača povećava tako se povećava i broj poruka koje se šalju između računara. Za igru od 8 igrača, svaki računar bi na ovaj način slao do 420 poruka u sekundi, a takođe primao 420 poruka, po 60 za svakog od ostalih računara.

U tabeli 1. prikazan je ukupan broj poruka koje šalju i primaju računari za običajeni broj igrača u multiplayer igrama.

Tabela 1. Ukupan broj poslatih i primljenih poruka po računar, u sekundi, sa frekvencijom od 60 puta u sekundi - bez servera

Broj igrača	Broj poslatih poruka	Broj primljenih poruka
2	60	60
4	180	180
8	420	420
16	900	900
32	1860	1860
64	3780	3780

3.2. Prenos informacija preko servera

Ukoliko se za komunikaciju u igri koristi server količina poslatih poruka se može smanjiti (Tabela 2.). Nezavisno od broja igrača broj poslatih poruka ostaje isti zbog toga što se uvek šalje samo jedna poruka ka serveru, a server kasnije svakom od igrača dalje prosledi informacije. Broj primljenih poruka se povećava zbog toga što server šalje primljene informacije svim igračima, pa će računar primiti i informacije koje je on poslao ka serveru.

Tabela 2. Ukupan broj poslatih i primljenih poruka po računar, u sekundi, sa frekvencijom od 60 puta u sekundi - sa serverom

Broj igrača	Broj poslatih poruka	Broj primljenih poruka
2	60	120
4	60	240
8	60	480
16	60	960
32	60	1920
64	60	3840

Igra može da funkcioniše i sa ovolikim brojem poruka koje se kroz mrežu šalju u sekundi. Za jednostavnije igre obrada ovih poruka ne predstavlja problem. U zahtevnijoj igri poželjno je smanjiti količinu informacija koja pristiže do računara kako bi se za ostale komponente igre oslobodio procesor. Na taj način se procesor može posvetiti ostalim elementima igre.

3.3. Smanjenje količine protoka

Kako broj poruka koje se šalju preko mreže raste tako se povećava i potreban protok kroz mrežu. Smanjenje protoka se postiže smanjenjem broja poruka koji svaki igrač šalje. Taj broj se može menjati u zavisnosti od potrebe igre.

Jedan od načina smanjenja je da računar šalje poruku o poziciji svog karaktera manji broj puta u sekundi. Ukoliko je taj interval od npr. 0,1 sekunde (što bi značilo 10 poslatih poruka u sekundi) onda se ukupan broj poruka značajno smanjuje (Tabela 3.).

Ovakav pristup donosi veliko poboljšanje u protoku informacija, ali pošto se informacije šalju ređe može se videti razlika između stanja na računarima, gde se jedan igrač pomerio u trenutku t , a drugi igrač bi u proseku poruku o tome dobio u trenutku $t + 0,05s$. To znači da se stanja igre razlikuju, jedan igrač kasni za drugim u proseku za 0,05s. Ova razlika nije zanemarljiva jer drugi

igrač može u tih 0,05 sekundi da uradi akciju koja bi sprečila ili ometala kretanje prvog igrača.

Tabela 3. Ukupan broj poslatih i primljenih poruka po računar, u sekundi, sa frekvencijom od 10 puta u sekundi - sa serverom

Broj igrača	Broj poslatih poruka	Broj primljenih poruka
2	10	20
4	10	40
8	10	80
16	10	160
32	10	320
64	10	640

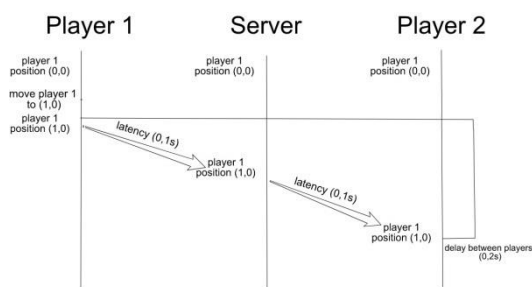
Na ovaj način iako se smanji protok kroz mrežu igra se vidljivo pogorša. Frekvencija slanja poruka može se podešavati kako bi i protok i igrivost bili na zadovoljavajućem nivou. Igra teče bez problema ukoliko je slanje poruka što približnije intervalu od 0,017 sekundi (što predstavlja 60 poslatih poruka u sekundi).

Jedno od vrlo uspešnih rešenja je slanje poruke samo u trenutku kada se kretanje karaktera promeni. Igrač je fizički u stanju da menja kretanje svog karaktera samo nekoliko puta u sekundi tako da se broj poruka za slanje višestruko smanjuje. Umesto 60 puta u sekundi poruke o kretanju se šalju desetinu puta ređe. Osim toga ukoliko računar čeka određeni interval vremena da bi poslao poruku događa se da informacija o kretanju kasni. Slanjem poruke u trenutku kada se menja kretanje karaktera sprečava nepotreban gubitak vremena.

4. KAŠNENJE INFORMACIJA

Pored količine protoka veliki problem prilikom sinhronizacije igre je razlika u trenutku slanja informacije sa jednog računara i trenutka primanja informacije na drugom računaru. Prenos bilo kakve informacije se vrši preko mreže. Prilikom prenosa, informacije moraju fizički da se prenesu iz jednog računara u drugi. U zavisnosti od fizičke udaljenosti računara od servera vreme prenosa informacije je reda veličine 0,1 sekunde. Vremenski period od trenutka slanja informacije, do trenutka primanja informacije zove se kašnjenje (latency). Zbog kašnjenja informacije, potpunu sinhronizaciju igre je veoma teško ostvariti.

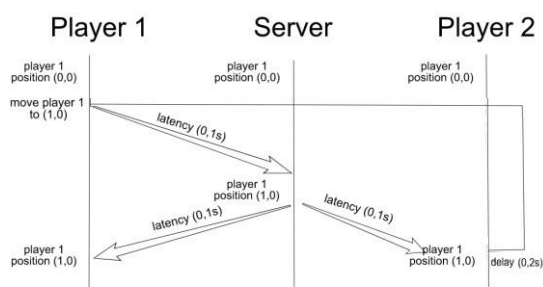
Svaka akcija koju izvrši igrač 1, će putovati do servera 0,1s a zatim do igrača 2 još 0,1s. Igrač 2 na ovaj način vidi akciju prvog igrača 0,2 sekunde nakon što je ona izvršena. Za to vreme stanje dve igre se razlikovalo. Ukoliko se igrač 1, pomeri sa pozicije (0, 0) na poziciju (1,0) a igrač 2 pokuša da gađi igrača 1, koji se na njegovom računaru još uvek nalazi na poziciji (0, 0), jer jos uvek nije dobio informaciju o kretanju (Slika 1.). U periodu od tih 0,2 sekunde kašnjenja računar 2 će registrovati pogodak a računar 1 će registrovati promašaj.



Slika 1. Kašnjenje u prenosu informacija o poziciji igrača

4.1. Uvođenje osnovnog kašnjenja

Jedan od načina na koji se ovaj problem može rešiti je uvođenje osnovnog kašnjenja za svaku akciju. Ovo osnovno kašnjenje omogućava da se isto stanje vidi na svim računarima istovremeno. Akcija se šalje ka serveru, i tek nakon odgovora od servera karakter igrača se pomera. Na slici 2 prikazano je kako izgleda stanje na računarima uz uvedeno osnovno kasnjenje koje u ovom slučaju iznosi 0,2 sekunde.



Slika 2. Uvođenje osnovnog kašnjenja u prenos informacija

Na ovaj način se rešava problem različitih stanja između dva igrača jer svaki igrač u istom trenutku dobija informaciju o kretanju. Međutim, interaktivnost igrača se smanjuje [2].

Ukoliko neki igrač hoće da se pomeri, mora da sačeka 0,2s kako bi računar dobio odgovor od servera.

U slučaju da se kašnjenje za igrača 1 i igrača 2 razlikuje, server može da pošalje informaciju u različitim vremenskim trenucima tako da svakom igraču informacija stigne u isto vreme, čime stanje igre na oba računara ostaje identično.

U igrama koje zahtevaju brze reakcije, zbog smanjenja interaktivnosti igrača, ovakvo rešenje nije idealno.

4.2. Odabir pogodnog stanja igre

Ukoliko se igra bazira na kooperaciji igrača protiv računara problem sinhronizacije se može i zamaskirati tako što se različita stanja pošalju serveru, a on odluči koje od tih stanja je ispravno.

Razlike u stanju igre zbog kašnjenja dovede do bitnih razlika kao što je registrovanje pogotka na jednom računaru i promašaja na drugom računaru prilikom pucanja. Pošto je cilj igrača isti, da zajedno pobeđu, server

može odlučiti šta se dogodilo tako da odluka uvek bude u korist igrača.

Ukoliko postoje 2 stanja igre, jedno u kojem su igrači pogodili neprijatelja, i jedno u kojem nisu, server uvek može da odabere stanje gde su neprijatelji pogodoeni. Informacija o odabranom stanju se mora proslediti svim računarima kako bi pogodak bio registrovan na svim računarima.

Pošto ovakva odluka ide u korist svim igračima nijedna strana se neće osetiti oštećenom. Na ovaj način se izbegava problem sprovede interakcije koja bi nastala uvođenjem osnovnog kašnjenja, kada igrači moraju da čekaju odgovor od servera kako bi njihova akcija bila realizovana.

Pristup kada server bira koje stanje je ispravno može se koristiti i u igrama gde igrači igraju jedni protiv drugih ali se tada posebna pažnja mora obratiti na to koje stanje će biti izabrano za ispravno.

Ukoliko se ciljevi igrača podudaraju, bira se ono stanje koje odgovara svima. Ukoliko igrač 1 na svom računaru vidi da je pogodio karakter igrača 2, a igrač 2 vidi da je njegov karakter izbegao metak karaktera 1 dolazi do dva stanja igre [3]. Stanje u kojem je karakter 2 pogodoen, koje odgovara igraču 1, i stanje u kojem je karakter 2 izbegao metak, koje odgovara igraču 2.

U ovakvim situacijama nije moguće odabrati stanje koje odgovara svim igračima. Tada se mora odlučiti na koji način se bira stanje. Jedno rešenje je da svaki računar šalje samo podatke o svojim pogocima. Svaki računar prati svoje metke, i ukoliko oni pogode nekog od igrača šalje se informacija ka serveru.

Na ovaj način nijedan od igrača objektivno nije oštećen pri izboru, jer su kriterijumi odabira stanja isti za oba igrača, ali se u pojedinačnim situacijama igrači mogu osetiti oštećeno, jer su na svom ekranu mogli primetiti kako su izbegli metak a računar je registrovao pogodak jer je tu informaciju dobio od servera.

5. ZAKLJUČAK

U multiplayer video igrama potrebno je uspostaviti jedinstveno stanje igre koje bi se nalazilo na svim računarima u isto vreme. Kako se igra izvršava na više računara potrebno je potrebno je proslediti sve bitne informacije kako bi svi igrači videli isto stanje igre.

Osnovni problemi u sinhronizaciji igre ogledaju se u količini informacija koje treba preneti i vremenskoj razlici u trenutku slanja i primanja informacija.

Da bi igra tekla nesmetano potrebno je obezbediti pristizanje dovoljne količine informacija. Tehnika u kojoj se slanje informacija realizuje direktno između računara zahteva veliki protok informacija kroz mrežu. Uvođenje servera omogućava smanjanje količine informacija koju računar šalje.

Ukoliko je protok i dalje prevelik može se podešavati frekvencija slanja poruka uz vođenje računa o zadržavanju kvaliteta igre. Odabir informacija i trenutaka u kojima se one šalju može doprineti velikoj uštedi u količini protoka uz održanje visokog kvaliteta igre.

Kašnjenje informacija je problem koji fizički postoji zbog međusobne udaljenosti računara i može se samo smanjiti ili prikriti. Jedno rešenje za ovaj problem je uvođenje osnovnog kašnjenja, čime se stanje igre održava jedinstvenim ali se interaktivnost igrača smanjuje. Drugo rešenje je odabir pogodnog stanja igre, kada server odlučuje koje od različitih stanja će se odabrati za ispravno i prikazati na svih računarima.

Problem sinhronizacije igre je veliki problem za sve moderne multiplayer video igre i od njegovog rešenja zavisi i kvalitet igre.

6. LITERATURA

- [1] Maciej Świechowski, HyunSoo Park, Jacek Mańdziuk, and Kyung-Joong Kim, "Recent Advances in General Game Playing," The Scientific World Journal, vol. 2015, Article ID 986262, 22 pages, 2015. doi:10.1155/2015/986262
- [2] Alf Inge Wang, Martin Jarrett, and Eivind Sorteberg, "Experiences from Implementing a Mobile Multiplayer Real-Time Game for Wireless Networks with High Latency," International Journal of Computer Games Technology, vol. 2009, Article ID 530367, 14 pages, 2009. doi:10.1155/2009/530367
- [3] Bernier W. Yahn, (2001) Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization, at VALVe Developer Community Retrieved at https://developer.valvesoftware.com/wiki/Latency_Compensating_Methods_in_Client/Server_In-game_Protocol_Design_and_Optimization#Footnotes

Kratka biografija:

Marko Tot rođen je u Zrenjaninu 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2016.god.

**IOT MERNO-INFORMACIONI SISTEM ZASNOVAN NA HEXIWEAR
PLATFORMI U OKRUŽENJU PAMETNE KUĆE****IOT INFORMATION MEASUREMENT SYSTEM BASED ON THE HEXIWEAR
PLATFORM IN A SMART HOME ENVIRONMENT**

Dorđe Novaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj—U radu je dat opis implementacije rešenja merno informacionog sistema baziranog na Hexiwear platformi i njegove komunikacije sa web serverom. Daje se opis hardvera i implementiranog firmvera Hexiwear uređaja, a takođe opis implementacije web servera i baze podataka u koju se smeštaju rezultati merenja.

Abstract—This paper contains the description of the implementation of information measurement system based on the Hexiwear platform and it's communication with the web server. A description of the hardware and the implemented Hexiwear device firmware is provided, and also a description of the implementation of the web server and the database in which the results of measurements are stored.

Cljučne reči : Hexiwear, FreeRTOS, pametna kuća, merno-informacioni sistem, Internet of Things.

1. UVOD

Brz razvoj merno informacionih sistema uslovio je razvoj novih koncepata među kojima se *IoT* ističe kao jedan od najzastupljenijih. Sve veći zahtevi korisnika ka automatizaciji uobičajenih radnji veliki su izazovi za razvoj *IoT* sistema. Jedan od najzastupljenijih sistema predstavlja sistem pametne kuće, koji ima zadatak da vrši automatsko upravljanje kućnim aparatima i obaveštavanje korisnika o određenim veličinama od interesa. Pored merenja i upravljanja, neophodno je vršiti i skladištenje podataka čijom se analizom može doći do određenih zaključaka. Na ovaj način pored *IoT* koncepta nameće se i problem rukovanja velikom količinom podataka (*Big Data*).

Današnji sistemi pametnih kuća zahtevaju rad u realnom vremenu kao i pravovremeni odziv na određeni događaj, pa se u tu svrhu kao veoma koristan alat za firmver mikrokontrolera koristi operativni sistem *FreeRTOS*. Komunikacione tehnologije primenjene u *IoT* konceptu zahtevaju bežične protokole. Najzastupljeniji način bežičnog povezivanja predstavlja *WiFi* komunikacija. U tu svrhu je razvijen celokupan *embedded* modul *WiFi ESP* modul. Pravovremeni prikaz podataka na računaru predstavlja podjednako veliki izazov, pa je u tu svrhu za nadgledanje podataka neophodno napraviti klijentsku aplikaciju. Pored toga neophodno je i napraviti svojevrstu bazu podataka iz koje se po potrebi mogu koristiti prikupljeni podaci. Današnji sistemi pored nabrojanih elemenata iziskuju korišćenje baterijskog napajanja zato se mora obratiti pažnja da potrošnja celokupnog sistema bude što je moguće manje.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Platon Sovilj.

2. OPIS SISTEMA

Primarni predmet istraživanja ovog rada predstavlja razvoj merno-informacionog sistema koji uz pomoć *Hexiwear* platforme i *WiFi* modula vrši komunikaciju sa računaru. Uloga ovog sistema je očitavanje podataka sa senzora temperature, relativne vlažnosti vazduha, atmosferskog pritiska i senzora iluminacije, njihovo prikazivanje na ekranu *Hexiwear* uređaja, kao i slanje podataka ka računaru. Takođe ovaj sistem ima ulogu da simulira uključivanje određenih uređaja u domaćinstvu na zahtev korisnika, uključivanjem određene *LED* diode ili pokretanje servo motora. Uloga računara je da uz pomoć web servera, primljenim podacima pohranjuje bazu podataka, vrši prikazivanje podataka korisniku, kao i da na osnovu zahteva generisanih od korisnika vrati poruku ka merno-informacionom sistemu. Omogućen je pristup podacima u bazi podataka, što olakšava analizu podataka.

3. HARDVER

Kao ključna komponenta celokupnog sistema ističe se *Hexiwear* platforma. *Hexiwear* platforma predstavlja prenosivi uređaj razvijen od strane Mikroelektronika, za razvoj aplikacija koje iziskuju *embedded* uređaje sa velikim brojem senzora. Na *Hexiwear* platformi razlikujemo dve ključne komponente *Hexiwear* uređaj (slika 1. levo) i *Hexiwear docking station* (slika 1. desno).



Slika 1. Komponente *Hexiwear* platforme

Hexiwear uređaj u sebi sadrži integrisana dva mikrokontrolera, oba dostupna korisniku na reprogramiranje. Prvi mikrokontroler predstavlja *MK64* mikrokontroler, čija procesorska jedinica predstavlja *ARM Cortex-M4* procesor i brzinom takta od 120 MHz, odlikuje ga veliki broj hardverski implementiranih interfejsa. Drugi mikrokontroler integrisan u ovaj uređaj je *KW40Z* mikrokontroler, sadrži procesorsku jedinicu u *ARM Cortex-M0* arhitekturi, ali njegova prvenstvena uloga je u vršenju komunikacije sa računaru uz pomoć *bluetooth* komunikacije, kao i

praćenje statusa kapacitivno osetljivog panela. U *Hexiwear* uređaj je implementirano šest senzora i to: troosni žiroskop, troosni akcelerometar, senzor za merenje pulsne oksimetrije, senzor iluminacije, senzor temperature i relativne vlažnosti vazduha i barometar sa altimetrom. Zbog pomenutog broja senzora i integrisanih mikrokontrolera, ovaj uređaj se nametnuo kao idealno rešenje za razvoj sistema pametne kuće. Takođe *Hexiwear* uređaj sadrži i dodatne komponente poput OLED 1,1 inch ekrana, *haptic* vibracione povratne sprege, dodatni čip *flash* memorije, integrisanu *Li-Ion* bateriju kao i odgovarajući regulator punjenja baterije. Uređaj je moguće da radi bez konekcije sa *Hexiwear docking station* pločom, i punjenje se obavlja posredstvom *USB micro* kabla. *Hexiwear* uređaj je predviđen da radi i kao prenosivi uređaj pa se u tu svrhu uz pomoć specijalnih adaptera može koristiti kao sat ili kao ogrlica.

Ako pomenutih šest senzora integrisanih u ovom uređaju nisu dovoljni, ostavljena je mogućnost korisniku da uz pomoć *Hexiwear docking station* vrši dodavanje još tri modula. Dodatni moduli moraju biti kompatibilni sa *microBUS* šesnaestopinskim standardom čiji je tvorac Mikroelektronika. U ovom radu iskorišćena su dva dodatna modula jedan predstavlja *WiFi ESP* modul, dok drugi modul predstavlja prilagođenje napona za kontrolu servo motora. Takođe uz pomoć ove ploče je moguće vršiti reprogramiranje mikrokontrolera, kao i debugovanje u realnom vremenu. Ovo je omogućeno korišćenjem *OpenSDA* (*Open-standard Serial and Debug Adapter*) adaptera koji za svoj rad koristi *K20* mikrokontroler. Selekcija mikrokontrolera na kome se žele vršiti izmene koda ili debugovanje u realnom vremenu vrši se uz pomoć odgovarajućeg kontrolnog prekidača. Korisniku je omogućeno da može dodati i *microSD* karticu u predviđeni adapter na ploči, čime se znatno mogu proširiti memorijski kapaciteti mikrokontrolera. Ostavljena je mogućnost i korišćenja spoljašnjih programatora uz pomoć *JTAG* konektora.

Priključeni modul *WiFi ESP* modul, predstavlja mikrokontroler *ESP8266* sa integrisanom antenom i koristi se za *WiFi* komunikaciju. *WiFi ESP* pločica je predviđena da se napaja preko linije 3,3 V, a komunikacija se vrši posredstvom *UART* komunikacije slanjem *AT* komandi. Ovaj modul je predviđen da radi u tri moguća moda *station mode*, *softAP* (*software enabled access point*) i u oba moda paralelno.

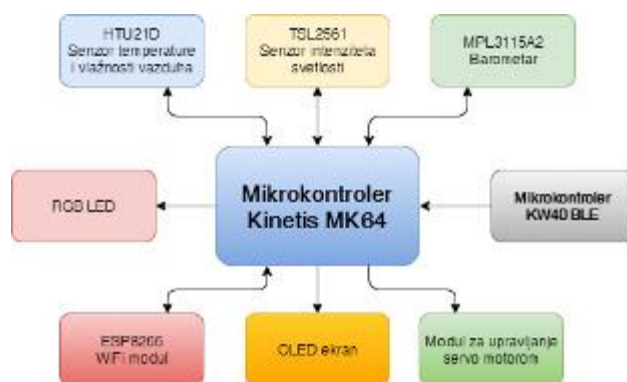
4. FIRMVER

Za pisanje firmvera korišćeno je razvojno okruženje mikroC PRO for ARM, razvijeno od strane kompanije "Mikroelektronika". MikroC PRO for ARM je programsko okruženje za programiranje mikrokontrolera u programskom jeziku C, predviđeno za programiranje ARM Cortex M0, M0+, M3, M4 i M7 serije mikrokontrolera. Prednost ovog okruženja je veliki broj napisanih i testiranih biblioteka poput SPI, I²C, UART biblioteka za komunikaciju koje su u mnogome olakšale izradu ovog rada. Pored velikog broja biblioteka ovo okruženje podržava hardversko debugovanje, čime se olakšalo nadgledanje rada celokupnog firmvera, kao i otklanjanje potencijalnih grešaka. Celokupan firmver je predviđen da za svoj rad koristi FreeRTOS, čime kod postaje čitljiviji i jednostavniji za razumevanje. MikroC PRO for ARM nudi podršku za rad sa FreeRTOS operativnim sistemom.

Pomenuti firmver se može podeliti u nekoliko jasno definisanih celina. Izvršavanje programa počinje generisanjem inicijalnog zadatka, koji ima ulogu da prilagodi sve senzore i korišćene periferije radu, kao i konfigurisanja hardverskih prekida i tajmera u sklopu FreeRTOS-a. Nakon toga se generišu tri zadatka koji se paralelno izvršavaju i imaju različit nivo prioriteta. Zadatak najnižeg prioriteta predstavlja zadatak koji ima ulogu da upravlja prikazom podataka na ekranu. Drugi zadatak ima ulogu pravovremenog očitavanja podataka sa senzora, kao i konverziju pročitanih podataka i njihovo skladištenje u red. Zadatak najvišeg prioriteta ima ulogu da čeka sve dok ne pristignu podaci za aktuatora od strane klijenta posredstvom ESP8266 WiFi modula.

4.1. Inicijalni zadatak

Pre nego što se počne sa korišćenjem bilo koje od periferija neophodno ih je konfigurisati. Korišćene periferije date su na slici 3.2.



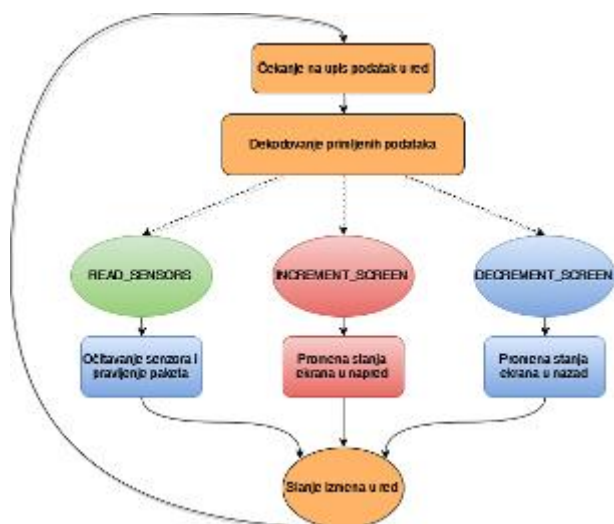
Slika 2. Inicijalizovane periferije

Za komunikaciju sa sensorima neophodno je konfigurisati I²C komunikacioni interfejs uz pomoć ugrađenih biblioteka. Firmver za svoj rad koristi tri interapta, koje je neophodno inicijalizovati i konfigurisati. Za komunikaciju sa *WiFi ESP* modulom neophodno je inicijalizovati *UART2* modul koji koristi pinove PD2 kao RX tj. PD3 kao TX liniju. Brzina prenosa podataka je 115000 bps sa jednim stop bitom, bez bita parnosti. Za komunikaciju između MK64 i KW40 mikrokontrolera koristi se *UART4* modul konfigurisan brzinom 230400 bps, bez bita pariteta, ali sa dva stop bita. Na svaki od ovih *UART* modula je konfigurisan interapt, pa će pri prijemu podataka na bilo koji od postojećih *UART* modula biti generisana interapt rutina sa ulogom da primi određene podatke. Za rad sa servo-motorom neophodno je koristiti tajmerski interapt, za generisanje *PWM* signala.

Nakon konfigurisanja perifernih jedinica, inicijalizuju se tri zadatka koji se paralelno izvršavaju na mikrokontroleru. Nakon generisanja ovih zadataka, inicijalni zadatak se završava.

4.1. Zadatak koji očitava senzore

Zadatak koji očitava senzore u ovom radu je nazvan *sensorDriver* zadatak, i njegova uloga je da vrši očitavanje sa svih senzora, a zatim da vrši njihovo slanje u red ka drugim zadacima koji obrađuju te podatke. Ovaj zadatak je konfigurisan kao mašina stanja prikazana na slici 3.



Slika 3. *sensorDriver* zadatak

Zadatak se nalazi u stanju čekanja dok se ne prime podaci u red. Nakon prijema podataka vrši se raspoznavanje koje od mogućih stanja je primljeno. *INCREMENT_SCREEN* stanje ima zadatak da promeni trenutni prikaz ekrana na sledeći, dok *DECREMENT_SCREEN* stanje ima ulogu da vrati na prethodni prikaz ekrana. Prikazi ekrana smenjivati automatski na svakih 20 s, ako to korisnik ne zahteva drugačije. Uloga *READ_SENSORS* stanja je da očitava sve senzore a zatim sve pročitane podatke stavi na raspolaganje ostalim zadacima upisom očitanih podataka u red.

4.2. Zadatak koji upravlja ekranom

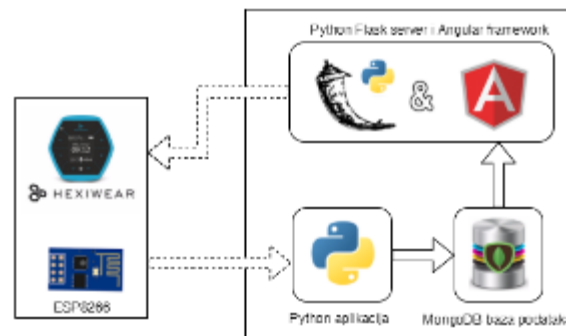
Uloga ovog zadatka je da prima podatke od strane *sensorDriver* zadatka, da vrši njihovu interpretaciju i ispis na *OLED* ekranu. I ovaj zadatak se ponaša kao mašina stanja (*state machine*), gde se sistem može naći u četiri moguća stanja tj. četiri moguća prikaza na ekranu. Prva slika predstavlja prikaz temperature, druga slika predstavlja prikaz relativnu vlažnost vazduha, treća slika prikazuje intenzitet atmosferskog pritiska i četvrta slika prikazuje vrednost iluminacije. Svaka od tih slika je sačuvana u fleš memoriji mikrokontrolera kao *bitmap* slika, koja se u određenom stanju prosleđuje odgovarajućoj funkciji koja ima ulogu da je prikaže na ekranu. Nakon prikaza slike na ekranu, vrši se dekodovanje očitanih podataka sa odgovarajućeg senzora. Očitana vrednost se prelepljuje preko slike, čime korisnik dobija informaciju o podacima sa senzora jednog od četiri stanja. Nakon dekodovanja i prikaza odgovarajućih podataka na ekranu, zadatak se vraća u blokirano stanje gde čeka na nove elemente u redu. Osvežavanje podataka na ekranu se vrši svake sekunde.

4.3. Zadatak prijema preko WiFi modula

Uloga ovog zadatka je da čeka na podatke koje prosleđuje klijent i da izvrši određenu akciju. Po prijemu podataka na *UART* interapt rutina postavlja daje semafor, čime ovaj zadatak biva odblokiran. Prvo se vrši proverava primljenih podataka preko *UART*-a, upravo onih koje prosleđuje klijent. Primljeni kod je u csv formatu, pa ga je neophodno parsirati. Iz primljenih podataka izvlače se četiri promenljive: tri su bitske i odgovaraju stanjima *LED* dioda, dok četvrto stanje predstavlja celobrojnu vrednost pozicije koju treba da dosegne servo motor. Nakon izvršenja odgovarajućih akcija, zadatak se vraća u blokirano stanje čekajući na novi semafor od strane interapt rutine.

5. SOFTVER

Da bi se celokupan sistem koristio u IoT konceptu, neophodno je prikupljati podatke od hexiwear uređaja i skladištiti ih na računaru. Iz tog razloga je na računaru napravljen web server, koji korisniku omogućava pregled primljenih podataka u prethodnih trideset sekundi. Takođe korisniku je omogućen uvid u sve primljene podatke koji se nalaze smešteni u *MongoDB* bazi podataka. Na slici 4. prikazana je struktura softverskog dela.



Slika 4. Struktura softverskog dela

Na računaru se paralelno izvršavaju dve python aplikacije. Jedna aplikacija je zadužena za prijem podataka sa *Hexiwear* uređaja i njihovo skladištenje u *MongoDB* bazi podataka. Druga aplikacija predstavlja *backend* web servera, koji je napravljen uz pomoć *Flask framework*-a. *Frontend* web servera je napravljen uz pomoć *Angularframework*-a.

5.1. Python aplikacija za prijem podataka

Da bi podaci bili skladišteni u *MongoDB* bazi podataka neophodno je da prethodno budu primljeni od *ESP8266* mikrokontrolera, posredstvom *WiFi* komunikacije i *TCP* (*Transmission Control Protocol*) protokola. Pošto je *ESP8266* podešen da radi u modu *softAP*, neophodno je ostvariti vezu sa njime sa računara. Kada se računar konektuje na *WiFi ESP* modul, moguće je početi sa prijemom podataka. Realizacija prijema podataka se vrši uz pomoć soketa. Primljeni podaci su predstavljeni kao string, pa da bi bili skladišteni u *mongoDB* bazu neophodno ih je formatirati u *JSON* format. Nakon prijema i formatiranja podataka, na njih se dodaje trenutno vreme i vrednost celokupnog paketa se upisuje u bazu podataka.

5.2. Python aplikacija web servera

Za realizaciju backend aplikacije web servera korišćen je *Flask framework*. Kada klijent šalje *GET* zahtev vrši se izdvajanje poslednjih 15 podataka iz *MongoDB* baze podataka, i prikazuju se klijentu u web browseru. Kada klijent generiše *POST* zahtev, vrši se direktno slanje podataka ka *WiFi ESP* modulu, koji vrši konverziju podataka i preduzima određenu akciju.

5.3. Java script kontroler forntend-a

Uloga ovog kontrolera je da vrši prijem podataka iz baze i njihovo prikazivanje u browseru, kao i pravovremeno detektovanje *POST* zahteva od strane korisnika. Kada se šalje *GET* zahtev, od servera se dobija odgovor u vidu podataka pročitanih iz baze podataka. Primljeni podaci predstavljaju 15 najskorijih merenja u *JSON* formatu. Ti podaci se razvrstavaju u podgrupe za podatke o temperaturi, vlažnosti vazduha, pritisku i iluminaciji. Ovaj kontroler ima relaizovanu funkciju koja obezbeđuje slanje

novog GET zahteva svake sekunde, čime se vrši automatsko osvežavanje sadržaja stranice.

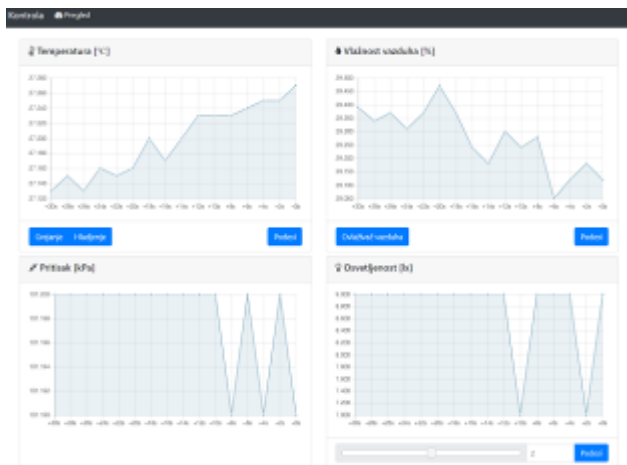
5. TESTIRANJE SISTEMA I REZULTATI

IoT merno-informacioni sistem zasnovan na hexiwear platformi u okruženju pametne kuće, testiran je u kućnim uslovima. Sistem se pokazao stabilno i u svrhu testiranja je izvršeno prikupljanje podataka, sa više od 100 000 merenja. Takođe sistem je preduzimao zadatu akciju, kada bi klijent prosledio podatke hexiwear platformi. Na slici 5., date su slike četiri moguća stanja ekrana hexiwear uređaja, takođe je izvršeno uključivanje pojedinih aktuatora što reprezentuje uključena LED dioda.



Slika 5. Testiranje rada Hexiwear uređaja

Ispitivanje rada web servera na računaru kao i prikaz dobijenih rezultata dat je na slici 6.



Slika 6. Rezultati prikazani na web serveru

6. ZAKLJUČNA RAZMATRANJA

Realizovani IoT merno-informacioni sistem zasnovan na hexiwear platformi u okruženju pametne kuće, omogućava korisniku da vrši nadzor osnovnih parametara okruženja u kući, kao i preduzimanje odgovarajućih akcija. Pošto je sistem za merenje integrisan kao gotov proizvod, ovaj uređaj se može koristiti i u komercijalne

svrhe. Sistem odlikuje stabilan rad i relativno malo vreme odziva. Celokupan sistem bi mogao biti povezan na internet tj. globalnu mrežu, ali bi se u tom slučaju javili dodatni problemi. Najvažniji problem predstavlja problem zaštite i bezbednosti sistema da ne bi došlo do zloupotrebe od strane trećih lica. Potencijalni problem predstavlja ponašanje ako više lica želi da vrši zadavanje komandi. Ako bi se sistem povezao na globalnu mrežu, moglo bi se pristupiti i izradi mobilne aplikacije. Sistem bi se mogao nadograditi da vrši kontinualan nadzor svih pomenutih parametara, i ako bi neki od parametara izlazio iz zadatog opsega korisnik bi o tome bio obavešten putem notifikacija. Moguće je takođe dodati još nekoliko senzora čime bi se poboljšao kvalitet samog uređaja. Takođe mogao bi se unaprediti sistem tako da automatski vrši prepoznavanje dodatnih elemenata, čime bi se omogućila plug&play tehnika.

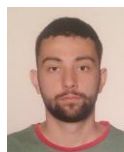
Ono što predstavlja još jednu bitnu karakteristiku jeste da se hexiwear uređaj ne mora koristiti na docking station modulu za svoj rad. U tom slučaju se gubi veza između hexiwear uređaja i web servera, ali se podaci mogu pratiti na OLED ekranu. U ovakvom režimu rada hexiwear uređaj koristi svoju internu bateriju i na taj način može raditi nezavisno i do 3 sata. Još jedan od potencijalnih problema sa kojima se korisnik može susresti je problem napajanja. Potrošnja celokupnog sistema u radnom stanju ne prelazi 150 mA. Što znači da se na komercijalno dostupnim baterijama uređaj može raditi do 4 dana. Ovo može predstavljati problem, ako se sistem želi nadzirati sa udaljenih mesta. Rešenje bi bilo prelazak na napajanje iz mreže, ali se na taj način uređaj ne može prenositi.

Dakle, uređaj se pokazao kao veoma stabilan za rad i može se koristiti kako u naučne, tako i u komercijalne svrhe. Uz malo izmena i dodataka, sistem bi mogao parirati sličnim uređajima na tržištu.

7. LITERATURA

- [1] Branko Milosavljević, Milan Vidaković, „Java i internet programiranje“, Fakultet Tehničkih nauka u Novom Sadu, Novi Sad 2014.
- [2] Richard Barry, „Mastering the FreeRTOS Real Time Kernel“, Real Time Engineers Ltd., 2016.
- [3] Miguel Grinberg, „Flask Web Development, developing web applications with python“, O'Reilly, 2014.
- [4] Cuno Pfister, „Getting started with internet of things“, O'Reilly, 2011.
- [5] <https://www.mikroe.com> (15.09.2017.)

Kratka biografija



Dorde Novaković je rođen u Loznici 1993. godine. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, na katedri za električna merenja 2016. godine. Master rad iz oblasti memo-informacionih sistema, odbranio je 2017. godine.

ALGORITMI ZA GENERISANJE LAVIRINATA U VIDEO IGRAMA**MAZE GENERATING ALGORITHMS IN VIDEO GAMES**

Aleksandar Manasijević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – Ovaj rad ilustruje problem stvaranja lavirinata u video-igrama. Takođe, u ovom radu će biti opisani i detaljno objašnjeni najpoznatiji algoritmi za rešavanje ovog problema.

Abstract – This paper illustrates a problem for generating mazes in video-games. Also, in this paper, the most well-known algorithms for solving this problem will be described and explained in detail.

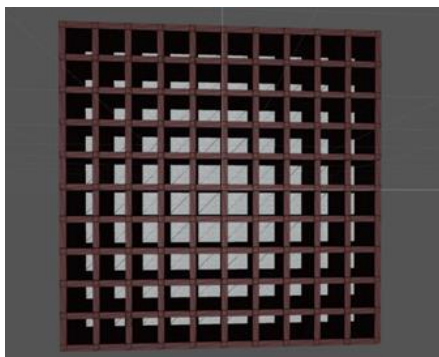
Ključne reči: Lavirint, Video-igre, generisanje, algoritmi

1. UVOD

Video-igre su jedan od osnovnih vidova zabave u današnje vreme, bilo da se igraju na kompjuteru, telefonu ili konzoli. Postoji mnogo žanrova video-igara, a samo neki od njih su RPG-ovi, akcione igre, pućačine (iz prvog i trećeg lica) i platformerske igre. Mnoge od njih su napravljene sa akcentom na zabavni karakter, dok postoje i neke koje služe u edukacione svrhe, tako da je njihova primena veoma rasprostranjena.

2. ALGORITMI ZA GENERISANJE LAVIRINTA**2.1. Rekurzivni Backtrack algoritam**

Pripada algoritmima za pretraživanje, tačnije predstavlja samo jednu od verzija algoritma Prvi u dubinu (eng. Depth-First Search). Najčešće se implementira sa stekom. Predstavlja jedan od najjednostavnijih pristupa za generisanje lavirinta. Sam početak generisanja predstavlja velika mreža ćelija (poput velike šahovske table). Svaka ćelija prvobitno ima sva četiri zida kao što je prikazano na slici 1.



Slika 1. Ćelije lavirinta sa sva 4 zida

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinać, van.prof.

Algoritam kreće od slučajno izabrane ćelije. Sledeći korak jeste biranje jedne susedne ćelije koja nije posećena i to na slučajan način. Zatim sledi uklanjanje zida između ove dve ćelije i označavanje novoizabrane ćelije kao posećene pri čemu se ona dodaje na stek kako bi se olakšao povratak. Ovaj proces se nastavlja dok se ne stigne do ćelije koja nema neposećenih suseda.

U ovoj situaciji, navedena ćelija se smatra "čorsokakom". Kada se ovo dogodi, algoritam se vraća po jedan korak unazad sve dok ne naiđe na ćeliju koja je na steku i koja ima neposećenih suseda, a zatim sledi faza uklanjanja zida između te dve ćelije (kao što je ranije bilo opisano). Ovaj proces se nastavlja sve dok se ne poseti svaka ćelija, što dovodi do toga da će se algoritam vratiti u početnu ćeliju (odakle je i krenulo generisanje).

Kao što je već navedeno, ovaj algoritam podrazumeva duboku rekurziju koja može prouzrokovati probleme nedostatka memorije kod nekih kompjuterskih arhitektura. Naravno, do ovog problema može doći i zbog dimenzija lavirinta. Algoritam se može optimizovati ukoliko dođe do ovakvih problema.

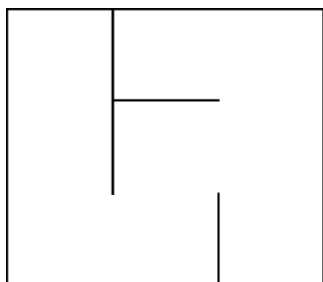
Lavirint generisan pomoću algoritma Prvi u dubinu ima nizak faktor grananja i sadrži mnogo dugačkih putanja jer se istražuje koliko god je moguće duž svake grane pre povratka.

2.1.1 Pseudokod rekurzivnog Backtrack algoritma za generisanje lavirinta

Algoritam pretrage Prvi u dubinu generisanja lavirinta se često realizuje korišćenjem backtrack-a (povratak unazad):

1. Postavite početnu ćeliju kao trenutnu i označite je kao posećenu
2. Dok postoje neposećene ćelije
 1. Ako trenutna ćelija ima susede koji nisu posećeni
 1. Odaberi slučajno jednog od neposećenih suseda
 2. Stavite trenutnu ćeliju na stek
 3. Uklonite zid između trenutne ćelije i izabrane ćelije
 4. Postavite izabranu ćeliju za trenutnu i označite je kao posećenu
 2. U suprotnom, ako stek nije prazan
 1. Uklonite ćeliju sa steka
 2. Postavi je za trenutnu ćeliju

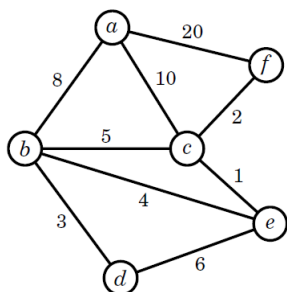
Na slici 2 prikazan je primer izgenerisanog lavirinta korišćenjem rekurzivnog Backtrack algoritma.



Slika 2. Primer izgenerisanog lavirinta rekursivnim Backtrack algoritmom

2.2. Kruskalov algoritam

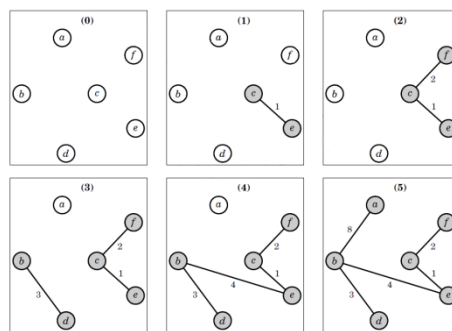
Ovaj algoritam je od velikog značaja kada je neophodno naći takav kostur grafa da je njegov zbir težina grana, od koji je on sastavljen, što manji. Ovakav kostur naziva se minimalno povezujuće stablo (eng. minimal spanning tree). Treba napomenuti da broj kostura tj. načina povezivanja stabla može biti veliki. U ovakvim situacijama, primena nekog od algoritama za pronalaženje minimalnog povezujućeg stabla jeste rešenje. Takođe, neophodno je napomenuti da se problem minimalnog povezujućeg stabla razmatra samo za slučaj stabla sa neusmerenim grafovima. Primer ovakvog stabla je prikazan na slici 3. Ovaj algoritam pripada grupi "pohlepnih" algoritama jer njegovo rešenje zavisi od lokalnog. Ipak, njegov postupak zaista dovodi do minimalno povezujućeg stabla. Kruskalov algoritam koji se koristi za generisanje lavirinata je malo izmenjen u odnosu na pravi Kruskalov algoritam za pronalaženje minimalnog povezujućeg stabla.



Slika 3. Primer stabla sa neusmerenim grafovima

Originalna verzija Kruskalovog algoritma kreće od šume koja se sastoji od svih čvorova grafa. Neophodno je naglasiti da na početku, u ovom prvom koraku ne postoje grane. Dakle, navedena šuma grafova sastoji se od pojedinačnih čvorova. Svaki čvor je izolovan sam za sebe i nema nijednu vezu sa nekim drugim čvorem (nema nijednu granu). Sledeći korak jeste ispitivanje svih grana i traženje one koja ima najmanju težinu. Ukoliko je više takvih grana, bira se jedna koja će biti polazna.

Svaki sledeći korak jeste biranje sledeće grane koja ima najmanju težinu od svih ostalih neizabranih grana, pri čemu je neophodno voditi računa da se ne napravi ciklus sa ostalim, već izabratim granama. Tačnije, ukoliko novoizabrana grana povezuje dva čvora iz dva različita stabla, onda se ona dodaje skupu izabranih grana, a stabla se objedinjuju u jedno. Ukoliko novoizabrana grana spaja dva čvora tako da su oba iz istog stabla, onda se ta grana odbacuje jer bi se na ovaj način formirao ciklus. Na slici 4 je postupno prikazano izvršavanje Kruskalovog algoritma za graf sa slike 3.



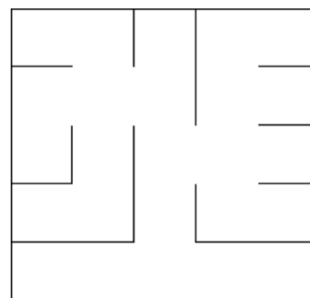
Slika 4. Primer izvršavanja Kruskalovog algoritma

2.2.1 Pseudokod Kruskalovog algoritma za generisanje lavirinta

Sledi pseudokod izmenjenog Kruskalovog algoritma koji se koristi za generisanje lavirinata:

1. Kreirati lista svih zidova i kreirati skup za svaku ćeliju posebno, tako da svaki skup sadrži samo jednu ćeliju.
2. Za svaki zid, po nekom nasumičnom redu:
 1. Ako ćelije koje su podeljene nasumično izabranim zidom pripadaju različitim skupovima:
 1. Ukloniti trenutno izabrani zid
 2. Spojiti skupove ćelija (koje su prethodno delile uklonjen zid)

Na slici 5 prikazan je primer izgenerisanog lavirinta Kruskalovim algoritmom.



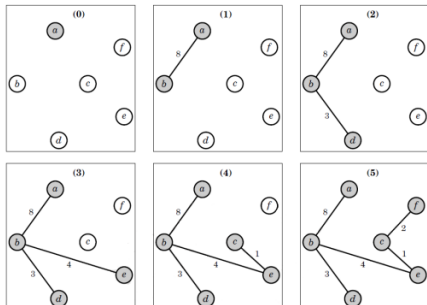
Slika 5. Primer izgenerisanog lavirinta Kruskalovim algoritmom

2.3. Primov algoritam

Ovaj algoritam se takođe koristi za nalaženje minimalnog povezujućeg stabla. Za razliku od Kruskalovog algoritma, gde se grane najmanje težine biraju „pohlepno” neuzimajući u obzir odnos sa prethodno izabranim granama (osim što se pazi na formiranje ciklusa), kod Primovog algoritma u svakoj iteraciji postoji stablo koje se postepeno proširuje. Kao i kod Kruskalovog algoritma, za generisanje lavirinta, neophodno je malo modifikovati originalnu verziju.

Originalna verzija Primovog algoritma kreće od nasumičnog biranja čvora grafa i njegove grane sa najmanjom težinom. U svakoj sledećoj iteraciji, stablu koje još uvek nije u potpunosti konstruisano dodaje se novi čvor. Dodavanje se vrši na drugom kraju grane sa najmanjom težinom u celokupnom stablu. Na ovaj način

stablo grafova se uvećava za po jedan čvor u svakoj iteraciji, pri čemu je ta cena minimalna s obzirom da je dodat graf na grani sa trenutno najmanjom težinom. Algoritam se završava formiranjem minimalnog kostura stabla tj. dok se ne spoje svi čvorovi na naveden način. Postupno izvršavanje Primovog algoritma za graf sa slike 2 je prikazan na slici 6 u nekoliko iteracija. Za početni čvor grafa izabran je čvor a.



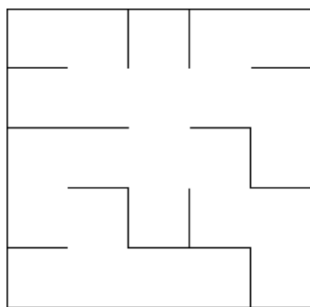
Slika 6. Primer izvršavanja Primovog algoritma

2.3.1 Pseudokod Primovog algoritma za generisanje lavirinta

Pseudokod izmenjenog Primovog algoritma koji se koristi za generisanje lavirinta:

1. Krenuti od mreže ćelija, gde svaka ćelija sadrži sva četiri zida.
2. Izabrati ćeliju i označiti je kao deo lavirinta. Dodati zidove ćelije u listu zidova.
3. Dok postoje zidovi u listi zidova:
 1. Izaberite slučajni zid iz liste zidova. Ako je posećena samo jedna od ćelija koje dele zid, onda:
 1. Ukloniti zid i od njega napraviti prolaz, a neposećenu ćeliju obeležiti kao deo lavirinta.
 2. Dodati susedne zidove ćelije u listu zidova.
 2. Izbrisati zid iz liste zidova.

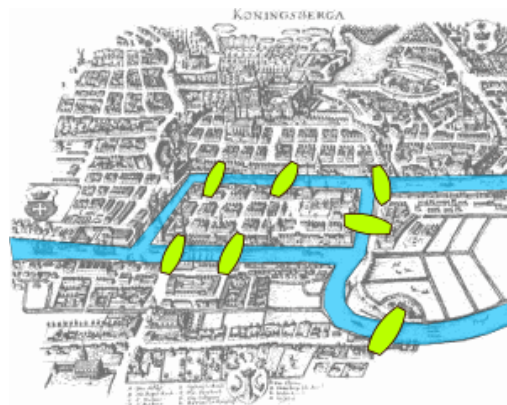
Na slici 7 je prikazan lavirint izgenerisan korišćenjem Primovog algoritma za generisanje lavirinata.



Slika 7. Primer izgenerisanog lavirinta Primovim algoritmom

2.4. Ojlerov algoritam

Glavni grad Pruske, Kenigsberg, danas Kalinjingrad, nalazio se na reci Pregel. Njegova teritorija prostirala se na dva velika rečna ostrva. Ostrva su bila povezana sa ostatkom grada, kao i međusobno, pomoću sedam mostova. Građani su često postavljali pitanje da li je moguće poći sa određenog mesta i vratiti se na isto, tako da se, prilikom obilaženja grada, svaki most pređe tačno jednom.



Slika 8. Ilustracija problema "Sedam mostova Keninsberga"

Ojler se smatra jednim od osnivača teorije grafova. On je 1735. godine Sang Petersburškoj akademiji nauka prezentovao ovaj problem, a već 1736. godine objavljuje rad pod nazivom "Sedam mostova Kenigsberga" gde je priložio rešenje. U ovom radu formulisao je potrebne i dovoljne uslove za rešavanje ovog problema i napomenuo kako se njegov metod može proširiti na proizvoljan raspored ostrva i mostova. Takođe se navodi kako je pod zadatim uslovima prethodni problem nemoguće rešiti, što znači da ne postoji Ojlerov put. Navedeno rešenje se smatra prvom teoremom teorije grafova odnosno teorije planarnih grafova, a 1736. godina godinom začetka teorije grafova. Navedeni problem prikazan je na slici 8.

Ojlerov algoritam za generisanje lavirinta zasniva se na pojmu Ojlerovog puta. To je zapravo putanja gde se kroz svaku granu grafa, koji je neusmeren, prolazi tačno jednom. Dakle, Ojlerov put postoji ukoliko graf ili multigraf sadrži dva čvora koja su neparnog stepena. Specijalni slučaj Ojlerovog puta jeste kada ne postoji ni jedan jedini čvor u grafu koji je neparnog reda. Ovaj slučaj se naziva zatvoren Ojlerov put. Ono što je neophodno naglasiti jeste to da je prilikom obilaženja grafa po Ojlerovom putu moguće posetiti neki čvor i više puta ako je to neophodno. Kako svaki čvor (osim početnog i krajnjeg) mora imati po jednu izlaznu granu za svaku ulaznu, iz toga sledi da stepen svih čvorova mora biti paran (naravno, osim početnog i krajnjeg).

Ojlerov algoritam je jedan od najneobičnijih, kao i jedan od najbržih za generisanje lavirinata. Ovo je jedini algoritam koju može izgenerisati lavirint beskonačne veličine i to u linearnom vremenu, što ga čini vrlo neobičnim.

Lavirint se izgrađuje postepeno i to tako što se izgeneriše jedan ceo red odjednom. Prilikom generisanja reda, ćelije se klasifikuju u skupove kako bi se pratila njihova povezanost. Posebno je interesantno to što algoritam nikada ne prati više od jednog reda. Na kraju, kada se završi generisanje, rezultat je uvek savršeno izgenerisan lavirint.

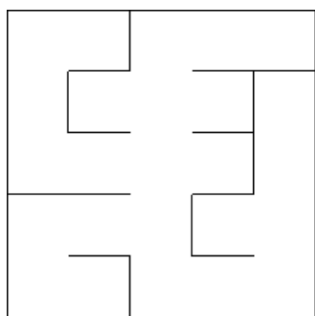
2.4.1 Pseudokod Ojlerovog algoritma za generisanje lavirinta

Ojlerov algoritam za generisanje lavirinta se sastoji od određenog broja koraka:

1. Inicijalizacija ćelija prvog reda tako da za svaka ćelija pripada svom sopstvenom skupu.

2. Vršiti se spajanje susednih ćelija na slučajan način, ali samo ako ne pripadaju istom skupu. Kada se neke ćelije spoje, spajaju se i skupovi ovih ćelija što znači da su sve ćelije sada povezane (postoji put koji povezuje bilo koje dve ćelije u skupu).
3. Za svaki skup, slučajno se kreira vertikalna veza nadole do sledećeg reda. Svaki preostali skup mora imati najmanje jednu vertikalnu vezu. Ćelije sledećeg reda ovako povezane moraju deliti skup ćelije iznad njih.
4. Sledeći red se popuni stavljanjem svih preostalih ćelija u svoje zasebne setove.
5. Ponavljati prethodne korake dok se ne dostigne poslednji red.
6. Za poslednji red pridružuju se sve susedne ćelije koje ne dele skup i izostavljaju se vertikalne veze.

Na slici 9 kao rezultat prethodno opisanog algoritma, prikazan je izgenerisan lavirint.



Slika 9. Primer izgenerisanog lavirinta Ojlerovim algoritmom

3. ZAKLJUČAK

Rekurzivni backtrack algoritam se od svih algoritama za generisanje lavirinata najčešće koristi. Razlog je očigledan – jednostavan je za implementaciju i vrlo lako je shvatiti način njegovog funkcionisanja. Pored toga, ovaj algoritam generiše možda čak i najlepše lavirinte jer oni uvek sastoje od dosta dugih putanja, za razliku od Kruskalovog i Primovog algoritma.

Kruskalov algoritam jeste zanimljiv za implementaciju i posmatranje njegovog iterativnog razvoja, ali način generisanja lavirinata nije zadovoljavajući. On teži ka stvaranju puno kratkih puteva bez izlaza (eng. dead-ends), što i nije baš tako poželjno zbog zanimljivosti same igre. Jedna oblast u kojoj Kruskalov algoritam radi bolje od algoritma kao što je rekurzivni backtrack algoritam jeste kada je neophodno napraviti lavirint sa dva ili više nepovezanih područja npr. pravljenje dva ili više lavirinta koji su određenog oblika (oblika slova npr.). U suštini, pravljenje tako složenog lavirinta je isto kao i pravljenje više manjih lavirinata, ali sa Kruskalovim algoritmom se mogu izgenerisati svi odjednom pošto se on generiše samo pomoću ivica, a ne direktnom vezom.

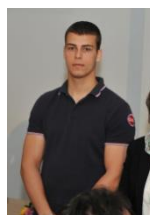
Lavirint generisan Primovim algoritmom deli mnoge karakteristike sa onima kreiranim pomoću Kruskalovog algoritma.

Glavna karakteristika jeste obilje kratkih putanja. Brzina, kao i mogućnost generisanja "beskonačnog" lavirinta jesu glavne odlike Ojlerovog algoritma za generisanje lavirinata. Ovaj algoritam je ubedljivo najbrži od ostalih opisanih. Takođe, jedna od zajedničkih karakteristika sa Kruskalovim i Primovim algoritmom jeste generisanje kratkih putanja.

4. LITERATURA

- [1] Maze generation algorithm on https://en.wikipedia.org/wiki/Maze_generation_algorithm
- [2] Dr. Željko Jurić: Diskretna matematika (novi kurs) Predavanje 10, Radna skripta za istoimeni kurs na Elektrotehničkom fakultetu u Sarajevu (akademska godina 2016/17)
- [3] Uvod u Algoritme i Strukture PODATAKA, Autor: Prof. dr Dejan Živković, Izdavač: UNIVERZITET SINGIDUNUM
- [4] Maze Generation: Recursive Backtracking on <http://weblog.jamisbuck.org/2010/12/27/maze-generation-recursive-backtracking>
- [5] Maze Generation: Kruskal's Algorithm on <http://weblog.jamisbuck.org/2011/1/3/maze-generation-kruskal-s-algorithm.html>
- [6] Maze Generation: Prim's Algorithm on <http://weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm.html>
- [7] Maze Generation: Eller's Algorithm on <http://weblog.jamisbuck.org/2010/12/29/maze-generation-eller-s-algorithm>
- [8] Leonhard Euler on https://sh.wikipedia.org/wiki/Leonhard_Euler

Kratka biografija:



Aleksandar Manasijević rođen je u Vranju 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2016.god.

**SOFTVERSKA PODRŠKA ZA PREDIKCIJU KOLIČINE VLAGE U ZEMLJIŠTU U
SISTEMIMA ZA NAVODNJAVANJE****A SOFTWARE FOR HUMIDITY LEVEL PREDICTION IN SOIL IN THE IRRIGATION
SYSTEMS**

Nikola Todorović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu predstavljen je sistem za predikciju budućeg kretanja vlage u zemljištu. Radi uspostavljanja ovakvog sistema, osmišljena je i implementirana arhitektura sistema koja podržava skladištenje i analizu velikih masiva podataka generisanih od strane senzora postavljenih na njivama. Nakon toga, odabrana je odgovarajuća predikciona tehnika, čijom se primenom dobijaju predikcioni modeli koji s dovoljnom preciznošću predviđaju buduće vrednosti količine vlage u zemljištu.

Abstract – A software for humidity level prediction in soil in the irrigation systems is presented in this work. The architecture for storage and analysis of data generated by soil moisture sensors was designed and implemented as a part of this software solution. A prediction technique that produces sufficiently accurate predictive models was selected.

Ključne reči: ECTL proces; Sistemi skladištenja podataka; Metode za predikciju.

1. UVOD

Projekat Agronet, u okviru kojeg je obuhvaćen i projekat Irignet [1], čije proširenje je tema ovog rada, ima za cilj unapređenje tradicionalnih tehnika obrade useva korišćenih u poljoprivredi. Način na koji se unapređenje pokušava postići jeste primena koncepta *Internet of Things (IoT)* u poljoprivrednoj proizvodnji. Cilj je da se pomoću podataka dobijenih korišćenjem različitih senzora poveća kvalitet useva, povećaju prinosi, smanji potrošnja vode i korišćenje pesticida i smanji negativan uticaj poljoprivrede na životnu sredinu. Ovakav način obrade useva u poljoprivredi poznat je pod nazivom „Precizna poljoprivreda“ (Precision Agriculture).

Jedan od ciljeva projekta Irignet predstavlja optimizacija navodnjavanja useva. Motivacija ovog projekta proizilazi iz činjenice da se najčešće korišćene tehnike za navodnjavanje, pri njihovom korišćenju u realnim uslovima, u velikoj meri zasnivaju na gruboj proceni vrednosti parametara njive, ključnih za razvoj biljaka, umesto na egzaktnim činjenicama, čime može doći do preteranog ili nedovoljnog navodnjavanja useva. Takav pristup navodnjavanju može dovesti do povećanja troškova proizvodnje, smanjenja prinosa, uništavanja zemljišta, i, na kraju, do gubitka profita.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Ivan Luković, red. prof.

Cilj ovog projekta je da omogući prevazilaženje navedenih problema, tako što će obezbediti kreiranje recepta za navodnjavanje, na osnovu vrednosti parametara njive dobijenih sa senzora. Kreirani recept za navodnjavanje će krajnjem korisniku dati informaciju o tome da li je i kada potrebno zaliti njivu, i, ako jeste, sa kojom količinom vode. Pri kreiranju recepta uzimaju se u obzir podaci o trenutnoj vlažnosti zemljišta, stadijumu razvoja u kojem se nalazi biljka, podaci o tipu zemljišta na kom je posejan usev i vremenska prognoza.

Podaci sa senzora korišćeni u ovom radu nastali su kao rezultat testiranja postojećeg rešenja projekta na više različitih slučajeva korišćenja. Testiranje je ukazalo na nedostatak postojećeg rešenja. Uočeni nedostatak postojećeg sistema jeste to što on nije u mogućnosti dovoljno rano da predvidi da li je i kada potrebno zaliti usev, pri njegovom korišćenju u okviru velikih sistema za navodnjavanje. Problem jeste to što postojeći sistem korisnika obaveštava da je potrebno zaliti usev tek onda kada količina vlage u zemljištu dostigne kritičnu vrednost, pri kojoj postoji opasnost od uvenuća biljke. Ovakvo rešenje nije odgovarajuće kod sistema za navodnjavanje kod kojih je potrebno nekoliko dana, a nekada čak i do nedelju dana, kako bi se zalila cela njiva. Kod takvih sistema za navodnjavanje, proces zalivanja potrebno je pokrenuti dovoljno rano, kako se ne bi dogodilo da u toku trajanja tog procesa na još uvek nezalivenom delu njive količina vlage u zemljištu padne ispod minimuma potrebnog za normalan rast i razvoj biljke.

Kako bi ovaj nedostatak bio prevaziđen, potrebno je uspostaviti sistem za predikciju budućeg kretanja količine vlage u zemljištu, koji bi na osnovu postojećih istorijskih podataka bio u stanju da predvidi kada će vlažnost zemljišta opasti na graničnu vrednost. Na osnovu takvog previđanja, proces zalivanja bilo bi moguće pokrenuti dovoljno rano, čime bi u svakom trenutku bila osigurana potrebna količina vlage na svakom delu njive.

Za uspostavljanje ovakvog sistema, potrebno je najpre osmisliti i implementirati arhitekturu sistema koja bi podržala skladištenje i analizu velikih masiva podataka generisanih od strane senzora postavljenih na njivama. Potom je potrebno odabrati odgovarajuću predikcionu tehniku, čijom bi se primenom nad istorijskim podacima dobio model koji bi s velikom preciznošću mogao da predvidi buduće vrednosti količine vlage u zemljištu.

Prvi cilj ovog rada je obezbeđenje takve arhitekture sistema za skladištenje i analizu velikih masiva podataka kojom će biti obezbeđeno skladištenje podataka u obliku koji je podesan za vršenje predikcije budućih vrednosti količine vlage u zemljištu i za sprovođenje kompleksnih multi-dimenzionih analiza. Kako bi ovo bilo moguće ostvariti,

najpre je potrebno izvršiti analizu postojeće arhitekture za skladištenje podataka projekta Irignet, kao i analizu postojećeg skupa podataka, nastalog u okvirima slučajeva korišćenja na kojima je ovaj projekat testiran. Potom je potrebno isprojektovati zvezdastu šemu baze podataka, kao dela sistema skladišta podataka (*DW*) za podršku procesu predikcije količine vlage u zemljištu. Krajnji korak u obezbeđenju arhitekture jeste projektovanje *ECTL* (eng. *ECTL* – *Extraction, Cleaning and Transformation, and Loading*) procesa za preuzimanje podataka iz postojeće *OLTP* baze podataka i njihovo smeštanje u bazu podataka ciljnog sistema skladišta podataka.

Drugi cilj ovog rada jeste pronalaženje odgovarajuće predikcione tehnike za predikciju količine vlage u zemljištu. U tom cilju, potrebno je, na osnovu izvršene analize postojećeg skupa podataka, odabrati više različitih prediktivnih tehnika, čija bi svojstva odgovarala primeni na ovakvom skupu podataka. Odabrane tehnike je potom potrebno primeniti na postojeće istorijske podatke, i na taj način kreirati prediktivne modele. Dobijene prediktivne modele potrebno je analizirati s ciljem odabira najpogodnije prediktivne tehnike za probleme ovakvog tipa. Nakon odabira prediktivne tehnike, potrebno je izvršiti tumačenje rezultata dobijenih predikcijom, kako bi se došlo do zaključaka koji će obezbediti znanje o njenim mogućnostima i ograničenjima.

Autori rada [2], kroz svoje istraživanje, vrše predikciju kretanja vrednosti količine vlage u zemljištu na osnovu prognoziranih vrednosti nezavisnih promenljivih. Za predikciju su korišćene *Support Vector Machines (SVM)* i *Relevance Vector Machines (RVM)* metode mašinskog učenja. Nažalost, ovakvu predikciju u slučaju projekta Irignet nije moguće vršiti zbog toga što su dostupni samo istorijski podaci o količini vlage u zemljištu i o temperaturi vazduha. Usled toga, u ovom radu će biti vršena predikcija koja se zasniva na analizi predhodnih vrednosti zavisne promenljive.

Pored Uvoda i Zaključka, rad sadrži još pet poglavlja. U poglavlju 2, dat je prikaz analize postojeće arhitekture za skladištenje podataka projekta Irignet, kao i analize postojećeg skupa podataka ovog projekta. U okviru poglavlja 3, dat je opis sistema koji je projektovan tako da se osigura ostvarenje ciljeva i zadataka ovog rada. Zatim je u poglavlju 4 prikazana šema baze podataka projektovana prema principima dimenzionog modelovanja sistema za skladištenje podataka. U poglavlju 5, dat je prikaz *ECTL* procesa koji predstavlja softversku podršku za izdvajanje podataka iz različitih izvora, validaciju, prečišćavanje, integraciju i vremensko označavanje tih podataka i punjenje *DW* baze podataka. Potom je u poglavlju 6 dat detaljan opis principa na osnovu kojih je vršen odabir tehnika za predikciju kretanja vrednosti količine vlage u zemljištu, nakon čega je izvršeno poređenje rezultata dobijenih primenom odabranih tehnika.

2. ANALIZA POSTOJEĆE ARHITEKTURE ZA SKLADIŠTENJE PODATAKA

U ovom poglavlju prikazana je analiza postojeće arhitekture za skladištenje podataka projekta Irignet, kao i analiza postojećeg skupa podataka ovog projekta.

Konceptualna šema postojeće *OLTP* baze podataka projekta, izražena *ER* modelom podataka, sastoji se od sledećih tipova entiteta: Klijent (*Client*), Grad (*City*), Vremenska zona (*Time zone*), Tip useva (*Crop Type*), Stanica (*Station*),

Njiva (*Field*), Tip uređaja (*Device Type*), Uređaj (*Device*), Podaci (*Data*) i Recept za navodnjavanje (*Irrig Recipe*).

Analizom postojeće šeme uočene su određene nelogičnosti pri povezivanju tipova entita. Uzrok ovih nelogičnosti jeste to što je projektant *OLTP* šeme baze podataka pri projektovanju imao u vidu potrebu za optimizovanjem pristupa velikoj količini podataka koju generišu senzori. Kreiranjem *OLAP* sistema za skladištenje podataka, prilagođenog za obradu velike količine podataka, nestala bi potreba za ovakvim prinudnim rešenjima. Ovo je posebno značajno kada se uzme u obzir da je, radi bolje predikcije, postojeće podatke potrebno dopuniti podacima o padavinama, navodnjavanju i temperaturi, čime će se količina podataka u bazi podataka višestruko uvećati.

Podaci sačuvani u *OLTP* bazi podataka nastali su pri primeni sistema na šest različitih slučajeva korišćenja. Analizom sačuvanih podataka utvrđeno je da su senzori, usled izloženosti spoljnim faktorima, u određenim trenucima očitavali vrednosti količine vlage u zemljištu koje značajno odstupaju od predviđenih. Ovaj nedostatak podataka biće nadomešćen pri izvršavanju *ECTL* procesa.

3. ARHITEKTURA SISTEMA

Kako bi bilo osigurano ostvarenje ciljeva i zadataka opisanih u uvodnom poglavlju, osmišljen je sistem čija se arhitektura sastoji od sledećih komponenti:

- *OLTP* baze podataka – komponenta koja predstavlja bazu podataka implementiranu korišćenjem *Microsoft SQL Server* sistema za upravljanje bazama podataka. Služi za skladištenje transakcionih podataka projekta Irignet;
- Spoljnih izvora podataka – ovu komponentu sistema čini datoteka koja sadrži podatke o vrednostima temperature vazduha za područja Novog Sada i Sombora za 2016.god. Ove dve lokacije su odabrane zbog njihovog geografskog položaja u odnosu na lokacije testnih slučajeva korišćenja. Podaci o temperaturi dobijeni su metodom linearne interpolacije primenjenom nad podacima o minimalnim i maksimalnim dnevnim vrednostima temperature vazduha za pomenuta područja, preuzetim sa Internet prezentacije Republičkog hidrometeorološkog zavoda Republike Srbije (RHMZ). Predviđeno je da se ova komponenta sistema u budućnosti proširi izvorima podataka o padavinama i navodnjavanju;
- *ECTL* komponente – predstavlja komponentu sistema realizovanu korišćenjem softverskog alata *Microsoft SQL Server Integration Services*, zaduženu za realizaciju *ECTL* procesa za preuzimanje podataka iz *OLTP* baze podataka i spoljnih izvora podataka, njihovu transformaciju i smeštanje u bazu podataka ciljnog sistema skladišta podataka. Isprojektovani *ECTL* proces detaljno je opisan u poglavlju 5, *ECTL* proces;
- Šeme skladišta podataka – komponenta sistema koja će biti detaljno opisana u narednom poglavlju ovog rada. Na osnovu nje je kreirana *DW* baza podataka;
- *DW* baze podataka – ova komponenta sistema predstavlja bazu podataka implementiranu u okviru *Microsoft SQL Server* sistema za upravljanje bazama podataka. Osmišljena je kako bi se unutar nje čuvala velika količina podataka, u obliku koji je podesan za obradu velike količine podataka, i kako bi pružila podršku za analizu ovih podataka od strane komponente za analizu podataka i predikciju; i

- Komponente za analizu podataka i predikciju – komponenta čija je svrha vršenje predikcije budućeg kretanja vrednosti količine vlage u zemljištu. Pored toga, ona je zadužena i za pružanje odgovora na analitička pitanja o uticaju temperature vazduha, tipa useva i dubina na kojoj se nalazi korenov sistem biljke na promenu količine vlage u zemljištu. U ovom radu biće opisan isključivo deo komponente koji se tiče predikcije.

Predstavljena arhitektura sistema će biti u mogućnosti da odgovori postavljenim ciljevima ovog rada zbog toga što se sastoji od skupa prilično nezavisnih komponenti. Ovakva arhitektura omogućava primenu inkrementalnog modela životnog ciklusa razvoja softvera. Primenom inkrementalnog modela životnog ciklusa razvoja se omogućava fazni razvoj sistema, deo po deo, uz postupnu integraciju komponenti. Na ovaj način je obezbeđen nezavisan razvoj i testiranje svake komponente arhitekture sistema.

4. ZVEZDASTA ŠEMA BAZE PODATAKA

Zvezdasta šema baze podataka projektovana je prema principima dimenzionog modelovanja sistema za skladištenje podataka, opisanim u [3]. Zvezdasta šema, dobijena kao rezultat projektovanja, sadrži sledeće dimenzije:

- Njiva (Dim_Field) – dimenzija koja predstavlja kompletno denormalizovanu hijerarhijsku strukturu u kojoj su svi hijerarhijski nivoi predstavljeni jednom šemom relacije. U njoj su sadržani podaci o samoj njivi, zatim o klijentima, stanicama, vremenskim zonama i receptima.
- Tip useva (Dim_CropType) – klasifikaciona (*Bracketing*) dimenzija koja klasifikuje činjenične podatke po tome za koji tip useva su mereni. Izdvojena je kao zasebna dimenzija kako bi se podržalo analiziranje uticaja tipa useva na kretanje vrednosti količine vlage u zemljištu.
- Vrsta zemljišta (Dim_SoilType) – takođe predstavlja klasifikacionu dimenziju, čija je namena podrška analizi uticaja vrste zemljišta na kretanje vrednosti količine vlage u zemljištu.
- Dubina (Dim_MoistureDepth) – još jedna klasifikaciona dimenzija, koja klasifikuje činjenične podatke po tome na kojoj dubini su izmerene vrednosti količine vlage u zemljištu. Namena ove dimenzije jeste podrška analizi uticaja dubine na kojoj se nalazi korenov sistem biljke na kretanje količine vlage u zemljištu.
- Vremenska dimenzija (Dim_DateTime) – dimenzija koja daje vremenski kontekst činjeničnim podacima, odnosno koja se koristi za vremensko označavanje podataka činjenične tabele.

S obzirom na to da se radi o zvezdastoj šemi baze podataka, šema sadrži jednu činjeničnu tabelu – Merenja (Fact_Measurements), pri čemu je u pitanju činjenična tabela sa uključenim merama (Regular Fact Table), kod koje postoje atributi koji reprezentuju mere, odnosno vrednosti činjenica. U okviru ove tabele postoje sledeće mere činjenica: vlažnost zemljišta (Moisture), temperatura (Temperature), količina navodnjavanja (Irrigation) i količina padavina (Precipitation).

Ključevi svih šema relacija projektovani su kao veštački ključevi (surrogate keys).

Uvažavajući vremenski aspekt granularnosti sistema skladišta podataka, za nivo agregacije činjeničnih podataka

odabran je vremenski period od jednog sata. Razlog za odabir ovog nivoa agregacije je taj što je bilo potrebno obezbediti dovoljnu količinu podataka kako bi bila moguća primena svih odabranih prediktivnih tehnika. Podizanjem nivoa agregacije na vremenski period od jednog dana, broj podataka u činjeničnoj tabeli bi se značajno smanjio.

5. ECTL PROCES

ECTL proces, prikazan u ovom poglavlju, implementiran je u okviru *ECTL* komponente sistema. Kreirani proces predstavlja softversku podršku za zadatke: 1) ekstrakcije (*Extraction*) – izdvajanja podataka iz različitih izvora, 2) transformacije (*Transformation*) – validacije, prečišćavanja, integracije i vremenskog označavanja podataka i 3) punjenja (*Loading*) *DW* baze podataka. Realizovan je upotrebom *SQL Server Integration Services (SSIS)* softverskog rešenja za podršku pomenutim zadacima. Komponente toka kontrole ovog procesa realizovane su, u zavisnosti od svoje namene, korišćenjem toka podataka, *SQL* skripte ili korišćenjem *R* skripte.

Tok kontrole *ECTL* procesa započinje paralelnim izvršavanjem komponenti za punjenje dimenzija *DW* baze podataka. Nakon što se završi punjenje dimenzija Njiva, Tip useva, Vrsta zemljišta i Dubina, kreira se agregirani pogled na podatke iz tabele Podaci *OLTP* baze podataka, pri čemu se ovim pogledom podaci agregiraju na vremenski period od jednog sata. Pri agregiranju podataka u obzir se uzimaju samo podaci kod kojih je izmerena količina vlage u zemljištu veća od 15% vol.%, zbog toga što sve vrednosti ispod ove donje granice predstavljaju grešku pri merenju. Punjenje Vremenske dimenzije se nastavlja zasebno, u paralelnom procesu, zbog toga što je za punjenje ove tabele potrebno značajno više vremena nego za punjenje ostalih dimenzija. Nakon kreiranja agregiranog pogleda, agregiranim podacima sa senzora dodeljuju se odgovarajuće vrednosti stranih ključeva za sve dimenzije sem vremenske, kako bi, potom, dobijene torke bile upisane u privremenu tabelu.

Naredni korak koji se izvršava u okviru toka kontrole ključan je za podršku vršenju analize i predikcije nad postojećim podacima. Svrha ovog koraka jeste kreiranje kompletnih vremenskih serija, jer većina metoda za analizu vremenskih serija polazi od pretpostavke da su podaci razmaknuti u fiksnim vremenskim intervalima. Cilj agregiranja podataka na nivo od sat vremena predstavlja pokušaj da se ovo postigne. Međutim, dešava se da senzori šalju netačne podatke, ili da čak uopšte ne šalju podatke u određenom vremenskom periodu. Netačni podaci biće uklonjeni prethodnom komponentom kontrole toka. Kako bi se odredile vrednosti nedostajućih podataka, u okviru ove komponente će biti sprovedena interpolacija. Do ovakvih događaja dolazi najčešće usled kvara na senzoru ili na mrežnoj infrastrukturi. Kao posledica ovakvih događaja, može se desiti da vremenske serije ne budu kompletne.

U cilju obezbeđenja kompletnosti vremenskih serija, izvršena je procena vrednosti količine vlage u zemljištu za tačke u kojima ta vrednost nije bila poznata. Procena je izvršena korišćenjem *Kalman Smoothing Estimator* metode [4].

U okviru narednog koraka toka kontrole *ECTL* procesa, izračunavaju se vrednosti temperature vazduha za trenutke u kojima su senzori slali podatke o količini vlage u zemljištu. Ovo se postiže korišćenjem aproksimacionih

funkcija koje vrše linearnu interpolaciju. Aproksimacione funkcije su dobijene pomoću podataka preuzetih sa Internet prezentacije Republičkog hidrometeorološkog zavoda Republike Srbije. Pri izračunavanju temperature vazduha, za testne slučajeve realizovane na području Futoga i Feketića koriste se podaci o temperaturi vazduha za Novi Sad, dok se za testne slučajeve postavljene u Srpskom Miletiću koriste podaci o temperaturi vazduha za Sombor. Ovakva raspodela je izvršena na osnovu geografskog položaja testnih slučajeva korišćenja u odnosu na lokacije mernih stanica RHMZ-a.

Izvršavanjem poslednjeg koraka toka kontrole vrši se opretnije podataka vremenskom dimenzijom i učitavanje podataka u činjeničnu tabelu *DW* baze podataka.

6. PREDIKCIJA KOLIČINE VLAGE U ZEMLJIŠTU

Usled nedostatka odgovarajućih nezavisnih promenljivih, na osnovu kojih bi se mogla vršiti precizna predikcija budućeg kretanja zavisne promenljive, odlučeno je da se za vršenje predikcije koriste tehnike koje vrše prognoziranje budućeg kretanja vremenske serije. Analizom osobina komponenti vremenske serije, koja sadrži podatke o kretanju količine vlage u zemljištu, došlo se do zaključka da je teško obezbediti dovoljno preciznu predikciju na nivou cele vremenske serije. Zbog toga je odabrano rešenje koje podrazumeva da se pri kreiranju i testiranju modela ne koriste sve vrednosti vremenske serije, već da se koriste samo opadajući intervali, nastali nakon nekog od naglih skokova vrednosti količine vlage u zemljištu. Posledica ovakvog rešenja jeste da sistem nije u mogućnosti da predvidi skokove u vrednostima količine vlage u zemljištu. Ovakav sistem za predikciju je prihvatljiv za zainteresovane strane, jer se vrši predikcija za najgori slučaj korišćenja – kada nema padavina.

U odnosu na odabrano rešenje, primenjene su tri prediktivne tehnike koje bi mogle dati zadovoljavajuće rezultate pri prognozi: *Drift* metoda, *ARIMA* modeli i autoregresija korišćenjem neuronskih mreža. Analizom modela dobijenih primenom ovih tehnika došlo se do zaključka da *ARIMA* model [5] daje najprecizniju prognozu, sa najmanjim vrednostima mera grešaka. Pored toga što su dobijene greške *ARIMA* modela najmanje, njegovim korišćenjem se dobijaju i predikcioni intervali, koji omogućavaju da se linijsko prognozirovanje proširi rasponom vrednosti koje bi zavisna promenljiva mogla zauzeti sa određenom verovatnoćom. Na osnovu svega navedenog, odlučeno je da se za korišćenje u realnim uslovima koriste *ARIMA* modeli.

7. ZAKLJUČAK

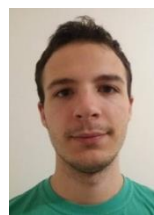
U ovom radu prikazan je sistem za predikciju budućeg kretanja količine vlage u zemljištu. Za uspostavljanje ovakvog sistema, potrebno je bilo osmisliti i implementirati arhitekturu sistema koja bi podržala skladištenje i analizu velikih masiva podataka generisanih od strane senzora postavljenih na njivama. Potom je bilo potrebno odabrati odgovarajuću predikcionu tehniku, čijom bi se primenom nad istorijskim podacima dobio predikcioni model koji bi s velikom preciznošću mogao da predvidi buduće vrednosti količine vlage u zemljištu.

Dalji razvoj softverske podrške može se kretati u pravcu unapređenja komponente sistema za analizu podataka i predikciju. Iako trenutno rešenje daje zadovoljavajuće rezultate sa stanovišta prevencije uvenuća biljaka, njegovim korišćenjem nije moguće optimizovati navodnjavanje, zbog toga što na osnovu njega nije moguće predvideti koliki će skok u vrednosti količine vlage u zemljištu izazvati prognozirane padavine. Kao posledica ove činjenice, u određenim situacijama može doći do preteranog ili nedovoljnog navodnjavanja useva. Prikupljanjem podataka o padavinama i navodnjavanju bilo bi omogućeno korišćenje prediktivnih tehnika koje su u mogućnosti da obezbede preciznu predikciju na nivou cele vremenske serije. Ovakve prediktivne tehnike bi davale još bolje rezultate ukoliko bi se prikupljali i podaci o vlažnosti vazduha, vazdušnom pritisku, solarnoj radijaciji i brzini vetra, a prikupljanjem ovih parametara njive bi bilo moguće i vršenje znatno opsežnijih analiza podataka.

8. LITERATURA

- [1] "IrrigNet - rational irrigation using soil sensors" <http://agronet.solutions/irrignet> Web. 9. Sep. 2017.
- [2] Hong, Zhihao, Z. Kalbarczyk, and Ravishankar K. Iyer. "A Data-Driven Approach to Soil Moisture Collection and Prediction." *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE, 2016.
- [3] Kimball, Ralph, and Margy Ross. *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons, 2011.
- [4] Deluyker, H., et al. "Replacement of missing values in time series data by Kalman smoothed estimators using state-space modeling." *Acta Vet. Scand.(Suppl 84* (1988).
- [5] Hyndman, Rob J., and George A. "8.5 Non-seasonal ARIMA models" *Forecasting: principles and practice*. OTexts, 2014.

Kratka biografija:



Nikola Todorović rođen je u Subotici 1993. god. Fakultet tehničkih nauka upisao je 2012. god. Bečelorski rad iz oblasti Elektrotehnike i računarstva – Računarske nauke i informatika odbranio je 2016. god. Master rad iz iste oblasti odbranio je 2017. god.

**ALGORITMI ZA ANALIZU VREMENSKIH SERIJA NA PRIMERU ISPITIVANJA
DINAMIKE VLAŽNOSTI ZEMLJIŠTA U POLJOPRIVREDI****ALGORITHMS FOR TIME SERIES ANALYSIS IN THE DOMAIN OF INVESTIGATION
OF SOIL MOISTURE DYNAMIC IN AGRICULTURE**

Vladimir Ivković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu, predstavljen je skup podataka o vlažnosti zemljišta sa tri poljoprivredne lokacije u Bačkoj. Dat je pregled metoda i tehnika analize vremenskih serija, a neke od njih su primenjene nad sakupljenim podacima sa ciljem utvrđivanja dinamike vlažnosti tla, kao i otkrivanja režima i šablona ponašanja u skupu podataka. Takođe, predložena su moguća tumačenja dobijenih rezultata, kao i potencijalna poboljšanja postupka prikupljanja i analize podataka.

Abstract – In this work, a soil moisture data set collected from three agricultural locations in Backa is presented. A brief overview of time series analysis methods is presented. Some of them are applied to data set in order to determine soil moisture behavioural regimes and patterns. Finally, an interpretation of analysis results and potential improvements in data gathering and analytics are presented.

Cljučne reči: Vremenske serije; Analiza podataka; Otkrivanje šablona; Precizna poljoprivreda.

1. UVOD

Intenzivno prisustvo informacionih tehnologija u svim područjima ljudskog delovanja dovodi do nastanka enormnih količina podataka u elektronskom obliku. Takvi podaci nemaju direktno korisnu vrednost, ali se kroz niz postupaka, koji obuhvataju obradu, istraživanje i analizu velikih količina podataka, može doći do različitih saznanja bitnih za unapređenje delatnosti. Iako je primena savremenih informacionih tehnologija izuzetno primetna u većini privrednih grana, poljoprivreda je, kao najstarija privredna grana, ostala nedovoljno istražena i unapređena u tom pogledu. Sa ciljem rešavanja ovog problema definisan je pojam *Precizne poljoprivrede* (PP) koji obuhvata integrisani poljoprivredni sistem, informaciono zasnovan i projektovan tako da uveća efikasnost, produktivnost i profitabilnost celokupnog poljoprivrednog gazdinstva [1]. Karakterističan oblik PP je specifično upravljanje usevima (*Site-Specific Crop Management*) koje predstavlja koncept nove generacije u ratarstvu, razvijen na osnovu posmatranja, merenja i reagovanja na spoljašnje varijacije uslova u kojima se usevi uzgajaju. PP uključuje upotrebu različitih tipova uređaja, kao što su senzori, dronovi i sateliti za sakupljanje svih vrsta podataka o usevima i njihovom okruženju. Jedan od najvećih problema u ratarskoj proizvodnji predstavlja adekvatno snabdevanje usevima vodom. Potrebe useva za vodom se u današnjoj praksi određuju

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Ivan Luković, red. prof.

pomoću različitih metoda, a dobijene vrednosti se neretko značajno razlikuju i odstupaju od stvarnih vrednosti potrebe za vodom. S obzirom na to da specifičnosti uslova utiču na potrebe useva za vodom, do stvarnih vrednosti se može doći analizom velikih količina podataka sakupljenih sa senzora instaliranih na poljoprivrednim parcelama.

Autori rada [2], kroz svoje istraživanje, analiziraju dinamiku varijacije vlažnosti zemljišta pod usevima kukuruza u toku jedne vegetacione sezone u Belgiji. Uočena je zavisnost dinamike vlažnosti tla od dubine, odnosno, zaključeno je da je površinski sloj najaktivniji u pogledu varijacije količine vlage, te da se dinamika dubljih slojeva intenzivira sa porastom korena kukuruza.

Cilj ovog rada je doprinos unapređenju efikasnosti sistema za navodnjavanje u pogledu utvrđivanja koji od posmatranih faktora i u kolikoj meri utiču na količinu vlage u zemljištu, i određivanja dinamike potrošnje vlage određene ratarske kulture na određenom poljoprivrednom lokalitetu. Među posmatrane faktore spadaju količina padavina i navodnjavanja, maksimalne i minimalne dnevne temperature, uzrast biljne kulture tokom vegetacionog perioda i dubina merenja količine vlažnosti u zemljištu. Da bi se dostigao definisani cilj neophodno je izvršiti istražnu analizu vremenskih serija sačinjenih od podataka dobijenih sa senzora količine vlažnosti u tlu, utvrditi uticaj pomenutih faktora na dinamiku vlažnosti tla, i pronaći režime i specifične šablone ponašanja u vremenskim serijama prikupljenih podataka.

Osim uvoda i zaključka ovaj rad ima pet poglavlja. U poglavlju 2, dat je sažet pregled opštih karakteristika zemljišta i principa navodnjavanja. U poglavlju 3, predstavljen je skup podataka o vlažnosti tla i rezultati istražne analize nad njim. Zatim su u poglavlju 4 prikazani rezultati analize uticaja faktora na vlažnost zemljišta. Takođe, prikazana je primena tehnika za otkrivanje režima i šablona ponašanja u vremenskim serijama podataka.

**2. OPŠTE KARAKTERISTIKE ZEMLJIŠTA I
PRINCIPI NAVODNJAVANJA**

Zemljište je površinski rastresiti sloj Zemljine kore, koji sadrži hranljive materije neophodne za rast i razvoj biljaka. Od svih tipova zemljišta, na području Srbije, najplodnija je crnica. Ritska crnica je vrsta crnice nastala isušivanjem močvara i predstavlja tip zemljišta na kom su bile zasejane ratarske kulture posmatrane u ovom radu. Od izuzetne važnosti za rast i razvoj biljnih kultura je sadržaj vode u tlu. Hemijsko-fizičke karakteristike, među kojima su sadržaj gline i peska, određuju način i vreme zadržavanja vode u tlu. Voda se u tlu nalazi u različitim oblicima od kojih

najveći značaj za biljke ima kapilarna voda koja ujedno predstavlja osnovni faktor dinamike i plodnosti tla. Vodne konstante karakterišu sposobnost tla da zadržava, pokreće i premešta određenu količinu vode. Najznačajnije vodne konstante tla su poljski vodni kapacitet (PVK) i vlažnost venjenja (VV). PVK predstavlja sadržaj vode u tlu nakon maksimalnog zasićenja i proceđivanja slobodne vode. VV označava raspon količine vode od nivoa pri kome biljka pokazuje znake venjenja do potpunog uvenuća. Vodni režim tla obuhvata tri karakteristične faze: mokra faza (1) – potpuna zasićenost vodom, vlažna faza (2) – sadržaj vode od PVK do VV i suva faza (3) – sadržaj vode ispod VV.

Navodnjavanje predstavlja hidrotehničku meru za poboljšanje fizičkih osobina zemljišta dodavanjem vode kako bi se postigli visoki i stabilni prinosi useva nezavisno od vremenskih uslova. Norma navodnjavanja se definiše kao ukupna količina vode koju treba dodati usevima tokom vegetacionog perioda i predstavlja razliku ukupnih potreba za vodom biljke i atmosferskih padavina. Zalivna norma se definiše kao količina vode koja se dodaje prilikom jednog zalivanja. Postoje različite metode za proračun količine vode za navodnjavanje.

Potreba useva za vodom može se definisati kao količina vode koja je potrebna da usev nadoknadi gubitke vode do kojih dolazi evapotranspiracijom useva koji postiže maksimalni razvoj zelene mase. U ovom radu posmatrane su ratarske kulture: kukuruz, soja i šećerna repa. Kod većine biljnih vrsta potrebe za vodom variraju u zavisnosti od faze razvića biljke. Kod kukuruza i soje, kao i kod ostalih jednogodišnjih kultura, najveće potrebe za vodom su u periodu cvetanja i formiranja zrna. Odsustvo vode u ovim fazama ima daleko veće posledice po prinose ovih kultura nego nedostatak vode u drugim fazama razvića. Soja najveće potrebe za vodom zahteva prilikom intenzivnog rasta i formiranja korena i lisne mase u periodu od početka jula do sredina avgusta. Za područje Vojvodine, potrebe kukuruza za vodom variraju u opsegu 450-500 mm, soje u opsegu 440-510 mm, a kod šećerne repe u opsegu 450-550 mm.

3. SKUP PODATAKA I ISTRAŽNA ANALIZA

Skup podataka koji je korišćen u ovom radu nastao je beleženjem očitavanja senzora količine vlažnosti zemljišta na tri lokacije: u Feketiću, Srpskom Miletiću i Planti kod Futoga. Podaci su prikupljeni tokom jula, avgusta i septembra 2016. godine. Ovim podacima su pridodati podaci o količini padavina i navodnjavanja na dnevnom nivou, kao i o maksimalnim i minimalnim temperaturama vazduha u posmatranom periodu. Kalmanov filter je korišćen u slučaju nedostajućih i pogrešnih vrednosti.

Vremenska serija se može neformalno definisati kao skup podataka ili vektora podataka sa vremenskom odrednicom. Analiza vremenskih serija obuhvata četiri osnovna cilja:

- deskripcija – izučavanje osnovnih karakteristika kroz vizuelizaciju i sumarne statistike,
- objašnjenje – utvrđivanje međuzavisnosti varijacije u različitim vremenskim serijama,
- prognoziranje – identifikacija i ocena modela koji na osnovu prethodnih opservacija predviđa vrednosti budućih,
- kontrola – izgradnja modela za prilagođavanje ulazne serije tako da proces bude što bliži željenom cilju.

U ovom radu je pažnja usmerena na prva dva cilja – deskripciju i objašnjenje.

U sklopu istražne analize korišćeni su linijski i kutijasti dijagrami. Linijski dijagrami su pogodni za detektovanje skokova vrednosti količine vlage u zemljištu koji se dešavaju istovremeno na svim dubinama. S druge strane, kutijasti dijagrami su pogodni za detektovanje opadanja varijacija količine vlažnosti sa porastom dubine. Zbog nedostatka prostora ovi dijagrami se ne prikazuju, ali se mogu naći u master radu.

Autokorelacija predstavlja korelaciju vremenske serije u odnosu na samu sebe pomerenu u vremenu. Primenom autokorelacije nad posmatranim vremenskim serijama dobija se značajna zavisnost trenutne od prethodne tri opservacije.

Prvi izvod vremenske serije se definiše kao vremenska serija formirana od razlika susednih opservacija polazne vremenske serije. Diferenciranjem vremenske serije merenja na dubini od 10 cm primećuje se da je većina razlika negativna što upućuje na pad vlažnosti tokom perioda bez padavina. Takođe, uočava se da su pozitivne razlike u proseku veće od negativnih, što se objašnjava činjenicom da je porast količine vlage tokom dotoka vode jačeg intenziteta od pada kada dotoka vode nema.

Pod pojmom uravnanje vremenske serije podrazumeva se postupak promene vrednosti podataka u vremenskoj seriji sa ciljem otklanjanja slučajnih oscilacija kako bi se vremenska serija učinila pogodnijom za analizu i pronalaženje interesantnih oblika. Od brojnih metoda uravnanja vremenskih serija u ovom radu je odabrana ARIMA (*AutoRegressive Integrated Moving Average*).

Dekompozicija vremenske serije predstavlja postupak raščlanjivanja vremenske serije i identifikacije osnovnih obrazaca: trend (1) – dugoročni porast ili opadanje vrednosti opservacija, sezonalnost (2) – slično ponašanje u regularnim vremenskim intervalima i cikličnost (3) – slično ponašanje vremenske serije u neregularnim vremenskim intervalima. Primenom dekompozicije se vremenska serija najčešće predstavlja kao kombinacija sezonalne, trend-ciklične i komponente ostatka. Sezonalna komponenta identifikovana nad posmatranim vremenskim serijama ima period od 24h.

4. ANALIZA UTICAJA FAKTORA NA VLAŽNOST ZEMLJIŠTA

Prilikom upoređivanja vremenskih serija neophodno je definisati funkciju distance između dve vremenske serije. U literaturi su prisutne različite funkcije koje ispoljavaju sličnost vremenskih serija po strukturi [3]. Jedna od najpoznatijih funkcija distance je Euklidsko rastojanje koje se može primeniti jedino nad vremenskim serijama jednake dužine i razmatra sličnost između korespondentnih opservacija. Za razliku od Euklidskog rastojanja, DTW (*Dynamic Time Warping*) prepoznaje i sličnosti koje su pomerene u vremenu, a funkcioniše i nad vremenskim serijama različite dužine.

4.1. Klasterovanje po dubini, lokaciji i tipu kulture

Klasterovanje je jedna od osnovnih tehnika nenadgledanog učenja i predstavlja prepoznavanje grupa sličnih objekata u nekom skupu. Hijerarhijsko klasterovanje je oblik klasterovanja koje produkuje ugnježdene grupe sličnih objekata. U

ovom radu primenjeno je hijerarhijsko anglomerativno klasterovanje nad vremenskim serijama podataka na svim dubinama za posmatrane ratarske kulture u Feketiću i Srpskom Miletiću.

Kao mera sličnosti između vremenskih serija korišćeni su Euklidsko rastojanje i DTW. Kod oba klasteringa jedino se izdvajaju podstabla sa merenjima na 60cm dubine što ukazuje na to da je na većim dubinama uticaj korena biljke na dinamiku vlažnosti tla manji nego u gornjim slojevima zemljišta.

Euklidsko rastojanje, na dubinama od 10cm i 30cm favorizuju sličnost po lokaciji, pa se tako formiraju podstabla sa merenjima u Srpskom Miletiću i Feketiću bez obzira na kulturu. S druge strane, DTW daje veći značaj tipu ratarske kulture prilikom formiranja dendrograma hijerarhijskog klasterovanja.

Kod oba klasteringa jasno se uočava značajno odstupanje strukture vremenske serije merenja kod soje u Feketiću na 10 cm i 30 cm dubine.

4.2. Uticaj padavina i navodnjavanja

Posmatranjem linijskih dijagrama količine vlage u tlu zapažaju se reakcije zemljišta na pojavu kiše i navodnjavanje u vidu skokova vrednosti količine vlage na svim posmatranim dubinama. Intenzitet i oblik tih skokova se značajno razlikuje s obzirom na dubinu merenja. Analizom koeficijentata korelacije dnevnih promena u količini vlage u odnosu na količinu padavina ili navodnjavanja dolazi se do zaključka da najveću reakciju ispoljavaju gornji slojevi zemljišta, naročito u sušom periodu kada ti slojevi tla zadržavaju skoro svu količinu dodate vode.

Primenom linearne regresije nad dnevnim porastima količine vlage u odnosu na količinu dodate vode koeficijenti nagiba iznose od 0.05 do 0.08 zapreminskih procenata po litru padavina.

4.3. Uticaj temperature vazduha

Na proces evapotranspiracije, između ostalog, utiče temperatura vazduha. U ovom radu nije utvrđena direktna zavisnost dnevnih promena količine vlažnosti tla tokom sušnih dana u odnosu na maksimalne i minimalne dnevne temperature.

Međutim, sezonalna komponenta dobijena dekompozicijom vremenske serije ispoljava svoje dnevne minimume i maksimume u približno istim delovima dana kao i temperatura vazduha. Konkretnije, dnevni maksimum količine vlage u tlu se pojavljuje od 12.30 do 13.30 časova, a minimum u intervalu od 03.30 do 05.00 časova.

4.4. Uticaj faze u razviću kulture

Kao što je prethodno pomenuto, potrebe biljaka za vodom variraju u zavisnosti od faze razvića. Analizom dnevnih promena u količini vlage, tokom dana bez padavina i navodnjavanja, primećuje se pad intenziteta dnevnog gubitka vode iz posmatranih slojeva tla.

Takođe, sabiranjem dnevnih gubitaka vode za posmatrane mesece (jul, avgust i septembar) uočava se proporcionalan odnos sa očekivanim vrednostima evapotranspiracije za ove mesece, sa izuzetkom kukuruza u Srpskom Miletiću gde su vrednosti gubitaka vode u avgustu i septembru približno jednaki. S druge strane, apsolutne vrednosti gubitaka vode se razlikuju u zavisnosti od lokacije, pa se

takva pojava može objasniti kao posledica specifičnih hidroloških karakteristika poljoprivrednog lokaliteta.

5. OTKRIVANJE REŽIMA I ŠABLONA PONAŠANJA U VREMENSKIM SERIJAMA

Neretko vremenske serije prelaze iz stacionarnosti u nestacionarnost ili iz jednog trenda u drugi. Takva pojava se može primetiti i kod vremenskih serija posmatranih u ovom radu. Otkrivanje režima ponašanja, koje karakterišu stacionarnost ili specifičan trend, može doprineti boljem razumevanju vodnog režima zemljišta na određenoj lokaciji tokom vegetacionog perioda. U ovom radu su primenjene dve metode za otkrivanje režima ponašanja u vremenskim serijama: skriveni Markovljevi modeli i SPIRIT (*Streaming Pattern dIScoverY in multiple Time-series*) algoritam.

Skriveni Markovljevi modeli (*Hidden Markov Models*, HMM) pripadaju grupi Markovljevih modela koji predstavljaju stohastičke modele slučajne promenljive stanja. Za razliku od običnih Markovljevih modela, stanja HMM su skrivena, odnosno nemoguće ih je direktno uočiti iz opservacija. Umesto toga, opservacije variraju oko vrednosti skrivene promenljive stanja. Nakon izgradnje HMM nad posmatranim vremenskim serijama za važeći režim u određenom trenutku proglašava se skriveno stanje koje u tom trenutku ima najveću verovatnoću. Primenom HMM za tri skrivena stanja nad merenjima količine vlage za soju u Feketiću na 10 cm dubine dobijaju se režimi koji uglavnom odgovaraju pomenutim fazama u vodnom režimu tla: mokroj, vlažnoj i suvoj.

Dok se pomoću HMM otkrivaju režimi u jednoj vremenskoj seriji, SPIRIT (*Streaming Pattern dIScoverY in multiple Time-series*) algoritam predstavlja postupak pronalaženja korelacija i skrivenih promenljivih nad više poravnatih vremenskih serija. SPIRIT algoritam nastoji da sa što manjim brojem promenljivih opiše ponašanje multivariabilne vremenske serije, a broj promenljivih se dinamički određuje. U ovom radu, SPIRIT algoritam je primenjen nad familijama vremenskih serija iz Feketića i Srpskog Miletića. Režim se određuje kao broj prepoznatih promenljivih u određenom trenutku. U Feketiću se veći deo jula nalazi u režimu 4, dok se tokom avgusta i septembra pojavljuje režim 5. To upućuje na znatna odstupanja između oblika vremenskih serija tokom posmatranog perioda. S druge strane, u Srpskom Miletiću su tokom većeg dela jula i avgusta na snazi režimi 2 i 1, dok se u septembru pojavljuje režim 3 izazvan razlikama u reakcijama na padavinama kod kukuruza i soje.

S obzirom na to da vremenske serije često obuhvataju velike količine podataka to čini njihovu obradu i analizu zahtevnijom u pogledu potrebnog vremena. Stoga se pribegava diskretizaciji vremenske serije koja predstavlja postupak aproksimacije sa ciljem smanjenja obima podataka kao i što boljeg očuvanja strukture polazne vremenske serije. Jedna od metoda za diskretizaciju je SAX (*Symbolic Aggregate approXimation*) i predstavlja preslikavanje kontinualnih segmenata vremenske serije na alfabet zadate veličine. Diskretizacija vremenske serije doprinosi lakšem pronalaženju interesantnih šablona ponašanja.

Jedan od osnovnih izazova analize i istraživanja vremenskih serija je problem pronalaženja sličnih ponavljajućih podnizova u dugačkim vremenskim serijama. Poklapanje

između dva posmatrana podniza se može identifikovati ako je vrednost funkcije distance između njih manja od zadatog praga. Kroz algoritam vremenske projekcije određuju se potencijalni podnizovi – kandidati koji se podudaraju [4]. Kroz nekoliko iteracija se za slučajno odabrane parove znakova proverava da li sledi poklapanje. Na kraju se za podnizove sa najvećim brojem poparnih poklapanja računa funkcija distance, što predstavlja krajnji test sličnosti podnizova. Takvi ponavljajući podnizovi se nazivaju motivi. Primenom ovog algoritma dobijaju se motivi koji se mogu svrstati u četiri osnovne grupe: rastući i opadajući, konveksna i konkavna pomena trenda. Ovakvi motivi odgovaraju periodu sa padavinama i bez padavina, prestanku i početku padavina, redom.

Dok se klasična analiza vremenskih serija bavi određivanjem globalnih fenomena i karakteristika, metode i tehnika otkrivanja šablona ponašanja se fokusiraju na osobine lokalnog nivoa. Otkrivanje šablona ponašanja može se definisati kao traženje uzajamnih veza između pojava karakterističnih podnizova u vremenu. Asocijativno pravilo označava da jedan događaj sa određenom sigurnošću izaziva drugi u određenom vremenskom intervalu: ako se pojavi šablon A tada se pojavljuje šablon B najkasnije u vremenskom intervalu dužine T. Pronalaženje asocijativnih pravila nad vremenskom serijom nije primenjeno u ovom radu. Razlog tome je nepostojanje dovoljnog broja zapisa o padavinama, navodnjavanju i temperaturama vazduha tokom dela vegetacionog perioda posmatranih ratarskih kultura.

6. MOGUĆNOSTI PROŠIRENJA POSTOJEĆEG SISTEMA

Uvođenjem senzora za merenje meteoroloških parametara, za koje je dokazano da utiču na vrednost evapotranspiracije useva, omogućio bi se jasniji uvid u reakcije zemljišta na pojave u okolini. U te parametre spadaju vlažnost vazduha, brzina vetra i solarna radijacija. Raspoređivanjem senzora po celokupnoj površini parcele povećao bi se kvalitet podataka jer bi se otklonile potencijalne štetne varijacije među podacima izazvane kvarom senzora. Takođe, korišćenje lako prenosivih senzora osobina tla povećalo bi se znanje o sastavu zemljišta koje direktno utiče na njegov vodni režim.

Ukoliko bi neka od predloženih poboljšanja bila primenjena, naročito u pogledu proširivanja skupa merenih meteoroloških parametara i obezbeđivanja poravnanja vremenskog intervala na koji se vrše merenja, povećala bi se raznovrsnost i masovnost podataka. To bi otvorilo brojne mogućnosti korišćenja širokog spektra metoda i tehnika mašinskog učenja u domenu vremenskih serija koje nisu mogle biti adekvatno primenjene u ovom radu. S obzirom na relativno pristupačne cene senzora i očekivanu stabilizaciju i povećanje prinosa korišćenjem analiza sprovedenih nad podacima sa senzora, finansijska korist investiranja u specifično upravljanje usevima može biti uočljiva nakon 5-6 godina.

7. ZAKLJUČAK

U ovom radu sprovedena je analiza ponašanja nivoa vlage u zemljištu primenom algoritama za analizu vremenskih serija podataka. Utvrđeno je da pored padavina i navod-

njavanja jak uticaj na nivo vlage u tlu imaju i faze u razviću biljne kulture. Međutim, pored klimatskih i pedoloških faktora, veliku važnost ima i specifičnost poljoprivrednog lokaliteta, pa je režim navodnjavanja useva teško generalizovati.

Primenom istražne analize, analize trenda, dekompozicije i poređenja vremenskih serija dobijen je opšti uvid u vodni režim tla na posmatranim lokacijama. Primetne su razlike za različite kulture na istoj lokaciji, kao i razlike dinamike promene nivoa vlage u tlu za istu ratarsku kulturu na više lokaliteta. Specifičnosti poljoprivrednog lokaliteta ogledaju se u pojavi maksimalne količine vlage na različitim dubinama i različitoj osetljivosti na dotok vode za različite lokacije.

Osim ispitivanja uticaja različitih faktora na vodni režim tla, u ovom radu, ustanovljeni su i različiti režimi i šabloni ponašanja vremenske serije nivoa vlažnosti tla. Ustanovljeni režimi su uglavnom određeni količinom i rasporedom padavina i navodnjavanja. Pronađeni motivi se odnose na kraće intervale i predstavljaju karakteristične oblike u vremenskoj seriji koji se ponavljaju. Takvi šabloni su najčešće u vezi sa trenucima početka i prestanka padavina ili sa dnevnim oscilacijama temperature vazduha.

Korišćenjem dobijenih rezultata, ali i rezultata u uslovima poboljšanja navedenih o prethodnom poglavlju, moguće je unaprediti znanje o osobinama dinamike promene nivoa vlage u tlu za određene lokacije pod usevima i samim tim doprineti poboljšanju efikasnosti sistema za navodnjavanje.

8. LITERATURA

- [1] Zarco-Tejada, P. J., Hubbard, N., & Loudjani, P. (2014). Precision Agriculture: An Opportunity for EU Farmers—Potential Support with the CAP 2014-2020. *Joint Research Centre (JRC) of the European Commission*.
- [2] Hupet, F., & Vanclooster, M. (2002). Intraseasonal dynamics of soil moisture variability within a small agricultural maize cropped field. *Journal of Hydrology*, 261(1), 86-101.
- [3] Lin, J., Williamson, S., Borne, K., & DeBarr, D. (2012). Pattern recognition in time series. *Advances in Machine Learning and Data Mining for Astronomy*, 1, 617-645.
- [4] Chiu, B., Keogh, E., & Lonardi, S. (2003). Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 493-498). ACM.)

Kratka biografija:



Vladimir Ivković rođen je u Nevesinju 1993. god. Fakultet tehničkih nauka upisao je 2012. god. Bečelor rad iz oblasti Elektrotehnike i računarstva – Računarske nauke i informatika odbranio je 2016. god. Master rad iz iste oblasti odbranio je 2017. god.

WINDOWS PRESENTATION FOUNDATION CODE GENERATOR FOR SOFTWARE PRODUCT LINES**GENERATOR WINDOWS PRESENTATION FOUNDATION FORMI U OKVIRU SOFTVERSKIH PROIZVODNIH LINIJA**

Aleksandar Bošnjak, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

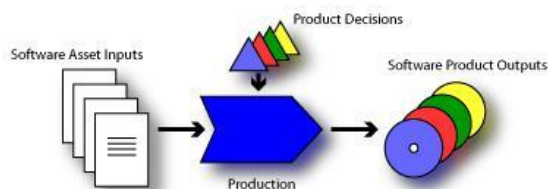
Kratak sadržaj – U radu je opisana okvirna struktura formiranja SPL (Softverske proizvodne linije) proizvoda poštujući SPL principe uvođenjem novih formi / polja u SPL proizvod putem WPF generatora formi. Opisan je proces izrade SPL proizvoda sa stanovišta izrade korišćenja jezika specifičnih za domen (JSD). Opisana je struktura meta-modela kao i korisnički interfejs koji prati opisan meta-model. Analizirana je struktura sistema, posmatrani su ulazi i izlazi kao i interna logika sačinjena od niza generatora gde je svaki analiziran zasebno.

Abstract – This paper presents structure describing SPL (Software product lines) product creation respecting form and field SPL generation principles relying on WPF forms. It also describes SPL product implementation process regarding DSL (Domain Specific Languages) technologies. Also it describes meta-model structure, as well as corresponding user interface (UI). System structure has been analyzed including inputs and outputs as well as intern logic composed of multiple separated generators.

Ključne reči: WPF, code generation, forms, SPL, DSL

1. UVOD**1.1 Softverske proizvodne linije**

Softverske proizvodne linije predstavljaju skup softverskih sistema koji kroz sinergiju formiraju nov proizvod (SPL Product) prilagođavanjem u cilju zadovoljavanja potrebe tržišta koristeći se resursima baznog proizvoda (slika 1.) [1].



Slika 1.1: Grafički prikaz procesa stvaranja SPL proizvoda

Na tržištu svaki klijent najčešće poseduje sopstvene mehanizme obavljanja poslovne logike u cilju stvaranja dobiti. Jednim sistemom je praktično nemoguće obuhvatiti sve poslovne logike čak i kada je u pitanju sistem specifične namene. Iz pomenutog razloga, obično je tendencija ka tome da se formira jedan bazni proizvod (SPL core asset) koji pokriva većinu poslovne logike za određeni domen. Zatim se vrši proces prilagođavanja gde svaki od klijenata iznosi svoje specifičnosti poslovanja, što spada pod produktne odluke (SPL Product Decisions).

1.2 Jezici specifični za domen

Jezici specifični za domen (DSL) su jezici prilagođeni i ograničeni na ciljani domen problema. Uspešni DSL-ovi se fokusiraju na uske i dobro definisane domene [2]. Svaki DSL se sastoji od: apstraktne sintakse (meta-modela), jedne ili više konkretnih sintaksi i semantike.

Opšta ideja DSL-a svodi se na:

1. inicijalno kreiranje konceptualnog modela rešenja
2. opisivanje rešenja problema putem jezika specifičnog za domen
3. generisanje/kompajliranje izvršnog koda rezultovanog opisanim modelom.

2. UVOD U WPF TEHNOLOGIJE I MVVM ŠABLON**2.1 Microsoft WPF tehnologije**

Microsoft Presentation Foundation (WPF) [3] predstavlja savremeni prezentacioni sistem u okviru .NET Framework-a [4] u cilju formiranja korisničkog interfejsa. WPF prezentacioni sistemi pružaju podršku za izradu kako desktop tako i web aplikacija. U daljem tekstu je analizirana upotreba WPF sistema u izradu desktop aplikacija.

Osnovna gradivna jedinica WPF aplikacija je WPF kontrola (WPF Control) [5], pod time se misli na ugrađene kontrole (slika 2.1). Na primeru nekih standardno korišćenih komponenti uočavaju se sledeće komponente:

1. padajući meni (Combobox)
2. tekstualno polje (Text input)
3. radio dugme (Radio button)
4. polje za potvrdu (Checkbox)
5. polje za duži unos (Textarea)
6. dugme (Button)
7. izbor datuma (Datepicker)

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

8. oznaka (Label)
9. traka za pomeranje (Scrollbar)

WPF kontrole predstavljaju komponente pomoću kojih se prikupljaju ili prikazuju podaci zadovoljavajući standarde korisničkog izgleda i korisničkog iskustva (UI/UX) [6]. WPF sistemi su uveli dodatna poboljšanja za proces razvoja desktop aplikacija tako što je uvedena podela prilikom implementiranja logike zadužene za:

- segmenta zaduženog za lokalizaciju resursa (RESX)
- vizuelnu implementaciju komponenti (XAML)
- logike koja opisuje ponašanje komponenti (Code-Behind)

Logika zadužena za lokalizaciju resursa (RESX) [7] je logički odvojena od prikaza i ponašanja UI aplikacije. Ovaj rad se isključivo bavi upotrebom statičkih resursa. Opšta ideja jeste da se u odvojene datoteke, koje je moguće referencirati iz ostatka sistema, skladište podaci o lokalizovanim vrednostima

Programska logika vizuelne implementacije komponenti (XAML) je implementirana pomoću posebnog razvijenog meta jezika isključivo za tu namenu - Extensible Application Markup Language (XAML)-a [8]. XAML je deklarativni jezik za označavanje (markup language) pomoću kog se opisuju komponente korisničkog interfejsa.

Logika koja opisuje ponašanje (Code-Behind) ima za cilj implementaciju funkcionalnih segmenata koji obrađuju interakciju sa korisnikom kroz obradu događaja.

2.2 Microsoft WPF MVVM šablon

Ideja MVVM šablona jeste uvođenje dodatnog sloja logike čija je primarna uloga razdvajanje nadležnosti.

ViewModel sadrži sve neophodne komponente modela čija vrednost se preuzima ili prikazuje na UI korisničke forme. Konvencija je da se za naziv svakog ViewModel-a dodaje postfix ViewModel. Budući da je vezivanje podataka između ViewModel-a i View-a obostrano (two way data binding).

3. PROJEKTOVANJE JEZIKA ZA OPIS FORMI

3.1 Rad sa kompozitnim formama

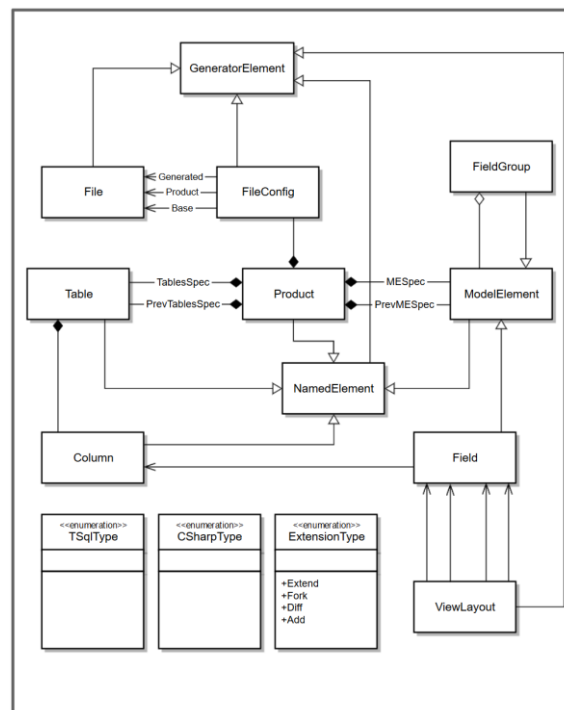
Kao što je već napomenuto u prethodnom poglavlju, bazni resursi su najčešće potpune aplikacije sa razvijenim modelom podataka koji se sastoji od složenih entiteta. To dalje implicira složenost korisničkih formi (forme koje poseduju veliki broj kontrola).

Predmet ovog rada je nadograđivanje tj. prilagođavanje zahtevima klijenata za novim poljima/grupama polja koja odgovaraju domenu njihovog poslovanja.

3.2. Metamodel za opis formi

Pomoću meta-modela jezika (slika 2) prezentovana je sintaksa koja opisuje proces prilagođavanja zahtevima klijenata SPL proizvoda u domenu SPL-a [14]. Pod prilagođavanjem se u ovom kontekstu misli na dodavanje

novih elemenata u postojeće forme/dokumente SPL baznog proizvoda.



Slika 2. Arhitektura meta-modela generatora koda

3.3 Sintaksa jezika generatora formi

U metodologijama jezika specifičnih za domen definisane su različite sintakse za opisivanje modela korišćenjem apstraktne sintakse – metamodela [9]. Neke od konkretnih sintaksi DSL-ova su:

- dialog based opis modela putem korisničkog interfejsa aplikacije
- opis modela putem grafičkog editora (pr. UML editori)
- opis modela putem tekstualnog editora i druge...

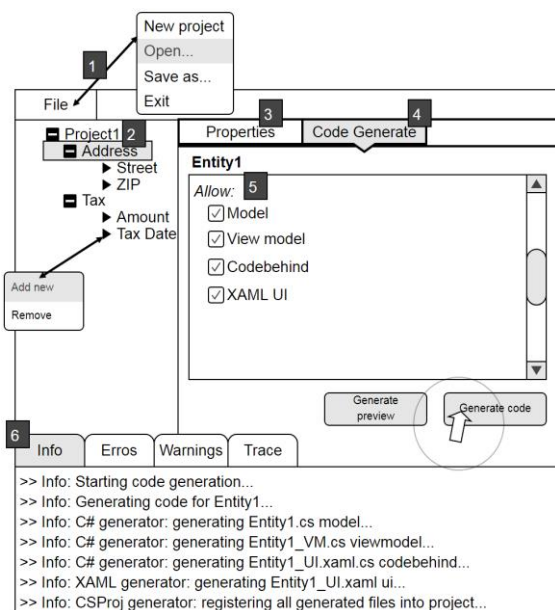
U radu je korišćena je *dialog based* sintaksa modela oslanjajući se na korisnički interfejs aplikacije za opisivanje konkretnog modela.

Korisnički interfejs (slika 3.) treba da obezbedi jednostavan, intuitivan i jednoznačan put za izbor gradivnih stavki modela u zavisnosti od domena problema. Usvojivši ova pravila, korisnički interfejs je podeljen na sledeće stavke:

1. Glavni meni
2. Stablo izbora
3. Sekcija svojstava označenog modela
4. Sekcija generisanja koda
5. Selekcioni paneli
6. Sekcija za pozicioniranje polja u formi
7. Paneli za ispis učinka i grešaka tokom generisanja.

Uloga glavnog menija jeste upravljanje projektima. Imajući u vidu da je podržan rad sa više različitih projekata, ova komponenta omogućava pristup formiranju novog projekta – funkcija „New project“, učitavanja novog

projekta – funkcija „Open“, serijalizaciju postojećeg – funkcija „Save as“ ili prekid rada – funkcija „Exit“.



Slika 3. Korisnički interfejs glavnog prozora - generisanje sadržaja

Označavanje određenog čvora u stablu ima značajnu ulogu u procesu generisanja. U zavisnosti od pozicije označenog čvora u stablu bira se sadržaj koji biva preuzet za dalje procesiranje.

Svi potomci označenog čvora uključivši i sam čvor ulaze u selekciju podataka za generisanje. Ukoliko je izabrani čvor list stabla, tada u selekciju podataka ulazi samo entitet kome on pripada. Na korisničkom interfejsu glavnog prozora (slika 3.3).

Sekcija svojstava označenog modela je korišćena istovetno i za dodavanje novog čvora i za izmenu postojećeg. Sadržaj polja u okviru sekcije se menja u zavisnosti od označenog čvora na stablu. U slučaju dodavanja novog čvora stabla pomoću kontekstnog menija (slika 3.3), sva polja su inicijalno prazna omogućivši unos novih modela u stablo.

Selekcioni paneli su sadržani u okviru sekcije za generisanje koda. Uloga komponente je da se omogući selektivno generisanje koda u zavisnosti od označenih stavki.

Sekcija generisanja koda sadrži funkcije za generisanje koda – „Generate code“ ili generisanje prikaza – „Generate preview“. U slučaju iniciranja funkcije generisanja prikaza otvara se novi modalni prozor sa izgenerisanim sadržajem.

4. SPECIFIKACIJA I STRUKTURA SISTEMA

4.1 Ulazi sistema

Ulaz u sistem generatora koda razdvajamo u dva toka:

1. Informacije o baznim produktim resursima
2. Projektne odluke / prilagođavanje zahtevima

Informacije o baznim proizvodnim resursima predstavljaju jedan od esencijalnih faktora u SPL-ma. Na osnovu ovih informacija o postojećim produktim resursima, moguće je vršiti prilagođavanje pomenutih resursa za tržište.

Odluke o prilagođavanju baznih resursa tržištu donosi menadžment matične organizacije.

Na osnovu prethodno donesenih odluka u skladu sa zahtevima tržišta, generator koda poseduje neophodne informacije kako da vrši prilagođavanje i formira novonastali projektni softverski proizvod u skladu sa zahtevima.

4.2 Izlazi sistema

Rezultat generisanja prestavlja bazni proizvod sa izmenama koje zadovoljavaju projektne odluke – SPL proizvod.

4.3 Opšta arhitektura sistema

U opštu arhitekturu sistema spadaju i *Parser-Template* generatori. Pored toga što generišu kod, poseduju implementiranu logiku za parsiranje različitih sadržaja. Generatori koji imaju ugrađenu logiku parsiranja XML dokumenata, razvrstavamo na osnovu tipa XML semantike:

- *XAMLGenerator-a*
- *RESXGenerato-a*
- *CSProjGenerator-a*

Pored *Parser-Template* generatora čija je namena obrada XML datoteka, definisan je i generator sa ugrađenom logikom parsiranja C# datoteka – *BaseCSharpGenerator*.

Template generatori na osnovu stanja modela koji referenciraju oslanjajući se na obrasce generatora koda generišu sadržaj. U ovu grupaciju najčešće spadaju C# generatori kao što su:

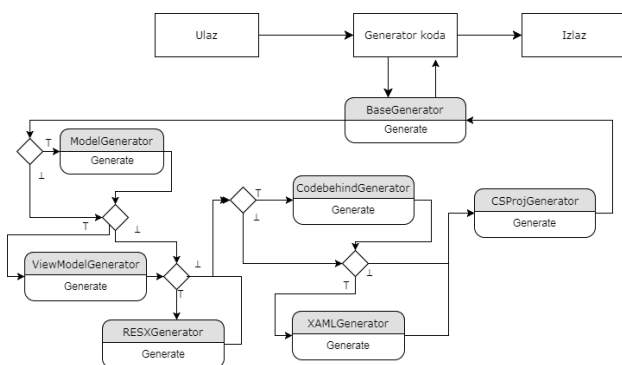
- *ModelGenerator*
- *ViewModelGenerator*
- *XAMLGeneratpr*
- *CodeBehindGenerator*

5. IMPLEMENTACIJA PROCESA GENERISANJA

Na dijagramu aktivnosti (slika 4.), prikazan je tok izvršavanja aktivnosti prilikom generisanja projektnog proizvoda. Budući da je omogućen izbor stavki koje će se generisati za svaki od generatora postoji alternativna putanja za nastavak procesa generisanja.

Svaki od prikazanih generatora se oslanja na na T4 obrasce prilikom procesa generisanja. Imavši u vidu da se model generisanja formira u vremenu izvršavanja programa putem UI aplikacije, u ovom radu su korišćeni run-time T4 obrasci.

Pozivanje run-time T4 šablona u toku izvršavanja aplikacije rezultuje vraćanjem izgenerisane klase na osnovu zadatog šablona koji se dalje serijalizuje u datoteku na zadatu poziciju.



Slika 4. Dijagram toka aktivnosti

6. ZAKLJUČAK

U radu je opisana okvirna struktura formiranja SPL proizvoda poštujući SPL principe uvođenjem novih formi/polja u SPL proizvod. Opisan je proces izrade SPL proizvoda korišćenjem jezika specifičnih za domen. Opisana je struktura meta-modela koja podržava generisanje tj. formiranje novih SPL proizvoda oslanjajući se na SPL bazne resurse. Dalje istraživanje u ovom smeru moglo bi da uključi generisanje validacije u okviru formi. Na ovaj način bi se povećala robusnost i kvalitet generisanog SPL proizvoda.

7. LITERATURA

- [1] Microsoft, "Introduction to WPF," 2017. [Online]. Available: [https://msdn.microsoft.com/en-us/library/aa970268\(v=vs.100\).aspx](https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx).
- [2] Microsoft, "Overview of the .NET Framework," Microsoft, [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview>. [Accessed 2017].
- [3] Microsoft, "XAML Overview," Microsoft, [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-overview-wpf>. [Accessed 2017].
- [4] C. Liu. [Online]. Available: <https://www.cs.nmsu.edu/~epontell/courses/XML/presentations/XMLparsers.ppt>. [Accessed 2017].

- [5] Microsoft, "Controls," [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/controls/>. [Accessed 2017].
- [6] L. M. Ming, "UI, UX: Who Does What? A Designer's Guide To The Tech Industry," [Online]. Available: <https://www.fastcodesign.com/3032719/ui-ux-who-does-what-a-designers-guide-to-the-tech-industry>. [Accessed 2017].
- [7] C. Eberhardt, "A SIMPLE PATTERN FOR CREATING RE-USEABLE USERCONTROLS IN WPF / SILVERLIGHT," [Online]. Available: <http://blog.scottlogic.com/2012/02/06/a-simple-pattern-for-creating-re-useable-usercontrols-in-wpf-silverlight.html>. [Accessed 2017].
- [8] Microsoft, "Layout," [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/layout>. [Accessed 2017].
- [9] Microsoft, "Code Generation and T4 Text Templates," [Online]. Available: <https://msdn.microsoft.com/en-us/library/bb126445.aspx>. [Accessed 2017].

Kratka biografija

Aleksandar Bošnjak rođen je 07.06.1993. godine u Novom Sadu. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, odsek Elektrotehnika i računarstvo, smer Računarstvo i automatika upisao je 2012. godine. Osnovne akademske studije završava 2016. Master akademske studije usmerenje Elektronsko poslovanje upisuje 2016/2017. Položio je sve ispise predviđene planom i programom.

**PRIMENA "ROSLYN" KOMPAJLERSKE PLATFORME I PROŠIRENOG XML
PARSERA NA SOFTVERSKO PROIZVODNE LINIJE****APPLICATION OF "ROSLYN" COMPILER PLATFORM AND EXTENDED XML
PARSER TO SOFTWARE PRODUCT LINES**

Aleksandar Lukić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je predstavljena primena .NET kompajlerske platforme „Roslyn“ i XML parsera za prilagođavanje dokumenata softverske proizvodne linije razvijene u programskom jeziku C#. Takođe, opisano je na koji način je proširen parser za potrebe rada sa XML dokumentima.

Abstract – This paper presents application of .NET compiler platform „Roslyn“ and XML parser for document customization in a software product line developed in programming language C#. It also describes the developments of XML parser that were needed for XML document handling.

Ključne reči: Roslyn, Software product line, code generation, XML

1. UVOD

Softverska proizvodna linija (eng. software product line - SPL) [1] predstavlja paradigmu razvoja softvera po kojoj se razvoj novih proizvoda svodi na prilagođavanje, integraciju i proširivanje elemenata već postojećeg baznog proizvoda, u cilju ispunjavanja korisničkih zahteva. Bazni proizvod se razvija tako da zadovolji zahteve jednog konkretnog tržišta, gde svaki novi proizvod predstavlja njegovo prilagođavanje novim zahtevima. Kako bi se omogućio ovakav razvoj, arhitektura cele proizvodne linije se mora projektovati tako da podrži jednostavnu modifikaciju i proširivost, što se najčešće postiže parametrizacijom i nasleđivanjem softverskih komponenti. Parametri se obično prosleđuju putem konfiguracionih datoteka, dok nasleđivanje zahteva dodatno programiranje. Postavljanjem dobre arhitekture, pravljenje proizvoda svodi se na konfiguraciju i integraciju, čime se smanjuje potreba za dodatnim programiranjem, pa time i vreme potrebno za njihov razvoj.

Iako se dobrom arhitekturom smanjuje vreme razvoja, konfigurisanje i proširivanje, često predstavljaju redundantan i kompleksan posao. To se može uočiti kada postoji veliki broj konfiguracionih datoteka ili kada je neophodno dodati neku kompleksniju funkcionalnost u sistem, ali i u jednostavnijim slučajevima, na primer, kada je potrebno uticati samo na grafički interfejs proizvoda. Jedan od načina kojim se ovaj proces može automatizovati je upotrebom generatora koda.

U ovom radu predstavljene su gradivne komponente generatora koda (CG), upotrebljenog u procesu upravljanja dokumentima, jednog konkretnog SPL-a razvijenog u programskom jeziku C#. Pošto je sadržaj dokumenata specifičan za svakog klijenta, CG se koristi za njihovo prilagođavanje novim specifičnim zahtevima. Detaljan opis njegove arhitekture prikazan je u [2].

Kako se potreba za razvoj CG-a javila u odmakloj fazi razvoja pomenutog SPL-a, pred njega su postavljeni specifični zahtevi. Prvo, bilo je potrebno podržati razvoj novih proizvoda, kao i održavanje postojećih, što ne spada u tipičan slučaj primene generatora koda. Drugo, CG je morao da se prilagodi postojećoj strukturi koja uglavnom nije dozvoljavala primenu dobre prakse razdvajanja generisanog i ručno pisanog koda.

Postojeća struktura je sa jedne strane zahtevala da se kod generiše u datoteke koje programeri ručno menjaju. U želji da se postigne veća sigurnost prilikom izmene koda iskorišćena je kompajlerska platforma „Roslyn“, koja deserijalizuje programski kod u sintaksno stablo i nudi niz operacija za njegovu izmenu. Sa druge strane, bilo je potrebno podržati izmenu konfiguracionih datoteka, na takav način da se očuva njihova potpuna struktura, uključujući komentare i sve bele karaktere (eng. whitespace).

Radi ispunjenja tih zahteva bilo je neophodno korišćenje XML parsera koji u potpunosti očuvava bele karaktere, što normalizacijom XML-a [3] većina njih ne podržava. U potrazi za odgovarajućim XML parserom pronađen je samo jedan koji potpuno čuva strukturu dokumenta. On je veoma sličan DOM parseru [4] zbog toga što od dokumenta pravi stablo, ali pošto je i dalje u razvoju, još uvek ne podržava njegovu izmenu.

2. KOMPAJLERSKA PLATFORMA „Roslyn“

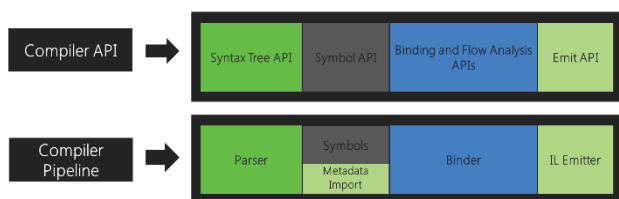
U želji da se pruži mogućnost korišćenja funkcija kompajlera za pravljenje sopstvenih alata, analizu i generisanje koda, Microsoft je razvio .NET kompajlersku platformu [5, 6, 7], poznatiju kao „Roslyn“. Ova platforma nudi kompajlere otvorenog koda (eng. open source), za programske jezike C# i Visual Basic, koji pružaju odgovarajuće aplikativne programske interfejse (API) za praćenje svake faze kompajliranja koda.

Na ovaj način kompajleri postaju servisi koji pružaju informacije o izvornom kodu.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

U „Roslyn“ platformi svaka faza kompajliranja se tretira kao zasebna komponenta koja informacije o svom izvršavanju nudi putem objektnog modela. Faze kompajliranja kroz koje prolazi kompajler i prikaz njihovih odgovarajućih API-ja su prikazani na slici 2.1.



Slika 2.1 Faze kompajliranja i odgovarajući API, preuzeto iz [5]

Parsiranje, u ovoj fazi kompajler tokenizuje izvorni kod i parsira ga u sintaksno stablo zasnovano na gramatici programskog jezika.

Simboli i metapodaci, u ovoj fazi generišu se imenovani simboli. Simboli su elementi deklarirani u izvornom kodu ili su uvezani u vidu metapodataka iz objektnih datoteka. Rezultat izvršavanja ove faze predstavljen je kao hijerarhijska tabela simbola.

Faza povezivanja, reprezentovana je putem objektnog modela, koji nudi rezultat semantičke analize kompajlera nad do tada prikupljenim informacijama.

Emiteri međujezika (eng. intermediate language - IL), na osnovu svih prikupljenih informacija iz prethodnih koraka generišu IL bajt kod.

2.1 Rad sa sintaksom

U „Roslyn“ platformi, sintaksno stablo je primarna struktura koja se koristi za kompajliranje, analizu, refaktorisiranje i generisanje koda. Nastaje analizom niza karaktera izvornog koda procesima skeniranja (tokenizacije) i parsiranja.

Listovi stabla su tokeni prepoznati od strane skenera, dok je struktura stabla određena gramatikom jezika. Sintaksna stabla imaju tri značajne osobine. Prvo, nepro-menljiva su, čime se gubi problem konkurentnosti i nameće specifičan način izmene.

Naime, svaki čvor u stablu nudi odgovarajuće metode za izmenu, koje na ulazu očekuju željene izmene, a kao rezultat vraćaju novo stablo. Drugo, u potpunosti čuvaju informacije o izvornom kodu. Treće, sintaksno stablo koje se dobije parsiranjem, ponovnom serijalizacijom u datoteku daje identičan izvorni kod.

Gradivni elementi ovog stabla su sintaksni čvorovi, tokeni i trivije (eng. syntax trivia). Svaka gradivna komponenta ima svoj tip na osnovu kog se kategorizuje i podatak o tome gde se tačno nalazi u izvornom kodu.

Sintaksni čvorovi reprezentuju deklaracije, izkaze i izraze, i imaju sledeće osobine:

- Svaki tip sintaksnog čvora je reprezentovan odgovarajućom podklasom klase *SyntaxNode*.
- Svaki čvor u stablu osim korena ima roditelja kome može pristupiti putem *Parent* obeležja.
- Uvek imaju potomke koje čine drugi čvorovi i tokeni. Svojoj deci mogu pristupiti putem *ChildNodes* metode.

Dok, svojim potomcima mogu pristupiti putem metoda *DescendantNodes* i *DescendantTokens*.

- Svaka podklasa klase *SyntaxNode* nudi pristup svojoj deci putem strogo tipiziranih obeležja.

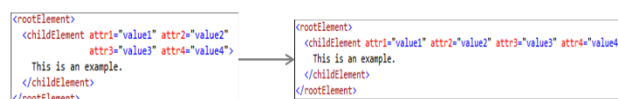
Tokeni predstavljaju terminalne simbole gramatike jezika, oni:

- Nikad nisu roditelji nijednom čvoru ili tokenu.
- Sastoje se od ključnih reči, identifikatora, literala i znakova interpunkcije.
- Zbog efikasnosti tokeni su reprezentovani C# strukturama. Za razliku od čvora, postoji samo jedna struktura koja reprezentuje sve tokene, koji se razlikuju na osnovu obeležja *RawKind*.

Sintaksnu triviju čine delovi izvornog koda koji nisu značajni za njegovo razumevanje, kao što su komentari, beli karakteri i predprocesorske direktive. One nisu uključene u sintaksno stablo kao potomak čvora, već im se može pristupiti putem dva obeležja *LeadingTrivia* i *TrailingTrivia*. Trivije, kao i tokeni, su reprezentovane putem strukture, ali za razliku od njih nemaju roditelja.

3. XML PARSER

U toku razvoja CG-a javila se potreba za korišćenjem specifičnog XML parsera. Naime, za izradu SPL-a za koji je razvijan CG korišćena je WPF tehnologija[8], koja se koristi za kreiranje desktop aplikacija u programskom jeziku C#. U WPF-u grafički prikaz se opisuje putem XML gramatike poznate kao XAML [9]. Jedna od karakteristika XAML datoteka jeste da XML elementi sadrže veliki broj atributa. To je slučaj zbog toga što se putem atributa povezuju delovi koda sa elementima i vrše dodatna podešavanja. Zbog čitljivosti takvih dokumenata, atributi elemenata se najčešće navode jedan ispod drugog. Konvencionalni parseri kao što je DOM ne vode računa o belim karakteristikama koji se nalaze između atributa elementa. Time se deserijalizacijom XML dokumenta u DOM stablo i njegovom ponovnom serijalizacijom u dokument ne dobija isti sadržaj. Na slici 2.1, prikazana je razlika između dokumenta deserijalizovanog putem DOM parsera i dokumenta dobijenog serijalizacijom DOM stabla.



Slika 2.1 Nestanak karaktera za novi red unutar XML elemenata [2]

Pošto je CG trebao da omogući promenu grafičkog prikaza dokumenta izmenom XAML datoteka, DOM parser nije zadovoljio zahteve, pa je za tu potrebu iskorišćen je *XmlParser* [10], razvijan od strane kompanije Microsoft. Ovaj parser je inspirisan „Roslyn“ platformom i kreira nepromenljivo stablo koje u potpunosti čuva sve informacije iz datoteke. To znači da se deserijalizacijom datoteke u stablo i njegovom ponovnom serijalizacijom dobija identičan sadržaj datoteke. Kao i stablo u „Roslyn“ platformi, stablo koje parser kreira se sastoji iz čvorova, tokena i trivija. Pošto je ovaj parser još uvek u razvoju nema implementiran API za izmenu, što ga u slučaju CG-a čini neupotrebljivim. Analizom načina rada i pregledom diskusije samih autora, stablo koje generiše

parser načinjeno je promenljivim, čime se omogućila njegova direktna izmena. Takođe, kreiran je API za izmenu, tako da bude sličan API-ju DOM parsera kako bi se korisnicima pružio familijaran način korišćenje. Naime, čvorovi stabla su prošireni metodama *AppendChild*, *InsertChildAt*, kojima se dodaju nova deca u čvor, zatim *RemoveChild*, *RemoveChildAt*, koje se koriste za uklanjanje čvora iz stabla, dok su za rad sa atributima kreirane *AddAttribute* i *RemoveAttribute* metode.

Kreiranje stabla putem ovog parsera je zbog njegove kompleksnosti veoma zahtevno. Naime, da bi se kreirao XML element potrebno je kreirati većinu njegovog podstabla. Na najvišem nivou hijerarhije XML element se sastoji iz početnog taga, sadržaja i krajnjeg taga. Početni i krajnji tagovi su neophodni za kreiranje elementa, dok je sadržaj opcion.

Da bi se kreirao početni tag, moraju se kreirati čvorovi koji čine njegovu decu, kao što su token za znak „<“, čvor koji predstavlja naziv elementa, čvor koji predstavlja kolekciju atributa i token za znak „>“. Slično je i za zatvarajući tag, zbog toga kreirana je fabrika putem koje je ponuđeno kreiranje čvorova u celini, na primer, *CreateElement*, *CreateAttribute* itd. Na ovaj način se korisniku olakšava kreiranje novih čvorova stabla.

4. PRIMER UPOTREBE

Kao što je već rečeno, CG je upotrebljen za prilagođavanje dokumenata novim klijentskim zahtevima, što najčešće obuhvata dodavanje novih polja u dokument. Da bi se omogućilo čuvanje vrednosti novog polja, potrebno je proširiti model podataka. Sa druge strane, da bi se korisniku omogućio unos i pregled tih vrednosti, potrebno je proširiti korisnički interfejs proizvoda.

Kako je pomenuti SPL kreiran u WPF tehnologiji, dokumenti su, sledeći dobru praksu, kreirani pomoću MVVM (*Model-View-ViewModel*) arhitektonskog šablona [11]. *View* komponenta predstavlja grafički prikaz dokumenta, koji se sastoji iz XAML dokumenta i njemu pridružene C# klase tzv. *code-behind*, u kojoj se takođe mogu definisati delovi korisničkog interfejsa.

Ranije je pomenuto da CG treba da podrži razvoj novih proizvoda i održavanje postojećih. Ukoliko je proizvod već u razvoju njegovi dokumenti su kreirani, pa njihovo prilagođavanje obuhvata samo izmenu sadržaja XAML dokumenata. Sa druge strane, ukoliko se počinje sa razvojem novog proizvoda, njegovi dokumenti nisu još uvek kreirani, već se preuzimaju i prilagođavaju dokumenti baznog proizvoda. Preuzimanje XAML datoteka zahteva i preuzimanje njihovog *code-behind* dela.

U CG-u podrška prilagođavanju dokumenta zahtevima klijenata je implementirana putem nekoliko generatora koda. Specifičnosti tih generatora su opisane u [12], dok je čitava arhitektura opisana u [2]. Ovde će biti pokazani samo delovi generatora, koji se oslanjaju na „Roslyn“ ili *XmlParser*.

4.1 Izmjena grafičkog prikaza dokumenta

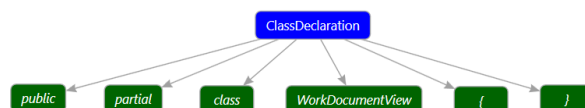
Pošto grafički prikaz dokumenta čine XAML dokument i C# klasa, za njihovu izmenu neophodno je upotrebiti i „Roslyn“ i *XmlParser*. Na osnovu arhitekture SPL-a, dodavanje i prilagođavanje dokumenta kod novih proizvoda obuhvata:

- Preuzimanje sadržaja XAML dokumenta baznog proizvoda, njegova eventualna izmena i čuvanje u novi XAML dokument.
- Preuzimanje *code-behind* dela i njegova izmena koje obuhvata: promenu naziva klase, što uključuje i promenu naziva konstruktora.

4.1.1 Promena naziva klase

Na slici 4.1, prikazan je deo sintaksnog stabla deklaracije klase. Plavom bojom označeni su čvorovi, dok su zelenom označeni tokeni.

Može se videti da čvor *ClassDeclaration* ima direktan pristup nazivu klase putem tokena, u ovom slučaju *WorkDocumentView*.



Slika 4.1 Deo sintaksnog stabla deklaracije klase

U listingu 4.1, prikazana je metoda koja je iskorišćena za izmenu naziva klase. Ova metoda kao parametre očekuje koren sintaksnog stabla i novi naziv klase. Na početku selektuje se prvi čvor koji je tipa *ClassDeclarationSyntax*. Zatim se kreira novi identifikator klase pomoću *SyntaxFactory* klase.

Čvor koji predstavlja deklaraciju klase nudi metodu *WithIdentifier* koja kreira novu deklaraciju klase sa prosledenim identifikatorom. Na kraju se u stablu zamene stara i nova deklaracija klase čime je uspešno promenjen naziv klase i vraća se identifikator kako bi se koristio u drugim metodama.

```

private SyntaxToken RenameClass(SyntaxNode root,
    string newName)
{
    var classDecl = root
        .DescendantNodes()
        .OfType<ClassDeclarationSyntax>()
        .FirstOrDefault();

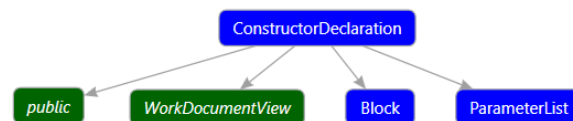
    SyntaxToken identifier = SyntaxFactory
        .Identifier(newName);
    var newClassDecl = classDecl.WithIdentifier(identifier);

    root = root.ReplaceNode(classDecl, newClassDecl);
    return identifier;
}
  
```

Listing 4.1 Metoda za izmenu naziva klase

4.1.2 Promena naziva konstruktora

Na slici 4.2, je prikazan deo sintaksnog stabla deklaracije konstruktora.



Slika 4.2 Deo sintaksnog stabla deklaracije konstruktora

Može se videti da kao i klasa, konstruktor ima direktan pristup svom nazivu. U listingu 4.2, prikazana je metoda za izmenu naziva konstruktora.

Pošto konstruktora u klasi može biti proizvoljno mnogo, potrebno omogućiti izmenu svih njih. U ovom primeru se vidi loša strana rada sa nepromenljivim stablima. Naime, prilikom promene naziva prvog konstruktora kreira se novo stablo, zatim se nad tim stablom mora menjati naziv narednog konstruktora, pošto originalno stablo ne sadrži izmene prvog.

Na ovaj način se povećava potrošnja memorije, pošto preimenovanje svakog konstruktora kreira novo stablo.

```
private SyntaxNode RenameConstructors(SyntaxNode root,
    SyntaxToken identifier)
{
    var constructorCount = root
        .DescendantNodes()
        .OfType<ConstructorDeclarationSyntax>().Count();

    for (var current = 0; current < constructorCount; current++)
    {
        var constructors = root
            .DescendantNodes()
            .OfType<ConstructorDeclarationSyntax>();

        var ctor = constructors.ElementAt(current);
        var newCtor = ctor.WithIdentifier(identifier);
        root = root.ReplaceNode(ctor, newCtor);
    }
    return root;
}
```

Listing 4.2 Metoda za izmenu naziva konstruktorima

4.1.3 Dodavanje novog polja u XAML dokument

Za organizaciju dokumenta korišćeni su XAML *Grid* elementi. Naime, oni u vidu mreže respoređuju polja dokumenta. Da bi se dodalo novo polje, potrebno je prvo proširiti *Grid* sa još jednim redom, što se čini dodavanjem elementa *RowDefinition* unutar *Grid.RowDefinitions* elementa koji je dete *Grid* elementa.

Takođe, potrebno je dodati samo polje kao dete *Grid* elementa. U listingu 4.3, je prikazana metoda koja dobija listu polja koje je potrebno dodati u dokument. Zatim za svako polje dodaje jedan *RowDefinition* element i polje se ubacuje kao dete *Grid* elementa. Čvorovi se selektuju putem Xpath izraza. Rad sa njima je podržan putem *SelectSingleNode* metode koja kao parametar očekuje Xpath izraz u vidu stringa.

```
public void AddNewElements(XmlNodeSyntax[] elements)
{
    XmlDocumentSyntax xmlDoc = LoadDocument();
    var grid = xmlDoc.SelectSingleNode("./Grid");
    var rowDefinitions = grid
        .SelectSingleNode("./Grid.RowDefinitions");
    foreach (var c in elements)
    {
        XmlNodeSyntax newRow = XmlFactory
            .CreateElement("RowDefinition");
        newRow["Height"] = "Auto";
        rowDefinitions.AppendChild(newRow);
        grid.AppendChild(c);
    }
}
```

Slika 4.3 Metoda za dodavanje novih polja u dokument

5. ZAKLJUČAK

Zadatak ovog rada je bio da prikaže na koji način su iskorišćene kompajlerska platforma „Roslyn“ i XmlParser u konkretnom SPL-u razvijanom u programskom jeziku C#, za podršku prilagođavanja dokumenata novim zahtevima klijenata.

Takođe, opisane su izmene funkcionalnosti XmlParsera načinjene u cilju da se omogućí kreiranje i izmena XML dokumenata. Te izmene obuhvataju implementaciju API-ja sličnom onom koji pruža DOM parser i kreiranje fabrike kojom se pojednostavljuje rad sa parserom.

Pošto parser trenutno ne podržava rad sa XML i DTD šemama, kao ni sa namespace-ovima, u daljem radu će se proširivati XML parser kako bi se podržao rad sa njima.

6. LITERATURA

- [1] A framework for software product line practice, version 5.0. http://www.sei.cmu.edu/productlines/frame_report/index.html
- [2] Nenad Todorović. Code Generation Architecture for Software Product Lines. Master's thesis, Faculty of Technical Sciences, 10 2017.
- [3] Xml normalization. <https://www.w3.org/2008/xmlsec/Drafts/xml-norm/Overview.html> Accessed: 01-Oct-2017.
- [4] Dom parsing and serialization. <https://www.w3.org/TR/DOM-Parsing/> Accessed: 01-Oct-2017.
- [5] .net compiler platform ("roslyn"). <http://roslyn.codeplex.com/wikipage?title=Overview> Accessed: 01-Oct-2017.
- [6] Nick Harrison. Code Generation with Roslyn. Apress, 1 edition, 2017.
- [7] Alessandro Del Sole. Roslyn Succinctly. Syncfusion, 1 edition, 2016.
- [8] Windows presentation foundation. <https://docs.microsoft.com/en-us/dotnet/framework/wpf/>. Accessed: 01-Oct-2017.
- [9] What is xaml? <https://msdn.microsoft.com/en-us/library/cc295302.aspx>.
- [10] Xmlparser. <https://github.com/KirillOsenkov/XmlParser>.
- [11] The mvvm pattern. <https://msdn.microsoft.com/enus/library/hh848246.aspx>.
- [12] Aleksandar Bošnjak. Windows Presentation Foundation Forms Code Generator for Software Product Lines. Master's thesis, Faculty of Technical Sciences, 10 2017.26

Kratka biografija

Aleksandar Lukić je rođen 14.9.1993. godine u Novom Sadu. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, odsek Elektrotehnika i računarstvo, smer Računarstvo i automatika upisao je 2012. godine. Osnovne akademske studije završava 2016. godine sa prosečnom ocenom 8,70. Master akademske studije Fakulteta tehničkih nauka, odsek Elektrotehnika i računarstvo, smer Računarstvo i automatika, usmerenje Inteligentni sistemi upisuje školske 2016/2017. Položio je sve ispite predviđene planom i programom.

ARHITEKTURA GENERATORA KODA ZA SOFTVERSKE PROIZVODNE LINIJE CODE GENERATION ARCHITECTURE FOR SOFTWARE PRODUCT LINES

Nenad Todorović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U okviru ovog rada prikazana je arhitektura generatora koda čija je svrha automatizacija procesa proširenja dokumenata kao dela radnog okvira SPL-a, za kojeg su već razvijene bazične softverske komponente. Bazične komponente su razvijene upotrebom .NET tehnologija (C#, WPF), sa SQL bazom podataka, i konfiguracionim datotekama pisanim u XML-u. Veliki uticaj na arhitekturu generatora je imala činjenica da u razvoju proizvoda referentnog SPL radnog okvira, pored generatora, učestvuju i programeri.

Abstract – This paper presents an architecture of code generator that automates the process of the expansion of document-based systems within an SPL framework, for which core components have already been developed. Core components were developed using .Net technologies (C#, WPF), with an SQL database, and configuration files that are written in XML. The architecture was greatly influenced by the fact that in the process of product development of reference SPL framework, the generator has to collaborate with programmers.

Cljučne reči: MDD, code generation, SPL

1. UVOD

Softverska proizvodna linija (*Software product line – SPL*) [1] predstavlja radni okvir po kom se softverski sistemi, namenjeni određenom delu tržišta, sastavljaju prilagođavanjem, proširivanjem i integracijom postojećih, bazičnih softverskih komponenti u klijentu prilagođen proizvod. Kako bi takva integracija bila moguća i olakšana, bazične softverske komponente obično imaju adaptivnu arhitekturu. Međutim, sastavljanje i prilagođavanje iziskuju iscrpan, često repetativan rad programera.

Generator koda (*Code Generator – CG*), čija arhitektura je tema ovog rada, nastao je sa ciljem da taj proces ubrza kod SPL-ova čija svrha je sastavljanje softverskih proizvoda baziranih na dokumentima za upravljanje proizvodnim procesom.

Deo kreiranja takvog proizvoda predstavlja postupak dodavanja novih polja u dokumente poslovnog procesa, i njihovo uključivanje u arhitekturu rešenja. Ovaj postupak je prepoznat kao jedna od procedura koju je moguće automatizovati i značajno ubrzati generatorom koda. Razvijeni generator ujedno predstavlja i alat za modelovanje dokumenata, prilagođen pre svega generisanju sistema kreiranih u C# tehnologijama.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

Činjenica da se CG kao alat uključuje u razvoj proizvoda kreiranog od već razvijenih bazičnih softverskih komponenti, čini ga atipičnim u odnosu na klasične generatore, jer postojeći softver poseduje strukturu koju generator ni na koji način ne sme da naruši. Generisani kod takvoj strukturi mora da se prilagodi, što posledično znači da generator pored modela mora da analizira i trenutno stanje softvera, te da interaguje sa korisnikom u slučaju kada rizik od narušavanja strukture postoji. To su ujedno i najbitniji zahtevi koje je potrebno ispuniti.

2. REFERENTNI SPL RADNI OKVIR

Domen referentnog SPL radnog okvira jesu sistemi koji uključuju funkcionalnost upravljanja radnim nalogima, kao uobičajenu funkcionalnost softvera koji se bavi upravljanjem proizvodnim procesima. Dokumenti koji se razmatraju su: radni zahtev, radni nalog i plan rada. Sadržina ovih dokumenata je specifična za svakog klijenta, s obzirom da klijenti imaju ustaljene poslovne procese, pa i dokumenti kao deo softvera moraju da im se prilagode. Ovakvo prilagođavanje softvera pojedinačnim klijentima predstavlja dobar primer upotrebe SPL radnog okvira, kod kojeg bi se bazični softver proširio i iskonfigurisao s ciljem dobijanja proizvoda u skladu sa zahtevima.

Navedeni sistemi imaju strukturu uobičajenu za poslovne softverske sisteme, i ona se sastoji od:

- Grafičkih klijentata, pod kojima se podrazumevaju desktop aplikacije. Oni pružaju grafički prikaz, omogućavajući korisniku da upravlja poslovnim procesima, što podrazumeva i upravljanje dokumentima;
- Servisa, koji opslužuju sve delove sistema. Oni vrše potrebne obrade, validaciju i upravljanje podacima i životnim ciklusom dokumenata. Za potrebe ovog primera, njihovo ponašanje je visoko konfigurabilno; i
- Baze podataka, čija je svrha da skladišti podatke sistema i aktivnosti korisnika.

Početna tačka za proširenje dokumenata jeste proširenje modela podataka. Takvo proširenje je potrebno uneti u odgovarajuće konfiguracione datoteke, kako bi sve funkcionalne celine strukture uzele takvo proširenje u obzir. Prilikom projektovanja generatora za proširenje jednog ovakvog sistema, uzeto je za činjenicu da je on realizovan u .NET tehnologijama, na programskom jeziku C# i da se konfiguriše XML datotekama.

Da bi se omogućio unos proširenih dokumenata, potrebno je proširiti korisnički interfejs, što iz ugla strukture sistema podrazumeva proširenje desktop aplikacija, u referentnom sistemu realizovanog u *Windows Presentation Foundation (WPF)* [2] tehnologiji. Radi omogućavanja perzistencije podataka unetih dokumentima, potrebno je proširiti modela

baze podataka (skladišta), u referentnom sistemu realizovanog u *SQL* tehnologiji.

Postoje četiri mehanizma kojima se sprovodi prilagođavanje u okviru navedenog primera *SPL*-a, a koja generator implementira:

- Nasleđivanje *C#* klasa bazičnih komponenti;
- Proširenje konfiguracije *XML* datoteka;
- Proširenje *XAML* dokumenata kao *XML* bazirane sintakse za opis grafičkog interfejsa *WPF* (Windows Presentation Foundation) tehnologije; i
- Proširenje *SQL* baze podataka generisanjem potrebnih *SQL* naredbi.

To su ujedno i četiri klase datoteka kojima generator može da rukuje.

3. PROJEKTOVANJE GENERATORA

Prilikom projektovanja, analizirani su zahtevi, isprojektovan je meta-model jezika za domen proširenja dokumenata u okviru ovakvog *SPL*-a, i definisana gruba arhitektura.

3.1 Analiza zahteva

Kako se generator uključuje u proces razvoja u kojem uporedo učestvuju i programeri, generator prilikom generisanja mora voditi računa o postojećem kodu, što je ujedno i najbitniji zahtev. Zatim, generator mora biti osposobljen da se uključi i u razvoj novog, tek uspostavljenog proizvoda, kao i u razvoj i održavanje već postojećeg proizvoda *SPL* procesa.

Njegove aktivnosti ni na koji način ne smeju da naruše strukturu konačnog proizvoda *SPL* procesa. Potom, zahtev je da se omogući korisniku da vidi izgenerisani sadržaj i pre nego što se on upiše u datoteke (preview).

Kada korisnik odluči da izgeneriše sadržaj, generisani softverski elementi bi trebalo u potpunosti da se integrišu sa proizvodom.

Uz to, generator mora omogućiti korisniku da bira u koje delove strukture i funkcionalnosti sistema bi novododata polja trebalo da se integrišu.

3.2 Gruba arhitektura generatora

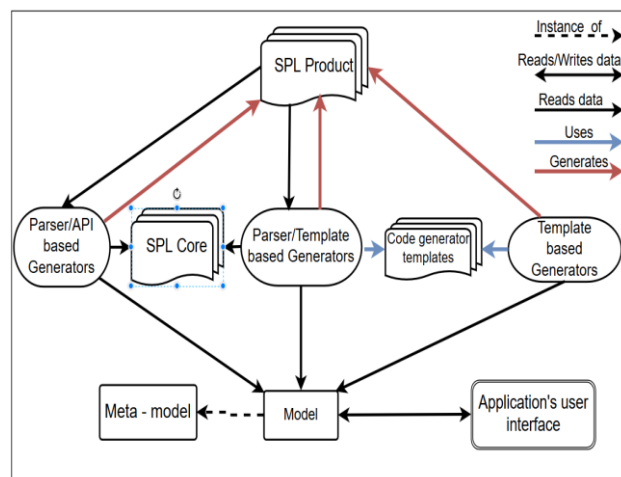
Kako bi se navedeni zahtevi ispunili, osmišljena je arhitektura koja ima izgled prikazan na slici 3.1. Njena osnova je meta-model koji definiše odgovarajuće koncepte za modelovanje proširenja *SPL* proizvoda zasnovanog na dokumentima.

Njemu odgovara model, kojeg korisnik kreira preko korisničkog interfejsa generatora. Tako kreiran model je ulaz u generator koda. Drugi ulaz je informacija o trenutnom stanju programskog koda proizvoda, do koje se dolazi parsiranjem.

Generator vrši generisanje koda korišćenjem hijerarhije generatorskih klasa, gde je svaki konkretni element hijerarhije zadužen za jedan tip datoteke potrebne za formiranje *SPL* proizvoda. Po načinu funkcionisanja, klase iz hijerarhije generatora su podeljene na tri grupe.

Prva grupa je bazirana na šablonima i parserima, koji radi analize stanja sistema moraju parsirati datoteke, kako baznog softvera *SPL* procesa, tako i proizvoda *SPL* procesa.

U drugu, manju grupu spadaju generatorske klase bazirane na šablonima koje vrše generisanje, bez prethodne analize trenutnog stanja. Članovi ove i prethodno pomenute grupe generisanje vrše pomoću obrađivača šablona (*template engine*). Treću, poslednju grupu generatora čine generatorski objekti bazirani na parserima i njihovim *API*-jima, koji kod generišu isključivo preko navedenog *API*-ja.

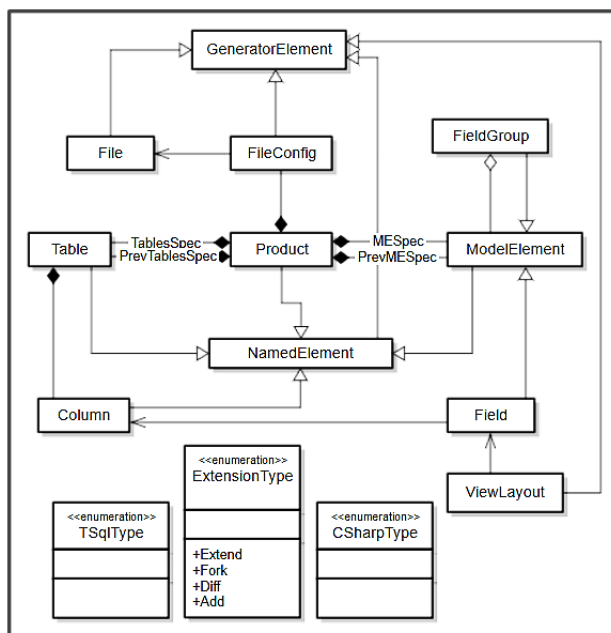


Slika 3.1 Gruba arhitektura generatorske aplikacije

3.2 Meta-model jezika

Najbitnije meta-klase modela, prikazane na slici 3.2 jesu sledeće:

- Meta-klasa *Product*, koja se mapira na proizvod *SPL* procesa, i čija instanca enkapsulira instance svih ostalih meta-klasa;
- Meta-klasa *FieldGroup*, koja se mapira na dokument;
- Meta-klasa *Field*, koja se mapira na polje, enkapsulirano dokumentom;
- Meta klasa *Table*, koja se mapira na tabelu;
- Meta klasa *Field*, koja se mapira na polje tabele, i u vezi je sa poljem, čime se određuje u okviru koje kolone se čuvaju podaci uneti kroz polje;
- Meta-klasa *FileConfig*, koja enkapsulira podatke o tome kojim datotekama, i na koji način, rukuje *CG* generator. Mogući načini rukovanja jesu nasleđivanje (*extend*), preuzimanje (*fork*), semantičko nastavljavanje (*diff*) datoteke iz baznog softvera u proizvodni softver. Uz to je prepoznato kao način i dodavanje (*add*) nove datoteke u proizvod;
- Meta-klasa *ViewLayout*, koja enkapsulira informacije o rasporedu na dokumentu; i
- Veze za verzionisanje (*TablesSpec*, *MESpec*, *PrevTablesSpec*, *PrevMESpec*), koje postoje kako bi se moglo zaključiti koji sadržaj je generator generisao, jer njega potom sme slobodno da menja, dok se on ne promeni ručno. U slučaju da se on promeni ručno, zamišljeno je da se konflikt razreši tako što generator konsultuje korisnika, što je jedan od budućih koraka u razvoju.



Slika 3.2 Meta-model jezika

3.3 Korišćene tehnologije

Generator je razvijen kao WPF generatorska aplikacija. Za generisanje zasnovano na šablonima je korišćen *Microsoft T4 Framework* [3]. Za parsiranje, a u nekim slučajevima i generisanje C# koda, korišćena je Microsoft .NET platforma za komplajlere [4]. Za parsiranje XML datoteka jednostavne strukture, u svrhu proširenja konfiguracionih datoteka, korišten je Microsoft-ov XML parser [5]. Pored navedenog standardnog parsera, za potrebe rukovanja Extensible Application Markup Language (XAML) [6] sintaksom baziranoj na XML-u, proširen je i upotrebljen XML parser opisan na stranici [7].

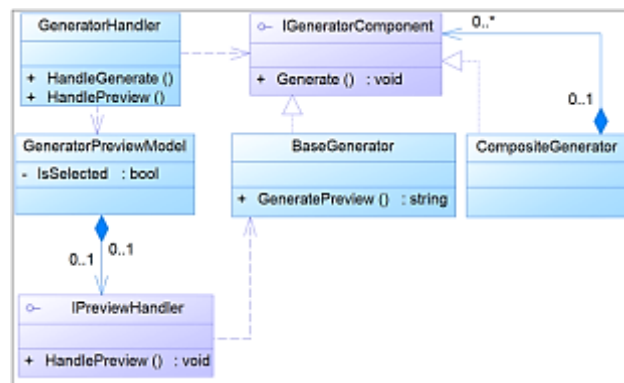
4. DETALJNA ARHITEKTURA GENERATORA

4.1 Osnovna arhitektura za generisanje

S obzirom da je jedan od zahteva da se korisniku omogući pregled koda koji će biti izgenerisan (*preview*), postoje dva procesa u toku kojih mora doći do generisanja koda: generisanje radi prikazivanja generisanog koda korisniku i generisanje radi snimanja generisanog koda u datoteke. Ova dva procesa se razlikuju, zbog čega je osnovna arhitektura za generisanje (slika 4.1), osmišljena tako da postoje dve relativno nezavisne strukture, po jedna za svaki proces.

Zajednička početna tačka za ova dva procesa jeste klasa *GeneratorHandler* - upravljač generatorskim klasama (u daljem tekstu samo *generatorima*). Bez obzira da li se deo koda generiše za potrebe snimanja ili prikazivanja, uvek ga generiše instanca iste generatorske klase. Generatorske klase su uređene u hijerarhijsku strukturu, koja je detaljnije objašnjena u narednoj sekciji. Na vrhu hijerarhije generatorskih klasa nalazi se bazni generator *BaseGenerator*.

Radi veće kontrole toka generisanja, upotrebljen je šablon kompozicije (eng. *Composite pattern*) radi kreiranja kompozitnog generatora. Time je omogućeno grupisanje generatora koji generišu međusobno zavisne komponente proizvoda. *BaseGenerator* i *CompositeGenerator* implementiraju interfejs *IGeneratorComponent*, koji definiše metodu *Generate*. Svrha ove metode je generisanje koda i njegovo snimanje u odgovarajuće datoteke.



Slika 4.1 Osnovna arhitektura za generisanje

Proces generisanja se inicira pozivom metode *HandleGenerate* upravljača generatorima. Postupak generisanja počinje instanciranjem pojedinačnih generatora. U slučajevima kada to ima smisla, instancirani generatori se grupišu u kompozitnu strukturu. Nad svim generatorima, kako pojedinačnim, tako i kompozitnim, iterativno se poziva metoda *Generate*, čime se kod generiše i snima u datoteke. Kako se generisanjem stanje čitavog modela sada preslikalo na proizvod, potrebno je to zabeležiti i u modelu. Zato se kolekcije *MESpec* i *TableSpec* kopiraju u *PrevMESpec* i *PrevTableSpec*, respektivno. Na kraju, novonastalo stanje modela se snima na disk.

Kako bi se omogućilo generisanje prikaza (*preview*), uvedena su dva nova elementa. Prvi jeste *IPreviewHandler*, predstavljen interfejsom kojeg implementiraju klase namenjene upravljanju generisanjem radi prikaza. On se vezuje za grupu semantički bliskih datoteka za generisanje. Interfejs definiše jednu metodu, *HandlePreview*, u okviru čije implementacije bi trebalo da budu pozvane metode *GeneratePreview* generatora. Arhitekturom nisu postavljena nikakva ograničenja u vezi sa tipom ili brojem pozvanih konkretnih generatora, zbog toga što su zahtevi za strukturom prikazanog generisanog koda raznoliki. Drugi element je *GeneratorPreviewModel*, koji enkapsulira *IPreviewHandler*, uz obeležje *IsSelected*, koje nosi informaciju da li korisnički odabir zahteva pokretanje enkapsuliranog *IPreviewHandler*-a.

4.2 Hijerarhija generatorskih klasa

S ciljem objedinjavanja zajedničkih funkcionalnosti generatorskih klasa, odnosno klasa koje generišu kod radi umanjenja redudanse koda, i definisanja strukture koju bi implementacije konkretnih generatora trebalo da imaju, osmišljena je hijerarhija baznih generatora. Hijerarhija ne prati u potpunosti grubu arhitekturu prikazanu u sekciji 3.2, već je vezana za tipove datoteka kojima CG generator rukuje. Tako imamo bazni generator *XAML* datoteka *BaseXAMLSGenerator*, bazni generator *XML* datoteka *BaseXMLGenerator*, bazni generator *C#* datoteka *BaseCSharpGenerator* sa dodatnom pot-hijerarhijom, i bazni generator *SQL* skript datoteka *BaseDBGenerator*. Na vrhu hijerarhije se nalazi bazni generator *BaseGenerator* koji definiše osnovno ponašanje koje bi konkretne implementacije trebalo da prate.

Predviđeno je da svi generatori imaju metodu pomenutu metodu *Generate*, čije izvršavanje predstavlja početni korak za mehanizam generisanja radi snimanja sadržaja na disk. Ona je implementirana u okviru baznog genera-

tora. U toku procesa generisanja, operacije se loguju kako bi korisnik imao uvid u tok. Potom, implementiran je mehanizam za rukovanje izuzecima, koji prepoznaje dve grupe izuzetaka:

- Kritične izuzetke, kada se obustavlja proces generisanja na nivou čitave aplikacije; i
- Regularne izuzetke, koji se loguju radi otkrivanja uzroka.

Prilikom inicijalizacije mora doći do nalaženja instance *FileConfig* meta-klase, kako bi metode u okviru generatorske klase mogle da rukuju, potrebnim datotekama. Većina generatorskih klasa zahteva učitavanje, ili proveru stanja sadržaja datoteke. Potrebne datoteke se prvo traže u proizvodnom, pa potom u baznom softveru, i predviđeno je da u toku inicijalizacije dođe do učitavanja njihovog sadržaja u dinamičku strukturu;

Metoda *GenerateContent* služi kako bi se dinamička struktura nastala u toku inicijalizacije izmenila sadržajem generisanim u skladu sa trenutnim stanjem modela. Tri su stanja elemenata između verzija na koja metoda reaguje:

- Stanje New, prilikom kojeg poziva apstraktnu metodu *AddNamedElement*;
- Stanje Edited, prilikom kojeg poziva apstraktnu metodu *UpdateNamedElement*;
- Stanje For removal, prilikom kojeg poziva apstraktnu metodu *DeleteNamedElement*.

Radi provere da li bi za određeno polje trebalo da dođe do generisanja definisana je metoda *ShouldGenerate*.

Kako bi došlo do serijalizacije dinamičke strukture u datoteku, mora biti implementirana za to predviđena apstraktna metoda *Flush*. Novokreirane datoteke se moraju registrovati, kako bi se upotpunosti integrisale u proizvod. U slučaju da aktivnosti generatora ne zahtevaju dinamičku strukturu, do snimanja u datoteku dolazi u okviru metode *GenerateContent*.

4.3 Rukovanje rasporedom

S ciljem omogućavanja rukovanja rasporedom, nastala je zasebna struktura. Takvu strukturu enkapsulira dokument, jer se rasporedom rukuje na nivou dokumenta. Struktura se sastoji od kolektor rasporeda (eng. *Layout Collector*) upravljača rasporedom. Svrha kolektora rasporedom je da učitati dokument iz datoteke, reflektuje stanje modela na učitani sadržaj, i da model rasporeda. Kolektor enkapsulira upravljač rasporedom, klasu koja menja model rasporeda prilikom izmene rasporeda od strane korisnika, u skladu sa ograničenjima.

Generatori koji generišu sadržaj datoteka kod kojih postoji potreba za rukovanjem rasporeda, ne inicijalizuju se samostalno, već u tu svrhu od dokumenta kontaktiraju potrebnog rukovaoca rasporedom, koji se prilikom prvog pristupa inicijalizuje. Uz sintaksno stablo dokumenta, generatori preuzimaju i model rasporeda. Generatorske klase potom menjaju sintaksno stablo, i snimaju ga na disk.

5. ZAKLJUČAK

U okviru ovog rada prikazana je arhitektura generatora koda čija je svrha automatizacija procesa proširenja dokumenata kao dela radnog okvira softverskih proizvodnih linija. Na primeru apstraktnog sistema su objašnjeni mehanizmi na kojima je ovakav proces zasnovan. Navedena su ograničenja postavljena pred generatorom. Prikazana je gruba globalna arhitektura generatora, i meta-model koji uključuje potrebne meta-klase iz domena, kao početnom tačkom za sve funkcionalnosti.

Potom je predložen detaljan opis arhitekture za generisanje. Prvo je predložen osnovni pogled na arhitekturu, koja je oblikovana prema zahtevima. Zatim je objašnjen vrh hijerarhijske strukture generatorskih klasa, koja je nastala s ciljem objedinjavanja zajedničkih funkcionalnosti generatorskih klasa radi smanjenja redudanse koda, i definisanja strukture koju bi implementacije konkretnih generatora trebalo da imaju. Na kraju je prikazana arhitektura stvorena kako bi se upravljalo rasporedom komponenti u okviru samih dokumenata, uz proces generisanja koda, koji zahteva saradnju komponenti..

6. LITERATURA

- [1] „A Framework for Software Product Line Practice, 5.0“ http://www.sei.cmu.edu/productlines/frame_report/Web, 6. Oct. 2017.
- [2] „Introduction to WPF“ <https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/introduction-to-wpf-in-vs> Web, 6. Oct. 2017.
- [3] M. Voelter, DSL Engineering: Designing, Implementing and Using Domain-Specific Languages, 2010-2013.
- [4] „Code Generation and T4 Text Templates“ <https://msdn.microsoft.com/en-us/library/bb126445.aspx>
- [5] „The .NET Compiler Platform (\"Roslyn\")“, <https://github.com/dotnet/roslyn>
- [6] „XML Processing“, https://msdn.microsoft.com/en-us/library/aa478996.aspx#aspnet-jspmig-xmlprocessing_topic3
- [7] „What is XAML?“, <https://msdn.microsoft.com/en-us/library/cc295302.aspx>
- [8] „XmlParser“, <https://github.com/KirillOsenkov/XmlParser>

Kratka biografija:

Nenad Todorović rođen je 18.12.1993. godine u Subotici. Završio je Osnovnu školu „Kizur Istvan“ u Subotici, kao dobitnik Vukove diplome, posle čega upisuje Gimnaziju „Svetozar Marković“, prirodno-matematički smer, takođe u Subotici. Školske 2012/13. godine upisuje osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, na studijskom programu Računarstvo i automatika. Diplomirao je 2016. godine sa prosečnom ocenom 9,70. Iste godine upisuje master akademske studije na Fakultetu tehničkih nauka, studijski program Računarstvo i automatika, smer Računarske nauke i informatika. Položio je sve ispite propisane planom i programom sa prosečnom ocenom 10.

GEJMING U SLUŽBI OČUVANJA ZDRAVLJA GAMING IN SERVICE OF HEALTHCARE

Jelica Kapetina, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – *Terapija afazije može biti dugotrajna i naporna za pacijente. Pored redovnih lekarskih tretmana, oni nemaju načina da vežbaju samostalno, u svom tempu. Tokom izrade ovog rada, kreirana je aplikacija u „Unity” pogonu, kao rešenje za samostalno vežbanje govora i prepoznavanja pojmova kod pacijenata sa afazijom, koje bi se moglo primenjivati uz redovnu lekarsku terapiju.*

Abstract – *Aphasia therapy can be longterm and hard for patients. Besides doctors treatments, they have no way of self-training in their own pace. During the making of this paper, an application is developed in “Unity” editor, and presented as a solution for self practice of speech and term recognition in patients with aphasia, which could be used with regular doctors therapy.*

Ključne reči: *Gejming, računarske igre, apopleksija, demencija, afazija, govorna terapija, prepoznavanje*

1. UVOD

Ranije je vladalo mišljenje da su računarske igre gubitak vremena, ali danas postoji mnoštvo studija koje ilustruju njihovu korisnost. Računarske igre generalno, dužom upotrebom, ubrzavaju donošenje odluka, brže reagovanje na promene u okolini, bolju percepciju u vidnom polju, finiju motoriku, ali se računarske igre u skorije vreme koriste i u borbi protiv mentalnih bolesti[1].

Privlačnost računarskih igara se, citirajući Barbaru Čemberlin, direktorku New Mexico State University Learning Games laboratorije, krije u tome što „Igre nude direktnu povratnu informaciju, gde možete videti svoje napredovanje, isprobati nešto i biti frustrirani, ali kasnije i naučiti još više.” („Games offer immediate feedback, you can see your progress, you can try something and be frustrated but later learn more... that’s why game play is so engaging to us.” — Barbara Chamberlin, project director at the New Mexico State University Learning Games Lab). Privlačnost igranja računarskih igara se može proceniti kroz prihode od njihove prodaje. U 2014 godini, cifra od prodaje računarskih igara, bila je 15,4 biliona dolara [2], pri čemu su u brojke uključene samo igre prodane na fizičkim medijima, poput CD-ova. Ukupan profit je zapravo mnogo veći.

Što se tiče namene igara, ona se deli na dve grupe: zabavne i ozbiljne igre. Termin „ozbiljne igre” je uveden za one igre koje su „dizajnirane za primarnu svrhu, a koja nije samo zabava” [3]. Ozbiljne igre su nastale iz želje da

se omogućiti učenje, podizanje svesti o nekoj temi, ili pomoć određenim grupama ljudi na jedan zabavan način.

Fokus rada će biti na oblasti mentalnih oboljenja koja nastaju kao posledica apopleksije, odnosno stanja moždanog udara. Uz rad je kreirana igra **Mozaik** koja cilja pacijente sa afazijom (poremećajem govora), poput postojeće komercijalne igre „Language Therapy”, čija je uspešnost naučno dokazana. Biće izneta istraživanja koja potvrđuju uspešnost igre „Language Therapy”, ali i njene mane koje su prevaziđene u igri **Mozaik**.

2. PREGLED PRIMENE GEJMINGA U PREVENCIJI I TERAPIJI

Računarske igre se intenzivno koriste u prevenciji i terapiji oboljenja, ali je jasan njihov nedostatak u oblasti detekcije. Prostor za primenu gejminga u oblasti detekcije oboljenja postoji sa velikim potencijalom, pa je to oblast na koju bi trebalo obratiti više pažnje u budućnosti.

2.1. Gejming u službi prevencije

Neosporna je činjenica da deca u današnje vreme provode mnogo vremena za računarom, igrajući računarske igre. U pokušaju da se to iskoristi, kreiran je veliki broj računarskih igara koji za cilj imaju edukaciju dece o štetnosti nekih materija, bolestima, lepom ponašanju, očuvanju okoline i vežbanju memorije, na primer. Pretraga interneta za pojmom „children education games” vraća veliki broj sajtova sa još većim brojem edukativnih računarskih igara.

2.2. Gejming u službi terapije

Mnogobrojna stanja dovode do toga da pacijent oseća veliki fizički ili emotivni bol. Same terapije mogu biti izuzetno teške za podneti, poput hemoterapije, pa često pacijenti vrlo brzo klonu duhom. Međutim, postoji niz studija koje pokazuju da računarske igre mogu da se upotrebe kao distrakcija od bola. Na primer, jedna studija [4] je pokazala da su deca na hemoterapiji, koja su igrala računarske igre, pokazala ređe i manje intenzivne mučnine u odnosu na kontrolnu grupu. Time je smanjena i potreba za lekovima protiv bolova ili potreba za medicinskim nadzorom.

Apopleksija (šlog) često ima za posledicu poremećaje motorike, govora i razumevanja. U najvećem slučaju su te posledice reverzibilne: izgubljene funkcije je moguće povratiti, ali samo ukoliko se intenzivno radi na njihovom obnavljanju. Fizička rehabilitacija je podjednako bitna koliko i kognitivna. Neki pacijenti izgube sposobnost komunikacije (u jednom ili više oblika), snalaženja u prostoru, pamćenja, prepoznavanja lica i predmeta, itd. Za svakog od njih je najbitnije da radi na izgubljenoj sposobnosti. Medicinsko osoblje i okolina često nema dovoljno vremena da svakodnevno radi sa pacijentom na

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red.prof.

njegovoj rehabilitaciji. Za tako nešto je potreban izuzetan napor i terapija od po više sati, svakog dana.

Afazija je poremećaj govora i sposobnosti komunikacije, do koga dolazi kada prilikom šloga izlivena krv zahvati levu hemisferu mozga. Ovo može čak dovesti do toga da oboleli nije u stanju da razume jezike koje je pre šloga razumevao. Štaviše, ovim pacijentima predstavlja napor govoriti smisleno, ali iako im može biti izuzetno teško da se sporazumevaju sa okolinom, bitno je naglasiti da njihova inteligencija ostaje netaknuta. Postoji niz aplikacija uz koje pacijent može vežbati komunikaciju i kao najbolja pokazala se aplikacija „LanguageTherapy“ (TactusTherapy) [5]. Korisnik pri pokretanju igre bira koji vid terapije želi da pokrene, a izbori su: razumevanje, imenovanje, čitanje i pisanje. Postoji studija [6] koja je istraživala uticaj ovakvih aplikacija na pacijente, a koja pokazuje da one mogu značajno da pomognu obolelim pacijentima u povratku normalnih sposobnosti za komunikaciju.

U svrhe terapije demencije, postoji razvijen sistem nazvan Cognimat [7], čiji je cilj da se fokusira na prostornu orijentaciju, prostornu memoriju, radnu memoriju i društvenu interakciju. Kao što se navodi u radu [7], njegova prednost je u tome što ga korisnik može koristiti iz svog doma, samostalno, te time uštedeti troškove lečenja. Pritom, softver daje osećaj igre i pruža zabavu svojim korisnicima, što ih podstiče na dalju terapiju i brže učenje. Projekat Cognimat je nastao jer je utvrđeno da je većina postojećih softvera dvodimenzionalne prirode, koja ne omogućava treniranje prostorne orijentacije. Zbog toga je VR (Virtual Reality) savršeno prilagođen za dati problem, ali očevidan je nedostatak ovakvih sistema na tržištu.

3. RAZVOJ RAČUNARSKIH IGARA

Na razvoj jedne računarske igre utiču mnogi faktori: ciljna publika, tema igre, njena namena, budžet za proizvodnju, veličina tima koji radi na igri, ciljna platforma, konfiguracije uređaja na kojima se ta igra pravi itd. Osnovni faktor razvoja računarske igre je ciljna publika, odnosno pitanje: kome je igra namenjena. To pitanje od samog starta oblikuje proces stvaranja igre i svih njenih aspekata, sa ciljem da se ta igra u potpunosti prilagodi svojim korisnicima.

Istraživanja [2] su pokazala da prosečan igrač ima 35 godina. Grupa igrača koja je najmanje zastupljena, upravo je grupa starosti od 36 do 49 godina, jer je to radno stanovništvo, a grupa koja je posebno zanimljiva je grupa starosti veće od 50 godina. Vidi se da po brojnosti stariji igrači ne zaostaju za mlađim. To je jasan pokazatelj da su računarske igre privlačne i starijoj populaciji, upravo onoj koja je pod najvećim rizikom od nastanka mentalnih oboljenja ili moždanog udara. Imajući ovo u vidu, uz činjenice o pozitivnom uticaju igara na ljudski mozak, može se primetiti jasna mogućnost primene računarskih igara u terapijama različitih oboljenja.

3.1. Razvoj zabavnih igara (entertainment)

Razvoj zabavne igre se odvija kroz nekoliko faza: faza kreiranja ideje i koncepta igre, faza dizajniranja mehanike i nivoa, faza preprodukcije, faza produkcije, faza testiranja i faza objavljivanja. Uvek se prvo kreće od postavke ideje igre, koja se u narednoj fazi dizajna

postepeno razvija i poboljšava. Faza preprodukcije predstavlja kreiranje plana rada. Sastavlja se kompletan tim ljudi koji će biti uključeni u razvoj i marketing. Preprodukcija obuhvata i pronalaženje fizičkog prostora za rad i opreme, dok kreiranje plana rada podrazumeva formiranje budžeta, pronalaženje investitora, određivanje novčanih naknada i satnica za sve osobe uključene u razvoj igre. U toj fazi se određuju uloge svih učesnika, definišu dužnosti i vrši se raspodela zadataka. U fazi preprodukcije se takođe donosi odluka oko objavljivanja igre: da li to samostalno uraditi ili uposliti drugu firmu. Tek kada se sve isplanira, onda se kreće sa fazom produkcije koja predstavlja programiranje i kreiranje art elemenata igre. Programiraju se nivoi i implementira se njihova kompletna mehanika. Art podrazumeva 3D modele koji će se koristiti u igri za likove i objekte, modele terena za nivoe, dekoraciju nivoa, muziku i animaciju.

U najboljem slučaju, faza testiranja se povezuje sa fazom produkcije, tako što se svaki nivo odmah po svom završetku testira. Time je obezbeđeno blagovremeno pronalaženje grešaka. Nakon završene faze produkcije, radi se intenzivno testiranje i potraga za propustima i greškama u programiranju, gde se rade stres testovi i igra namerno pokušava da se dovede u nepredviđeno stanje. Vrlo često se za fazu testiranja ne ostavi dosta vremena, što dovede do toga da se igra objavi sa velikim brojem propusta. Generalno govoreći, faza testiranja određuje kvalitet igre, pa se stoga ne sme zanemariti. Nakon testiranja, sledi faza objavljivanja kreirane računarske igre. Objavljivanje podrazumeva dobijanje svih potrebnih licenci, autorskog prava, zaštita imena igre i njenog brenda, a potom i masovnu distribuciju i reklamiranje te igre. Igra može da se reklamira u časopisima, na televizijskim kanalima i radio emisijama, međutim moderni načini marketinga uključuju socijalne mreže i gejming konferencije.

3.2. Razvoj ozbiljnih igara (serious games)

Razvoj ozbiljnih igara odstupa od modela razvoja zabavnih igara u fazi dizajna. Nakon osmišljavanja ideje, razrađuje se mehanika i različitost nivoa igre. Upravo u osmišljavanju mehanike leži najveći izazov u kreiranju ozbiljnih igara, jer je potrebno dizajnirati mehaniku tako da se igra učini što privlačnijom, a istovremeno što približnijom glavnoj ideji i poruci same igre.

3.3. Proces razvoja Mozaik ozbiljne igre

Ideja iza igre koja je razvijena u ovom radu, bila je pomoći ljudima koji su pretrpeli šlog, da rehabilituju svoje kognitivne sposobnosti i fine pokrete ruku, u sopstvenom tempu, uz minimalni angažman drugih ljudi. Sa psihološke strane, takvi pacijenti se često osećaju kao dodatan teret za ljude u svom okruženju, ili čak osećaju nelagodu jer im je neophodna tuđa pomoć u obavljanju svakodnevnih aktivnosti. Tipične terapije su dosada podrazumevale da pacijenti redovno posećuju lekara, kako bi sa njim vežbali govor ili fine pokrete ruku. To je znalo predstavljati opterećenje za pacijente koji žive dalje od bolnice, ili one koje nemaju nekog u okruženju ko bi mogao da im pomaže dvadeset četiri sata dnevno. Za takve pacijente bi bila izrazito korisno da mogu vežbati sami, u bilo koje doba dana i noći, koliko god često to oni

želeli. Uzimajući sve u obzir, koncepti igre **Mozaik** su: vežbanje fine motorike u rukama i vežbanje prepoznavanja raznih pojmova. Fizička rehabilitacija je ostavljena za kasnije unapređivanje igre.

Moždani udar može imati razne posledice po čoveka. Može biti narušena motorika ruku, sposobnost pamćenja, sposobnost govora, kao i sposobnost razumevanja. Bilo bi izuzetno teško da se jednom igrom pokuša pomoći u svakoj od tih oblasti. Zbog toga je **Mozaik** kreiran kao omnibus devet podjednako važnih igara, od kojih svaka pokriva po jednu ili više oblasti primene.

Razvoj mozaika, odvijao se po već iznetom modelu razvoja ozbiljnih igara. Formiran je tim, pronađen prostor i oprema za rad, a odlučeno je da se igra neće objavljivati dok se ne završi faza testiranja od strane pacijenata. U razvoj su bili uključeni specijalista rehabilitacije, neuropsihijatar i logoped, kao savetnici u domenskoj oblasti. Dizajn mehanike igara se odvijao iterativno, uz stalno savetovanje sa uključenim medicinskim osobljem. Iglad početnog menija **Mozaik-a** se može videti na slici 1.



Slika 1. Početni meni **Mozaik-a**

Mozaik je uključivao igru „Pronađi korpu”, gde je cilj igrača navesti sliku da padne u korpu sa imenom pojma sa slike. Podešena težina igre utiče na brzinu padanja slike. Pokrenuta igra se može videti na slici 2. Sledeća igra je „Pogodi reč”, gde korisnik treba da pogodi sva slova koja se nalaze u imenu pojma sa slike. Težina igre utiče na broj azbučnih slova koja su ponuđena korisniku.



Slika 2. Pokrenuta igra „Pronađi korpu”

Igra „Sastavi rečenicu” traži od korisnika da sastavi rečenicu koja verno opisuje situaciju sa prikazane slike. Težina igre utiče na broj ponuđenih reči, koje korisnik može iskoristiti u rečenici. Potom sledi igra „Uхвати reč”, gde korisnik mora da iz pokretne trake sa rečima, uhvati naziv pojma sa date slike i potom je prevuče na sliku tehnikom drag&drop. Težina igre utiče na brzinu proticanja reči na pokretnoj traci. Igra „Memorija” traži da korisnik zapamti spisak pojmova, a potom mu se

postavljaju pitanja da li je neki pojam bio na spisku ili ne. Težina igre utiče na broj pojmova koji budu prikazani korisniku. Sledeća igra je „Pojmovi”, koja ima za cilj da korisniku omogući vežbanje izgovora. Istovremeno su prikazani slika i naziv pojma, kao i dugme koje pušta glasovni zapis izgovaranja naziva pojma. Izabrana težina nema uticaj na ovu igru. Izgled pokrenute igre se može videti na slici 3.



Slika 3. Pokrenuta igra „Pojmovi”

Igra „Nađi kategoriju”, ima za cilj da korisnik prepozna kategoriju kojoj pripada pojam sa slike. Izabrana težina nema uticaj na ovu igru. Igra „Pogodi sliku”, ima za cilj da korisnik prepozna sliku koja pripada datoj kategoriji. Izabrana težina nema uticaj na ovu igru. Igre nemaju poseban redosled kojim se igraju, ali lekar propisuje koje igre i koliko dugo treba da se igraju, na osnovu stepena oštećenja određenih funkcija. Testiranje je zbog obimnosti, ostavljeno za buduće istraživanje.

4. ZAKLJUČAK

Tokom izrade ovog rada, kreiran je omnibus igara, nazvana **Mozaik**. U njegovom sastavu nalazi se devet manjih igara, koje ciljaju različite poremećaje kod posledica apopleksije ili šloga. Jedne ciljaju kognitivne sposobnosti, druge delikatne pokrete ruku, dok treće ciljaju oboje. Namera iza kreiranja **Mozaik-a**, bila je da se pomogne pacijentima koji su pretpeli šlog, a kao posledicu između ostalog imaju poremećaj govora i prepoznavanja pojmova, poznat kao afazija. Takvi pacijenti uz **Mozaik** imaju mogućnost da između poseta lekaru samostalno vežbaju izgovaranje pojmova, kao i njihovo prepoznavanje. To im omogućava vežbanje u sopstvenom tempu, bez osećaja stida, ili nametanja svojoj okolini. Pacijent može da vežba svakodnevno, u trajanju u kom se on oseća prijatno i može da zaustavi ili potpuno prekine igru u bilo kom trenutku. Dugotrajno vežbanje, gde korisnik ima želju za oporavkom, pa svakodnevno vežba, pomaže da se učvrste odgovarajuće neuronske mreže i da se pojmovi ispravno prepoznaju. **Mozaik** će pomoći svojim korisnicima da brže razmišljaju, brže reaguju, uvežbaju finu motoriku prstiju i ruku, ali će im pomoći i da vežbaju svoju memoriju i prepoznavanje. Pomoću dijaloga za podešavanje opcija, težina igre može da se prilagodi korisniku i njegovim trenutnim sposobnostima. Prvi nivo težine je skoro potpuno statičan, pa je prigodan za osobe koje tek započinju kognitivnu rehabilitaciju. **Mozaik** se za sada fokusira samo na kognitivnu rehabilitaciju i vežbanje finih pokreta ruku, ali bi mogao da se dodatno unapredi na više načina.

Za korisnike bi najbolje bilo da kao pomoć imaju video zapis osobe koja izgovara dati pojam, fokusiran na usne i oblikovanje slova. Korisnikovo glasovno ponavljanje za audio snimkom bi bilo poželjno snimati, kako bi lekar mogao proceniti da li je korisnik dobro izgovorio dati pojam.

Predlog bi bio da se u neku od igara ubaci i dugme za snimanje glasa, pa da korisnik kada je spreman može sam ili uz pomoć okoline snimiti svoj izgovor. Te snimke lekar bi mogao kasnije da reprodukuje i oceni da li je bilo napredovanja između samih snimaka, ili između snimaka i terapije u njegovom prisustvu.

Mozaike se može unaprediti i tako da podrži fizičku rehabilitaciju, gde se na ekranu demonstriraju vežbe koje korisnik treba da reprodukuje. Integracija aplikacije sa konzolama za video igre poput Microsoft Kinect i Wii, omogućila bi prepoznavanje pokreta korisnika, a samim tim i utvrđivanje tačnosti izvedene vežbe. Doktor bi na osnovu aplikacije imao uvid u statistiku, broj ponavljanja i vrstu izvedenih vežbi, kao i spisak onih vežbi koje su predstavljale problem za korisnika, a koje vežbe su mu bile lagane za izvođenje.

Za igranje dinamičkih igara, za korisnika sa težim stepenom poremećaja pokreta, bilo bi lakše da se postave veliki tasteri – jedan crveni i drugi zeleni, a koji bi vršila ulogu kao unapred određena dugmad na tastaturi.

Za samog lekara koji nadgleda terapiju, objedinjena statistika za sve pacijente, dobijena na osnovu ove igre, znatno bi olakšala posao. Selekcijom pacijenta bi se mogli dobiti podaci o tome koliko je vremena izabrani pacijent proveo koristeći aplikaciju, kolika je njegova uspešnost u igranju, u kojem vremenu je prelazio određene pojmove, a koji su mu bili najteži itd. Lekar bi imao kontrolu nad težinama svake igre, pa bi na osnovu izveštaja mogao da prilagodi težinu određenih igara.

Kreirana aplikacija je veliki korak ka pružanju pomoći osobama kojima je ona najpotrebnija. Pokazano je putem istraživanja da igranje zabavnih igara donosi poboljšanje u bržem donošenju odluka, bržem reagovanju na iznenadne situacije, bržem prihvatanju informacija iz okoline, boljoj memoriji, boljoj koordinaciji pokreta okoruka, većem stepenu samopouzdanja i osećaju dostignuća, posebno kada se nakon dužeg vremena napokon pređe problematičan nivo.

Računarske igre pomažu i u oslobađanju od stresa, jer dozvoljavaju malu distrakciju od stvarnosti, kada se život čini težak. Ipak, treba biti umeren i u igranju računarskih igara. Istraživanja [8] su pokazala da igranje do 21 sat nedeljno, može doneti korist po zdravlje, dok prekoračenje ove granice može da škodi.

Zanimljivo je da vojska danas koristi računarske igre za lečenje posttraumatskog sindroma kod ratnih veterana.

Takođe, igre se uveliko koriste kao distrakcija tokom hemoterapije, ali i za prevazilaženje kliničke depresije. Naravno, koriste se i u sistemu obrazovanja, radi lakšeg učenja gradiva. Njihova prisutnost je sve veća, jer ne samo da su korisne, igre imaju mogućnost da probude pozitivne emocije i optimizam.

Računarske igre pobuđuju radoznalost i avanturistički duh u toku igranja, a ponos po njenom završetku. Igre podstiču i na udruživanje sa drugima, kako bi mogli zajedničkim snagama da prevaziđu prepreke i ostvare izuzetne ciljeve, koje bi bilo skoro nemoguće ostvariti u

realnom životu, poput spašavanja čitave planete od apokalipse [8]. Računarske igre nas motivišu, inspirišu i leče, pa ih ne treba olako shvatiti.

5. LITERATURA

- [1] S. N. Merry, K. Stasiak, M. Shepherd, C. Frampton, T. Fleming and M. F. G. Lucassen, „The effectiveness of SPARX, a computerised self help intervention for adolescents seeking help for depression: randomised controlled non-inferiority trial“, *BMJ*, vol. 344, e2598, April 2012.
- [2] Entertainment Software Association, „Essential facts about the computer and video game industry“, <http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf>, Stranica sa izveštajem o stanju u gejmu industriji.
- [3] D. Djaouti, J. Alvarez and J. P. Jessel, „Classifying serious games: the G/P/S model“, P. Felicia „Handbook of research on improving learning and motivation through educational game“, ch. 6, pp. 118—136, April 2011.
- [4] W. H. Redd, P. B. Jacobsen, M. Die-Trill, H. Dermatis, M. McEvoy and J. C. Holland, „Cognitive/attentional distraction in the control of conditioned nausea in pediatric cancer patients receiving chemotherapy“, *Journal of Consulting and Clinical Psychology*, vol. 55(3), pp. 391-395, June 1987.
- [5] TactusTherapy, „4 Aphasia Apps in 1: Language Therapy“, <http://tactustherapy.com/app/language/>, Aplikacija za afaziju.
- [6] B. C. Stark and E. A. Warburton, „Improved language in chronic aphasia after self-delivered iPad speech therapy“, *Neuropsychological rehabilitation*, February 2016.
- [7] B. Buss, „Virtual Reality Training System for Patients with Dementia“, *ETH Swiss Federal Institute of Technology and Institute of Neuroinformatics*, September 2009.
- [8] J. McGonigal, *Reality is Broken: Why Games Make Us Better and How They Can Change the World*, New York: Penguin Press, 2011.

Kratka biografija:



Jelica Kapetina rođena je u Ilidži (Sarajevo, BiH) 1992. god. Diplomski-master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke odbranila je 2017.god.

RAZVOJ RAČUNARSKE IGRE ZA STICANJE OSNOVNIH PROGRAMSKIH VEŠTINA DEVELOPMENT OF A COMPUTER GAME FOR ACQUIRING BASIC PROGRAMMING SKILLS

Aleksandar Josić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad sadrži dizajn kao i uvid u tehničku implementaciju dve edukativne video igre čiji je cilj naučiti igrača osnovnim upravljačkim strukturama programiranja. Prikazan je teorijski osnov psiholoških kognitivnih teorija učenja na koje se one oslanjaju, a zatim pregled nekih od već postojećih igara koje se istim problemom bave. Objašnjene su neke od naprednih tehnika razvoja video igara koje se koriste kako bi se postigao efikasan rad aplikacije neophodan za ugodno korisničko iskustvo. Ove tehnike su objašnjene koristeći Unity3D pogon za video igre.*

Abstract – *This paper contains a design as well as insight in some of the technical aspects of developing two video games which have a purpose to teach basic programming structures. It has an overview of the psychology theories upon which they are made, as well as examples of already built games. Some of the more advanced techniques of video game development aimed at mobile platform optimizations are described in greater detail. These techniques are implemented in the Unity3D game engine.*

Cljučne reči: *edukativne tehnologije, razvoj video igara, vizualni programski jezici, mobilne aplikacije*

1. UVOD

Edukativne video igre su se pokazale kao veoma moćan alat u školskim učionicama širom sveta. Pored mogućnosti da motivišu učenike i podignu im nivo koncentracije, naučeni sadržaj deca duže pamte i kasnije lakše primenjuju u praksi. Ovaj rad pruža uvid u neke od kognitivnih teorija učenja na koje se takav tip igara oslanjaju.

Opisan je Okvir za ocenu dizajna ozbiljnih igara, koji se koristi kako bi se igra raščlanila na svoje osnovne elemente koji se zatim usklađuju sa osnovnim ciljem igre. Predstavljeni su neka od multimedijalnih okruženja koja se koriste prilikom podučavanja programiranja, kao i neke od već postojećih igara na tu temu.

Detaljno je opisan dizajn dve edukativne igre čiji je cilj da uče osnovnim kontrolnim strukturama programiranja. Alat koji se koristio za samo tehničku implementaciju je Unity3D pogon za pravljenje video igara, i on je takođe opisan u radu. Jedan od najvećih problema kod razvijanja mobilnih igara jesu performanse na uređajima koji su slabijih hardverskih mogućnosti.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red.prof.

Stoga su detaljno objašnjene neke od naprednih tehnika razvoja video igara koje su se koristile prilikom optimizacije igre za mobilne platforme.

2. UČENJE POMOĆU RAČUNARA

Učenje se može definisati kao proces koji spaja lična iskustva kao i uticaje iz sredine, u cilju sticanja i obogaćivanja znanja, veština, vrednosti, stavova, ponašanja ili pogleda na svet.

Uvid u osnovne teorije koje se bave proučavanjem kognitivnih procesa koji se odvijaju prilikom usvajanja novih znanja mogu značajno pomoći prilikom kreiranja edukativnih aplikacija.

Tako konstrukcionizam, teorija najznačajnija za ovaj rad, nalaže da projektovanje naših unutarnjih osećanja i ideja u javnost dovodi do njihovog daljeg oblikovanja i izoštravanja pa samim tim i poboljšavanja kroz iteraciju. Široka rasprostranjenost digitalnih tehnologija, sa naglaskom na mobilne platforme koje imaju mogućnost povezivanja na internet, omogućuje deljenje ovih ideja među učenicima, kao i predavačima. Ove mogućnosti dovele su do razvitka širokog spektra platformi koje se koriste u svrhu obrazovanja.

Edukativne igre, podvrsta ozbiljnih igara, kao svoj osnovni zadatak imaju sticanje nekog znanja, odnosno veštine. Njihova funkcija jeste naučiti korisnika kako da koristi alate za učenje, da pristupa radu aktivno, te se uči samostalnom, kao i timskom radu. Najistaknutija pozitivna svojstva učenja putem ovog alata jesu gotovo automatski fidbek, posedovanje kontrole od strane učenika nad sopstvenim učenjem, te decentralizacija rukovodeće uloge nastavnika u procesu učenja. Istraživanje iz [1] ukazuje na još neke od benefita primene igara u obrazovanju.

Međutim, tehnologija i sam akt njene primene u učionici nikako ne bi trebao da bude primarni cilj edukatora, a pogotovo ne cilj sam za sebe. Današnji alati, iako mnogo kompleksniji i moćniji u tehničkom pogledu, i dalje postoje samo zato da bi pružili podršku samom gradivu i trebaju biti potpuno podređeni efikasnosti i kvalitetu učenja.

3. OSNOV RAZVOJA VIDEO IGRE KOJA PODUČAVA OSNOVNIM KONTROLNIM STRUKTURAMA PROGRAMIRANJA

Ovo poglavlje prvo definiše koje su to osnovne kontrolne strukture programiranja, a zatim način na koji se one mogu predstaviti kroz vizualne programske jezike. Opisan je Okvir za procenu dizajna ozbiljnih igara i kroz njega su ocenjene neke od već postojećih igara koje su

kreirane u svrhu podučavanja osnovnih kontrolnih struktura programiranja.

3.1 Böhm-Jacopini teorema

Ova teorema, poznatija pod nazivom Teorema strukturiranog programa, nalaže da klasa grafova kontrole toka može da izračuna bilo koju funkciju podložnu računanju, ukoliko kombinuje potprograme u samo tri specifična načina, tj kontrolne strukture:

1. Sekvenca – izvršavanje jednog potprograma, pa zatim narednog potprograma
2. Selekcija – izvršavanje jednog od dva moguća potprograma, gde logički izraz definiše koji će se potprogram izvršiti.
3. Iteracija – izvršavanje potprograma sve dok je neki logički izraz tačan

Navedene strukture su osnovni sadržaj najvećeg dela edukativnih igara koje su opisane u ovom radu.

3.2 Vizualni programski jezici

U računarstvu postoji čitav niz programskih jezika koji koriste vizualne izraze, spacijalnu organizaciju teksta i grafičkih simbola da bi predstavili sintaksu samog jezika, kao i sekundarnu notaciju programa. Jedna od najvećih prednosti ovakvih tipova jezika je što su jednostavni za naučiti, budući da same karakteristike vizualnih elemenata mogu navoditi korisnika na korektnu sintaksu, dok kod tekstualne manipulacije izvornim kodom programa to nije slučaj. Stoga su svoju primenu pronašli najviše upravo u obučavanju novonastalih programera.

3.3 SGDA, Okvir za procenu dizajna ozbiljnih igara

Kroz ovaj okvir okvir pokušava se sagledati celokupan formalni konceptualni dizajn igre, njeni konkretni elementi, kao i njihov međusobni odnos. Definiše se šest bazičnih elemenata dizajna igre, i proučava povezanost između njih, kao i mera u kojoj su ti elementi poravnati sa ciljem same igre. Svaki poseban element dizajna mora da bude veoma dobro usklađen sa ciljem igre, a isto tako mora da se slaže sa svim ostalim elementima. Ovi elementi su :

1. Cilj tj svrha same igre - Naspram igara koje se primarno dizajniraju i koriste u zabavne svrhe, gde je sam cilj upravo aktivirati i zabaviti igrača kroz iskustvo koje proživljava igrajući, edukativne igre su eksplicitno dizajnirane sa ciljem koji je van same igre. Ovaj dizajn element je i najbitniji.
2. Sadržaj – Pod ovim elementom se misli na ceokupan material koji se nalazi unutar igre i na neki način je prezentovan igraču.
3. Mehanike – Element koji definiše metode pozvane od strane agenata kako bi interagovali sa svetom igre.
4. Fikcija/narativ – Ovaj element daje igri njen fikcionalni kontekst.
5. Estetika i grafika - Ovaj element se odnosi na audiovizualni jezik(estetske karakteristike, slikovito izlaganje, odabir stila, umetničke medije, tehnike računarske grafike poput tipa iscrtavanja koje se koristi) konceptualizovan, odabran i razvijen od strane dizajnera za

vizualizaciju, kao i prikaz elemenata uključenih u igre.

6. Okvir – Element koji se odnosi na uokviravanje postojećih elemenata u vidu ciljne grupe igrača.

Svaki od ovih elemenata neophodno je dobro ispitati prilikom dizajna jedne edukativne video igre. Kroz njega su opisane neke od glavnih karakteristika već postojećih igara koje uče programiranju.

3.4 LightBot

Ovo je video igra sa jedinstvenim i prepoznatljivim stilom, element koji je jedan od glavnih razloga njene široke rasprostranjenosti. Mehanike igre verno oslikavaju process programiranja koristeći vizualni programski jezik. Element na koje se najviše može dodatno raditi jeste sadržaj. Iako kreće sa jednostavnim konceptom sekvence, pa prelazi na concept pozivanja potprograma, teži nivoi zadaju poteškoće velikom broju učenika. Koncept iteracije se uvodi prvenstveno i jedino kroz primenu rekurzije koja ne sadrži eksplicitan uslov zaustavljanja definisan od strane korisnika. Jedini način da se iz funkcije izade jeste da se kroz njeno korišćenje pređe ceo nivo. Ovo je potencijalno jedan od razloga zbog kojeg učenici imaju problem prilikom nailaska na teže nivoe. Element fikcije i narativa je zapostavljen od strane dizajnera. Generalno, ova igra veću vrednost ima kao zabavna, budući da se sam sadržaj veoma teško može verodostojno oslikati na kontekst van same igre.

3.4 Code.org

Ova platforma sadrži mnoštvo edukativnih kurseva namenjenih raznim uzrastima učenika u okviru jednog internet sajta. Kroz kurseve se učenicima u vidu vežbi omogućava da kroz prevlačenje kontrola jednog vizualnog programskog jezika, upravljaju raznim karakteristikama i na taj način prelaze nivoe. Glavne pozitivne karakteristike ove platform su lakoća njene upotrebe, mogućnost automatskog praćenja napretka učenika, kao i veoma kvalitetno osmišljen sadržaj koji je u skladu sa zvaničnim školskim kurikulumom. Mehanike samih igara takođe veoma verno oslikavaju process programiranja koristeći vizualni programski jezik. Element na kojem bi se najviše moglo dodatno poraditi jeste njen fikcionalni karakter. Igrač kroz virtualne svetove upravlja likovima koji su praktično preuzeti iz već postojećih i veoma uspešnih video igara. Tako ova aplikacija praktično svoj fikcionalni karakter ni nema, već pokušava asociirati na narativ iz nekih drugih aplikacija. Generalno, ova platform ima veoma kvalitetnu osnovu u smislu sadržaja koji pokušava da presene igraču, kao i empirijski dokazane mehanike u igrama koje su dostupne u vidu vežbi. Kako bi svoj potencijal iskoristila maksimalno, potrebno je dodatnu pažnju posvetiti kreiranju teme koja bi bila ista u svim igrama, kao i jedinstvena za celu platformu.

3. DIZAJN I IMPLEMETACIJA POSTMAN I ROBBIE APLIKACIJA

Dve teme su se pokazale kao interesantne za dizajn igre koja uči osnovnim konceptima programiranja. Od obe je nastao po jedan projekat i za vreme pisanja ovog rada, oba su u fazi implementacije. Ovo poglavlje sadrži opis dizajna obe aplikacije, kao i neke od tehničkih detalja

koji su se pokazali kao značajni u određenim fazama razvoja.

3.1 Robbie

Robbie je dečak koji se nakon napornog dana u školi vraća kući da igra video igre na svom računaru. Ubrzo saznaje da je računar zaražen virusom, koji je glavni negativac u igri. Robbie zatim ulazi u računar i zadatak mu je da ga očisti od virusa, igrajući njegovu igru. Nivoi se prelaze usvajajući osnovne kontrolne strukture programiranja definisane Teoremom strukturiranog programa, kao i upoznavanjem sa nekim od osnovnih koncepata računarstva generalno. Tako Robbie uči da piše i čita sa hard diska, koristi RAM memoriju za skladištenje podataka, uči da je indeksira kako bi te podatke zatim pročitao itd.

Osnovni dizajn elementi su kreirani imajući na umu Okvir za procenu dizajna ozbiljnih igara. Rezultat toga je da je narativ usko povezan sa sadržajem same igre, što većina igara koji se ovim problemom bave nema. Mehanike se sastoje iz prevlačenja kontrola vizualnog programskog jezika, a ciljna grupa još ostaje da se utvrdi kroz testiranje.

3.2 PostMan

Narativ ove aplikacije nije naročito kompleksan, budući da ciljna publika za koju je zamišljena, deca od 3 do 5 godina, nisu u mogućnosti da ga razumeju. Glavni lik je poštar kojem je zadatak da raznosi pakete po gradu. Kako bi to mogao da uradi, mora imati precizno definisanu putanju kojom se kreće kroz grad, a zadavanje te putanje je posao za korisnika igre. Sam izgled igre je prikazan na Slici 1.



Slika 1. Izgled ekrana PostMan aplikacije

Koncept koji se odmah uvodi jeste sekvenca, budući da je potrebno definisati precizan redosled kojim auto treba da skreće.

Ovo je jedina struktura iz osnovnih struktura Teoreme strukturiranog programa koja je implementirana u igri. Pored nje, uvodi se koncept uzročnopsledične veze na više načina. Tako igrač mora da na nekim nivoima pokupi ključ kako bi prešao preko mosta.

Igra po svom sadržaju nije veoma bogata, odluka koja je doneta zbog mlade publike za koju je namenjena. Kontrole se prevlače kao i u svakoj igri do sada prodiskutovanoj.

3.3 Sistem za smenu scena u aplikaciji

Prilikom smene scena unutar bilo koje aplikacije, dolazi do pristupa tim scenama koje se inicijalno nalaze na dugotrajnoj memoriji računara. Kako su ovi uređaji uvek mnogo sporiji od radne memorije, pristup njima uvek zahteva određeno vreme. Video igre se trude ovaj pristup zamaskirati na različite načine. Tako je za potrebe PostMan aplikacije razvijeno rešenje koje se ovim problemom bavi. Kada se detektuje smena scene, prvo se poziva animacija prelaza u narednu scenu koja naliči "spuštanju zavesa". Jednom kada je animacija izvršena, otpočinje asinhrono učitavanje scene, koje dovodi do pada frekvencije iscrtavanja. Budući da korisnik sada vidi samo prelazni ekran, ovaj pad mu nije uočljiv. Kada se scena u pozadini učita, animacija otkrivanja scene se poziva i nova scena je vidljiva i spremna za korišćenje.

3.4 Optimizacija iscrtavanja za mobilne platforme

Neophodno je omogućiti ugodno korisničko iskustvo na širokom spektru hardverskih mogućnosti koje današnji mobilni uređaji imaju. Kako bi se ovo uradilo, potrebno je implementirati igru na način koji uzima u obzir mehanizam preko kojeg alati visokog nivoa poput Unity3D pogona, zapravo iscrtavaju sadržaj na ekran.

Ukratko, glavno zaduženje centralne procesorske jedinice (eng. CPU) jeste da iz scene koja se prikazuje, odredi objekte koji će se iscrtati na ekranu, a zatim te objekte pretvori u naredbe za iscrtavanje koje prosleđuje grafičkoj procesorskoj jedinici (eng. GPU). Dve su glavne vrste naredbi koje mogu biti poslate:

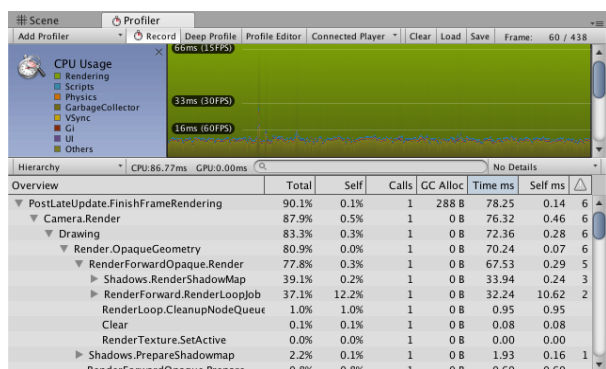
- Promena stanja - naredba koja se šalje u situacijama kada je potrebno npr promeniti teksturu koja se trenutno iscrtava
- Iscrtavanje geometrije - Ovde se šalje paket konkretne geometrije koju je potrebno iscrtati po trenutno aktuelnom stanju

GPU po prijemu ovih naredbi ima zadatak da ih izvrši. Ukoliko bilo koja strana, CPU ili GPU, ne uspeva brzo da izvrši svoj deo zadatka, rezultat je manja frekvencija iscrtavanja. Korisnici su jako osetljivi na frekvenciju kojom se ekran osvežava i da bi imali ugodno iskustvo potrebno je osvežiti im ekran barem 30 puta u sekundi. Pored apsolutnog broja, takođe je potrebno ovu frekvenciju stabilizovati.

Alat koji se koristio za merenje performansi same igre je Unity-ev Profiler prozor, namenjen nadgledanju alokacije hardverskih resursa. Nakon pokretanja ovog alata, PostMan aplikacija je na iPhone 4s uređaju osvežavala ekran samo 20 puta u sekundi. Kako bi se ovaj broj povećao, potrebno je optimizovati proces iscrtavanja. Generalno, optimizacija se svodi na smanjenje opterećenja na strani koja diktira frekvenciju iscrtavanja (CPU ili GPU).

Nakon pregleda grafikona sa Profiler prozora prikazanog na Slici 2, dolazi se do zaključka da je u slučaju ove aplikacije više opterećen procesor. Naredba koja mu oduzima 90.3% vremena za iscrtavanje jednog prikaza igre ima naziv PostLateUpdate.FinishFrameRendering

što znači da procesorska strana podele posla iscrtavanja mora da se optimizuje.



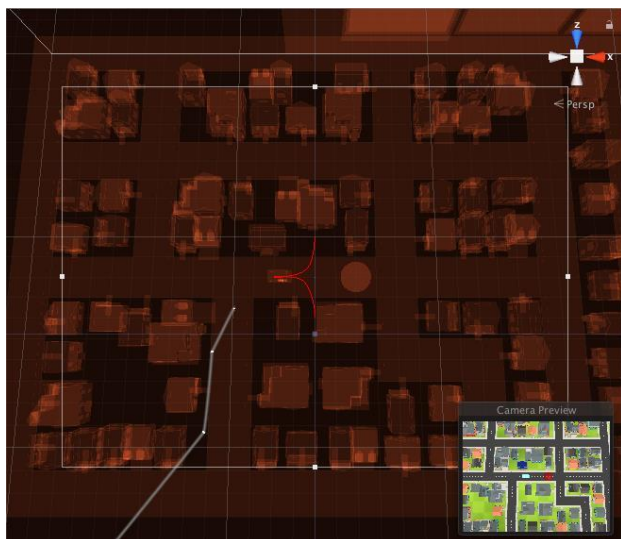
Slika 2. Profiler prozor

Kako bi se podigle performanse, jedan od koraka jeste uklanjanje objekata koji se iscrtavaju na sceni.

Unity3D pogon automatski vrši zanemarivanje objekata koji se ne nalaze unutar vidnog polja kamere(eng. Frustum Culling).

Tehnika koja se takođe može ubrojati u smanjivanje broja iscrtanih objekata jeste zanemarivanje elemenata koji su već sakriveni od kamere sa nekim drugim objektima(eng. Occlusion Culling). Ova tehnika, iako veoma efektivna u nekim slučajevima, ovde čak dovodi do lošijih performansi.

Razlog za to je što se najveći deo same igre odvija iz ptičije perspektive, ugao iz kojeg kada se scena posmatra, ne postoji puno preklapanja objekata. Prikaz preklapanja može se videti na Slici 3, svetliji delovi slike označavaju više preklapanja.



Slika 3. Prikaz preklapanja geometrije

Kako ta tehnika sada mora za svaki objekat proveriti da li je zaklonjen, i za svaki dobija rešenje da nije, krajnji rezultat je lošiji performans.

3.5 Grupisanje (eng. batching)

Ova tehnika nastoji da spoji što više postojećih objekata unutar jednog istog paketa geometrije koji se šalje grafičkom čipu. Umanjuje se količina komunikacije između procesora i grafičke kartice, što drastično podiže

performanse. Najskuplji pozivi jesu pozivi promene stanja. Kako bi se njihov broj umanjio, potrebno je objekte kreirati tako da mogu da se iscrtaju sa istim stanjem grafičke, tj da se iscrtavaju sa istim materijalom. Zatim, budući da svi ti podaci koriste isto stanje za iscrtavanje, nije potrebno slati dodatne pozive promene stanja koji zapravo škode performansama, već se celokupno iscrtavanje radi sa jednim pozivom za promenu stanja i nizom jednostavnijih poziva za iscrtavanje.

U slučaju PostMan aplikacije, nakon njene implementacije dolazi do drastičnog skoka u efikasnosti izvršavanja igre. Frekvencija osvežavanja ekrana dostiže vrednost od prosečno 40 osvežavanja u sekundi, što je dovoljno za ugodno korisničko iskustvo.

4. ZAKLJUČAK

Kroz osnovne kognitivne teorije učenja objašnjeno je zašto ljudi imaju benefit od primene računara u obrazovanju. Video igre su se u ovom polju pokazale kao izuzetno korisne. Prikazan je Okvir za dizajn ozbiljnih igara i kroz njega su opisane dve igre koje se bave podučavanjem programiranja. Takođe je prikazan dizajn, kao i tehnička implementacija igre PostMan.

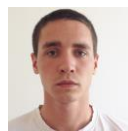
Dok occlusion culling nije doneo značajne rezultate u smislu podizanja performansi, primena grupisanja objekata je rezultovala u drastičnom skoku frekvencije osvežavanja ekrana.

Finalni rezultat rada su aplikacije koje su ukorenjene u kognitivnim teorijama učenja, dizajnirane kako bi što bolje podučavale programiranje na zanimljiviji i efikasniji način, a implementirane na način koji omogućava ugodno korisničko iskustvo i na uređajima slabijih hardverskih mogućnosti.

5. LITERATURA

- [1] Jelena Popović, "Motivisanost nastavnika za primenu video igara u nastavi", master rad na Filozofskom fakultetu u Novom Sadu

Kratka biografija:



Aleksandar Josić rođen je u Somboru 1992. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarske nauke i informatika, odbranio je 2015.god.

ŽIVOTNI CIKLUS MODELA PODATAKA ELEKTROENERGETSKOG SISTEMA

LIFECYCLE OF ELECTRIC POWER SYSTEM DATA MODEL

Marko Razumenić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu je predstavljen proces održavanja modela podataka elektroenergetске mreže u odnosu na arhitekturu elektroenergetskog sistema (EES). Na osnovu analize, utvrđeni su formati za: predstavljanje životnog ciklusa i čuvanje životnog ciklusa izmene modela podataka elektroenergetskog sistema. Implementirano je programsko rešenje za grafičko predstavljanje životnog ciklusa modela mreže, kao i generisanje odgovarajućih SQL skripti koje definišu stanja modela.

Abstract – Architecture of the electric power system, and the process of maintaining the power grid data model is presented in this paper. Based on the results of the analysis, formats were defined for representing and storing the lifecycle of the electric power system data model changes. A software solution for graphical representation of the lifecycle of a network model has been implemented, as well as the generation of appropriate SQL scripts that define the state of the model.

Ključne reči: Model elektroenergetskog sistema, životni ciklus, CIM (Common Information Model), SQL (Structured Query Language)

1. UVOD

Elektroenergetski sistemi (EES) predstavljaju jedan od ključnih nosioca razvoja i napretka jedne zemlje, zato njihov razvoj i održavanje čine jedan od važnijih zadataka današnjice. Usavršavanje EES se u poslednje vreme, velikim delom, svodi na poboljšanje softverskog dela sistema u cilju efikasnijeg upravljanja električnom energijom. Pred proizvođačima električne energije se nameću sve veći zahtevi u potrebnoj količini i načinu proizvodnje - zahtevi za električnom energijom su u porastu kao i težnja za očuvanjem prirode i usled toga težnja za iskorišćavanjem obnovljivih resursa kao što su voda, vetar i sunčeva energija [1]. Svaki model elektroenergetskog sistema – model mreže, s obzirom na svoje namene je drugačiji. Stoga razvoj modula je primarno orijentisan za određen sistem i prilagođen njegovim specifičnostima. Ovaj rad se bavi problemima kreiranja i predstavljanja životnog ciklusa izmena modela podataka elektroenergetskog sistema. Implementirano je programsko rešenje za grafičko predstavljanje životnog ciklusa izmena modela mreže i mapiranje podataka o definisanim stanjima modela na SQL skriptu - transformacija podataka iz oblika modela (dijagram stanja) u izlazni oblik tj. SQL skriptu.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Milan Gavrić.

2. MODEL PODATAKA

ELEKTROENERGETSKOG SISTEMA

Model mreže (eng. *Network Model*) ili statički model - u literaturi se još naziva i model izgrađenosti mreže (eng. *As-Built Network Model*) obuhvata elemente EES od izvora energije, prenosne mreže, distributivne mreže i predstavlja kolekciju podataka koji potpuno opisuju fizički deo mrežne opreme i način na koji je oprema povezana u EES. To je objektno orijentisani model, dokumentovan UML-om, podeljen na logičke pakete, zasnovan na CIM (eng. *Common Information Model*) seriji standarda IEC 61970 (eng. *International Electrotechnical Commission*) kojim se predstavljaju i balansirane i nebalansirane elektroenergetске mreže. Statički model je osnova u procesu izvođenja studija analize mreže (eng. *Network Analysis Data Management*), izvršavanja proračuna i vizuelizacije DMS/EMS/OMS aplikacija, topološke analize mreže (eng. *Network Topology Analysis*) i formiranja normalnog uklopnog stanja mreže [2]. U elektroenergetskim kompanijama dve aktivnosti se izdvajaju u radu nad mrežnim modelom:

- upravljanje: sve akcije nad mrežom u vremenskom domenu dok se energija isporučuje potrošačima i
- planiranje: sve inženjerske aktivnosti nad mrežom u cilju postizanja efikasnog upravljanja [2].

Fizički model se sastoji od dva glavna dela:

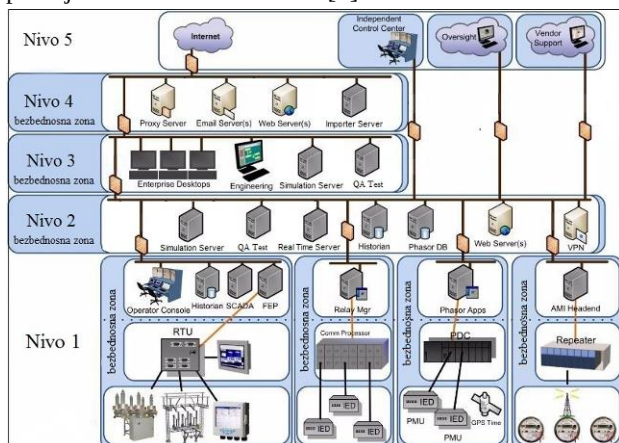
- fizičke komponente (elektroenergetski resursi sistema) i njihova svojstva.
- električna konektivnost komponenti (kako su komponente u mreži međusobno povezane – izgrađenost mreže) [2].

3. OPIS TIPIČNE ARHITEKTURE SISTEMA

Elektroenergetski softverski sistemi, pripadaju klasi sistema koji se nazivaju sigurnosno-kritični sistemi (*safety critical systems*). Kod ove klase sistema nije dovoljno da svi podsistemi u sistemu funkcionišu besprekorno svaki za sebe, jer greške mogu biti veoma skupe (ne samo u materijalnom smislu; ove greške mogu dovesti i do gubitka ljudskih života), već je i neophodno da sva interakcija (uključujući i razmenu podataka) između podsistema bude ispravna [3] obezbeđujući na taj način korektan rad kompletnog sistema.

U prošlosti elektroenergetski sistemi funkcionisali su na nestandardnim protokolima u izolovanim mrežama [4]. Dok je ovakav pristup eliminisao brojne potencijalne pretnje, takođe je ograničavao pristup proizvedenim informacijama za druge zainteresovane strane u sistemu. Moderni elektroenergetski sistemi pružaju nove industrijske standarde i dizajn otvorenih sistema (eng. *open-system design*) omogućavajući vezu sa ostalim delovima mreže celokupnog preduzeća (eng. *corporate network*) koji je povezan na Internet. NERC (eng. *North*

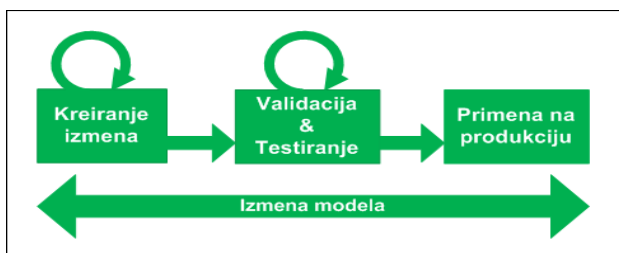
American Electric Reliability Corporation) je publikovao standard: CIP (Critical Infrastructure Protection) koji je preporuka elektroenergetskim preduzećima širom sveta kako bi moderni SCADA, OMS, EMS, DMS trebalo da u svojim operacijama zadovolje minimalne zahteve sigurnosti i bezbednosti. CIP standardizacija ne propisuje tehnologiju kojom postići bezbednosne ciljeve - ona jednostavno definiše ciljeve, ali ne i način kako doći do njih [5]. Današnja moderna ideja u postizanju cilja - siguran i bezbedan elektroenergetski sistem je zasnovana na konceptu dubinske odbrane uvođenjem višeslojne zaštite. Moderna arhitektura softverskog EES prikazana na slici 1. se zasniva na arhitekturi sistema koji je podeljen na zone bezbednosti [6].



Slika 1. Moderna arhitektura softverskog EES [6]

Bezbednosna zona je skup informacionih sistema povezanih u jednu ili više unutrašnjih mreža pod jedinstvenom upravom, jedinstvenom bezbednosnom regulativom i lokacijski je nezavisna logička celina; logička struktura bezbednosne zone ne mora da se poklapa sa fizičkom strukturom [7], npr. inženjerski departman i departman za menadžment mogu biti locirani na istom fizičkom mestu (u istoj zgradi) ili u različitim (objektima), ali funkcionalno oni pripadaju istoj bezbednosnoj zoni. Bezbednosna zona mora posedovati jasno definisane granice i komunikacija sa ostalim zonama mora biti filtrirana u skladu sa bezbednosnim pravilima. Posledica navedene organizacije sistema uvodi novu vrstu bezbednosne zone: DMZ (eng. Demilitarized Zone) slika 1, Nivo 2. Demilitarizovana zona DMZ predstavlja mrežni segment koji se logički nalazi između dve mreže: spoljašnje i unutrašnje i igra ulogu zaštitnog oklopa unutrašnje mreže u komunikaciji sa spoljašnjom mrežom. DMZ je filter za eksterne neproverene izvore informacija u smeru ka internoj mreži, dok u isto vreme ograničava pristup samo onim podacima koji treba da su dostupni u obrnutom smeru komunikacije [7].

4. OPIS TIPIČNOG BIZNIS PROCESA



Slika 2. Redosled operacija pri izmeni modela mreže

Proces izmene modela mreže se sastoji iz tri glavna koraka u kojima se promene modela unose u sistem, validiraju, testiraju, i na kraju primenjuju na model elektroenergetske mreže u produkcionom okruženju, prikazan je slici 2.

Prvi korak procesa izmene model elektroenergetske mreže je kreiranje skupa izmena, tj. *Changeset*-a koji može biti kreiran EMS, DMS, OMS aplikacijama za održavanje elektroenergetske mreže ili uvoženjem podataka iz spoljašnjeg informacionog sistema.

Validacija i testiranje je naredna faza u procesu izmene modela u kojoj svaki *Changeset* prolazi kroz sopstveni ciklus validacije i testiranja. Mogu se razlikovati dva tipa validacije prema sadržaju koji se validira:

- osnovna - obavezna validacija koja se odnosi na validaciju statičkog modela, tj. modela izgrađenosti mreže i
- dinamička validacija - testiranje na serverskim instancama nivo 2. slika 1. koje su konfigurisane da primaju *Real Time* podatke iz zone nivo 1; slika 1.

Poslednji korak je primena izmene modela na model u *Real Time* (slika 1; nivo 1) okruženju i odvija se na dva načina: sinhronizacija modela: mehanizam u kojem se sve izmene, koje su validirane i testirane, primenjuju na pojedinačne modele u celom sistemu i direktna primena izmena na model - *Real Time* energizacija na nivou 1; slika 1.

4.1 Učesnici u izmenama modela elektroenergetske mreže

U mehanizmu održavanja i ažuriranja modela učestvuju više nosilaca funkcionalnosti - rola (eng. *user roles*) i pojednostavljeno gledano, za učesnike se mogu smatrati grupe korisnika u elektroenergetskom preduzeću koje učestvuju u manipulacijama nad modelom mreže. Primeri grupa korisnika i njihove pozicije u arhitekturi EES su sledeći:

- Model koordinator predvodi glavnu funkcionalnost i njegove akcije se izvršavaju kroz sve zone u sistemu u smislu pripreme izmena modela. Pozicija Model koordinatora može biti u zoni nivo 2; slika 1.
- Operator izvršava komande u produkcionoj zoni (*Real Time* komande) slika 1; nivo 1. Komande su najvišeg prioriteta i odgovoran je za funkcionisanje elektroenergetskog sistema.
- Departman za planiranje je lociran u delu sistema nivo 3; slika 1. i zadužen je za unos podataka u model mreže i izradu planova za održavanje modela mreže.
- Departman za testiranje izmena modela lociran u zoni nivo 2; slika 1. vrši finalne provere pred primenu izmene modela u produkcionu zonu nivo 1; slika 1.

4.2 Dijagrami stanja modela elektroenergetske mreže

Životni ciklus izmene modela podataka elektroenergetske mreže se predstavlja stanjima i prelazima (eng. *State Machine*) kroz koje model prolazi. U zavisnosti od vrste izmene i načina na koji se izmena izvršava sledi i klasifikacija izmena po tipovima:

- *Network Plan Changeset* - Izmene modela elektroenergetske mreže nastale u procesu analize mreže (eng. *Network Analysis*), planiranja i kreiranja scenarija.
- *External Changeset* - Izmene modela elektroenergetske mreže importovane iz spoljašnjeg sistema, npr. GIS.

- *Network Replacement Changeset* - Izmene nastale od strane aplikacija za održavanje modela mreže.

U dijagramu stanja izmene modela, svako stanje poseduje svoju semantiku. Primeri stanja su: inicijalno, primenjeno na model, testiranje u toku, uspešno testiranje, arhivirano, nevalidno, sinhronizovano, itd. Svako stanje predstavlja status u kojem se *Changeset* trenutno nalazi u odgovarajućem delu sistema, to jest zoni, a tranzicije predstavljaju dozvoljene prelaze stanja prilikom izmene modela. Svakom stanju se dodeljuje semantika kao na slici 7. koja predstavlja dijagram stanja *External Changeset* tipa. Slučaj korišćenja (eng. *use case*) takvog *Changeset* tipa bi se izvršavao sledećim redosledom:

- *Changeset* je kreiran, nalazi se u bazi podataka (*DB*) u inicijalnom stanju *STARTED* u nivou 2. slika 7. (Importovana je izmena modela mreže iz spoljašnjeg sistema npr. *GIS*).
- Ako su importovane izmene modela zastarele, one postaju nedostupne za dalje analize i sinhronizaciju. *Changeset*-u se dodeljuje stanje *OBSOLETE*. Iz ovog stanja se *Changeset* ne može prevesti ni u jedno stanje.
- Ukoliko je *Changeset* neuspešno testiran/validiran dodeljuje mu se stanje *DUMP*.
- *Changeset*-u je dodeljeno stanje *TEST* za vreme validiranja/testiranja na *QATest* instanci.
- Nakon završenog ciklusa validiranja/testiranja *Changeset*-u je dodeljeno stanje *PREPARED*. Prilikom pokretanja sinhronizacije sistema, izmena modela će biti primenjena na model u *Real Time* zoni nivo 1.
- Stanje *APPLIED* je dodeljeno *Changeset*-u ukoliko je izmena modela primenjena na model u zoni nivo 1. *Changeset* je u stanju *APPLIED* nakon testiranja tj. nakon prelaska iz nivoa 2. u nivo 1. kao na slici 7.
- Stanje *STORED* je dodeljeno *Changeset*-u u trenutku kad je arhiviran.

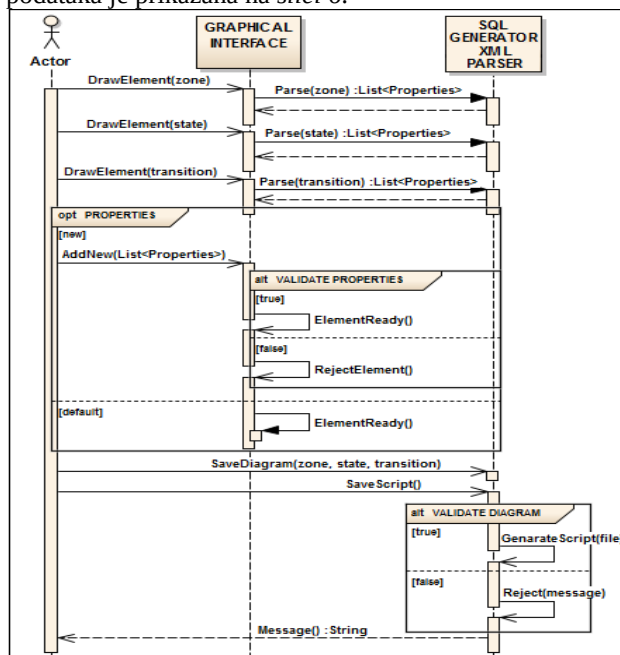
U poslovnom procesu *ChangeSet* je smešten u distribuiranom sistemu za održavanje modela elektroenergetske mreže. Dijagram stanja svojom semantikom definiše dozvoljene operacije nad *ChangeSet*-om slika 7. Svako dodavanje stanja, definisanje prelaza između stanja i dostupnost *ChangeSet*-a u različitim nivoima arhitekture sa slike 1. je definisano dijagramom stanja. Sistem za održavanje modela mreže sve izmene nad modelom izvršava u skladu sa dijagramom stanja.

5. IMPLEMENTACIJA REŠENJA

5.1 Grafički Interfejs

Implementirana aplikacija omogućuje kreiranje dijagrama stanja, kao i njegovo održavanje. Korisnik ima mogućnost crtanja proizvoljnog broja stanja i prelaza između njih. U svaki od njih se u toku crtanja dijagrama učitavaju podrazumevana svojstva koristeći *SAX XML Parser*. Svojstva elemenata su definisana poslovnim procesom. Svojstvima se predstavljaju: naziv stanja, početno i završno stanje tranzicije, dozvoljene promene stanja kroz zone, tip *Changeset*-a, jedinstveni identifikator, sistem koji kreira *Changeset* itd. Svaka restrikcija poslovnog procesa opisanog u poglavlju 4. je zabeležena u svojstvima. Pored podrazumevanih, korisnik može da dodaje i nova svojstva u zavisnosti od poslovnog modela koji kreira. Za vreme dodavanja svojstva vrši se validacija podataka čime je onemogućeno da korisnik unese

nevalidne podatke. Korisnik može da sačuva dijagram stanja u *XML* formatu, ili da na osnovu prelaza stanja generiše *SQL* skriptu čije izvršavanje rezultuje kreiranjem odgovarajućih tabela i relacija na osnovu definisanog dijagrama stanja. Aplikacija je dizajnirana i implementirana tako da podržava zaštitu od korisničkih grešaka za vreme crtanja dijagrama. Greške koje se detektuju su: kreiranje stanja izvan zone, kreiranje Tranzicije koja ne sadrži Stanja kao početnu i krajnju tačku ili sadrži presek dve različite Zone, itd. Funkcionalnost aplikacije *Changeset Editor Tool* je implementirana entitetima: Grafički Interfejs, *XML Parser*, Serijalizacija/Deserijalizacija i *SQL Skript Generator*. Dijagram sekvenci na slici 3. prikazuje crtanje dijagrama stanja i generisanje *SQL* skripti. Korisnik kreira dijagram stanja zadavanjem elemenata dijagrama. Za to vreme se u svaki element dijagrama parsiranjem učitavaju podrazumevana svojstva. Moguće je zadavati nova svojstva koja će biti validirana. Ukoliko je dijagram validan, moguće ga je sačuvati, tj. serijalizovati. Pokretanjem komande za generisanje skripte, korisnik generiše *SQL* skriptu. Izvršavanjem generisane skripte se kreiraju i popunjavaju tabele sa podacima iz svojstva elementa na dijagramu stanja. Relaciona šema baze podataka je prikazana na slici 6.



Slika 3. Dijagram sekvenci aplikacije 5.2 Generisanje SQL upita

Generisanje tabela baze podataka je realizovano kao na relacionom dijagramu sa slike 7. Naziv svojstva elementa predstavlja naziv kolone, a tip svojstva je tip kolone u *SQL* naredbi *create* u skripti. Stanje je predstavljeno tabelom *changeset_state*, element Tranzicija je predstavljen tabelom *changeset_state_transition*, a tip izmene tj. *ChangeSet*-a tabelom *changeset_type*. Kardinaliteti, ključevi, i strani ključevi su prikazani na slici 7.

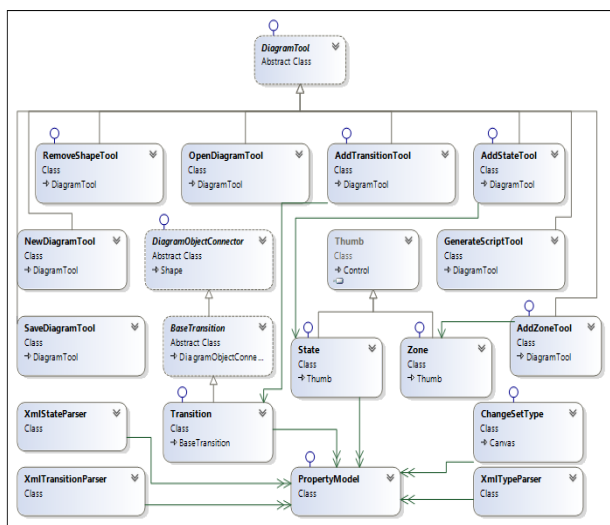
Generisanje *SQL* skript fajlova za popunjavanje tabela je implementirano mapiranjem na dijagram stanja. Implementacija je realizovana mapiranjem: jedan element dijagrama - jedna *SQL* naredba u skript fajlu. Naziv svojstva elementa predstavlja naziv kolone, a vrednost

svojstva je vrednost kolone u SQL naredbi *insert* u skripti - prikazano na slici 4.

```
--6/18/2017 10:50:31 PM
--ChangeSet scripts generated by Changeset Editor Tool:
--Inserts in: CHANGESET_STATE TABLE
INSERT [dbo].[CHANGESET_STATE]
([ID], [STATE_NAME], [DATASET], [ARCHIVED], [APPLIED], [VALIDATE], [SYNTAX], [EDITABLE])
VALUES (N'1', N'STARTED', N'ChangeSetTesting', N'false', N'false', N'false', N'1', N'true')
INSERT [dbo].[CHANGESET_STATE]
([ID], [STATE_NAME], [DATASET], [ARCHIVED], [APPLIED], [VALIDATE], [SYNTAX], [EDITABLE])
VALUES (N'5', N'PREPARED', N'ChangeSetTesting', N'false', N'false', N'false', N'5', N'false')
INSERT [dbo].[CHANGESET_STATE]
([ID], [STATE_NAME], [DATASET], [ARCHIVED], [APPLIED], [VALIDATE], [SYNTAX], [EDITABLE])
VALUES (N'2', N'DUMP', N'ChangeSetTesting', N'false', N'false', N'false', N'2', N'false')
INSERT [dbo].[CHANGESET_STATE]
([ID], [STATE_NAME], [DATASET], [ARCHIVED], [APPLIED], [VALIDATE], [SYNTAX], [EDITABLE])
VALUES (N'4', N'TEST', N'ChangeSetTesting', N'false', N'true', N'false', N'4', N'false')
INSERT [dbo].[CHANGESET_STATE]
([ID], [STATE_NAME], [DATASET], [ARCHIVED], [APPLIED], [VALIDATE], [SYNTAX], [EDITABLE])
VALUES (N'3', N'OBSELETE', N'ChangeSetTesting', N'true', N'false', N'false', N'3', N'false')
INSERT [dbo].[CHANGESET_STATE]
([ID], [STATE_NAME], [DATASET], [ARCHIVED], [APPLIED], [VALIDATE], [SYNTAX], [EDITABLE])
VALUES (N'9', N'APPLIED', N'ChangeSetEnergization', N'false', N'true', N'false', N'6', N'false')
INSERT [dbo].[CHANGESET_STATE]
([ID], [STATE_NAME], [DATASET], [ARCHIVED], [APPLIED], [VALIDATE], [SYNTAX], [EDITABLE])
VALUES (N'11', N'STORED', N'ChangeSetEnergization', N'true', N'true', N'false', N'9', N'false')
```

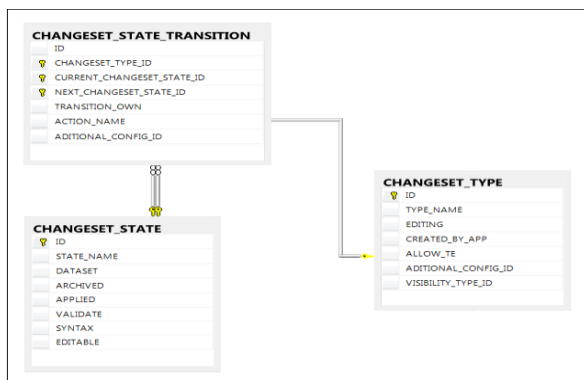
Slika 4. SQL skripta za popunjavanje tabele State elementa u dijagramu stanja.

5.3 Organizacija entiteta u programskom rešenju



Slika 5. Klasni dijagram aplikacije

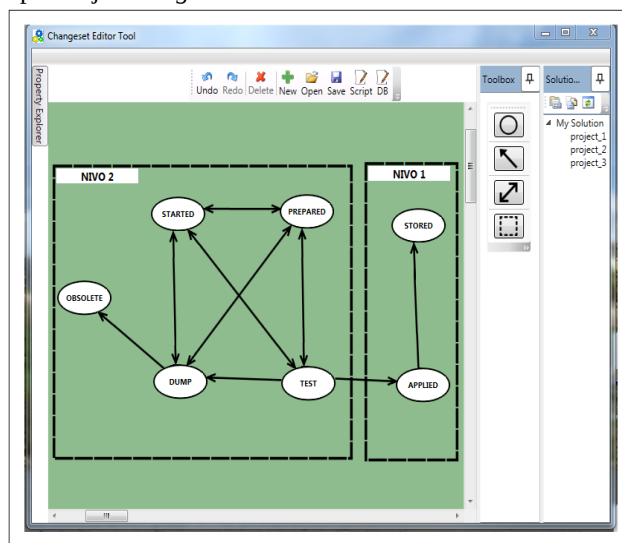
Aplikacija *Changeset Editor Tool* je izrađena u C# programskom jeziku, korišćenjem .NET platforme, WPF (Windows Presentation Foundation) tehnologije i Visual Studio 2013 alata.



Slika 6. Relacioni dijagram SQL šeme

Klasni dijagram je prikazan na slici 5. koja predstavlja veze između entiteta u aplikaciji. Pored grafičkih entiteta koji predstavljaju simbole dijagrama stanja, na slici 5. su i entiteti kojima je implementirana funkcionalnost: pomeranje elemenata dijagrama na radnoj površini (eng. *drag & drop*), enkapsulacija svojstva, selektovanje elemenata, generisanje SQL skripti, brisanje elemenata, opcija poništenja izvršavanja komande (eng. *undo*), opcija ponovnog izvršavanja komande - (eng. *redo*) i čuvanje i ponovno učitavanje dijagrama stanja.

Na slici 7. je prikazan izgled korisničkog interfejsa aplikacije *Changeset Editor Tool*.



Slika 7. Korisnički interfejs aplikacije.

6. ZAKLJUČAK

Aplikacija je namenjena grupi korisnika koji učestvuju u kreiranju izmena, održavanju i ažuriranju modela mreže. Cilj rada je predlog aplikativnog rešenja koje bi korisnicima olakšalo unos izmena modela u bazu podataka i propagiranje izmena na šemu baze. Ideja je automatizovati proces izmena. Aplikacija je testirana u praksi nad realnim modelom elektroenergetskog sistema.

7. LITERATURA

- [1] Dejan Ivezić, "Nacrt strategije razvoja energetike Republike Srbije za period do 2025. godine sa projekcijama do 2030. godine", Ministarstvo energetike, razvoja i zaštite životne sredine, Jun 2013.
- [2] P. Brown, "Using the Common Information Model for Network Analysis Data Management", A CIM Primer Series Guide, November 2014.
- [3] Ervin Varga, Beleške sa predavanja iz "Održavanje i kontrola kvaliteta elektroenergetskog softvera", Univerzitet u Novom Sadu, FTN, 2016.
- [4] "Creating a Reliable and Secure Advanced Distribution Management System", Schneider Electric USA Inc, June 2012.
- [5] E. Hayden, J. Freese, "Technical guide NERC COMPLIANCE", North American Electric Reliability Council, 2010.
- [6] Re. Mahan, JR Burnette, "Secure Data Transfer Guidance for Industrial Control and SCADA Systems", U.S. Department of Energy, 2011.
- [7] Kissel, R, "Glossary of Key Information Security Terms", National Institute of Standards and Technology Interagency, May 2013.

Kratka biografija:



Marko Razumenić je rođen u Vršcu. Školske 2004/2005 je upisao na FTN. Za vreme školovanja imao je petogodišnju pauzu, jer je radio na prekookeanskom brodu. 2011. je nastavio studije a 2014. godine diplomirao. Školske 2014/2015 je upisao master studije na smeru Elektroenergetski softverski inženjering na istom fakultetu.

**РЕКОНСТРУКЦИЈА ФАЗНИХ СТРУЈА ШЕСТОФАЗНЕ АСИНХРОНЕ МАШИНЕ
КОРИШЋЕЊЕМ СТРУЈА ЈЕДНОСМЕРНОГ МЕЋУКОЛА****SIX PHASE INDUCTION MACHINE CURRENT RECONSTRUCTION
USING DC BUS CURRENT**

Никола Вукајловић, Драган Милићевић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА

Кратак садржај – У овом раду излажу се теоријска грађа и симулациона верификација могућности и ограничења алгорита за реконструкцију фазних струја шестофазне асиметричне асинхроне машине. Принцип реконструкције фазних струја се базира на мерењу струја у једносмерном међуколу претварача који погони шестофазну машину. Приказан је кратак преглед основних теоријских начела на којима почива алгоритам за реконструкцију, као и симулациони резултати, на основу којих се може донети закључак о адекватности предложеног алгорита и могућностима његове примене.

Abstract – In this paper theoretical basis as well as verification using appropriate simulation model, are shown with aim to examine all the possibilities as well as limitations of algorithm for double three phase induction motor current reconstruction based on DC link current measurement. Herein, the brief theoretical principles are introduced, followed by simulation results, which form a suitable representation of proposed algorithm and possibilities of its practical application.

Кључне речи: шестофазна машине, реконструкција фазних струја, смањење броја струјних сензора, једносмерно међуколо

1. УВОД

Новија истраживања која се баве предностима вишефазних система у односу на класичан трофазни, дошла су до закључка да постоји низ предности вишефазних система међу којима су мања осцилација момента, већа сигурност, боља и равномернија расподела снаге (струје) по фазама машине [1].

Развој и употреба вишефазних асинхронних машина били су подстакнути између осталог и захтевом за повећање толеранције на кварове, као и захтевом за смањења фазних струја чиме би се постигао бољи однос снаге према фазној струји машине док се истовремено смањује и струјно напрезање полупроводничких елемената који напајају фазне намотаје машине. Као резултат напора у циљу превазилажења неких од поменутих проблема, појавиле су се најпре петофазне и шестофазне асинхроне машине, а недуго затим и машине са још већим бројем фаза [1], [2].

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је био доц. др Драган Милићевић.

Једна од конструкција вишефазне асинхроне машине, која је међу најчешће теоријски разматраним, али и међу најчешће практично коришћеним и то у електричним и хибридним возилима, бродским погонима, ветроелектранама као и у различитим вучним погонима, је шестофазна асиметрична асинхрона машина (енг. *Double Three-Phase Induction Machine*). Ова асинхрона машина се конструкцијски састоји од два трофазна намотаја, који су међусобно електрично померени за 30° , а чија су звездиста међусобно изолована. Оваква конструкција машине у првом реду побољшава сигурност, односно приликом испада једне фазе машине или претварача, погон може несметано да настави са радом, али са смањеним оптерећењем.

Како је циљ овог рада анализа могућности реконструкције фазних струја шестофазне асинхроне машине, али и целокупног погона, са циљем постизања управљања и регулације машине, у овом раду биће предложена и на симулационом моделу тестирана методологија добијања величина потребних за реализацију реконструкције фазних струја. Ова методологија ће бити анализирана са аспекта техничких захтева погона, али и економске оправданости, са циљем налажења оптималног решења које се огледа у употреби минимално потребног броја сензора, давача и друге аквизиционе и сензорне опреме уз постизање што бољих перформанси целог погона.

Методологија које ће бити разматрана подразумева коришћење само два струјна сензора постављена у једносмерном међуколу претварача који напаја машину. На основу мерења струја у једносмерном међуколу, уз употребу напредног алгорита, могу се реконструисати фазне струје у намотајима шестофазне асинхроне машине.

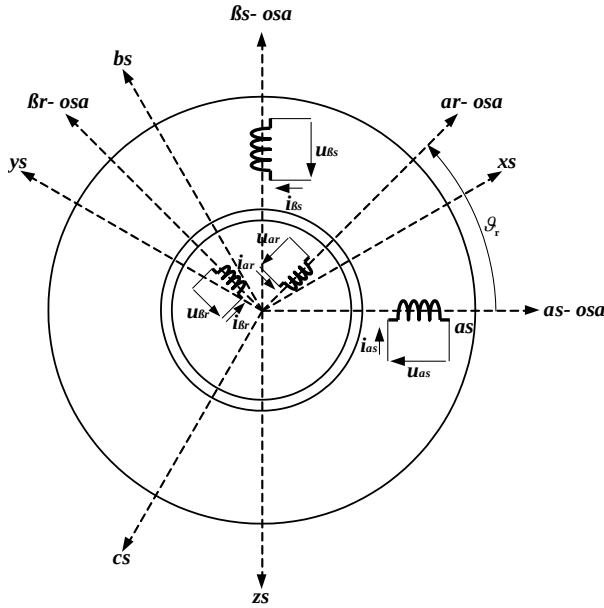
2. МАТЕМАТИЧКИ МОДЕЛ

Како би се омогућило истраживање различитих аспеката рада било које електричне машине и вршила симулација њеног рада под различитим условима коришћењем софтверских алага, потребно је извршити синтезу математичког модела те машине.

Математички модел електричне машине састоји се од сета диференцијалних једначина које дефинишу електричне и механичке процесе који се дешавају у машини. Међутим систем диференцијалних једначина у временском домену, који би описивао све процесе у једној машини био би сувише компликован и самим тим неупотребљив. Због тога се користе одређена

упрошћења, која поједностављују математички модел и доводе га до нивоа у коме он довољно добро описује процесе у машини, а степен његове сложености је прихватљив.

Математички модел који је овде представљен је дат након извршене трансформације распрезања, која за последицу има смањење комплексности модела у виду свођење шестофазне машине на двофазну, уз елиминацију електромагнетне спреге између два намотаја. Принциуска шема шестофазне машине након обављене трансформације распрезања приказан је на слици 1 [3]–[5].



Слика 1. Шематиски приказ шестофазне асиметричне асинхроне машине након трансформације распрезања

На основу теорије декомпозиције просторних вектора (енг. *Vector Space Decomposition*), шестофазна машина се не може једноставно свести на двофазну, већ се трансформационом матрицом шестодимензионални математички модел своди на три независна модела смештена у три дводимензионална међусобно ортогонална потпростора. Ови потпростори означени су као $\alpha\beta$ потпростор, у коме се налазе компоненте које учествују у електромагнетној конверзији енергије, $\mu_1 - \mu_2$ потпростор, у коме се налазе компоненте које доприносе губицима у машини, и $z_1 - z_2$ потпростор, у коме се налазе нулте компоненте, а које се неутралишу у случају када су звездишта намотаја машине међусобно изолована.

Матрични запис једначина напонске равнотеже машине у сва три потпростора дат је једначинама (1)–(3) [3], [6]:

$$\begin{bmatrix} u_{\alpha s} \\ u_{\beta s} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 \\ 0 & 0 & R_s & 0 \\ 0 & 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \\ i_{\alpha r} \\ i_{\beta r} \end{bmatrix} + \frac{d}{dt} [L_{\alpha\beta}] \begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \\ i_{\alpha r} \\ i_{\beta r} \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} v_{\mu 1s} \\ v_{\mu 2s} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 \\ 0 & 0 & R_s & 0 \\ 0 & 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} i_{\mu 1s} \\ i_{\mu 2s} \\ i_{\mu 1r} \\ i_{\mu 2r} \end{bmatrix} + \begin{bmatrix} L_{ls} & 0 & 0 & 0 \\ 0 & L_{ls} & 0 & 0 \\ 0 & 0 & L_{ls} & 0 \\ 0 & 0 & 0 & L_{ls} \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_{\mu 1s} \\ i_{\mu 2s} \\ i_{\mu 1r} \\ i_{\mu 2r} \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} v_{z1s} \\ v_{z2s} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 \\ 0 & 0 & R_s & 0 \\ 0 & 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} i_{z1s} \\ i_{z2s} \\ i_{z1r} \\ i_{z2r} \end{bmatrix} + \begin{bmatrix} L_{ls} & 0 & 0 & 0 \\ 0 & L_{ls} & 0 & 0 \\ 0 & 0 & L_{ls} & 0 \\ 0 & 0 & 0 & L_{ls} \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_{z1s} \\ i_{z2s} \\ i_{z1r} \\ i_{z2r} \end{bmatrix} \quad (3)$$

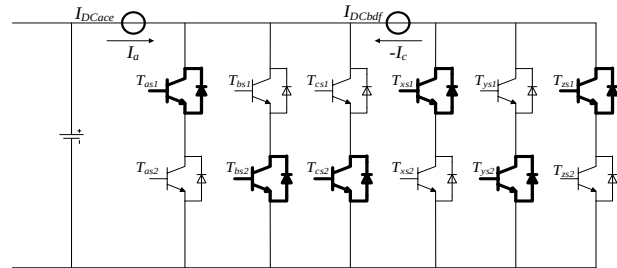
Анализом математичког модела добијеног применом теорије декомпозиције просторних вектора, као и трансформационе матрице којом се постиже распрезање намотаја машине, може се доћи до следећих закључака [3]:

1. Једначина показује да је модел шестофазне асиметричне асинхроне машине у $\alpha\beta$ потпростору, који је значајан јер величине у њему учествују у електромагнетној конверзији енергије, идентичан моделу трофазне асинхроне машине,
2. Једначине напонске равнотеже статорског и роторског кола $\mu_1 - \mu_2$ и $z_1 - z_2$ се међусобно не разликују по форми, и представљају једначине напонске равнотеже RL кола,
3. Како би се смањило генерисање губитака услед хармоника, варијабле у $\mu_1 - \mu_2$ потпростору морају бити што мање, односно компоненте напона статора које се трансформишу у $\mu_1 - \mu_2$ потпростор морају да имају што мању амплитуду.

На основу модела машине у $\alpha\beta$ потпростору, биће приказана теоријска основа за реконструкцију фазних струја на основу мерења струја једносмерног кола.

3. ТЕОРИЈА РЕКОНСТРУКЦИЈА ФАЗНИХ СТРУЈА ШЕСТОФАЗНЕ МАШИНЕ

Да би се према алгоритму који ће бити приказан могла вршити реконструкција фазних струја, на основу мерења струја у једносмерном међуколу, морају струјни сензори бити распоређени као што је приказано на слици 2.



Слика 2. Распоред струјних сензора у једносмерном међуколу

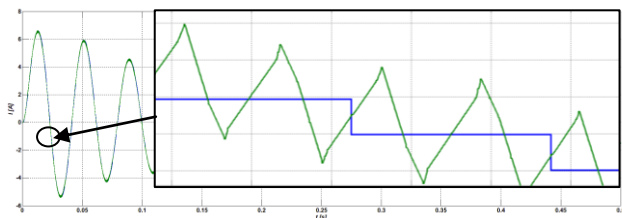
Струјни сензори су распоређени у једносмерном међуколу, тако да I_{DC1} мери укупну струју која иде према машини, док I_{DC2} мери струју према другом трофазном намотају. На овај начин може се одузимањем I_{DC2} од I_{DC1} добити струја која одлази ка првом трофазном намотају, те на тај начин добити информација о струјама једносмерног међукола које кореспондирају сваком трофазном намотају, што је неопходан предуслов за реконструкцију струја у сваком од трофазних намотаја.

Развијање алгоритма за реконструкцију фазних струја у намотајима машине у свим режимима рада, представља први корак у формирању поузданог, робусног, сигурног и квалитетног погона [7], [8].

4. РЕЗУЛТАТИ СИМУЛАЦИЈЕ

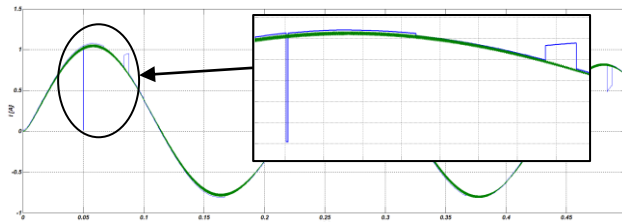
Коришћењем *Matlab* модела шестофазног претварача, као и шестофазне машине, уз имплементацију алгоритма укратко изложеног у претходном одељку добијени су резултати очитавања и реконструкције фазних струја, уз поређење са стварним фазним струјама.

На слици 5 приказано је поређење мерене струје и струје добијене из алгоритма за реконструкцију.



Слика 5. Реконструисана (плаво) и мерена (зелено) струја машине

Међутим резултати на слици 5 приказани су уз претпоставку да АД конверторима не треба време за очитавање струја, што пракси није тачно. Струјним сензорима треба време за очитавање које је реда величине μs . Ако се ово узме у обзир онда су резултати реконструкције другачији, а резултати симулације у којој се уважава претходно приказани су на слици 6.



Слика 6. Реконструисана (плаво) и мерена (зелено) струја машине

Са слике 6 постаје јасно да уважавање реалне природе АД конвертора реконструкција није истог квалитета, односно у неким ситуацијама није могуће извршити реконструкцију фазне струје. Ово је последица тога што се током неких PWM циклуса јављају вектори чије је време трајања краће од времена потребног да АД конвертор изврши очитавање, а што се дешава при малим брзинама или преласку између сектора.

Током периода када је трајање напонског вектора сувише мало, потребно је развити алгоритам за процену струје на основу неког другог параметра, или на основу еквивалентне шеме електричне машине.

5. ЗАКЉУЧАК

У раду је изложен алгоритам за реконструкцију фазних струја шестофазне машине, који се ослања на мерењу само две струје у једносмерном међуколу. Адресирана је и проблематика која се односи на време потребно за аналогно-дигиталну конверзију, комерцијално доступних конвертора.

Сврха изложеног решења може се посматрати кроз два независна аспекта, економског који за циљ има смањење цене реализованог погона, али и оног значајнијег техничког који уз минимална додатна улагања повећава толерантност вишефазног погона на отказ струјних сензора.

Будућа унапређења изложеног алгоритма, могу се базирати на покушају решавања проблема реконструкције струје током ситуација у којима се јавља кратко трајање напонских вектора. Решавањем овог проблема добила би се регулациона структура која обезбеђује робустан, сигуран, поуздан и квалитетан рад погона.

6. ЛИТЕРАТУРА

- [1] E. Levi, "Multiphase Electric Machines for Variable-Speed Applications," *IEEE Trans. Ind. Electron.*, vol. 55, no. 5, pp. 1893–1909, May 2008.
- [2] R. Bojoi, F. Farina, A. Tenconi, F. Profumo, and E. Levi, "Stationary frame digital current regulation for dual-three phase induction motor drives," 2013, pp. 2121–2127.
- [3] D. Milicevic, "Razvoj naprednog upravljanja pogonom sa šestofaznim asimetričnim asinhronim motorom," *Doktorska disertacija*, Univerzitet u Novom Sadu, Novi Sad, Srbija, 2014.
- [4] D. Hadiouche, L. Baghli, and A. Rezzoug, "Space vector PWM techniques for dual three-phase AC machine: analysis, performance evaluation and DSP implementation," *Ind. Appl. IEEE Trans. On*, vol. 42, no. 4, pp. 648–655, 2003.
- [5] Yifan Zhao and T. A. Lipo, "Space vector PWM control of dual three-phase induction machine using vector space decomposition," *IEEE Trans. Ind. Appl.*, vol. 31, no. 5, pp. 1100–1109, Oct. 1995.
- [6] V. Vučković, *Opšta teorija električnih mašina*. Beograd: Nauka, Beograd, 1988.
- [7] E. Adžić, "Regulisani pogon asinhronog motora sa minimalnim brojem senzora," *Doktorska disertacija*, Univerzitet u Novom Sadu, Novi Sad, Srbija, 2013.
- [8] D. P. Marčetić and E. M. Adžić, "Improved Three-Phase Current Reconstruction for Induction Motor Drives With DC-Link Shunt," *IEEE Trans. Ind. Electron.*, vol. 57, no. 7, pp. 2454–2462, Jul. 2010.

Кратка биографија:



Никола Вукајловић рођен је у Ужицу 1993. год. Дипломски рад на Факултету техничких наука из области Електротехнике и рачунарства-Енергетска електроника и електричне машине, одбранио је 2016.



Драган Милићевић рођену је 1977. год. у Тузли. У звање доцента Универзитета у Новом Саду је изабран 2014.год. Област интересовања су вишефазне машине, електромоторни погони и обновљиви извори електричне енергије

ADAPTACIJA DUBOKE NEURONSKE MREŽE NAMENJENE SINTEZI GOVORA NA KARAKTERISTIKE POJEDINAČNOG GOVORNIKA**ADAPTATION OF A DEEP NEURAL NETWORK FOR TEXT-TO-SPEECH SYNTHESIS TO THE CHARACTERISTICS OF A PARTICULAR SPEAKER**

Simona Đurić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**1. UVOD**

Kratak sadržaj – U ovom radu analizirana je sinteza govora na osnovu teksta zasnovana na modelu više govornika, kao i adaptacija ovog modela na govor pojedinca. Modelom više govornika pri tom se naziva duboka neuronska mreža obučavana na govornom materijalu različitih govornika, dok se modelom pojedinačnog govornika naziva mreža obučavana na govornom materijalu pojedinca. U okviru analize sinteza i adaptacija modela više govornika upoređene su sa sintezom i adaptacijom modela pojedinačnog govornika, koji je opšeprihvaćen, na osnovu čega je procenjena adekvatnost pritupa korišćenog u ovom radu. Objektivne mere, korišćene kao merilo kvaliteta sintetizovanog govora, ukazuju na to da se pristupom obući i adaptaciji modela više govornika korišćenim u ovom radu dobija sintetizovani govor lošijeg kvaliteta u odnosu na govor dobijen sintezom zavisnom od pojedinca. Ovakvi rezultati nisu bili očekivani i pretpostavlja se da je osnovni razlog za njihovo dobijanje to što je model više govornika obučavan na govornom materijalu svega dva govornika.

Abstract – In this paper an analysis of multi-speaker modeling for text-to-speech (TTS) synthesis was performed along with the analysis of a multi-speaker model adaptation to speech of a particular speaker. A deep neural network trained on speech material of different speakers is considered to be a multi-speaker model, while the network that is trained on the material of one speaker is considered to be a speaker-dependent model. Since using a speaker-dependent model is the state of the art in deep neural networks based TTS synthesis, the comparison between speaker-dependent model and multi – speaker model was performed. By comparing the objective measures of the synthesised voice quality in case of a multi-speaker and speaker-dependent model, one can conclude that the approach that is used in this paper for training and adapting the multi-speaker model is not adequate. However, it is assumed that the main problem is not in the approach itself, but in the limitation of speech material that is used and that comes from only two speakers.

Ključne reči: sinteza govora, duboke neuronske mreže, model više govornika, model pojedinačnog govornika, adaptacija modela.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Sečujski.

U poslednjih nekoliko godina najzastupljenija metoda za sintezu govora na osnovu teksta (engl. *Text to Speech Synthesis - TTS*) zasniva se na dubokim neuronskim mrežama (engl. *Deep Neural Network – DNN*). Različita istraživanja pokazuju da se sintezom zasnovanom na dubokim neuronskim mrežama dobija govor koji je blizak prirodnom [1], kao i to da je prirodniji od govora sintetizovanog metodom zasnovanom na HMM (engl. *Hidden Markov Model*) koja je do skora bila najzastupljenija u sintezi govora na osnovu teksta [2][3]. U pomenutim istraživanjima neuronska mreža najčešće se obučava na govoru nekog pojedinca, pa se shodno tome sinteza naziva zavisnom od pojedinačnog govornika (engl. *speaker-dependent*), i teži se da se sintetizom dobije govor što sličniji prirodnom govoru tog pojedinca [4].

U ovom radu pristup sintezi je drugičiji i pretežno se oslanja na koncept predstavljen u radu [5]. Mreža se obučava na govorima više govornika sa ciljem da se sintezom dobije prirodan govor čije karakteristike zavise od parametra sinteze koji se naziva *kodom govornika*. Ovakva sinteza takođe se naziva i nezavisnom od pojedinačnog govornika (engl. *speaker-independent*), a pristup omogućava da se promenom vrednosti koda sintetizuju govori sa karakteristikama različitih govornika što najčešće jesta poželjna osobina sintetizatora [5]. Ukoliko je potrebno da se sintetizuje govor sa karakteristikama nekog novog govornika, mreža se *adaptira*, tj. naknadno obučava na njegovom govornom materijalu i koristi se specijalna vrednost koda koja ukazuje na to da je u pitanju novi govornik. S obzirom na to da je za adaptaciju najčešće dostupno relativno malo govornog materijala (reda veličine nekoliko minuta), očekivano je da će se inicijalna obuka raditi na više materijala (reda veličine nekoliko sati) kako bi se kompenzovao ovaj nedostatak. Prednostima ovog pristupa smatraju se veća robustnost i veća fleksibilnost, o čemu će više reči biti u narednim odeljcima.

Ciljevi ovog rada jesu uporediti prirodnost govora dobijenog sintezom nezavisnom i sintezom zavisnom od pojedinca i ispitati da li prilikom sinteze informacije o govoru jednog govornika mogu kompenzovati nedostatak informacija o govoru nekog drugog govornika. Istraživanje je urađeno na osnovu raspoloživih govorno – jezičkih resursa za srpski i hrvatski jezik, ali se očekuje da dobijeni zaključci, zbog relativne nezavisnosti karakteristika glasa govornika od jezika, važe za sve jezike.

Rad je organizovan na sledeći način: u uvodnom delu definisani su osnovni pojmovi, nakon čega je predložena motivacija i cilj ovog rada.

2. SINTETIZATOR GOVORA ZASNOVAN NA DUBOKIM NEURONSKIM MREŽAMA

3. MODEL POJEDINAČNOG GOVORNIKA

lingvistička obeležja → [blue bar] → [green bar] → [green bar] → [green bar] → [green bar] → [red bar] → trajanja / akustička obeležja

lingvistička obeležja

kôd govornika

trajanja / akustička obeležja

na to da je u pitanju osoba čiji govor nije korišćen prilikom inicijalne obuke modela.

5. EKSPERIMENTI

U okviru eksperimentalnog dela ovog rada porede se kvalitet sinteze i adaptacije modela pojedinačnog govornika sa kvalitetom sinteze i adaptacije modela više govornika. Model čine dve duboke neuronske mreže sa četiri skrivena sloja, na čijem ulazu se nalazi 451 lingvističko obeležje teksta; svaki skriveni sloj sastoji se od po 1024 čvora, pri čemu je aktivaciona funkcija čvorova prva tri sloja tangens hiperbolički, dok je poslednji sloj tzv. sloj sa dugom kratkotrajnom memorijom (engl. *long short-term memory* - *LSTM*); izlaz modela trajanja čine predviđena trajanja 5 stanja glasa (preporuka alata *Merlin*), a izlaz akustičkog modela čine akustička obeležja. Akustičkim obeležjima u ovom radu smatraju se statička obeležja u koja spadaju mel-generalizovani kepsstralni koeficijenti (engl. *MGC* - *Mel-Generalised Cepstrum Coefficients*), logaritam osnovne frekvencije (*lf0*) i koeficijenti aperiodičnosti (engl. *BAP* - *Band Aperiodicity*), odnosno njihovi prvi i drugi izvodi, koji se takođe nazivaju i dinamičkim obeležjima, kao i obeležje zvučnosti glasa (engl. *VUV* - *Voiced/Unvoiced*).

Modeli su obučavani i testirani na govornom materijalu dva ženska govornika srpskog jezika i ženskog govornika hrvatskog jezika. Ovi materijali deo su govornih baza na srpskom i hrvatskom jeziku snimljenih u saradnji između Fakulteta tehničkih nauka i istraživačko-razvojnog centra „AlfaNum“. Ukupni govorni materijal na srpskom jeziku čine audio snimci i odgovarajuće fonetske transkripcije 3678 rečenica, u ukupnom trajanju od 6 sati i 49 minuta, dok materijal na hrvatskom jeziku čini 109 rečenica u ukupnom trajanju od 11 minuta. S obzirom da je količina raspoloživog materijala na srpskom jeziku značajno veća, ovaj materijal korišćen je za inicijalnu obuku modela, dok je materijal na hrvatskom jeziku korišćen prilikom adaptacije modela. Pri tom je potrebno naglasiti da se srpski i hrvatski smatraju srodnim jezicima, pa je pretpostavljeno da razlike među njima neće uticati na kvalitet adaptacije.

Celokupan govorni materijal podeljen je na dva disjunktna seta: set za obuku i set za testiranje, koji su nadalje korišćeni u fazi obuke modela odnosno provere kvaliteta govora koji je sintetizovan korišćenjem obučanih modela. Merilo kvaliteta generisanog govora u ovom radu čine vrednosti tzv. *objektivnih mera* u koje spadaju: koren sume kvadrata razlika stvarnih i predviđenih vrednosti trajanja (engl. *RMSE* - *root mean square error*) i Pirsonov koeficijent korelacije stvarnih i predviđenih trajanja (*Corr*) za model trajanja; kao i mel - kepsstralna udaljenost (engl. *MCD* - *mel-cepstrum distance*), greška procene koeficijenta aperiodičnosti (*BAP*), procenat pogrešnih procena zvučnosti frejma (*VUV*), koren sume kvadrata razlika stvarnih i predviđenih vrednosti osnovne frekvencije (*RMSE*) i Pirsonovim koeficijentom korelacije stvarnih i predviđenih vrednosti osnovne frekvencije za akustički model.

6.1. Prvi eksperiment

Cilj ovog eksperimenta jeste ispitati robustnost modela više govornika, odnosno njegovu sposobnost da manjak govornog materijala za jednog govornika prilikom obuke

nadomesti govornim materijalom nekog drugog govornika. U okviru eksperimenta model više govornika obučavan na 180 minuta govornog materijala (*M_180*) poreden je sa modelima pojedinačnih govornika obučavanih na 90 minuta (*S1_90* i *S2_90*), odnosno na 180 minuta govornog materijala (*S1_180* i *S2_180*). Rezultati ovog eksperimenta, odnosno vrednosti objektivnih mera dobijene prilikom testiranja ovih modela prikazani su u Tabeli 1.

Očekivani rezultati ovog eksperimenta jesu da se sintezom zasnovanom na modelu pojedinačnog govornika obučavanom na 180 minuta govornog materijala dobije najkvalitetniji govor, zatim da se sintezom zasnovanom na modelu više govornika, takođe obučavanom na 180 min govornog materijala, dobije govor nešto manjeg kvaliteta, dok se za govor dobijen na osnovu modela pojedinačnog govornika obučavanog na 90 minuta govornog materijala očekuje da bude najlošijeg kvaliteta. U Tabeli 1 crvenom bojom označene su vrednosti objektivnih mera koje u slučaju modela više govornika odgovaraju ovim očekivanjima.

	Model trajanja				
	RMSE (frejm/glasu)		Corr		
M_180 (1)	4.795		0.815		
M_180 (2)	4.486		0.861		
S1_180	4.383		0.824		
S2_180	4.042		0.876		
S1_90	5.162		0.753		
S2_90	4.418		0.862		
	Akustički model				
	RMSE (Hz)	Corr	MCD (dB)	BAP (dB)	VUV (%)
M_180_1	18.020	0.844	4.459	0.289	4.011
M_180_2	11.948	0.831	4.603	0.227	4.747
S1_180	16.678	0.876	4.423	0.286	4.531
S2_180	11.975	0.840	4.246	0.212	4.320
S1_90	17.414	0.854	4.442	0.293	4.442
S2_90	11.924	0.818	4.613	0.234	4.542

Tabela 1 Rezultati prvog eksperimenta: brojevi (1) i (2) pored oznake modela više govornika *M_180* označavaju da je u fazi testiranja korišćen govorni materijal (1) govornice *S1*, odnosno (2) govornice *S2*

Predstavljeni rezultati pokazuju da vrednosti objektivnih mera odstupaju od očekivanih, preciznije, da određene objektivne mere ukazuju na to da je govor dobijen sintezom nezavisnom od pojedinačnog govornika najmanje kvalitetan. Na osnovu toga može se zaključiti da u opštem slučaju dobijeni model više govornika nije robustniji u odnosu na model pojedinačnog govornika.

Međutim, takođe je potrebno uočiti da određene vrednosti odstupaju od očekivanih i time ukazuju na to da je sinteza govora zasnovana na modelu pojedinačnog govornika obučavanom na 180 min govornog materijala najmanje kvalitetna – u Tabeli 1 ove vrednosti su podebljane (**Bold**). One svakako nisu u skladu sa osnovanom pretpostavkom da bi se za model govornika obučavanog na 180 min materijala pojedinačnog govornika očekivalo da daje najpovoljnije rezultate. Shodno tome pretpostavljeno je da objektivne mere ne moraju biti apsolutni pokazatelj kvaliteta sinteze te da bi za adekvatniju procenu bilo potrebno uzeti u obzir i tzv. *subjektivne mere* koje izražavaju mišljenje slušaoca o kvalitetu sintetizovanog govora.

6.2. Drugi eksperiment

Cilj ovog eksperimenta jeste ispitati fleksibilnost modela više govornika, odnosno njegovu sposobnost da se adaptira na govor novog govornika kada je količina materijala za doobuku relativno mala. U okviru eksperimenta upoređena je adaptacija modela više govornika obučavanog na 180 minuta govornog materijala i adaptiranog na 10 minuta govornog materijala (Ma_180) sa adaptacijama modela pojedinačnih govornika takođe obučanih na 180 minuta govornog materijala i adaptiranih na istih 10 minuta govornog materijala (S1a_180, S2a_180). Količina materijala korišćenog za adaptaciju odgovara količini za koju se u radu [10] pokazalo da je dovoljna za uspešnu doobuku modela pojedinačnog govornika.

Model trajanja					
	RMSE (frejm/glasu)		Corr		
Ma_180	10.213		0.877		
S1a_180	4.131		0.824		
S2a_180	3.976		0.836		
Akustički model					
	RMSE (Hz)	Corr	MCD (dB)	BAP (dB)	VUV (%)
M_180	24.440	0.612	5.099	0.318	4.666
S1_180	24.709	0.592	4.985	0.320	4.934
S2_180	23.189	0.619	4.996	0.319	4.532

Tabela 2 Rezultati drugog eksperimenta

Očekivani rezultati ovog eksperimenta jesu da adaptacija modela više govornika prema objektivnim parametrima bude uspešnija u odnosu na adaptaciju modela pojedinačnih govornika. Dobijeni rezultati ukazuju upravo na suprotno, tj. da je adaptacija modela više govornika manje uspešna u odnosu na adaptaciju pojedinačnog govornika što dalje ukazuje na to da model više govornika obučavan na govornom materijalu svega dva govornika nije fleksibilniji u odnosu na model pojedinačnog govornika. Međutim, s obzirom na rezultate prethodnog eksperimenta, za adekvatniju procenu kvaliteta adaptacije osim objektivnih mera takođe bi bilo potrebno uzeti u obzir i subjektivne mere i tek nakon toga doneti pouzdanije zaključke.

7. ZAKLJUČAK

U ovom radu analizirana je sinteza govora zasnovana na dubokim neuronskim mrežama, preciznije, pristup ovoj sintezi zasnovan na modelu više govornika i adaptiranom modelu više govornika. Cilj rada bio je utvrditi da li je model više govornika robustan, tj. da li se prilikom sinteze nedostatak podataka o karakteristikama jednog govornika može kompenzovati podacima o karakteristikama nekog drugog govornika. Cilj je takođe bio utvrditi da li je model više govornika fleksibilan, odnosno pogodan za adaptaciju na karakteristike govora nekog novog govornika kada je količina govornog materijala za adaptaciju relativno mala.

Predloženi pristup sintezi zasnovan na modelu više govornika bazira se na korišćenju koda govornika. Rezultate ovog rada čine vrednosti objektivnih mera dobijene prilikom testiranja modela više govornika, kao i vrednosti dobijene prilikom testiranja modela pojedinačnog govornika. Poređenjem ovih vrednosti može se zaključiti da

odbrani pristup zasnovan na modelu više govornika nije pogodan za dobijanje robustnog i fleksibilnog modela.

Za adekvatniju procenu ovog modela, međutim, bilo bi potrebno koristiti veću govornu bazu, sa više od dva govornika i subjektivne mere procene.

U daljim istraživanjima bilo bi moguće ispitati različite pristupe sintezi zasnovanoj na modelu više govornika, a samim tim i različite pristupe njegovoj adaptaciji.

8. LITERATURA

- [1] T. Delic, M. Secujski, „Sinteza govora na srpskom jeziku zasnovana na veštačkim neuralnim mrežama“, 24. Telekomunikacioni forum, TELFOR 2016, November 2016, pp. 403-406.
- [2] H. Zen, A. Senior and M. Schuster, “Statistical Parametric Speech Synthesis Using Deep Neural Networks,” in Acoustic, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, May 2013, pp. 7962–7966.
- [3] Y. Qian, Y. Fan, W. Hu, F. K. Soong, „On the Training Aspects of Deep Neural Network (DNN) for Parametric TTS Synthesis“, in Acoustic, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, May 2014, pp. 3829-3833.
- [4] A. Lazaridis, B. Potard, P. N. Garner, „DNN-Based Speech Synthesis: Importance of Input Features and Training Data“, International Conference on Speech and Computer, SPECOM 2015, September 2015, pp. 193-200.
- [5] B. Potard, P. Motlicek, D. Imseng, „Preliminary Work on Speaker Adaptation for DNN-Based Speech Synthesis“, Idiap Research Report, January 2015.
- [6] Istraživačko-razvojni centar AlfaNum, <http://www.alfanum.ftn.uns.ac.rs/>
- [7] Z. Wu, O. Watts, S. King, "Merlin: An Open Source Neural Network Speech Synthesis System" in Proc. 9th ISCA Speech Synthesis Workshop (SSW9), September 2016, , Sunnyvale, CA, USA
- [8] “Reference manual for Speech Signal Processing Toolkit Ver. 3.9,” December 25, 2015, <https://ftp.netbsd.org/pub/pkgsrc/distfiles/SPTKref-3.9.pdf>
- [9] M. Morise, F. Yokomori, and K. Ozawa, “WORLD: a vocoder-based high-quality speech synthesis system for real-time applications,” IEICE transactions on information and systems, Vol. E99-D, pp. 1877-1884, July 2016.
- [10] T. Delic, S. Suzic, M. Secujski, D. Pekar, „Rapid Development of New TTS Voices by Neural Network Adaptation“, ISSPIT 2017 (na recenziji)

Kratka biografija:



Simona Đurić rođena je u Novom Sadu, 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Telekomunikacije i obrade signala odbranila je 2016. god., dok je Master rad odbranila 2017. god.

VIZUELIZACIJA AŽURIRANJA MODELA PAMETNIH MREŽA
VISUALIZATION OF SMART GRID MODEL'S UPDATES

Milan Kostić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu je analiziran proces održavanja mrežnih modela u okviru softverskog sistema čiji je zadatak upravljanje pametnim mrežama. Analiza uključuje aktivnosti u kontekstu definisanja i načina predstavljanja modela, kao i aktivnosti u procesu unošenja inkrementalnih izmena u model. Opisana je implementirana aplikacija koja vizuelno prikazuje proces ažuriranja proizvoljnog modela..

Abstract – Process of maintaining network data model within the software system responsible for managing smart grids is analysed in this paper. Analysis includes activities in the context of defining and presenting model, as well as activities in process of introducing incremental changes into the model. An implemented application which visualizes process of updating model of a smart grid is described.

Cljučne reči: Ažuriranje modela, Sinhronizacija modela, Vizuelizacija

1. UVOD

Elektroenergetski sistem je sistem u okviru kojeg se izvode sve elektroenergetske transformacije, te električna energija prenosi i distribuira [1]. Sastoji se od podistema proizvodnje, prenosa, distribucije i neposredne potrošnje.

Od savremenih elektroenergetskih sistema se zahteva da sigurno i pouzdano vrše isporuku električne energije [1]. Pored toga je poželjno da funkcionišu na ekonomičan način [1]. Da bi se ispunili pomenuti zahtevi i niz drugih zahteva, pristupa se korišćenju različitih softverskih sistema koji se zaduženi za upravljanje elektroenergetskom mrežom. Ovakve mreže, za čije se upravljanje koriste različiti računarski sistemi kao i najsavremenija znanja na polju informacionih tehnologija, se nazivaju pametne mreže (eng. *Smart Grids*). Održavanje i blagovremeno ažuriranje modela pametne mreže predstavljaju ključne aktivnosti od kojih zavisi proces upravljanja. Zbog toga je ažuriranje modela veoma složen proces, praćen nizom različitih provera koje sprečavaju pojavu potencijalnih grešaka.

Održavanje i ažuriranje modela se obavlja u okviru distribuiranog sistema koji uključuje veliki broj fizički nezavisnih hardverskih komponenti. Kao posledica ovako kompleksne fizičke arhitekture, ažuriranje modela predstavlja veoma složenu operaciju, koja može biti vremenski veoma zahtevna.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Milan Gavrić.

Da bi se korisniku, koji inicira proces ažuriranja modela, obezbedile adekvatne povratne informacije o tom procesu, implementirana je aplikacija koja vizuelno prikazuje proces ažuriranja (sinhronizacije) modela.

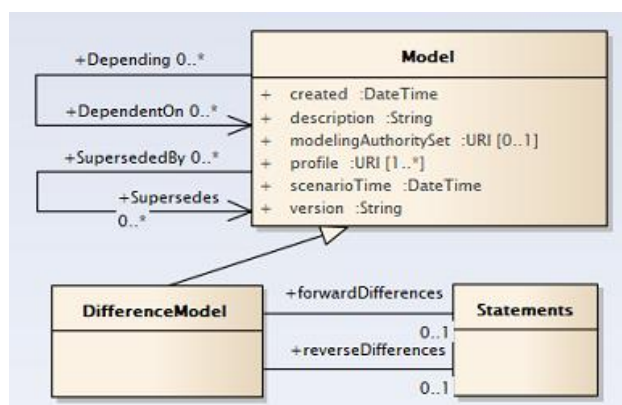
2. ODRŽAVANJE MODELA PAMETNE MREŽE
2.1 Struktura modela

U elektroenergetskom sistemu modelovanje uključuje predstavljanje fizičkih elemenata koji izgrađuju taj sistem kao i opis procesa koji se odvijaju u okviru tog sistema [2]. Fizički elementi koje je neophodno modelovati se nalaze u svim podsystemima elektroenergetskog sistema. Neki elementi su zajednički za sve podsysteme i predstavljaju sastavne delove, na primer provodne, prekidačke ili zaštitne opreme. S druge strane, određeni elementi su specifični samo za pojedine podsysteme, kao što su generatori, smešteni u podsystemu proizvodnje, a danas sve češće i u podsystemu neposredne potrošnje (distributivni generatori) ili pametna brojila (eng. *Smart Meter*) koja se nalaze u podsystemu neposredne potrošnje. Pored fizičkih elemenata, modelovanje uključuje i opis različitih aktivnosti koje čine sastavni deo procesa upravljanja pametnim mrežama [2].

2.2 Ažuriranje modela

U ovom radu se pojam ažuriranja modela odnosi na izmenu konkretnih podataka koji su skladišteni u okviru softverskog sistema zaduženog za upravljanje pametnom mrežom. U tom kontekstu, model predstavlja skup instanci klasa koje održava određena aplikacija.

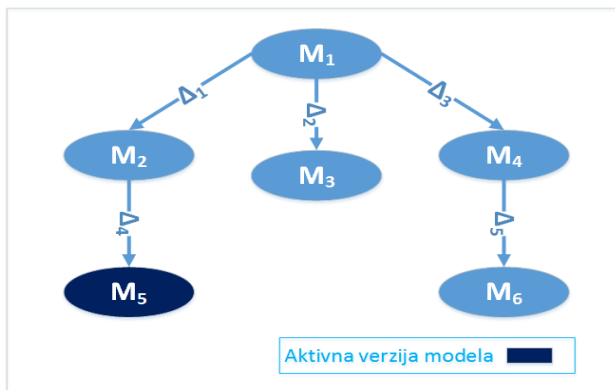
Način ažuriranja modela je standardizovan od strane organizacije *International Electrotechnical Commission (IEC)* [2]. Konceptualni opis načina ažuriranja modela je prikazan na slici 1.



Slika 1. Opis modela u skladu sa IEC standardom [2]

DifferenceModel predstavlja inkrementalnu promenu modela koja sadrži uređen skup operacija ažuriranja. Operacije ažuriranja su sastavljene od operacija koje obavljaju dodavanje objekata, brisanje objekata ili ažuriranje atributa objekata u modelu. Pomenute operacije su opisane klasom *Statements*, koja je povezana sa *DifferenceModel* klasom relacijama *forwardDifferences* i *reverseDifferences*. Relacija *forwardDifferences* opisuje skup operacija koje ažuriraju model i dovode ga u buduće stanje, dok relacija *reverseDifferences* sadrži operacije koje opisuju zatečeno stanje modela pre primene odgovarajućih operacija ažuriranja.

Na slici 2 je prikazan generički način održavanja modela opisan pomoću usmerenog grafa. Čvorovi grafa (označeni simbolima M_i) predstavljaju verzije modela, dok grane grafa (označene simbolima Δ_i) opisuju skup operacija dodavanja, brisanja ili ažuriranja objekata. Verzija modela opisuje novo stanje modela koje nastaje atomskom primenom operacija ažuriranja. Grane grafa opisuju zavisnost između verzija modela, što je na slici 1 opisano relacijom *DependingOn*. Zavisnost uvodi ograničenje da se izmene sadržane u jednoj verziji moraju primeniti pre primene izmena sadržanih u drugoj verziji modela.



Slika 2. Kreiranje verzija modela

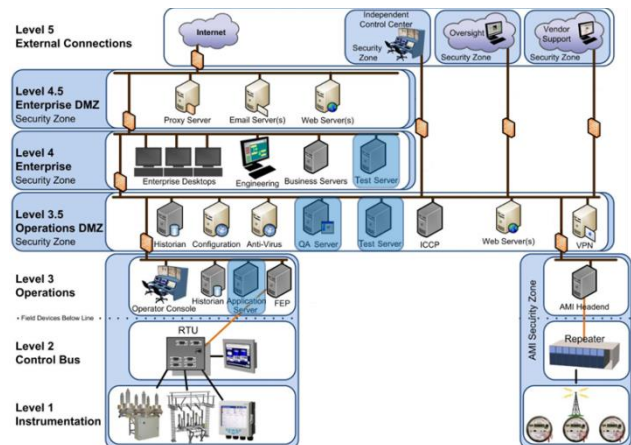
U jednom trenutku samo jedna verzija modela može biti aktivna. Aktivna verzija predstavlja trenutno stanje modela, koje se koristi za proračune i koje nastaje primenom operacija ažuriranja na prethodno aktivno stanje. Prema tome, ažuriranje modela se svodi na kreiranje nove verzije i promenu aktivne verzije. Na slici 2 ulogu aktivne verzije modela ima verzija M_5 .

2.3 Fizička arhitektura sistema odgovornog za održavanje modela

Održavanje modela se u praksi realizuje u okviru distribuiranog sistema sastavljenog iz više nezavisnih računarskih komponenti, koje mogu biti raspoređene u više bezbednosnih zona [4]. Podela sistema u zone omogućava lakšu kontrolu pristupa, čime se sprečava neovlašćena izmena podataka kao i unošenje grešaka u sistem [4]. Slika 3 prikazuje prilagođenu generičku arhitekturu distribuiranog sistema, koji kontroliše određeni industrijski proces. Primer takvog procesa može da bude upravljanje pametnom mrežom, što znači da se distribuirana arhitektura sa slike 3 može upotrebiti u tu svrhu.

U ovom radu se analizira podela sistema na zone sa stanovišta održavanja i ažuriranja modela. U skladu s tim,

u svakoj zoni treba da postoji definisan skup operacija koje se mogu izvršavati nad modelom, što znači da se ažuriranje modela može izvršavati samo na određenom broju servera smeštenih u različitim zonama. To implicira



Slika 3. Arhitektura industrijskog kontrolnog sistema [4]

da će se povremeno razlikovati modeli na serverima, zbog čega je periodično potrebno izvršiti poravnavanje modela na svim serverima, što se naziva sinhronizacija modela. Sinhronizacijom koordinira softverska komponenta smeštena u zoni najveće odgovornosti, koja se u kontekstu ovog rada zove *Synchronization Manager Service* (SyMS). Da bi se mogao realizovati proces sinhronizacije potrebno je da na svakom serveru postoji instalirana softverska komponenta koja obavlja lokalno ažuriranje modela. Ova komponenta se u kontekstu ovog rada zove *Model Manager Service* (MoMS). Sinhronizacija će izvršiti poravnavanje modela, tako da će izmene modela sa *Application* i QA servera primeniti na svim serverima koji ne poseduju te izmene.

Na slici 3 su plavom bojom označeni serveri na kojima se vrši održavanje modela. To znači da je *MoMS* instaliran na *Application* serveru, QA (Quality Assurance) serveru i na Test serverima. *MoMS* održava i ažurira modela na način koji je opisan u poglavlju 2.2. *Application* server se koristi za upravljanje pametnom mrežom, što znači da je SyMS instaliran na ovom serveru. Na QA serveru se vrši priprema izmena modela koje treba primeniti na *Application* serveru. Modeli na Test serverima se mogu ažurirati po potrebi, ali se te izmene ne propagiraju do *Application* servera.

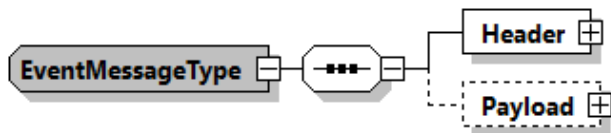
3. VIZUELIZACIJA PROCESA AŽURIRANJA MODELA

3.1 Implementacija aplikacije za vizuelizaciju

Predviđeno je da aplikacija bude integrisana u proizvoljan softverski sistem koji vrši upravljanje pametnom mrežom ili nekim podsistemom u okviru nje. Prikazana arhitektura na slici 3 predstavlja generičko okruženje u okviru kojeg bi aplikacija funkcionisala. Arhitektura konkretnog sistema se može razlikovati u određenoj meri, po broju zona, broju servera ili tipovima servera. Isto tako se može razlikovati i implementacija SyMS i MoMS komponenti.

Da bi aplikacija funkcionisala na odgovarajući način, potrebno da komunicira sa SyMS komponentom. SyMS generiše značajne događaje tokom sinhronizacije koje

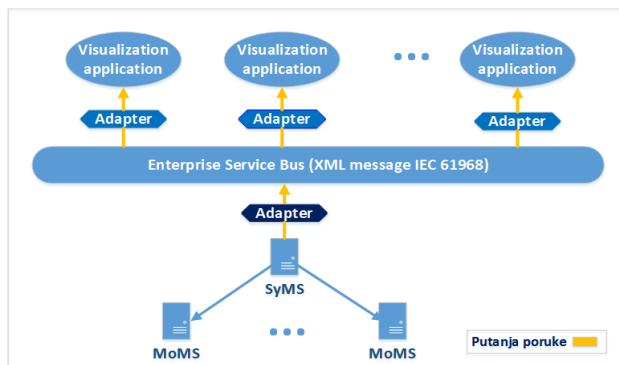
aplikacija obrađuje i na osnovu njih generiše vizuelni prikaz. Pošto nije poznata implementacija konkretne SyMS komponente, realizovan je generički oblik komunikacije, upotrebom poruka definisanih u skladu sa standardom IEC 61968. Ovaj standard definiše način razmene podataka između aplikacija u obliku poruka opisanih *Extensible Markup Language (XML)* jezikom [3]. Struktura tih poruka je definisana *XML Schema Definition (XSD)* jezikom [3]. Postoji više tipova poruka, ali je za potrebe implementacije podsistema za vizuelizaciju izabran *Event* tip poruke. Struktura *Event* poruka je prikazana na slici 4.



Slika 4. Struktura Event poruke [3]

Event poruka mora imati *Header* element koji sadrži metapodatke, dok *Payload* element prenosi podatke od interesa. *Payload* može biti definisan proizvoljnom XML šemom, što je iskorišteno da se u okviru *Payload* elementa prenose podaci koje će SyMS slati aplikaciji za vizuelizaciju.

Predviđeno je da se *Event* poruke transportuju posebnim softverskim podsistemom koji se naziva *Enterprise Service Bus (ESB)*. U opštem slučaju *ESB* omogućava povezivanje i komunikaciju različitih aplikacija kod kojih je otežana, a ponekad i nemoguća direktna komunikacija [3].

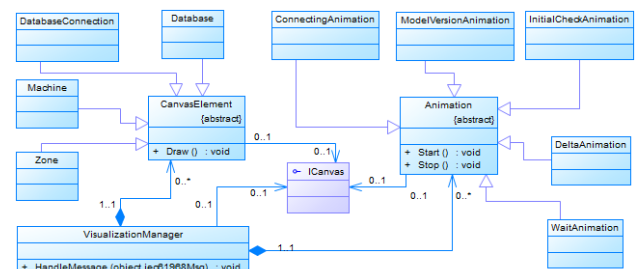


Slika 5. Arhitektura sistema za vizuelizaciju

Na slici 5 je prikazana arhitektura sistema za vizuelizaciju. Ključnu ulogu u ovom sistemu predstavlja *ESB*, jer povezuje *SyMS* komponentu sa proizvoljnim brojem aplikacija koje obavljaju vizuelni prikaz sinhronizacije. Aplikacija na taj način može komunicirati sa proizvoljnom implementacijom *SyMS*-a. *Event* poruke nastaju na *SyMS*-u koji se nalazi u zoni najveće odgovornosti. To znači da neće biti narušena bezbednost celokupnog sistema, jer će *Event* poruke putovati ka zonama istog ili nižeg nivoa bezbednosti.

Aplikacija je implementirana u *C#* programskom jeziku, upotrebom *Windows Presentation Foundation (WPF)* framework-a. Na slici 6 su prikazane najvažnije klase koje obavljaju vizuelizaciju. *VisualizationManager* predstavlja centralnu klasu, koja poseduje kolekciju animacija i kolekciju statičkih vizuelnih elemenata (zone, serveri,

baze podataka, veze između baza). Poseduje metodu *HandleMessage* u okviru koje se obrađuje poruka pristigla sa *SyMS*-a, što rezultuje kreiranjem animacije odgovarajućeg tipa. Tu metodu poziva *Adapter* koji ima ulogu da poveže *ESB* sa aplikacijom za vizuelizaciju, što je ilustrovano na slici 5.

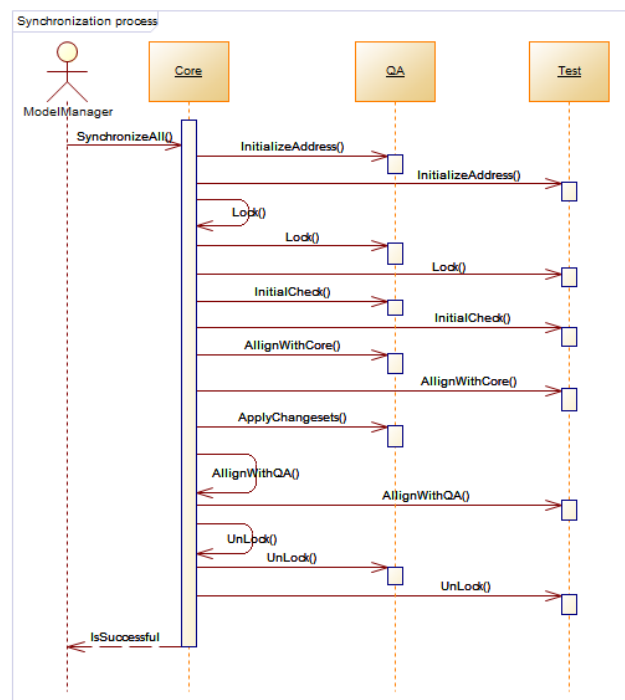


Slika 6. Dijagram klase

Celokupna logika koja realizuje prikaz animacija se nalazi u *Animation* klasi i klasama koje nasleđuju tu klasu. To omogućava jednostavno dodavanje novog tipa animacije, što se svodi na kreiranje nove klase koja nasleđuje klasu *Animation*.

3.2 Rezultati rada aplikacije za vizuelizaciju

U ovom poglavlju su prikazani rezultati rada aplikacije. Aplikacija vizuelno prikazuje proces sinhronizacije koji je na slici 7 opisan dijagramom sekvence.



Slika 7. Sequence diagram uspešnog procesa sinhronizacije

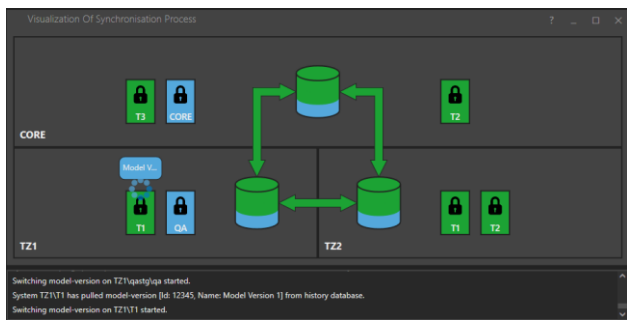
Core server sa dijagrama predstavlja *Application* server prikazan na slici 3. *Test* serveri sa slike 3 imaju isto ponašanje tokom sinhronizacije pa je na dijagramu zbog jednostavnosti prikazan samo jedan *Test* server.

Korisnik inicira proces sinhronizacije pozivom operacije *SynchronizeAll*. U nastavku *SyMS* prvo izvršava pripremne aktivnosti koje uključuju uspostavljanje veze sa *MoMS* komponentama (*InitializeAddress*),

zaključavanje modela na svim serverima (*Lock* – čime se sprečava konkurentna izmena modela) kao i obavljanje niza provera koje mogu biti značaja za nastavak sinhronizacije (*InitialCheck*). Nakon toga se vrši poravnavanje sa modelom na *Core* serveru (*AlignWithCore*) i sa modelom na *QA* (*AlignWithQA*) serveru. Operaciji *AlignWithQA* prethodi operacija *ApplyChangesets* u kojoj se primenjuju izmene na model *QA* servera. Podrazumeva se da su izmene na *Core* serveru primenjene pre početka sinhronizacije. Na kraju se pozivom operacije *UnLock* vrši otključavanje modela.

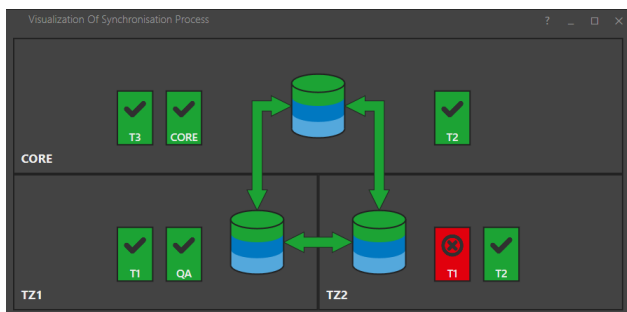
Aplikacija vizuelno prikazuje u obliku animacija sve korake tokom sinhronizacije koji su definisani na slici 7. Na slikama 8 i 9 su prikazana dva ilustrativna primera koji opisuju dva perioda tokom sinhronizacije.

Arhitektura sistema sa slike 3 je za potrebe testiranja proširena dodatnim *Test* serverima, kojih u opštem slučaju može postojati proizvoljan broj. Način na koji aplikacija vizuelno prikazuje ovako definisanu arhitekturu je ilustrovan na slici 8. *Core* zona odgovara zoni na nivou 3, a zone *Test Zone 1 (TZ1)* i *Test Zone 2 (TZ2)* odgovaraju respektivno zonama 3.5 i 4 sa slike 3. *Core* server predstavlja *Application* server, dok su *Test* serveri označeni sa *Ti*.



Slika 8. Vizuelizacija sinhronizacije

Sa slike 8 se zaključuje da je model na *QA* serveru poravnat sa modelom *Core* servera. Još se zapaža da je započeto izvršavanje operacije *AlignWithCore* na *T1* serveru u zoni *TZ1*. Verzija modela sa *Core* servera je preuzeta iz baze u *TZ1* zoni, a nakon toga je pozicionirana iznad *T1* servera, što pokazuje da se primenjuju izmene iz te verzije na *T1* serveru. Simboli baza podataka u svim zonama su delom obojeni bojom verzije modela koja potiče sa *Core* servera. To znači da je ta verzija upisana u sve baze, čime je omogućeno preuzimanje pomenute verzije na svim serverima.



Slika 9. Ilustracija završetka sinhronizacije

Na slici 9 je ilustrovan završetak sinhronizacije. Serveri, čiji su modeli uspešno poravnati, su označeni simbolom

koji uopšteno pokazuje uspešan završetak određene aktivnosti. Na *T1* serveru u *TZ2* zoni nije uspešno izvršena primena izmena na model koje potiču sa *Core* i *QA* servera, pa je taj server označen crvenom bojom i adekvatnim simbolom, koji označava neuspešan završetak određene aktivnosti.

Važno je napomenuti, da su sve vizuelne aktivnosti praćene odgovarajućim objašnjenjima u vidu poruka koje se prikazuju ispod prostora za vizuelizaciju.

Ponašanje *SyMS*-a i *ESB*-a je simulirano upotrebom *Windows Communication Foundation (WCF)* servisa, implementiranih u *C#* programskom jeziku.

4. KONTEKST UPOTREBE APLIKACIJE ZA VIZUELIZACIJU

Implementirana aplikacija se može koristiti u okviru sistema zaduženog za upravljanje pametnim mrežama. Međutim, generički pristup pri implementaciji omogućava da se aplikacija primeni i u drugim softverskim sistemima koji održavaju modele u distribuiranom okruženju. Kao primer takvog sistema se može navesti softverski sistem koji kontroliše distribuciju gasa.

5. ZAKLJUČAK

Softverski sistemi zaduženi za upravljanje pametnim mrežama su po svojoj strukturi veoma složeni. To se najbolje zaključuje na osnovu analize, prikazane u okviru ovog rada, koja se odnosi na podsistem odgovoran za održavanje modela. Neprekidni zahtevi, u smislu proširivanja novih funkcionalnosti, dovode do toga da pomenuti sistemi postaju vremenom sve kompleksniji, te je teško ispratiti aktuelno stanje modela. Implementirana aplikacija za vizuelizaciju predstavlja pokušaj da se korisniku olakša pristup informacijama o aktuelnom stanju modela tako kompleksnog sistema.

6. LITERATURA

- [1] V. Strezoski: Osnovi elektroenergetike, Univerzitet u Novom Sadu, FTN, 2014
- [2] EPRI: Common Information Model Primer, Third Edition, 2015
- [3] EPRI: Enterprise Service Bus Implementation Profile, 2009
- [4] U.S. Department of Energy: Secure Data Transfer Guidance for Industrial Control and SCADA Systems, 2011

Kratka biografija:



Milan Kostić je rođen 03.01.1993. godine u Bijeljini. Upisao se 2012. godine na Fakultet tehničkih nauka, studijski program Računarstvo i automatika. Osnovne akademske studije je završio 2016. godine. Iste godine upisuje master akademske studije, studijski program Primenjeno softversko inženjerstvo.

U realizaciji Zbornika radova Fakulteta tehničkih nauka u toku 2017. godine učestvovali su sledeći recenzenti:

Aco Antić	Dušan Gvozdenac	Milan Trifković	Slavko Đurić
Aleksandar	Dušan Kovačević	Milan Trivunić	Slobodan Dudić
Erdeljan	Dušan Uzelac	Milan Vidaković	Slobodan Krnjetin
Aleksandar Ristić	Duško Bekut	Milena Krklješ	Slobodan Morača
Bato Kamberović	Đorđe Ćosić	Milica Kostreš	Sonja Ristić
Biljana Njegovan	Đorđe Lađinović	Milica Miličić	Srđan Kolaković
Bogdan	Đorđe Obradović	Milinko Vasić	Srđan Popov
Kuzmanović	Đorđe Vukelić	Miloš Slankamenac	Srđan Vukmirović
Bojan Batinić	Đura Oros	Milovan Lazarević	Staniša Dautović
Bojan Lalić	Đurđica Stojanović	Miodrag	Stevan Milisavljević
Bojan Tepavčević	Filip Kulić	Hadžistević	Stevan Stankovski
Bojana Beronja	Goran Sladić	Miodrag Zuković	Strahil Gušavac
Branislav Atlagić	Goran Švenda	Mirjana	Svetlana Nikoličić
Branislav Nerandžić	Gordana	Damnjanović	Tanja Kočetov
Branislav Veselinov	Milosavljević	Mirjana Malešev	Tatjana Lončar
Branislava Kostić	Gordana Ostojić	Mirjana Radeka	Turukalo
Branislava	Igor Budak	Mirko Borisov	Todor Bačkalić
Novaković	Igor Dejanović	Miro Govedarica	Toša Ninkov
Branka Nakomčić	Igor Karlović	Miroslav	Uroš Nedeljković
Branko	Ivan Beker	Hajduković	Valentina Basarić
Milosavljević	Ivana Katić	Miroslav Plančak	Velimir Čongradec
Branko Škorić	Ivana Kovačić	Miroslav Popović	Velimir Todić
Cvijan Krsmanović	Jasmina Dražić	Mitar Jocanović	Veran Vasić
Damir Đaković	Jelena Atanacković	Mladen Kovačević	Veselin Avdalović
Danijela Lalić	Jeličić	Mladen Radišić	Veselin Perović
Darko Čapko	Jelena Borocki	Momčilo Kujačić	Vladimir Katić
Darko Marčetić	Jelena Kiurski	Nađa Kurtović	Vladimir
Darko Reba	Jelena Radonić	Nebojša Pjevalica	Radenković
Dejan Ubavin	Jovan Petrović	Neda Pekarić Nađ	Vladimir Strezoski
Dragan Ivanović	Jovan Vladić	Nemanja	Vladimir Škiljajica
Dragan Ivetić	Jovanka Pantović	Stanisavljević	Vlado Delić
Dragan Jovanović	Karl Mičkei	Nenad Katić	Vlastimir
Dragan Kukolj	Katarina Gerić	Nikola Brkljač	Radonjanin
Dragan Mrkšić	Ksenija Hiel	Nikola Đurić	Vuk Bogdanović
Dragan Pejić	Laslo Nađ	Nikola	Zdravko Tešić
Dragan Šešlija	Leposava Grubić	Jorgovanović	Zoran Anišić
Dragana Bajić	Nešić	Nikola Radaković	Zoran Brujić
Dragana	Livija Cvetičanin	Ninoslav Zuber	Zoran Jeličić
Konstantinović	Ljiljana Vukajlov	Ognjen Lužanin	Zoran Mijatović
Dragana Šarac	Ljiljana Cvetković	Pavel Kovač	Zoran Milojević
Dragana Štrbac	Ljubica Duđak	Peđa Atanasković	Zoran Mitrović
Dragiša Vilotić	Maja Turk Sekulić	Petar Malešev	Zoran Papić
Dragoljub	Maša Bukurov	Predrag Šiđanin	Željen Trpovski
Novaković	Matija Stipić	Radivoje Dinulović	Željko Jakšić
Dragoljub Šević	Milan Rackov	Radovan Štulić	
Dubravka Bojanić	Milan Rapajić	Rastislav Šostakov	
Dušan Dobromirov	Milan Simeunović	Slavica Mitrović	

