УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

ЗБОРНИК РАДОВА
ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке - зборници
Година: XXXVIII
Број: 7/2023

Нови Сад

Едиција: „Техничке науке – Зборници“
Година: XXXVIII Свеска: 7

Издавач: Факултет техничких наука Нови Сад
Главни и одговорни уредник: проф. др Срђан Колаковић, декан Факултета техничких наука у Новом Саду

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
Срђан Колаковић. – 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 седма 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 web сајту ФТН-а (www.ftn.uns.ac.rs) и штампаном, који је пред вама. Обе верзије публикују се сваки месец, у оквиру промоције дипломираних мастера.

У овом броју штампани су радови студената мастер студија, сада већ мастера, који су радове бранили у периоду од 7.11.2022. до 18.04.2023. год., а који се промовишу 18.05.2023. 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 7. објављени су радови из области:

- електротехнике и рачунарства.

У свесци са редним бројем 6. објављени су радови из области:

- машинства,
- грађевинарства,
- саобраћаја,
- графичког инжењерства и дузајна,
- архитектуре,
- управљања ризиком од катастрофалних догађаја и пожара и
- инжењерства информационих система.

У свесци са редним бројем 8. објављени су радови из области:

- инжењерског менаџмента,
- мехатронике,
- математике у техници,
- геодезије и геоматике и
- биомедицинског инжењерства.

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

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4
DOI: https://doi.org/10.24867/23BE02Nosovic

МИКРОСЕРВИСНО РЕШЕЊЕ ЗА ПРЕДВИЂАЊЕ ПОТРОШЊЕ ЕЛЕКТРИЧНЕ ЕНЕРГИЈЕ У *CLOUD* ОКРУЖЕЊУ

A MICROSERVICE SOLUTION FOR PREDICTING ELECTRICITY POWER CONSUMPTION IN A CLOUD ENVIRONMENT

Душан Носовић, *Факултет техничких наука, Нови Сад*

Oblast – ПРИМЕЊЕНО СОФТВЕРСКО ИНЖЕЊЕРСТВО

Kratak sadržaj – *Опис, анализа и приказ резултата времена извршавања тренинга модела машинског учења и предвиђања потрошње електричне енергије. Приказани резултати времена извршавања су тестирани на различитим Cloud computing платформама које се заснивају на PaaS моделу Cloud computing-а.*

Ključne reči: *Cloud computing, Микросервисна архитектура, Platform as a Service, Машинско учење.*

Abstract – *Description, analysis and presentation of machine learning model training execution time results and prediction of electricity power consumption. Presentation of execution time results tested on various Cloud computing platforms based on the PaaS model of Cloud computing.*

Keywords: *Cloud computing, Microservice arhitecture, Platform as a service, Machine learning.*

1. УВОД

У претходних неколико година долази до убрзаног развоја технике и технологије, чија је последица настанак нових технологија, што доводи до промена у свим сферама софтверске индустрије. Једну од најважнијих и најпримењенијих нових технологија представља *Cloud computing*, који доводи до есенцијалних промена у рачунарству. *Cloud computing* омогућава стварање нових архитектура и апликација, и утиче на начин на који размишљамо о развоју и примени софтвера. Битност *Cloud computing* технологије потврђују водеће светске компаније које су се усресредиле на његов развој и примену.

Поред бенефита у софтверској индустрији, *Cloud computing* увео је многе промене које су утицале и на бизнис. Капиталне инвестиције у локални хардвер сведене су на минимум и замењене су оперативним улагањима у *Cloud computing*. Такође, он је увео и нов начин плаћања по принципу: "плати колико потрошиш", што је резултирало драстичним смањењем трошкова.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Себастијан Стоја.

Један од могућих начина имплементације *Cloud computing* окружења представља микросервисни архитектурални приступ. Филозофија микросервисног архитектуралног стила суштински се изједначава са *Unix* филозофијом: "Ради једну ствар и ради је добро". Историја и порекло микросервиса подразумева сталне напоре да се обезбеди боља комуникација између различитих платформи, једноставнијих система и система једноставнији за корисника.

Главни задатак овог рада јесу детаљан опис, анализа и приказ резултата времена извршавања тренинга модела машинског учења и предвиђања потрошње електричне енергије.Приказани резултати времена извршавања тестирани су на различитим *Cloud computing* платформама које се заснивају на *PaaS* моделу *Cloud computing*-а. У раду се директно пореде *AWS Lambda* и *Azure Functions*. Поређење укључује резултате извршавања, анализу резултата, приказ могућности, као и предности и мане оба решења.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1 Cloud Computing

Cloud computing [2] је све заступљенији приступ пружања рачунарских услуга на флексибилан, ефикасан и лако доступан начин. Представља суштинску промену у начину на који компаније и појединци предају одговорност компанијама које пружају *cloud* услуге за обезбеђивање *IT* инфраструктуре и услуга. Традиционални *IT outsourcing* аранжмани обухватају уговоре са јасно дефинисаним објектима за складиштење и обраду података. За разлику од класичне инфраструктуре, количина *IT* ресурса може да се мења током времена, често брзо и динамично као одговор на тренутну потражњу за *IT* ресурсима. Купац ресурса углавном није свестан где се налази услужна инфраструктура.

Cloud-base пословни модел обухвата следеће:

- Смањење трошкова: дељењем ресурса између групе купаца и куповином инфраструктуре на велико, гигантске мултинационалне компаније које пружају услуге *Cloud computing*-а могу постићи уштеде које ће се пренети на купце ресурса,
- Трансформисање *CAPEX* у *OPEX*: премештање пословних операција у *Cloud* омогућава смањење трошкова за интерну *IT* инфраструктуру,
- Краће време до пословне примене: могућност брзог добављања нових капацитета скраћује време од развоја до пословне примене,

- Заштита животне средине: иако велики центри података изгледају као интезивни потрошачи енергије, у пракси се показало да је управљање једним таквим центром података енергетски ефикасније од еквивалетног капацитета састављеног од појединачних рачунара и сервера.

Амерички аналитичар Гартнер дефинише *Cloud computing* као: *"Стил рачунарства где су скалабилне и еластичне IT могућности обезбеђене као услуга за више корисника који користе интернет технологије."*

Иако ова дефиниција није савршена, она обухвата кључне атрибуте које чине *Cloud computing* привлачним купцима. Под скалабилним подразумева се количина коришћених рачунарских ресурса, било да се ради о обради или складиштењу података. Под појмом еластична, Гартнер подразумева да се скалирање може обавити брзо као одговор на промене у потражњи за ресурсима.

Једну од главних предности *Cloud computing* представља дељење скупа рачунарских ресурса између клијената. Клијенти обично имају променљиве захтеве за ресурсима, који ретко достижу максимум у исто време, тако да провајдери *Cloud* услуга обезбеђују базно оптерећење свих клијената које је мање од максималног оптерећења свих клијената.

Cloud computing је веома широк појам који се користи за описивање различитих аспеката у рачунарству. Модели *cloud computing*-а се деле на различите нивое услуга према степену виртуелизације [1], а то су *Platform as a service* (PaaS), *Infrastructure as a Service* (IaaS) и *Software as a Service* (SaaS).

2.2 Infrastructure as a Service

IaaS [3] је облик *hosting*-а, који укључује приступ мрежи, услуге рутирања, и складиштење. *IaaS* провајдер ће генерално обезбедити *hardware*, административне услуге потребне за чување апликација и платформу за покретање апликација. Добављач услуга поседује опрему и задужен је за њено складиштење, вођење и одржавање. *Infrastructure as a service* је еволуција традиционалног *hosting*-а који не захтева никакву дугорочну посвећеност и омогућава корисницима да обезбеде ресурсе на захтев. За разлику од *PaaS*, *IaaS* провајдери су задужени за управљање само у случају одржавања оперативног *Data* центра. Корисници примењују софтверске услуге и управљају њима баш онако како би то радили у својим *Data* центрима.

2.3 Platform as a Service

PaaS [4] је платформа која се гради на врху *IaaS* слоја, и обезбеђује комплетну инфраструктуру потребну за рад апликације преко интернета. Корисници *PaaS* не размишљају о позадинским процесима и сложености позадинског система. Она пружа платформу за креирање софтвера и омогућава програмерима да праве прилагођене апликације на мрежи без потребе да се баве сервирањем, складиштењем и управљањем података. У суштини, *PaaS* модел услуга омогућава предузећима да се концентришу на сам софтвер без промене постојеће инфраструктуре. Као и услужни програми, *PaaS* модел и даље пружа клијентима сервере и центре података у којима могу да похрањују своје информације, али у случају *PaaS* развијају једну апликацију која ће потом бити испоручена преко интернета. *PaaS* омогућава лак прелазак на хибридни модел *Cloud*-а, који представља комбинацију различитих окружења, а такође обезбеђује и приступ различитим ресурсима у групи апликација, укључујући програмске језике, оперативне системе и базе података. *PaaS* је заснован на моделу мерења или претплате, тако да корисници плаћају оно што потроше (енг. *Pay as you go model*).

2.4 Software as a Service

SaaS [1] је најчешће коришћен од свих слојева на *Cloud*-у гради се на врху *IaaS* и *PaaS* слојева. *SaaS* представља алтернативу локално покренутим апликацијама и стога је најпривлачнији крајњим корисницима. Потпуно је скалабилан и не мора да се преузима или инсталира на појединачним уређајима да би се применио на читав тим или компанију. Ова функција је посебно корисна за дистрибуиране глобалне тимове и хибридна радна окружења.

2.5 Микросервисна архитектура

Микросервисна архитектура [5] је архитектура сервисно-оријентисаног архитектуалног стила који организује апликацију у колекцију слабо повезаних услуга. Састоји се од колекције малих, аутономних услуга. Свака услуга је самостална и треба да имплементира једну пословну способност у оквиру система. Свака услуга је засебна, којом може да управља мали развојни тим. Сервиси се развијају независно, омогућавајући на тај начин унапређење одређеног сервиса без утицаја на друге делове система. Услуге комуницирају преко јасно дефинисаних *API*-ја скривајући на тај начин детаље интерну имплементацију других услуга.

3. ARHITEKTURA

Уводни део рада је послужио за представљање *Cloud computing*-а, микросервисне архитектуре, као и коришћених технологија. У овом поглављу биће анализирана архитектура имплементираног решења. Посебно ће бити објашњена свака компонента *Azure functions* и *AWS Lambda* система како би се што боље стекао увид у улогу сваке компоненте.

Апликација је намењена за тренирање модела машинског учења употребом *ML.NET*-а, предвиђање потрошње електричне енергије, као и приказ резултата предвиђања. Улазни подаци тренинга представљају временске сатне податке и сатне податке потрошње електричне енергије у периоду од 2016. до 2019. године. У оквиру једног микросервиса имплементиран је алгоритам машинског учења врши корелацију временских услова и потрошње електричне енергије за наведени период. Такође, имплементиран је алгоритам за филтрирање и парсирање података, као последњи корак подаци се деле на независне (временски) и зависне (потрошња електричне енергије) прослеђују алгоритму машинског учења. Резултат тренинга је модел машинског учења, пошто се ради о неструктуираним подацима, неопходно је њихово складиштење на *S3 Bucket* или *Azure Blob*. Предвиђање резултата потрошње уз коришћење сачуваних и истренираних

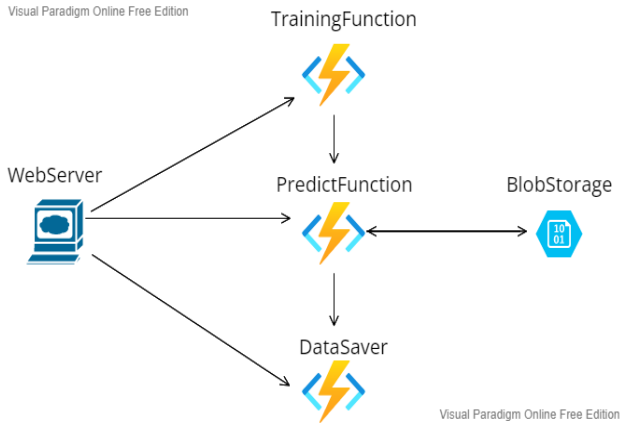
модела представља посебан микросервис. Приликом предвиђања у оквиру прослеђеног документа садржани су само независни подаци, а зависни подаци представљају резултат предвиђања на основу независних података.

Чување резултата предвиђања имплементирано је као посебан микросервис.

У оквиру микросервиса локално се чувају резултати последњег предвиђања. Клијент приликом тренинга и предвиђања податке шаље путем *HTTP* захтева у *(.xlsx)* документу.

3.1. Архитектура решења Azure functions

Имплементирано решење заснива се на микросервисној архитектури коришћењем *Azure functions* сервиса и приказана је на слици 1.



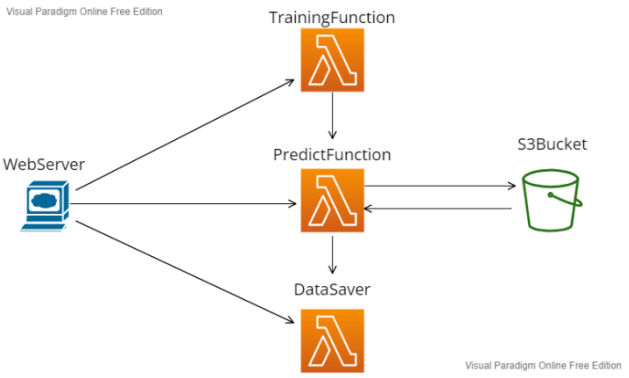
Слика 1. *Архитектура Azure решења.*

Компоненте система су следеће:

- *Web Server – On-premise* компонента која представља везу између *Azure functions* и корисника. Омогућава графички интерфејс за крајњег корисника као и комуникацију између корисника и одговарајућих функција система,
- *Training Function* – Прослеђени документ од стране корисника парсира и филтрира, затим са тим подацима врши тренинг података. Добијени модел машинског учења прослеђује функцији *Predict Function*,
- *Prediction Function* - Добијен модел чува локално и на *Blob Storage*-у. Приликом предвиђања парсира и филтрира улазни документ затим уз помоћ сачуваног модела машинског учења врши предикцију података. Добијене податке прослеђује функцији *Data Saver*,
- *Data Saver* – Добијене податке чува локално. На захтев клијента прослеђује податке на *Web Server*, и
- *Blob Storage* – Чува последње истренирани модел у *.Zip* формату.

3.2 Архитектура решења AWS Lambda

Имплементирано решење заснива се на микросервисној архитектури коришћењем *AWS Lambda* сервиса и приказана је на слици 2.



Слика 2. *Архитектура AWS решења.*

Компоненте система су следеће:

- *Web Server – On-premise* компонента која представља везу између *AWS Lambda* и корисника. Омогућава графички интерфејс за крајњег корисника као и комуникацију између корисника и одговарајућих функција система,
- *Training Function* – Прослеђени документ од стране корисника парсира и филтрира, затим са тим подацима врши тренинг података. Добијени модел машинског учења прослеђује функцији *Predict Function*,
- *Prediction Function* - Добијен модел чува локално и на *S3*. Приликом предвиђања парсира и филтрира улазни документ затим уз помоћ сачуваног модела машинског учења врши предикцију података. Добијене податке прослеђује функцији *Data Saver*,
- *Data Saver* – Добијене податке чува локално. На захтев клијента прослеђује податке на *Web Server*, и
- *S3 Bucket* - Чува последње истренирани модел у *.Zip* формату.

4. РЕЗУЛТАТИ ТЕСТИРАЊА

У овом поглављу биће анализирани резултати времена извршавања *Azure functions* и *AWS Lambda.* Да би добијени резултати били што релевантнији апликације су биле подигнуте у оквиру својих *Cloud* окружења, улазни подаци приликом тестирања увек су били исти, као и број итерација приликом тренинга.

Потребан услов сваког теста: Софтверско решење за тренирање и превиђање и тренутно чување резултата је подигнуто на *Cloud*, покренуто и подешено за рад. У овом раду су извршена три тест сценариа:

Сценарио тест (1):

- Покретање одговарајућег веб сервера,
- Одабир тренинга на графичком интерфејсу,
- Одабир одговарајућег документа за тренинг, и
- Иницирање функције *Training.*

Сценарио тест (2):

- Покретање одговарајућег веб сервера,

- Одабир предвиђања на графичком интерфејсу,
- Одабир одговарајућег документа за предвиђање података, и
- Иницирање функције *Predict.*

Сценарио тест (3):

- Покретање одговарајућег веб сервера,
- Одабир функције *Data Saver* и иницирање исте, и
- Приказ података предвиђања.

Табела 1. Поређење времена извршавања фунцкија у зависности од Cloud платформе.

Тестирана функција	**Просечно време извршавања Azure**	**Просечно Време извршавања AWS Lambda**
Сценарио тест (1), време извршавања функције[s]	38.4999	75.2485
Сценарио тест (1), HTTP Захтев и одговор [s]	4.3318	3.4144
Сценарио тест (2), време извршавања функције[s]	2.0031	2.5968
Сценарио тест (2), HTTP Захтев и одговор[s]	0.6934	0.617
Сценарио тест(3), време извршавања функције[s]	0.0000633	0.0002166
Сценарио тест(3), HTTP Захтев и одговор[s]	0.66176	0.1959

Мерењима из табеле 1 показана је разлика у брзини извршавања функција у зависности од платформе. Треба приметити да приликом извршавања процесорски захтевних функција као што је тренирање модела машинског учења *Azure functions* остварују 48.99% боље време извршавања функције. Приликом преноса веће количине података, као што су подаци за тренинг *AWS Lambda* је у просеку за 21.2% остварио бољи резултат. Мање процесорски захтевно предвиђање резултата доводи до смањења разлике у времену извршавања, па приликом предвиђања *Azure functions* остварује 22.86% бољи резлутат. Слање мање количине података у захтеву приликом предвиђања резултата одржава се на мању разлику између *AWS Lambda* и *Azure Functions* у реализацији захтева и одговара, у случају предвиђања *AWS Lambda* је просечно 0.0765 секунди брже реализовала захтев и одговор у односу на *Azure Functions* што представља просечно 11% мање времена за реализацију захтева и одговара. Добављање података представља процесорски најмање захтевну операцију, иако се ради о делићима секунде *Azure Functions* је приликом извршавања функције остварио 70% боље просечно време извршавања. Треба приметити да *AWS Lambda* треба просечно 0.4648 секунди мање за реализацију захтева, извршањање фунцкије и одговор што представља 70.35 % мање времена потребног за извршавање захтева у односу на *Azure Functions.* Иако је време укупно време извршавања краће треба узети у обзир више времена потребног за извршење функције што може довести до већих трошкова у односу на *Azure Functions.*

5. ЗАКЉУЧАК

Модерна софтверска решења све чешће користе микросервисни архитектурални стил у оквиру *Cloud computing* платформи. За разлику од монолитне архитектуре која се састоји од само једног сервиса, микорсервисна архитектура представља колекцију малих независних сервиса. Рад детаљно описује и анализира резултате два микросервисна сотверска решења у оквиру две различите платформе: *PaaS* модела и *Cloud computing*-а. Анализиране су технологије и стандарди коришћени приликом израде одговарајућих софтверских решења и предочене су предности коришћења *Cloud computing* микросервисне архитектуре.

6. ЛИТЕРАТУРА

[1] S. Nagaprasad, A. VinayaBabu, K. Madhukar, D. Marlene G Verghese, V. Mallaiah and A. Sreelatha, "Reviewing some platforms in cloud computing", *International Journal of Engineering and Technology*, vol. 2, no.5, pp. 348-353, 2010.

[2] S. Bradshaw, C. Millard, and I. Walden, "Contracts for clouds: Comparison and analysis of the terms and conditions of cloud computing services", *International Journal of Law and Information Technology*, vol. 19, no. 3, pp. 187–223, 2011.

[3] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (IAAS)" *International Journal of engineering and information Technology*, vol. 2, no. 1, pp. 60-63, 2010.

[4] K. Gurudatt, P. Khatawkar, and J. Gambhir. "Cloud computing-platform as service", *International Journal of Engineering*, vol. 1, 2011.

[5] EdPrice-MSFT, "Microservice architecture style - azure architecture center," Azure Architecture Center | Microsoft Learn. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices. [Accessed: 24-Oct-2022].

Kratka biografija:

Душан Носовић рођен је 1996. године у Шапцу. Завршио је Шабачку гимназију 2015. године. Факултет техничких наука у Новом Саду уписао је 2015. године, смер Примењено софтверско инжењерство. Завршио је основне студије 2021. године и након уписао мастер студије на Факултету техничких наука у Новом Саду, смер Примењено софтверско инжењерство.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE03Jenjic

УТИЦАЈ *REACT LAZY* И *WEBPACK* КОНФИГУРАЦИЈА НА ПЕРФОРМАНСЕ АПЛИКАЦИЈЕ

IMPACT OF REACT LAZY AND WEBPACK CONFIGURATION ON APPLICATION PERFORMANCE

Јован Јењић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У овом раду је представљен утицај одређених Webpack конфигурација и React lazy-a на перформансе апликација. Поред теоријских концепата који служе да би тема рада била јасна, овај рад прати развој веб апликације. Кроз развој апликације у неколико корака је приказан напредак у перформансама апликације што је документовано кроз одређене Google-ove метрике.*

Кључне речи: *Веб апликација, React, Webpack, lazy*

Abstract – *This paper presents the impact of certain Webpack configurations and React lazy on application performance. In addition to theoretical concepts that serve to make the topic of the work clear, this paper follows the development of a web application. Through the development of the application in several steps, the progress in the performance of the application is shown, which is documented through certain Google metrics.*

Keywords: *Web application, React, Webpack, lazy*

1. УВОД

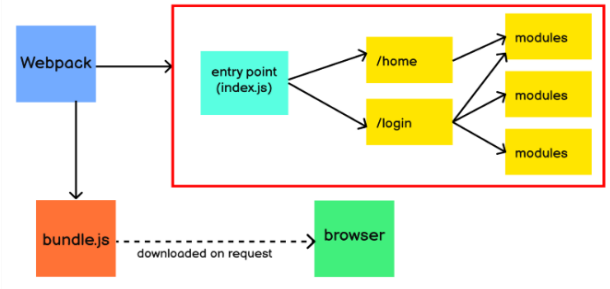
Овај рад се бави предностима које доносе *React* лењо учитавање (енг. lazy loading) и *Webpack* конфигурације у перформансама при имплементацији одређених веб апликација (енг. web application). Осврће се на модерне приступе у изради веб апликација, на предности појединих приступа у конфигурисању архитектуре веб апликација. Описани су појмови и дефиниције чије је разумевање неопходно. Обухваћене су основне теме који су уско повезане са темом овог рада. Обухваћени су концепти веб апликације, паковања (енг. bundling) и преузимање и извршавање апликације на интернет претраживачу (енг. web browser). Такође овај рад се бави и прегледом метрика и мерења перформанси веб апликација. Описано је чему је *Webpack* намењен, на чему се темељи и које су све могуће опције у конфигурисању *Webpack*-а у циљу израде што оптимизованијих апликација.

2. *WEBPACK*

Webpack је статички *bundler* модула за модерне *JavaScript* апликације. Ово је тренутно најзаступљенији алат за *bundling*. Преузима сав код из апликације и чини га употребљивим у веб претраживачу. Модули су комади *JavaScript* кода направљени за вишеструку употребу. Састављени су од кода, слика, екстерних билиотека које су увезене у апликацији или *CSS* стилова који су обједињени и упаковани како би се лако користили у веб претраживачу. *Webpack* одваја код на основу начина на који се користи у апликацији, а са особином модуларног расчлањивања одговорности постаје много лакши за управљати, отклањати грешке, верификовати и тестирати код.

У току *Webpack* обраде апликације он гради граф зависности од најчешће једне или више улазних тачака и затим комбинује сваки модул који је потребан апликацији у један или више пакета који су подељени у *chunk* датотеке. *Chunk* је посебна група повезаног кода који је компајлиран и трансформисан тако да га претраживач може покренути. Ако једна датотека директно зависи од друге, *Webpack* то третира као зависност. Слике, фонтови, стилови и друге ствари које се не кодирају директно у апликацији *Webpack* такође гледа као зависности [1]. Како би се процес *bundling*-а подешавао, *Webpack* користи плагине (енг. plugin) и учитаваче (енг. loaders).

Webpack пружа могућност за раздвајање кода (енг. code splitting) и лењо учитавање (енг. lazy loading) кода са уграђеним високо прилагодљивим плагинима. Такође подржава и дељење заједничких модула између различитих апликација, што посебно има значаја у *micro frontend* апликацијама [2]. На слици 1 је приказана шема креирања *bundle.js* датотеке на основу структуре кода подељених у фајлове где је улазна датотека приказана плавом бојом на слици.



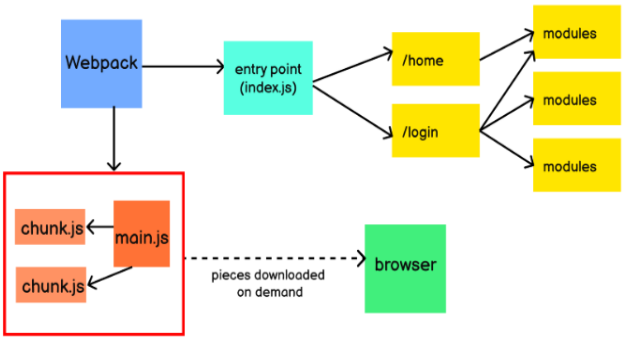
Слика 1. *Креирање више chunk датотека* [3]

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Александар Купусинац, ред. проф.

Са развојем апликације укупна величина изворног кода ће се повећати. Као резултат процеса *bundling*-а настаје датотека *bundle.js*. Претраживач мора у потпуности да преузме ову датотеку пре него што изврши код у њој, самим тим перформансе и брзина апликације опада са порастом величине ове датотеке, посебно ако су укључене *third-part* библиотеке.

Како би се решило дуго време учитавања које је изазвано великом *bundle.js* датотеком, *Webpack* тим је увео функцију раздвајања кода. Подела кода (енг. code splitting) омогућава да уместо једног великог *bundle.js* фајла, апликације буде подељена у неколико мањих датотека које претраживач може посебно и паралелно да преузима. Генерално, постојаће једна *bundle.js* датотека која ће се увек преузети при иницијалном учитавању сваке странице и постојаће неколико пратећих *bundle.js* фајлова, који су познати под називом *chunk.js* фајлови и они се учитавају на кориснички захтев за тим делом кода. Иако је укупна величина изворног кода остала иста, мања величина главне *bundle.js* датотеке ће убрзати почетно време учитавања и омогућиће бржу корисничку интеракцију са апликацијом а то директно води ка бољим перформансама.

Ако је *React* апликација креирана преко *Create React App (CRA)*, онда ће *Webpack* конфигурација која је генерисана од стране *CRA* подразумевано омогућити дељење кода, без додатног конфигурисања са стране програмера. Како у *React*-у постоје подразумеване конфигурације *Webpack*-а, тако да ће сваки динамични увоз који постоји у коду бити препознат са стране *Webpack*-а и за тај део увезеног кода ће бити креирана посебна *chunk* датотека [4]. На слици 2 је показана шема креирања више *chunk* датотека.



Слика 2. *Креирање више chunk датотека* [3]

Постоји више начина којом се постиже дељење кода. Један од начина је динамични увоз који користи синтаксу *import()*. Динамички увоз има другачију синтаксу од уобичајеног наичина за увоз неке датотеке или библиотеке. Када *Webpack* препозна синтаксу за динамични увоз он аутоматски покреће процес раздвајања кода.

Динамички увоз враћа *Promise* који се може користити само асинхроно. Уколико се не појави ниједна грешка, *Promise* ће да врати компоненту или функцију, у супротном ће вратити грешку коју можемо да ухватимо [5]. На слици 3 показан је пример динамичног увоза неке *text.js* датотеке. Коришћена је *async await* синтакса за динамички увоз.

```
```
// dynamic.js - import with async await
const loadDynamicImport = async () => {
  const blogContent = await import("./text.js");
  const str = blogContent();
  console.log(str);

  /* Output:
  Lorem ipsum dolor sit amet, consectetur adipiscing
  */
}
loadDynamicImport();
```
```

Слика 3. *Пример дунамичког увоза text.js компоненте* [3]

React лењо учитавање је техника оптимизације веб странице или веб апликације. Такође је позната као учитавање на захтев јер учитава само садржај видљив на екранима корисника. Када корисник посети неку веб страницу, уместо да учита целу страницу, учитава се само део странице. Затим лењо улитавање одлаже преузимање преосталог садржаја веб странице све док корисник не затражи тај део. На пример, ако веб страница садржи слику који није у фокусу након иницијалног учитавања странице, већ корисник мора да скролује доле да би видео слику, слика се учитава тек када корисник стигне до места где се налази слика. У *React* апликацијама поред слика могуће је лењо учитати и друге ресурсе, као на пример код. Заправо је *React* направио да лењо учитавање делова кода буде олакшано, тако што је омогућио да свака компонента буде стављена у посебан *chunk* фајл [5].

React lazy је релативно нова функција у *React*-у која омогућава лењо учитавање и дељење кода без коришћења неке *third-part* библиотеке као што је то био раније случај. Сада постоји *React lazy* која је интегрисана функција у саму основу *React* библиотеке. Ово уграђена функција омогућава да се лако изврши лењо учитавање компоненти.

Основна разлика између динамичног лењог учитавања и обичног увоза је у томе што у случају лењог учитавања се увози компонента или неки ресурс тек када се тај део захтева од стране корисника.

На слици 4 приказано је лењо учитавање компоненте које зависи од неког услова. Другим речима, *Webpack bundler* је препознао динамични увоз и креирао посебан *chunk.js* фајл за њега. Тај фајл ће шретраживач почети да преузима и обрађује так некон што услов буде испуњен. Услов најчешће буде испуњен као реакција на неку корисничку акцију клика и слично [5].

```
let OtherComponent = undefined;
// Never imported.
if (false) {
  OtherComponent = React.lazy(() => import('./OtherComponent.jsx'));
}
```

Слика 4. *Креирање више chunk датотека* [3]

Главна предност лењог учитавања су побољшавање перформанси. Учитавањем мањег *JavaScript* фајла претраживач ће смањити време учитавања *DOM*-а и побољшаће перформансе апликације. Корисници могу да приступе веб страници чак иако није све учитано. Неке од предности лењог учитавања [5]:

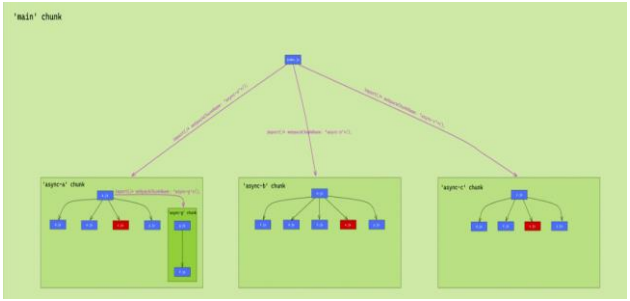
- **Брже почетно учитавање** - Лењо учитавање помаже да се смањи тежина странице се погледа величине, омогућавајући брже почетно учитавање.
- **Боље корисничко искуство** - Лењо учитавање побољшава корисничко искуство у апликацији. Добро корисничко искуство директно води ка повећању пословања и присутности корисника на апликацији, задржавајући посетиоце.
- **Мања потрошња пропусног опсега** - Лењо учитавање слика помаже у уштеди података и пропусног опсега. Ово посебно утиче на кориснике који немају брз интернет или имају слаб *CPU*.
- **Очување системских ресурса** - Лењо учитавање *React* компоненти помаже у очувању серверских и клијентских ресурса тако што захтева само делове читаве компоненте.
- **Смањени рад претраживача** - Претраживач не мора да обради слике или велики део кода све док слике или делови кода не буду захтевани од стране корисника као акција клика, скроловања или других акција.

SplitChunkPlugin аутоматски инфентификује модуле који би требали бити подељени у *chunk*-ove помоћу хеуристике користећи број понављања модула и категорије којима модули припарају.

Проблем који *SplitChunkPlugin* решава је дуплирање истог кода, што може довести до сувишног садржаја који се учитава преко мреже. *SplitChunkPlugin* је у стању да открије, на основу неких правила, модуле који су прескупи да би били дуплирани, затим их издваја у посебне *chunk* фајлове тако да се велики напор учитавања захтевних модула дешава само једном у оквиру једног *chunk* фајла. На пример, ако анализирањем *bundle*-а закључимо да се нека велика компонента понавља у више *chunk* фајлова, процес учитавања за сваки од тих *chunk* фајлова може да потраје дуго као и парсирање кода које следи након учитавања. Понављање овог скупог процеса није добро решење са становишта перформанси и неопходно је извршити одређене конфигурације *Webpack*-а како би се ово избегло. Управо такве ситуације решава овај плагин.

Дијаграм на слици 5 приказује 3 различита *chunk* фајла који су креирани на основу динамичког увоза. Такође, шематски је приказан садржај сваког *chunk* фајла, тј. који сви модули се налазе у њему. Занимљиво је приметити да постоји *x.js* датотека која се понавља у сва три *chunk* фајла. Као што је раније било споменуто, ово је случај који негативно утиче на перформансе веб апликације посебно ако је *x.js* нека велика и скупа датотека. Под претпоставком да *x.js* има 1000 линија кода, претраживач ће морати да учита *x.js* три пута јер је *x.js* део три различита *chunk* фајла. Узимајући у обзир ове информације, начин за решавање споменутог проблема би био да *x.js* буде смештен у посебан *chunk* фајл при чему би се овај садржај захтевао и обрађивао једном, уместо три пута.

Webpack-license-plugin издваја информације о лиценцама отвореног кода (енг. open source code) свих *npm* пакета у излазном *webpack* фајлу и помаже при идентификовању и решавању проблема са политиком лиценцирања отвореног кода. Овај *plugin* је подржан и тестиран у верзијама 2, 3, 4 и 5 *webpack*-а [6].



Слика 5. *Креирање више chunk датотека* [3]

3. ИМПЛЕМЕНТАЦИЈА

Приказане су и упоређене перформансе апликације пре и након укључивања лењог учитавања и након додатних *Webpack* конфигурација. За мерење перформанси ће бити коришћене основне *Google*-ове меторике из *Lighthouse* резултата.
За развијање *front-end* дела система коришћен је *Visual Studio Code* интегрисано развојно окружење и *React JavaScript* библиотека верзије 17.0.2 заједно са *Webpack*-ом који групише *JavaScript* фајлове и омогућава модуларну имплементацију система заједно са својим плугин решењем.

Сама апликација садржи неколико страница. *React* рутер (енг. Router) служи за навигирање између страница, свака страница се налази на посебној путањи и захтева посебан део кода који претраживач мора да учита како би страница била приказана, као што је приказано на слици 4.1. Свака страница садржи одређени део имплементиране логике у виду кода и увезене *third-part* библиотеке. Неке од библиотека се понављају неколико пута, тј. увезене су исте библиотеке у различитим страницама апликације. Циљ овог дела је да кроз неколико корака покаже напредак у перформансама апликације без превише детаљног залажења у само кодирање апликације.

Први корак садржи информације о апликацији без додавања лењог учитавања и без додатних *Webpack* конфигурација. Коришћен је стандардан начин увоза свих датотека. *JavaScript* код читаве апликације ће бити преузет од стране претраживача на иницијалном учитавању било које странице. Претраживач је у овој фази преузео пет *JavaScript* датотека које у укупном збиру чине приближно 2300kb.

Потребно је запазити да приликом навигирања на било који другу странице претражач неће преузети ниједан нови *JavaScript* фајл, јер је сав потребан *JavaScript* код преузет на првом иницијалном учитавању апликације. Жељени исход да претраживач преузме само довољан код који је потребно да изврши да би тренутна страница била видљива и функционална. *Lighthouse* резултата који је постигнут у овој фази развоја апликације је 76 бодова од максималних 100.

У другом кораку изкоришћена је предност подразумеваних *Webpack* конфигурација и уместо

уобичајеног начина увоза свих страница апликације у рутеру, коришћено је лењо учитавање који *Webpack* конфигурације препознају и аутоматски деле код на мање *chunk*-ове у зависности од сваке странице. Након поновног покретања апликације приликом иницијалног учитавања исте странице за разлику од претходне ситуације где је постојао само један *chunk* фајл, сада постоји осам мањих *chunk* фајлова. Након иницијалног учитавања странице, приликом навигирања на сваку следећу страницу апликације, претраживач преузима нове *chunk* фајлове и кешира их у своју меморију. Ово показује да свака страница која је лењо учитава преко рутера садржи свој део кода који се учитава тек приликом навигирања на ту страницу, а не иницијално. Укупан збир свих преузетих *JavaScript* датотека сада је приближно 1200 kb. *Lighthouse* резултат мерења је побољшан са 76 на 91 бод у овој фази.

Приликом анализирања садржаја сваког *chunk* фајла закључено је да постоји више *chunk* фајлова који садрже исту *third-part* библиотеку која долази из *node-module*-а. Другим речима различити *chunk* фајлови који ће бити преузети и извршени од стране претраживача на различитим страницама апликације садрже исти садржај. У следећем кораку имплементације оптимизованог решења ове апликације коришћене су додатне *Webpack* конфигурације. Како би било избегнута ситуација где се исти *JavaScript* код непотребно понавља на више места, циљ је креирати посебан *chunk* фајл који ће садржати баш такве делове ове апликације. На слици 6 је показан *cacheGroups* опција која је конфигурисана у оквиру *splitChunks* плагина. Са овом конфигурацијом омогућено је да сви делови већ креираних *chunk*-ова који се понављају у више од 3 *chunk* фајла, буду убачени у посебан *common chunk* фајл који ће се асинхроно учитати само једном, при иницијалном улитавању било које странице. На овај начин је решен проблем понављања истог садржаја у више различитих *chunk* фајлова.

```
splitChunks: {
  chunks: 'all',
  name: false,
  cacheGroups: {
    common: {
      name: 'common',
      minChunks: 3,
      chunks: 'async',
      priority: 10,
      reuseExistingChunk: true,
      enforce: true
    }
  },
},
```

Слика 6. *CacheGrpups konfiguracija*

Приликом даљег испитивања садржаја сваког *chunk* фајла закључено је да више нема истог садржаја који се понавља. Такође, овим је постигнуто додатно смањање укупне величине свих *JavaScript* датотека које је потребно да буду преузете и извршене од стране претраживача на истој страници. Након поновног тестирања перформанси веб апликације, очековано је постигнут нови напредак у перформансама. *Lighthouse* резултат мерења је побољшан са 91 на 94 бод у овој фази.

5. ЗАКЉУЧАК

У првом кораку имплементације приказано је неоптимизовано решење који је потпуно функционално, али кроз мерење перформанси и величину *JavaScript* кода који претраживач мора да преузме и изврши, било је приказано да постоји простор за побољшавање перформанси. У сваком кораку имплементације акценат је био на метрикама које су мериле различите приступе у мерењу перформанси апликације, као и осврtање на конкретан узрок који доводи до бољих или лошијих перформанси. Сваки следећи корак у имплементацији је уводио нове ствари које поспешују рад апликације и то је мерењем *Lighthouse* метрикама и документовано.

У корацима имплементације редом је укључено лењо учитавање а затим и додатне *Webpack* конфигурације где су перформансе апликације итеративно напредовале. Резултат је био функционална апликације високих перформанси. Иако у делу имплементације показано да је и на малим апликацијама могуће значајно побољшати перформансе апликације, важно је запазити да постоје и друге *Webpack* опције које могу да утичу на перформансе као и да споменуте опције могу дати различите резултате у зависности од постављених параметара.

6. ЛИТЕРАТУРА

[1] James Thomas, *What is Wabpack?*, 2019. Доступно: https://levelup.gitconnected.com/what-is-webpack-4fdb624597ae

[2] Morgan Persson, *JavaScript DOM Manipulation Performanse, Comparing Vanilla JavaScript and Leading JavaScript Front-end Frameworks*, 2020. Доступно: https://www.theseus.fi/bitstream/handle/10024/345959/Laurila_Sonja.pdf?sequence=2&isAllowed=y

[3] https://parceljs.org/features/scope-hoisting/

[4] Nathan Sebhastian, *Understanding Webpack's Code Splitting Feature*, 2021. Доступно: https://blog.bitsrc.io/understanding-webpacks-code-splitting-feature-3077768e4066

[5] https://www.copycat.dev/blog/react-lazy/

[6] https://www.npmjs.com/package/webpack-license-plugin

Кратка биографија:

Јован Јењић је рођен у Бањој Луци 1997. године. Основне академске студије завршио је 2020. године на Факултету техничких наука у Новом Саду. Мастер рад на Факултету техничких наука из области Рачунарство и аутоматика – Електронско пословање одбранио је 2022. године

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE04Budjen

АНОТИРАЊЕ И ОДРЕЂИВАЊЕ СЕНТИМЕНАТА ТВИТОВА ВЕЗАНИХ ЗА ПОЛИТИЧКУ СЦЕНУ РЕПУБЛИКЕ СРБИЈЕ

ANNOTATION AND SENTIMENT ANALYSIS OF TWEETS RELATED TO THE POLITICAL SCENE OF THE REPUBLIC OF SERBIA

Владимир Буђен, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У раду је приказано одређивање сентимента твитова и креирање алата који одређује позицију аутора твитова на политичком компасу. Циљ рада је асистеннција корисницима при политичким изборима. За одређивање сентимента твита коришћен је BERTić модел. Добијени сентимент, у комбинацији са темом твита, је коришћен као улаз у модел SVM и Random Forest моделе који служе за поларизацију аутора твита.*

Кључне речи: *твитер, сентимент, политика, вештачка интелигенција, BERT*

Abstract – *This paper describes tweet sentiment analysis and the development of a tool for placing tweet authors on a political map. The purpose of this system is to assist its users in political elections. BERTić model is used for tweet sentiment analysis. Sentiment and tweet's theme are used as inputs for SVM and Random-forest, which are used for user polarization.*

Keywords *tweeter, politics, artificial intelligence, sentiment, BERT*

1. УВОД

Друштвена мрежа „Twitter" својим корисницима нуди лак и јефтин начин да поделе лично мишљење великом броју људи. Због тога велики број корисника ове мреже чине политичке партије и удружења, као и њихови чланови. Објаве на твитеру су ограничене бројем карактера и због тога су погодне за анализу.

Једна од информација коју бисмо могли да сазнамо из твитова политички оријентисаних профила јесте позиција тих профила на политичком компасу. Та информација може да помогне гласачком телу да има реалнији увид у тачну политичку оријентацију странака. Потреба за тим постоји јер многе партије, због жеље да привуку што више људи различитих идеологије, не спомињу своје конкретно место на политичком компасу. Такође, постоје случајеви да странке потенцирају нестандардне идеје у односу на оријентацију којом се представљају.

Како би се аутоматизовао процес идентификације политичке оријентације аутора твитова, потребно је користити технике вештачке интелигенције. Као циљно обележје коришћени су сентимент твита, тема твита и позиција аутора твита на политичком компасу. Скуп података коришћен за обучавање ових модела је прикупљен са твитера и ручно анотиран од стране аутора рада. Текстови твитова су трансформисани у број који представља њихов сентимент. Налози корисника су искључиво са територије Републике Србије, што представља додатни изазов при одређивању сентимената, за шта је коришћен претренирани вишејезични BERT модел [1].

За предикцију сентимента твита коришћен је BERT, док су за поларизацију коришћени метод потпорних вектора (енг. *Support Vector Machine,* SVM) и модел насумичне шуме (енг. *Random Forest*). Излаз из SVM модела је позиција аутора твита са прецизношћу 0.1 на скали од -1 до 1, док је прецизност *Random Forest*-а 0.25.

Евалуацијом решења пројекта се може видети како политичке оријентације пишу твитове чији сентимент зависи од дате теме. Постоје теме у којим већина корисника испољавају исти сентимент, оне у којима нема корелације између оријентације и сентимента и оне чијим сентиментом се лако одреди позиција на политичком компасу (и обрнуто).

Тешко је директно поредити резултате приказане у овом раду са сличним радовима услед тога што користе различите скупове података. Међутим, за теме које се преклапају као и у овом раду, добијени су слични закључци о корелацији те теме и политичке оријентације корисника.

2. ПРЕТХОДНА РЕШЕЊА

У овом поглављу бавићемо се радовима сличне тематике, као и самом еволуцијом области и значајним резултатима и увидима.

Први значајан рад је „*Predicting the Political Alignment of Twitter Users"* [2]. Аутори овог рада се баве анализом твитова у циљу одређивања политичке оријентације корисника по економској скали на подручју Сједињених Америчких Држава у периоду од 14.9.2010. до 1.11.2010. Анализа твитова је извршена на два начина: на основу садржаја и на основу међукорисничких интеракција. На релевантност твитова је утицала појава хештегова (енг. *hashtag*). За класификацију је коришћен SVM, што је један од разлога због чега је тај модел коришћен и у експерименту који овај рад описује. Овај рад пријављује тачност од 0,92.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је била др Јелена Сливка, ванр. проф.

Још један значајан рад из ове области је „*A Sentiment Analysis System of Spanish Tweets and Its Application in Colombia 2014 Presidential Election*“ [3]. Идеја рада је базирана на анализи политичке струје у Колумбији 2014. године, с циљем предвиђања резултата председничких избора. Овај рад такође користи хештегове, али поред тога користи и ручно изабране кључне речи. Такође, један део података, везан за кориснике, је ручно анотиран. Подацима је додељен сентимент од стране волонтера и кроз јавне анкете, што је позитивно утицало на резултате. Због малог броја хештегова у политичким твитовима на српском језику, идеја да се користе изабране кључне речи је коришћена и у експерименту везаном за овај рад. Хештегови који су коришћени у наведеним радовима послужили су за бирање кључних речи у овом раду. Рад [3] пријављује F1 меру од 0,58.

3. МЕТОД

У наредним поглављима изложени су скуп података и начин на који је спроведен експеримент.

3.1. Скуп података

Почетна фаза експеримента је креирање упита који ће бити кориштени за прикупљање жељених твитова. Упити у себи садрже речи које желимо да се појаве у твитовима, као и кориснике које желимо да буду аутори. Речи које су кориштене у твитовима су груписане по темама. Листа аутора и тема је ручно креирана. Узор за одабир тема су радови „*Predicting the Political Alignment of Twitter Users“* [2] и „A Sentiment Analysis System of Spanish Tweets and Its Application in Colombia 2014 Presidential Election“ [3]. Конкретне теме коришћене за експеримент су:

- Косово
- Албанија
- Војска
- Црква
- ЛГБТ
- Београд
- Полиција
- Корупција
- Европа
- Украјина
- Путин
- Русија
- Нато
- Америка

Такође, политичка позиција аутора твитова (самим тим и њихових твитова) је ручно креирана. Она је представљена вредностима на социјалној и на економској скали. Вредност је број у распону од -1 до 1 са кораком 0.1. Политичка позиција је битна за фазу обучавања модела. Подаци су подељени на обучавајући и тест скуп у односу 1:5.

3.2. Одређивање сентимента твита

Прикупљање података даје информације о теми, садржају и осталим корисним информацијама. Проблем је у томе што о одређеној теми различити корисници говоре у различитом контексту, што представља битну информацију при одређивању политичког опредељења. Зато је потребно добити информацију да ли се тема у садржају спомиње у позитивном, негативном или неутралном контексту. За ово се користе претренирани модели за одређивање сентимента.

У овом експерименту коришћена је *fine-tune*-ована верзија Бертића [4], специјализована за одређивање сентимента твитова. Овај модел је доступан под називом *EMBEDDIA/bertic-tweetsentiment* на *huggingface* репозиторијуму [5]. Његов улаз је текст писан латиничним писмом.

Због овога је потребно ћирилични текст превести у латинични, за шта је коришћена библиотека „*SrbAi*“ [6]. Излаз из модела је сентимент (енг. *sentiment_label*) и сигурност у добијени резултат изражен бројем (енг. *Sentiment_score*).

3.3. Обучавање модела

Прикупљени подаци коришћени су за обучавање више модела применом два приступа. Оба приступа креирају посебне моделе за економску и за социјалну скалу. За креирање свих модела коришћена је „*scikit-learn*“ [7] библиотека.

Први приступ користи SVM, чији је вектор улаза сачињен од сентимената тема сваког појединачног твита. За позитиван сентимент је дат број 1, негативан -1, а неутралан сентимент 0.5. За све теме које нису везане за изабрани твит, унета је вредност 0. Разлог због којег је одабран овакав коефицијент за неутралан сентимент твита јесте могућност разликовања оних корисника који су имали неутрално мишљење о некој теми, од корисника који се о њој нису изјаснили.

Пошто су сви твитови везани за тачно једну тему улаз у модел је *one-hot* вектор. Излаз, као и у првом приступу, јесте вредност на политичкој скали (један од бројева између -1 и 1 са скоком 0,1). Исти принцип је коришћен при креирању *Random Forest*-а. Једина разлика јесте у томе што су узете вредности на скали са кораком 0,5 уместо 0,1.

Други приступ се разликује од претходног по самом начину формирања вектора, који се овог пута везује за корисника уместо за сам твит. Наиме, вектор се формира за сваког корисника, а као вредности садржи корисниково укупно мишљење на задату тему, при чему теме представљају димензије вектора. Корисниково мишљење на једну тему одређује се по формули:

$$\frac{k_{pos} * n_{pos} + k_{neg} * n_{neg} + k_{neu} * n_{neu}}{n},$$

где k представља коефицијент, а n укупан број твитова одговарајућег сентимента. Вредности коефицијента су 1 за позитиван сентимент, -1 за негативан и 0.5 за неутралан. Овако формирани вектори, употребљени су за обучавање две SVM (један за економски аспект и други за социјални аспект) [8]. За оптимизацију параметара коришћена је класа *GridSearchCV* из *scikit-learn* [7] библиотеке.

4. РЕЗУЛТАТИ И ДИСКУСИЈА

У овом поглављу су приказани експлоративна анализа, добијени резултати експеримента и предочене потенцијалне мане и предложена побољшања.

4.1. Експлоративна анализа

Обрада сентимента је показала да већина твитова политичке природе има негативан сентимент. Конкретно, од 3192 преузета твита, 1926 су негативног, 979 су позитивног, а 287 неутралног сентимента. Након вршења

анализе сентимента, могуће је видети да су неке теме поларизованије од других. Примери таквих тема су:

- Путин
- LGBT
- Русија
- Војска

4.2. Резултати експеримента

Резултати експеримента су приказани у табелама 4.1 – 4.3, где С представља праве координате на социјалној скали, Е представља праве координате на економској скали, С.П представа предикцију координата на социјалној скали, Е.П представа предикцију координата на економској скали, С.Г представа грешку на социјалној скали, Е.Г представа грешку на економској скали. Грешка се рачуна као апсолутна вредност разлике координата. У табелама су приказани насумично одабрани корисници из тест скупа података.

Резултат првог приступа при коришћењу SVM је приказан у табели 4.1. Просечна вредност грешке за економску скалу је 0,27, док је за социјалну скалу вредност 0,57.

Табела 4.1 *Мере евалуације првог приступа и коришћењем svm*

Аутор	С	Е	С.П	Е.П	С.Г	Е.Г
Марко Ђурић	0,9	0,3	0,3	0,2	0,6	**0,1**
Ана Брнабић	0,5	0,1	0,2	0	0,3	**0,1**
Небојша Стефановић	0,8	0,5	0,4	0,2	0,4	0,3
Александар Шапић	0,4	0	0,1	0,1	0,3	**0,1**

Први приступ при коришћењу *Random Forest* модела је тачно погодио половине квадранта у 48,27% података на економској и 37,93% на социјалној скали. Резултати овог приступа су приказани у табели 4.2.

Табела 4.2 *мере евалуације првог приступа и коришћењем Random forest ансамбла*

Аутор	С	Е	С.П	Е.П	С.Г	Е.Г
Марко Ђурић	0,9	0,3	0,3	0,2	0,6	**0,1**
Ана Брнабић	0,5	0,1	0,2	0	0,3	**0,1**
Небојша Стефановић	0,8	0,5	0,4	0,2	0,4	0,3
Александар Шапић	0,4	0	0,1	0,1	0,3	**0,1**

Други приступ, у ком су вектори формирани по упросеченим сентиментима тема појединачних корисника је одступао на економској скали за 0,28 у просеку, док је за социјалну скалу просечно одступање износило 0,39. Резултати овог приступа су приказани у табели 4.3.

Табела 4.3 *мере евалуације другог приступа*

Аутор	С	Е	С.П	Е.П	С.Г	Е.Г
Марко Ђурић	0,49	0,33	0,9	0,3	0,41	0,03
Ана Брнабић	0,54	0,45	0,5	0,1	0,04	0,35
Небојша Стефановић	0,51	0,31	0,8	0,5	0,29	0,19
Александар Шапић	0,14	0,19	0,4	0	0,26	0,19

4.3. Анализа грешака и потенцијална побољшања

Највећа мана система јесте недовољно велики скуп података. Укупан број твитова је 3191. Ово је мали број када се подели на 14 значајних тема. У просеку, то чини 228 твитова по теми. Последица овога је недовољно тренирани модел, као и немогућност балансирања скупа података.

Други извор грешака је то што претрага твитова по темама, за проналажење теме има једини услов да се тема спомиње у тексту, независно од контекста. Ово доводи до тога да тема није примарна ствар у твиту. Пошто се сентимент врши над целим садржајем твита, веза између теме и сентимента је врло слаба. Потенцијално решење за овај проблем јесте коришћење хештегова за избор теме. Овакво решење би било могуће у случају да постоји већа количина података и да их корисници више користе.

Садржај твита често садржи више реченица од којих су неке позитивног, а неке негативног сентимента. Један од начина на који ово може да се реши јесте филтрирањем података, тако да се користе само они твитови, чија вредност сигурности у сентимент (*sentiment_score*) је преко 0,9. Ово долази са ценом смањења скупа података.

Једна од највећих слабости система јесте и та што су вредности позиција на политичком компасу ручно анотирана од стране само једне особе, која притом није уско специјализована за то. Најбољи начин да се ово реши, јесте да се за ово ангажује више стручњака и да се вредности добију упросечавањем вредности које они задају.

4.4. Употребљивост система

Алат развијен овим експериментом се користи у ситуацијама када је потребно добити политичку позицију аутора твитова. Његова велика предност је у томе што је намењен српској популацији. Иако тренутно ради само са српским језиком, могућност проширења на друге језике није комплексна. Друга ствар која чини овај алат другачијим од осталих јесте квантификовани излаз, који је могуће користити као улаз за креирање других модела.

У поређењу са резултатима добијеним од стране људи, модел се показао бољим код људи са недовољним знањем о политици. Доказ о овоме јесте мала грешка при тестирању. Треба имати у виду да је и сам тест скуп ручно анотиран. Због овога модел ни у ком случају не може да превазиђе резултате који би дали људи специјализовани за политичке науке.

Имајући то у виду, тренутна употребљивост модела не улази у професионални домен. Помаже корисницима који нису у стању да сами одреде политичку позицију твита, али никако не може да замени мишљења професионалаца.

5. ЗАКЉУЧАК

У овом раду представљен је систем за одређивање политичке позиције аутора твитова. Мотивација за креирање овог система је била спознаја да би такав систем помогао корисницима да лакше донесу одлуку при гласању на политичким изборима. Систем је им-

плементиран из модула за одређивање сентимента твита и модула који садржи модел за препознавање политичке оријентације аутора твита.

Модул за одређивање сентимента твита је имплементиран коришћењем BERT-*ић* модела. Његовим коришћењем су добијени сентименти твитова. Комбинација добијеног сентимента и теме твита су информације потребне за одређивање политичке позиције. Тај процес се врши у другом модулу коришћењем модела SVM. Излаз из система су пар бројева који представљају координате позиције на политичком компасу.

Приступ у ком су вектори формирани по упросеченим сентиментима одређених тема појединачних корисника је дао најбоље резултате. Он је одступао на економској скали за 0,28 у просеку, док је за социјалну скалу просечно одступање износило 0,39. Директног поређења са другим радовима не може бити јер је коришћен нови скуп података са српског подручја, док су постојећи радови са подручја Сједињених Америчких Држава и Републике Колумбије. Још једна од препрека у поређењу резултата јесте та што су остали радови вршили бинарну класификацију, док је у овом раду вршена мултикласна класификација. Ово је разлог коришћења различитих мера евалуације. Међутим, поређења ради, рад [20] пријављује F1 меру од 0,58, а рад [19] тачност од 0,92.

Овом систему остаје простора за унапређење. Највише побољшање је могуће добити проширивањем скупа података, што би отворило могућност бољег балансирања скупа података по темама и корисницима, самим тим бољих резултата.

6. ЛИТЕРАТУРА

[1] J. D. M.-W. C. K. L. K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“.

[2] B. G. J. R. A. F. F. M. Michael D. Conover, Predicting the Political Alignment of Twitter Users.

[3] E. L.-G. Jhon Adrian Ceron-Guzman, A Sentiment Analysis System of Spanish Tweets and Its Application in Colombia 2014 Presidential Election.

[4] D. L. Nikola Ljubešić, „BERTić - The Transformer Language Model for Bosnian, Croatian, Montenegrin and Serbian“.

[5] „EMBEDDIA/bertic-tweetsentimen,“ [На мрежи]. Available: https://huggingface.co/EMBEDDIA/bertic-tweetsentimen.

[6] „SrbAi,“ [На мрежи]. Available: https://github.com/Serbian-AI-Society/SrbAI.

[7] „Scikit-learn,“ [На мрежи]. Available: https://scikit-learn.org/stable/.

[8] М. Кнежевић, „Позиционирање корисника друштвене мреже Twitter на мапи политичког спектра помоћу корисничких твитова“.

Кратка биографија:

Владимир Буђен рођен је 9.5.1998. године у Новом Саду, где је стекао своје основно и средње образовање. Школске 2017/18 године се уписисује на Факултет техничких наука на студијски програм Рачунарство и аутоматика, који је завршио школске 2021. године. Исте године и на истом факултету уписује мастер студије, конкретно, програм Електронско пословање. Положио је све испите предвиђене планом и програмом и стекао услов за одбрану завршног рада.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE05Dakic

ДЕТЕКЦИЈА И СЕГМЕНТАЦИЈА КАЈАКАША УПОТРЕБОМ КОНВОЛУЦИОНИХ НЕУРОНСКИХ МРЕЖА

DETECTION AND SEGMENTATION OF KAYAKERS USING CONVOLUTIONAL NEURAL NETWORKS

Никола Дакић, *Факултет техничких наука, Нови Сад*

Област – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *У раду је представљен систем за детекцију кајакаша на видео снимку. Систем врши парсирање видео записа и обрађује сваки фрејм. На сликама се детектују и сегментирају инстанце кајакаша. За решавање наведених задатака, коришћен је Mask R-CNN метод са конволуционом неуронском мрежом, ResNet101 архитектуре. Модел је направљен употребом технике преносног учења. Наведена техника преносног учења користи Mask R-CNN модел који је претходно обучен на Microsoft COCO скупу података. Као резултат система генерисан је излазни видео снимак на којем је детектован и сегментиран кајакаш.*

Кључне речи: *Детекција и сегментација објеката, кајакаш, Mask R-CNN, конволуционе неуронске мреже*

Abstract – *The paper presents a system for detecting kayakers on video. The system parses the video and processes each frame. In the images, instances of kayakers are detected and segmented. To solve the mentioned tasks, the Mask R-CNN method is used with ResNet101 architecture. The model was created using the transfer learning technique. The transfer learning technique uses a Mask R-CNN model previously trained on the Microsoft COCO dataset. As a result of the system, an output video was generated with the detected and segmented kayaker.*

Keywords: *Object detection and segmentation, kayakers, Mask R-CNN, convolutional neural networks*

1. УВОД

Детекција објеката је изозован задатак рачунарске визије, који је првобитно подразумевао предвиђање где се објекат тачно налази као и о ком типу објекта се ради. Нагли раст заједнице која се бавила овим проблемима, довела је до унапређења детекције објеката, који осим претходно споменута два задатка, подразумева и сегментацију инстанци. Сегментација инстанци је задатак који подразумева тачну детекцију свих објеката на слици, као и прецизну сегментацију сваке инстанце. Потпуније речено сегментација инстанци подразумева комбиновање класичних задатака детекције објеката у којем је циљ да се класификују појединачни објекти а затим да се они локализацију употребом граничних оквира, и класификовањем сваког пиксела у фиксни скуп категорија тако да се добију засебне инстанце објеката. Примена овог задатка је разнолика и неки од најчешћих употреба су: детекција објеката у малопродаји, аутономна вожња, детекција животиња у пољопривреди, откривање људи у безбедности, детекција возила у транспорту итд.

У овом раду приказана је имплементација фазе детекције и фазе сегментације појединачних објеката. Наведене фазе имплементиране су употребом сложених архитектура конволуционих неуронских мрежа.

За решавање проблема детекције у овом раду коришћен је Mask R-CNN [1] метод са конволуционом неуронском мрежом ResNet101 архитектуре. Модел је направљен употребом технике преносног учења. Наведена техника преносног учења користи Mask R-CNN модел који је претходно унапред обучен на Microsoft COCO [2] скупу података. Такав унапред обучен модел се затим користи као основа за генерисање модела над ручно прикупљеном скупу података. Финални модел представљен.у овом раду постиже *0.933 mAP* над тест скупом података.

2. ПРЕТХОДНА РЕШЕЊА

Компјутерски вид је мултидисциплинарно поље које је добило много пажње претходних година, наглим развојем конволуционих неуронских мрежа (енгл. Convolutional Neural Networks - CNN), а аутомобили који се сами возе (енгл. self-driving cars) заузимају централно место у овој области. Саставни део компјутерског вида је детекција објеката. Детекција објеката помаже у процени позе објекта, детекцији возила, надзору и сличним задацима. Разлика између алгоритама за детекцију објеката и алгоритама за класификацију је у томе што код алгоритама за детекцију покушавамо да нацртамо гранични оквир око објекта од интереса, како бисмо га лоцирали унутар слике. Такође на једној слици може постојати више граничних оквира који представљају различити објекте од интереса, што значи да не знамо унапред њихов тачан број. Стога главни разлог зашто се задаци детекције објеката не могу решити стандардом изградњом конволуционих неурноских мрежа, праћеним потпуно повезаним слојем, (енгл. fully connected layer) је тај што је дужина излазног слоја промењива. Број појављивања објеката од интереса није фиксан.

Наивни приступ решавању овог проблема би био да се из слике узму различити интересни региони и да се

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био проф. др Александар Ковачевић.

затим примени CNN ради класификације присуства објекта унутар региона. Проблем са овим приступом лежи у томе што објекти од интереса могу имати различите просторне локације унутар слике као и различите пропорције, што последично доводи до огромног броја региона које треба процесуирати. Стога су развијени алгоритми попут R-CNN [3] (Region Based Convolutional Neural Networks), YOLO [4] (You Only Look Once) и други, који ефикасно бирају и процесуирају регионе од интереса.

R-CNN метод превазилази проблем одабира огромног броја региона тако што уз помоћ селективне претраге издваја само 2000 региона из слике, и они се називају предложени региони (енгл. region proposals). Ови предложени региони се затим претварају у квадрате који се уносе у конволуциону неуронску мрежу која производи вектор карактеристика као излаз. CNN се понаша као екстрактор карактеристика а њен излаз се затим уноси у неки бинарни класификатор попут SVM (енгл. Support Vector Machine) ради класификације присуства објекта унутар предложених региона. Мане овог приступа су што и даље треба пуно времена за обуку неуронске мреже, јер се мора класификовати 2000 региона по слици, из чега следи да се не може применити у реалном времену.

Такође још једна значајна мана овог приступа јесте што је алгоритам селективне претраге фиксан алгоритам. Не постоји никакво учење у овој фази, што може довести до лошег генерисања самих предложених региона.

Fast R-CNN представља унапређење R-CNN методе, и то је постигнуто тако што се конволуционој неуронској мрежи шаље улазна слика уместо предложених региона, на основу које CNN генерише конволуциону мапу карактеристика. Разлог зашто је Fast R-CNN бржи од R-CNN методе је тај што се не мора сваки пут прослеђивати 2000 предложених региона конволуционој неуронској мрежи. Уместо тога, операције конволуције се ради само једном по слици и из ње се генерише мапа карактеристика.

Faster R-CNN представља унапређење Fast R-CNN методе. Унапређење је постигнуто тако што се избацује селективна претрага као начин генерисање предложених региона, јер она представља спор и дуготрајан процес који утиче на перформансе мреже. Уместо ње, користи се засебна мрежа за предвиђање предложених региона.

YOLO алгоритам за детекцију објеката представља алгоритам који се разликује од претходно наведених алгоритама. Сви претходних алгоритми користе регионе да локализују објекат унутар слике, и не гледају комплетну слику већ само делове слике који имају велику вероватноћу да садрже објекат. Код YOLO алгоритма, једна конволуциона неуронска мрежа предвиђа граничне оквире и њихове вероватноће да садрже циљни објекат. Из тог разлога YOLO алгоритам је доста бржи од претходно наведених алгоритама, ал и има потешкоћа са детекцијом малих објеката унутар слике.

Mask R-CNN представља унапређење Faster R-CNN методе, тако што поред предвиђања класе и оквира објекта, које Faster R-CNN даје као излаз, проширују излаз са додатном граном која предвиђа маску објекта.

3. МЕТОД

У наредним поглављима изложен је скуп података, начин креирања модела, начин евалуације решења и резултат система.

3.1 Скуп података

Скуп података коришћен у овом раду се састоји од 65 слика на којима се налазе кајакаши. Слике за потребе овог рада су скупљане ручно помоћу претраживача *Google, Bing*, и *Yandex*. Улазни видео снимак, који се обрађује у овом раду, снимљен је дроном из птичије перспективе, те су за потребе обучавања модела узимане углавном слике кајакаша из птичије перспективе.

С обзиром да је задатак Mask R-CNN модела да класификује објекте, предвиди њихове граничне оквире, као и да предвиди маске за детектоване објекте, неопходно је да ручно прикупљени скуп података који се користи за обучавање модела поседује координате свих полигона (полигон - многоугао који окружује сваког кајакаша на слици). За сваку слику која је коришћена приликом обучавања модела, уз помоћ *VIA* [5] софтвера, генерисан је адекватан *json* објекат. Сваки генерисани *json* објекат садржи информације које прецизно указују где се тачно налази кајакаш на слици. Сви *json* објекти су груписани и сачувани у склопу *annotations.json* фајла.

Прикупљени и анотирани скуп податак је подељен на скуп за тренирање и скуп за тестирање модела. Скуп за тренирање модела се састоји од 50 слика кајакаша и њихових анотација. Скуп за тестирање модела се састоји од 15 слика кајакаша и њихових анотација.

3.2 Креирање модела за детекцију и сегментацију кајакаша

Систем представљен у овом раду се базира на *Matterport Inc. Mask R-CNN* имплементацији [6]. Наведена имплементација даје основу за изградњу система детекције кајакаша.

Систем се може поделити у два модула:

- Модул за обраду видео снимка
- Модул за детекцију и сегментацију инстанце

3.2.1 Модул за обраду видео снимка

Модул за обраду видео снимка се бави учитавањем и парсирањем улазног видео снимка, као и генерисањем излазног. Модул је имплементиран употребом *OpenCV* библиотеке, и састоји се од *mark_kayaker* и *predict_kayaker* метода. Задатак *mark_kayaker* методе је да исцрта граничне оквире око сваке инстанце, текстуално прикаже називе детектованих класа, исцрта маску сваке инстанце и нумерички прикаже поузданост предикција.

Задатак *predict_kayaker* методе је да учита улазни видео снимак, издели улазни видео снимак на фрејмове, генерише и сачува излазни видео снимак.

3.2.2 Модул за детекцију и сегментацију инстанце

За решавање проблема детекције кајакаша и његове сегментације коришћен је Mask R-CNN модел и

техника преносног учења. Техника преносног учења користи Mask R-CNN модел претходно обучен на Microsoft COCO скупу података, за генерисање новог модела.

За кичму (енгл. backbone) Mask R-CNN модела која је задужена за екстракцију карактеристика објеката из слике, одабрана је ResNet-101 архитектура. За потребе сегментација инстанци коришћен је стохастички градијент и Адамов оптимизатор.

Модул за детекцију и сегментацију инстанце се састоји од метода *train* и *evaluate_model*. Задатак *train* методе је да покрене процес тренирања модела у зависност од одабраних параметара. Задатак *evaluate_model* методе је да евалуира колико су добре предикције модела.

4. ЕВАЛУАЦИЈА РЕШЕЊА И РЕЗУЛТАТИ

За евалуацију резултата модела коришћена је *mAP* (енгл. *mean Average Precision*) метрика. То је уобичајена метрика за проблем детекције. *mAP* се ослања на метрике прецизности (енгл. *precision*) и одзива (енгл. *recall*) који се рачунају по формули:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Где је *TP* број *True Positive* (тачно позитивних) детекција, *FP* број *False Positive* (нетачно позитивних) детекција, і *FN* број *False Negative* (нетачно негативних) детекција.

Прецизност одговара на питање колико од тога што је детектовано је релевантно, а одзив на питање колико од тога што је релевантно је детектовано.

Average Precision (AP) метрика односи се на једну класу и рачуна се тако што се узму све регије од интереса које је модел одредио за посматрану класу (укупно *n* регија). Регије се потом сортирају по сигурности тј. по *IoU* (енгл. *Intersection over Union*) детектоване регије (енгл. *RoI*) и истините лабеле (енгл. *GT- ground truth label*) по формули:

$$IoU = \frac{RoI \cap GT}{RoI \cup GT}.$$

На основу сортираних регија формира се график на ком x оса представља одзиве од 0, 0.1, 0.2 …, 1 (узимајући у обзир нулту регију) прву регију, прве две регије, …, свих *n* детектованих регија), а *y* оса прецизност детекција за одређену вредност одзива.

Нпр. за вредност одзива 0.5 рачуна се прецизност детекције првих 50% регија, где се детекција сматра успешном уколико је *IoU* већи од неке границе (нпр. 0.5). Од добијеног графика се потом формира нови график, који за сваку вредност одзива узима највећу вредност прецизности.

Коначно *Average Precision* се рачуна на основу графика као вредност површине испод зелене криве подељено са 11, а *mean Average Precision* као просечна вредност *Average Precision* метрике примењена над свим класама.

У табели 1. приказана је детаљна конфигурација параметара, коришћена приликом тренирања Mask R-CNN модела.

За потребе овог рада, тренирано је више модела са различитим параметрима. Модели и њихови резултати су приказани у табели 2.

Табела 1. *Приказ конфигурационих параметара који су коришћени приликом обучавања модела*

CONFIGURATIONS	
BACKBONE	resnet101
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	2
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	1
DETECTION_MIN_CONFIDENCE	0.7
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	1
GRADIENT_CLIP_NORM	5.0
IMAGES_PER_GPU	2
IMAGE_CHANNEL_COUNT	3
IMAGE_MAX_DIM	512
IMAGE_META_SIZE	14
IMAGE_MIN_DIM	512
IMAGE_MIN_SCALE	0
IMAGE_RESIZE_MODE	square
IMAGE_SHAPE	[512 512 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.001
LOSS_WEIGHTS	{'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE	14
MASK_SHAPE	[28, 28]
MAX_GT_INSTANCES	100
MEAN_PIXEL	[123.7 116.8 103.9]
MINI_MASK_SHAPE	(56, 56)
NAME	kayaker_cfg
NUM_CLASSES	2
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	1000
POST_NMS_ROIS_TRAINING	2000
PRE_NMS_LIMIT	6000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE	1
RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD	0.7
RPN_TRAIN_ANCHORS_PER_IMAGE	256
STEPS_PER_EPOCH	100
TOP_DOWN_PYRAMID_SIZE	256
TRAIN_BN	False
TRAIN_ROIS_PER_IMAGE	32
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	5
WEIGHT_DECAY	0.0001

Поред наведене конфигурације, модели који су обучавани за потребе овог рада, разликују се по почетним тежинама које су одабране, слојевима који се обучавају и укупном броју епоха који су постављени

приликом обучавања модела. Приказ одабраних параметара модела се може видети у табели 2, као и њихови резултати на тренинг и тест скупу података.

Табела 2. *Приказ модела и њихових резултата*

Модел	Почетне тежине	Тренирани слојеви	Број епоха	Тренинг скуп података *mAP*	Тест скуп података *mAP*
M1	COCO	heads	5	0.844	0.800
M2	COCO	heads	10	0.824	0.867
M3	COCO	heads	15	0.844	0.867
M4	COCO	heads	20	0.844	0.800
M5	COCO	all	5	0.865	0.867
M6	COCO	all	10	0.888	0.933
M7	COCO	all	15	0.865	0.867
M8	COCO	all	20	0.888	0.933
M9	M8	heads	5	0.865	0.933
M10	M8	heads	10	0.885	0.933
M11	M8	heads	15	0.865	0.867
M12	M8	heads	20	0.885	0.933

На основу резултата из табеле 2, може се уочити да најслабије резултате имају модели *M1, M2, M3 и M4*. Приликом обучавања ових модела, коришћена је техника преносног учења, а за почетне тежине одабран је претходно трениран модел који је обучаван на *MS COCO* скупу података. Наведеним моделима су обучаване само њихове главе а под тим се подразумева да је трениран део за класификовање објекта, део за одређивање граничног оквира и део за генерисање маске.

За почетне тежине *M5, M6, M7 и M8* модела је такође одабран модел који је обучаван на *MS COCO* скупу података, само су код њих тренирани сви слојеви модела. Овако тренирани модели су дали боље резултате из разлога што су тренирани и слојеви *Backbone*, *RPN RoIAlign*, који су задужени за екстракцију карактеристика објеката из слика.

На основу претходних резултата може се закључити да је за потребе успешне детекције и сегментације кајакаша, приликом тренирања модела, неопходно поново истренирати и делове модела који су задужени за екстракцију карактеристика.

За почетне тежине модела *M9, M10, M11 и M12* одабран је модел *M8* који је претходно дао најбоље резултате предвиђања. Приликом обучавања наведених модела, тренирани су само делови *M8* модела који се баве класификацијом, детекцијом и сегментацијом инстанци, са циљем добијања бољих резултата, међутим њихово поновно тренирање није резултовало побољшањем резултата модела.

5. ЗАКЉУЧАК

У овом раду представљен је систем који се бави детекцијом и сегментацијом кајакаша на видео снимку. За потребе овог рада коришћен је ручно прикупљен скуп података, и *Matterport Inc. Mask R-CNN* модел конволуционе неуронске мреже.

Приликом прављења модела коришћена је метода преносног учења. Најбољи резултати су се показали код модела који су за основу користили MS COCO скуп података и код којих су тренирани свих слојеви Mask R-CNN модела.

Представљено решење се може побољшати проширењем постојећег скупа података као и додатним подешавањем хиперпараметара модела.

6. ЛИТЕРАТУРА

[1] Abdulla, W.: Mask R-CNN for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask RCNN (2017), accessed 07-Oct-2020

[2] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in ECCV, 2014.

[3] Girshick, Ross & Donahue, Jeff & Darrell, Trevor & Malik, Jitendra. (2015). Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 38. 1-1. 10.1109/TPAMI.2015.2437384.

[4] Handalage, Upulie & Kuganandamurthy, Lakshini. (2021). Real-Time Object Detection Using YOLO: A Review. 10.13140/RG.2.2.24367.66723.

[5] VGG Image Annotator (VIA) https://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html [приступљено 24.11.2022.]

[6] Matterport Inc. Mask R-CNN https://github.com/matterport/Mask_RCNN [приступљено 24.11.2022.]

Кратка биографија:

Никола Дакић рођен је 1994. године у Новом Саду. Основне академске студије завршио је 2019. године на Факултету техничких наука у Новом Саду. Мастер рад је одбранио 2022. године из области Електротехнике и рачунарства, смер Софтверско инжењерство и информационе технологије - модул Интелигентни системи.
контакт: ndakic94@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4
DOI: https://doi.org/10.24867/23BE07Djuric

PRIMENA VEŠTAČKE INTELIGENCIJE ZA IDENTIFIKACIJU STANJA ELEKTRIČNOG BROJILA

APPLICATION OF ARTIFICAL INTELLIGENCE TO IDENTIFY THE STATE OF THE ELECTRIC METER

Aleksandar Đurić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Skeniranje slika i pretvaranje skeniranih informacija u digitalni format je aktivna oblast istraživanja. Skeniranje je automatizovan, brz i efikasan proces u poređenju sa tradicionalnim unosom podataka. Prepoznavanje cifara sa slika je izazovan zadatak. Tradicionalni pristupi za rešavanje ovog problema obično razdvajaju korake lokalizacije, segmentacije i prepoznavanja. Ovaj rad predstavlja jedinstven pristup koji integriše ova tri koraka korišćenjem duboke konvolucione neuronske mreže koja radi direktno na pikselima slike.*

Ključne reči: *Veštačka inteligencija, Mašinsko učenje, Duboko učenje, Neuronske mreže*

Abstract – *Scanning images and converting the scanned information into digital format is an active research area. Scanning is an automated, fast and efficient process as compared to traditional data entry. Recognizing digits from images is a challenging task. Traditional approaches to solve this problem typically separate the localization, segmentation, and recognition steps. This paper presents a unified approach that integrates these three steps via the use of a deep convolutional neural network that operates directly on the image pixels.*

Keywords: *Artifical intelligence, Machine learning, Deep learning, Neural networks*

1. UVOD

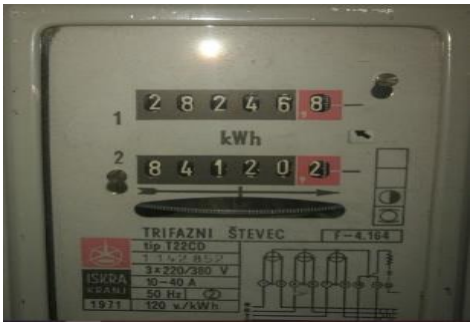
Problem prepoznavanja oblika je ljudima veoma blizak, sa njime se sreću tokom celog života, počevši od najranijeg detinjstva. Prepoznavanje oblika nije urođena osobina nego se stiče učenjem. Sposobnost prepoznavanja oblika ljudi razvijaju i usavršavaju gotovo celog svog života. Do nedavno su samo živi organizmi imali sposobnost prepoznavanja oblika. U poslednje vreme, zahvaljujući razvoju informacionih tehnologija i veštačke inteligencije, ostvareni su konkretni rezultati u oblasti prepoznavanja oblika. Pojam teorije prepoznavanja oblika podrazumeva matematičke metode, prvenstveno namenjene automatskom klasifikovanju konkretnih objekata. Teorija prepoznavanja oblika oslanja se uglavnom na upotrebu metoda veštačke inteligencije kao što su: neuronske mreže, rasplinuta logika (eng. fuzzy logic), matematička logika, statističke metode, itd.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Selakov, docent.

U ovom radu opisano je rešenje za identifikaciju stanja starih (nedigitalnih) brojila električne energije tj. identifikaciju tarifnih brojeva sa fotografija nedigitalnih električnih brojila. Tema ovog rada je opisivanje sistema koji funkcioniše na principu najnovijih tehnologija i koji se uz jednostavnu pripremu ulaznog skupa podataka i konfiguraciju samih parametara sistema može primeniti za rešavanje različitih slučajeva optičkog prepoznavanja karaktera. Cilj ovog rada jeste implementacija softvera koji pomaže popisivačima električne energije tako što ne moraju da prepisuju stanje brojila, već samo fotografišu brojilo, a softver prepozna stanje. Takođe, pomoću ovog softvera nedigitalna brojila je moguće pretvoriti u poludigitalna brojila, tako što bi se pored brojila stavila kamera koja na svakih 15 minuta snima stanje, koje zatim softver prepoznaje i prosleđuje dalje rezultate. Na ovaj način se simulira ponašanje modernih digitalnih brojila, a istovremeno i eliminiše potreba za popisivačima električne energije.

2. BAZA ULAZNIH PARAMETARA MREŽE

Na fotografijama u bazi prikazana su stara (nedigitalna) brojila električne energije (*Slika 1*). Cilj sistema je da za datu fotografiju brojila detektuje tarife i prikaže brojeve sadržane u okviru tarifa. Kako ne postoji javno dostupna baza podataka koja u sebi sadrži slike starih brojila, pre same realizacije sistema bilo je neophodno prikupiti odgovarajući broj fotografija. Uspešno je prikupljeno oko 2.500 fotografija. Nakon procesa prikupljanja fotografija, važno je bilo odraditi i proces filtriranja fotografija tj. uklanjanje fotografija sa šumovima (mutne fotografije ili fotografije kod kojih nisu jasno vidljive cifre na jednoj od tarifa). Posle procesa filtriranja, bazu podataka sačinjava skup od 2.330 fotografija.



Slika 1. *Primer fotografije za prepoznavanje*

Korišćene su tri metode preprocesiranja: konvertovanje kolor fotografija u monohromatske, smanjivanje dimenzija i augmentacija.

Za augmentaciju je korišćena skripta [1] pomoću koje je nad svakom ulaznom fotografijom izvršeno pet operacija: rotiranje u levo, rotiranje u desno, uveličavanje fotografije, iskrivljenje u levu ili desnu stranu i iskrivljenje unazad ili unapred, te je stoga i skup podataka uvećan pet puta. Proces filtriranja je pimenjen na fotografije dobijene nakon augmentacije zbog ponovne pojave fotografija sa šumom. Konačno, skup podataka za obuku i validaciju je imao 13.177 monohromatski fotografija starih brojila veličine 180x180 piksela.

Model obučen prethodno opisanim skupom podataka nije generisao zadovoljavajuće rezultate prepoznavanja.

Detaljnijom analizom ulaznih fotografija došlo se do zaključka da one sadrže, za svrhe ovog rada, veliki broj bespotrebnih informacija. Na osnovu slike (*Slika 1*) vidimo da pored tarifnih cifara, postoje delovi ulaznih fotografija (uglavnom je to donji levi ugao i/ili donji desni ugao) koji takođe sadrže određen niz cifara. Sistem sam vrši skeniranje u potrazi za karakterima za prepoznavanje, nema predefinisane regione. Ulazne fotografije koje pored tarifnih cifara poseduju dodatne regije u okviru kojih se nalaze određeni nizovi cifara, vrlo verovatno, će zbuniti model prilikom obuke tj. pomeriće fokus modela sa tarifnih cifara na druge delove fotografije, a to dalje rezultira lošim prepoznavanjem.

Kako bi povećali preciznost prepoznavanja potrebno je, na neki način, ukloniti suvišne informacije sa ulaznih fotografija. Ovaj problem je rešen izdvajanjem regija od interesa sa ulaznih parametara mreže. Ideja izdvajanja regija od interesa je preuzeta iz rada [2].

Originalni skup podataka ima 818 fotografija na kojima se nalazi jednotarifno brojilo i 1.512 fotografija na kojima su dvotarifna brojila. Na 8 fotografija jednotarifnih brojila, zbog pozicija iz kojih su snimljene, nije bilo izvodljivo izdvojiti tarifu. Nakon završetka izdvajanja regija od interesa sa originalnih slika (*Slika 1*), dobijen je novi skup podataka koji sadrži 3.834 fotografije na kojima se nalaze samo tarife (*Slika 2*).

Slika 2. *Primer izdvojene tarife*

2.1 Iteracija 1: 128x128 bez augmentacije

Prvobitnom izmenom, fotografije na kojima se nalaze tarife su sa originalne veličine smanjene na veličinu 128x128 piksela, a potom konvertovane u monohromatske fotografije. Značajan napredak je postignut u kvalitetu prepoznavanja. Ovo je razlog zbog kojeg proces augmentacije nije realizovan nad ovim skupom podataka. Pozitivan pomak u kvalitetu prepoznavanja bio je samo dodatni motiv za dalje eksperimentisanje sa dimenzijama ulaznih parametara mreže.

2.2 Iteracija 2: 96x96 bez augmentacije

Kako bi se našao kompromis između neophodnih resursa i kvaliteta prepoznavanja, originalne fotografije, na kojima se nalaze tarife, u ovoj iteraciji su smanjene na veličinu 96x96 piksela. Fotografije su nakon modifikacije dimenzija konvertovane u monohromatske. Rezultati prepoznavanja ove iteracije su poprilično dobri i iz tog razloga proces augmentacije nije primenjen.

2.3 Iteracija 3: 64x64 bez augmentacije

Kvalitetni rezultati prethodne iteracije bili su samo dodatni motiv za dalje eksperimentisanje sa dimenzijama ulaznih parametara mreže. Kako su rezultati prethodne iteracije poboljšani smanjenjem dimenzija fotografija, doneta je odluka da se nastavi sa istom paradigmom. Dakle, originalne fotografije, na kojima se nalaze tarife, su u okviru ove iteracije smanjene na veličinu 64x64 piksela. Osim smanjena veličine, smanjen je i kvalitet fotografija, što je uticalo na kvalitet same obuke. Fotografije su nakon korekcije dimenzija konvertovane u monohromatske. Odluka da se nastavi istom paradigmom se ispostavila kao pogrešna, jer su rezultati prepoznavanja ove iteracije za nijansu lošiji u odnosu na prethodnu iteraciju. Rezultati prepoznavanja ove iteracije, sami za sebe, su zadovoljavajući, te stoga nije bilo potrebe za realizovanjem procesa augmentacije.

3. ARHITEKTURE MREŽE

U cilju ostvarivanja što boljih rezultata procesi pretprocesiranja, učenja i testiranja su ponavljani u više navrata sa različitim parametrima i različitim arhitekturama neurosnke mreže. Od svih izmena napravljenih tokom pomenutih procesa izdvajaju se dve arhitekture modela.

Svaka korišćena arhitektura je posedovala konvolucionu neuronsku mrežu (KNM) čiji je zadataka da vrši segmentaciju fotografija i izdvajanje ključnih osobina koje sadrže karaktere za prepoznavanje, a zatim da ih prosledi, zavisno od arhitekture, potpuno povezanoj neuronskoj mreži (PPNM) ili povratnoj neuronskoj mreži sa dugotrajnom kratkoročnom memorijom (PNMDKM) koje za zadatak imaju klasifikaciju dobijenih ključnih osobina i da kao rezultat daju karaktere sa fotografije u digitalnoj formi.

3.1 KNM–PPNM arhitektura

Konvoluciona neuronska mreža (KNM) je izgrađena pomoću gradivnih blokova koji imaju sledeću strukturu:

- Konvolucioni sloj
- Sloj aktivacione funkcije
- Konvolucioni sloj
- Sloj aktivacione funckije
- Sloj sažimanja maksimalnom vrednošću po obe dimenzije

Arhitektura konvolucione neuronske mreže se sastoji od tri prethodno opisana gradivna bloka.
Potpuno povezana neuronska mreža kao ulaz zahteva vektor podataka. Iz tog razloga se izlazni tenzor konvolucione neuronske mreže dalje propušta kroz sloj za izravnavanje (eng. Flatten), koji od izlaza konvolucione mreže pravi jednodimenzionalni vektor.
Potpuno povezana neuronska mreža (PPNM) je izgrađena pomoću sledećih slojeva:

- Potpuno povezani sloj
- Sloj aktivacione funkcije
- Potpuno povezani sloj
- Sloj aktivacione funkcije
- Sloj za sprečavanje preobučavanja (eng. Dropout)

Skup podataka pored fotografija sa petocifrenim tarifama sadrži i nekoliko desetina fotografija na kojima se nalaze šestocifrene tarife, te iz tog razloga mreža ima šest

klasifikacionih slojeva. Broj klasa koje neuronska mreža treba da raspozna je 11, brojevi iz opsega [0,10]. Za identifikaciju tarifnih cifara koriste se brojevi iz opsega [0,9], a broj 10 služi kao indikator na osnovu kog se zna da li se radi o petocifrenoj ili šestocifrenoj tarifi.

3.2 KNM–PNMDKM arhitektura

Konvoluciona neuronska mreža i klasifikacioni slojevi su identični kao kod KNM-PPNM arhitekture. Prema tome, ovde se fokus stavlja na povratnu neuronsku mrežu.

Izlazni oblik (eng. output shape) tenzora konvolucione neuronske mreže nije upotrebljiv kao ulaz u povratnu neuronsku mrežu sa dugotrajnom kratkoročnom memorijom. Izlazni tenzor konvolucione neuronske mreže ima 4 dimenzije, a povratna neuronska mreža sa dugotrajnom kratkoročnom memorijom kao ulaz očekuje 3 dimenzije. Ovaj problem je rešen upotrebom sloja za promenu dimenzija (eng. Reshape).

Povratna neuronska mreža sa dugotrajnom kratkoročnom memorijom (PNMDKM) je izgrađena pomoću sledećih slojeva:

- Sloj povratne neuronske mreže sa dugotrajnom kratkoročnom memorijom
- Sloj aktivacione funkcije
- Sloj povratne neuronske mreže sa dugotrajnom kratkoročnom memorijom
- Sloj aktivacione funkcije

Izlaz povratne neuronske mreže sa dugotrajnom kratkoročnom memorijom se prosleđuje klasifikacionim slojevima.

4. PROCES UČENJA SISTEMA

U mašinskom učenju, cilj je postići dobru generalizaciju modela, odnosno da se model uspešno izvršava nad podacima koji nisu viđeni ranije. Ukoliko je trenirani model previše dobro prilagođen podacima kojima je treniran, može doći do stanja previše prilagođenog modela (eng. overfitting) [3-5]. S obzirom da se algoritam prostiranja greške unazad u praksi pokazao pouzdanim i računski isplativim, korišćen je za treniranje mreža u ovom radu. Prilikom treniranja mreža, računa se funkcija gubitka (eng. loss) te preciznost (eng. accuracy) upotrebom kategoričke unakrsne entropije (eng. categorical crossentropy). Takođe, tokom treniranja se koristi optimizacioni algoritam Adam.

Skup podataka je podeljen tako da je za obuku neuronske mreže odvojeno 95% od ukupnog skupa, dok je preostalih 5% služilo za validaciju mreže.

4.1 Iteracija 1: 128x128 bez augmentacije

Kao najbolji model u okviru ove iteracije pokazao se model sa KNM–PPNM arhitekturom. Za obuku je korišćeno 3.642 monohromatske fotografije veličine 128x128 piksela koje su podeljene u pakete (eng. batch) od 32 fotografije, a izabrani broj epoha je 50. Treniranje modela je trajalo 110 minuta.

4.2 Iteracija 2: 96x96 bez augmentacije

Za ovu iteraciju najbolji model je izgrađen pomoću KNM–PNMDKM arhitekture. Trening skup, od 3.642 monohromatske fotografije veličine 96x96 piksela, je podeljen na pakete od 32 fotografije, dok je izabrani broj epoha iznosio 100. Kriterijum zaustavljanja je postavljen na 100 epoha. Treniranje modela je trajalo 150 minuta.

4.3 Iteracija 3: 64x64 bez augmentacije

Za treću iteraciju skup za obučavanje, od 3.642 monohromatske fotografije veličine 64x64 piksela, je podeljen u pakete od po 32 fotografije, dok je definisan broj epoha bio 50. Trening faza je trajala 32 minuta. Za poslednju iteraciju, model sa KNM–PPNM arhitekutrom se pokazao kao najkvalitetniji.

5. TESTIRANJE MREŽA I REZULTATI

Testiranje je izvršeno nad skupom fotografija koje su naknadno prikupljene i koje nisu učestvovale u trening fazi, a isto tako ni u validacionoj fazi. Korišćenjem sasvim novog skupa podataka za testiranje se proverava da mreža nije pretrenirana i da nije naučila neke karakteristike specifične samo za podatke u okviru trening skupa.

Za svrhe testiranja prikupljeno je 44 fotografije brojila električne energije. Skup od 44 fotografije se sastoji od 38 fotografija na kojima se nalaze dvotarifna brojila i 6 fotografija na kojima su jednotarifna brojila. U slučaju fotografija sa dvotarifnim brojilima, na 5 fotografija su brojila sa šestocifrenim tarifama, a na preostale 33 fotografije su brojila sa petocifrenim tarifama. Svaka fotografija sa jednotarifnim brojilom sadrži petocifrenu tarifu. Dakle, može se zaključiti da ukupan broj karaktera koje sistem treba da prepozna iznosi 420. Odnos prepoznatih karaktera i ukupnog broja za prepoznavanje uzet je kao parametar uspešnosti sistema. Pored ovoga analizirane su još neke karakteristike: uspešnost prepoznavanja kompletne tarife, prepoznavanje u odnosu na cifru koja treba da se prepozna i u odnosu na poziciju u tarifi.

Kada uporedimo rezultate tri iteracije treniranja i testiranja mreže (*Tabela 1*) primetno je da pretposlednja izmena doprinosi unapređenju obučenosti mreže, dok poslednja izmena za nijansu narušava obučenost mreže.

Tabela 1. *Statistika uspešnosti prepoznavanja po iteraciji*

	Očekivano	**Prepoznato**	**Procenat uspešnosti**
Iteracija 1	420	396	94.29%
Iteracija 2	420	404	96.19%
Iteracija 3	420	400	95.24%

Sistem je u iteraciji 1 obučen da bolje uočava cifre na fotografijama u odnosu na preostale dve iteracije. Smanjenjem dimenzija ulaznih parametara sistema pojavljuju se fotografije na kojima sistem prepoznaje jednu ili dve cifre zavisno od iteracije (*Tabela 2*).

Tabela 2. *Uspešnost prepoznavanja kompletne tarife*

Prepoznato cifara	**Iteracija 1**	**Iteracija 2**	**Iteracija 3**
0	0	0	0
1	0	0	1
2	0	1	3
3	4	5	1
4	15	3	5
5	54	63	62
6	9	10	10
Ukupno:	**82**	**82**	**82**

U testnom skupu najmanje se pojavljuje broj osam, samo 18 puta. Sve tri iteracije su uspele da prepoznaju broj osam sa stoprocentnom tačnošću. U tabeli (*Tabela 3*) je pokazano da su iteracije 2 i 3 u potpunosti prepoznale četiri cifre, 1-2-8-9 u slučaju iteracije 2 i 4-6-8-9 u slučaju iteracije 3, dok je iteracija 1 u potpunosti prepoznala 3 cifre, 1-4-8.

Tabela 3. *Uspešnost prepoznavanja u odnosu na cifru koja se prepoznaje*

Očekivana cifra	Broj cifara	Iteracija 1	Iteracija 2	Iteracija 3
0	48	46	47	46
1	38	38	38	37
2	28	27	28	27
3	58	53	54	49
4	45	45	43	45
5	74	62	68	68
6	37	35	35	37
7	41	40	40	40
8	18	18	18	18
9	33	32	33	33
Ukupno:	**420**	**396**	**404**	**400**

Prilikom prepoznavanja stanja električnog brojila važno je da bar prve tri tarifne pozicije u slučaju petocifrenih brojila, a prve četiri tarifne pozicije u slučaju šestocifrenih brojila budu dobro prepoznate. Na primer, sa slike (*Slika 2*) se vidi da je stanje tarife: 28246, a neka je model prepoznao: 29246. Dakle, model jeste pogrešio samo jednu cifru ali pozicija na kojoj je došlo do zabune je važnija. U ovom slučaju, greška koja se propagira na stvarni svet je reda 1000 kWh, što nije malo ni sa elektroenergetske ni sa finansijske tačke gledišta. Ovo je razlog zbog kojeg je uveden kriterijum uspešnosti prepoznavanja broja u odnosu na poziciju (*Tabela 4*).

Tabela 4. *Uspešnost u odnosu na poziciju broja koji se prepoznaje*

Pozicija	Broj cifara	Iteracija 1	Iteracija 2	Iteracija 3
1	82	70	80	75
2	82	81	81	81
3	82	80	80	82
4	82	79	76	76
5	82	76	77	76
6	10	10	10	10
Ukupno:	**420**	**396**	**404**	**400**

6. ZAKLJUČAK

Glavni cilj ovog rada bio je implementacija softvera za identifikaciju stanja električnog brojila. Pod identifikacijom stanja se podrazumeva pretvaranje karaktera, u ovom slučaju brojeva, sa fotografija u njihove digitalne ekvivalente. Dakle, identifikacija stanja suštinski predstavlja problem koji je dosta rasprostranjen u modernom svetu – optičko prepoznavanje karaktera.

U radu su opisane metode za rešavanje ovog problema koje podrazumevaju pretprocesiranje ulaznih podataka, njihovu segmentaciju, zatim prepoznavanje ključnih osobina za detekciju i klasifikaciju karaktera sa fotografije. Prikazane su najbolje metode za rešavanje datog problema iz perspektive neuronskih mreža.

Na osnovu rezultata testiranja najboljih arhitektura svake iteracije ustanovljeno je da se neuronska mreža, sa KNM-PNMDKM arhitekturom iz iteracije 2, istakla po svim segmentima prepoznavanja, te je iz tog razloga izabrana kao najkvalitetnija neuronska mreža. Rezultati testiranja izabrane neuronske mreže su sledeći, preciznost prepoznavanja individualnih cifara iznosila je 96.19%, a preciznost prepoznavanja kompletnog niza cifara je iznosila 89.02%, mreža je od 82 tarife uspešno prepoznala 73. Dobijeni rezultati govore o tome da je cilj u uvodu postignut. Razvijen je softver koji, korišćenjem najnovijih tehnologija, sa velikom tačnošću identifikuje stanje električnog brojila.

7. LITERATURA

[1] https://github.com/mdbloice/Augmentor, skripta za augmentaciju, poslednji pristup 14.12.2022.

[2] Muhammad Asif, Maaz Bin Ahmad, Shiza Mushtaq *et al.* "Long Multi-digit Number Recognition from Images Empowered by Deep Convolutional Neural Network". The Computer Journal, Volume 65, Issue 10, October 2022, Pages 2815–2827. 13 September 2021.

[3] Tetko, I. V., Livingstone, D. J. and Luik, A. I. „Neural network studies. 1. Comparison of overfitting and overtraining," Journal of chemical information and computer sciences, 1995, 35.5: 826-833.

[4] Dietterich, T. „Overfitting and undercomputing in machine learning," ACM computing surveys (CSUR), 1995, 27.3: 326-327.

[5] Caruana, R., Lawrence, S., Giles, CL. „Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping." In: Advances in neural information processing systems. 2001. p. 402- 408.

Kratka biografija:

Aleksandar Đurić rođen je u Šapcu 1998. godine. Osnovne akademske studije na Fakultetu tehničkih nauka Univerziteta u Novom Sadu upisao je 2017. godine. Diplomirao je 2021. godine i iste godine upisuje master akademske studije.

Kontakt: acadjuric98@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/23BE08Lazic

ANALIZA MAGNETSKOG POLJA I TOPLOTNIH EFEKATA U OKOLINI TROFAZNOG ŠINSKOG RAZVODA

ANALYSIS OF MAGNETIC FIELD AND THERMAL EFFECTS AROUND THREE PHASE BUSBAR SYSTEM

Aleksa Lazić, Stevan M. Cvetićanin, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ENERGETIKA, ELEKTRONIKA I TELEKOMUNIKACIJE

Kratak sadržaj – *U radu su predstavljene opšte karakteristike elektroenergetskih sistema sa osvrtom na primenu bakarnih šinskih provodnika u razvodnim postrojenjima. Prikazani su rezultati proračuna magnetskog polja, gustine struje i toplotnih efekata bakarnih provodnika kao dela trofaznog sistema. Simulacija je rađena u COMSOL Multiphysics 5.5 programskom paketu koji numeričkim putem rešava parcijalne diferencijalne jednačine primenom metoda konačnih elemenata.*

Ključne reči: *Šinski razvod, vektor magnetske indukcije i vektor gustine struje, prenos toplote kroz čvrsta tela, temperatura provodnika.*

Abstract – *In this paper an electric power system and the application of power busbar system of copper conductors in switchyards is presented. The analysis of magnetic field, current density, and thermal effects of the copper rail conductors was performed. The simulation of these quantities was done in the COMSOL Multiphysics 5.5 software package, which solves partial differential equations using the finite element method.*

Keywords: *power busbar system, magnetic flux density and current density, heat transfer in solids, temperature of conductors.*

1. UVOD

Elektroenergetski sistemi predstavljaju osnovu snabdevanja električnom energijom počevši od njene proizvodnje, pa do snabdevanja krajnjih korisnika. Podsistem proizvodnje sastoji se od elektrana u kojima se primarni oblici energije pretvaraju u električnu.

U današnje vreme sve češće se koriste elektrane koje koriste obnovljive izvore energije, kao što su vetar (vetrogeneratori) i sunce (foto-naponski paneli). U podsistem prenosa spadaju dalekovodi koji prenose energiju do podsistema distribucije koja raspodeljuje energiju do krajnjih potrošača.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Stevan Cvetićanin.

2. OPŠTE KARAKTERISTIKE RAZVODNIH POSTROJENJA

Razvodno postrojenje je objekat koji je namenjen za transformaciju, odnosno, razvođenje električne energije. Predstavlja osnovnu celinu elektroenergetskog sistema. Može se podeliti na razvodno postrojenje na otvorenom prostoru i razvodno postrojenje unutar zgrade koje je smešteno u ćelijama.

Prema načinu izolacije samog postrojenje ono se može podeliti na: AIS – vazduhom izolovano postrojenje, GIS – gasom izolovano postrojenje (najčešće SF6) i HIS – hibridno izolovano postrojenje koje je sastavljano kombinacijom prethodne dve izvedbe.

Sabirnice i neizolovani provodnici

Sabirnice su sačinjene od neizolovanih provodnika i spadaju u glavne elemente razvodnih postrojenja. Šinski razvod se sastoji od provodnika koji mogu biti neizolovanog ili izolovanog tipa. Neizolovani provodnici se koriste za veze između aparata i za šinski razvod koji povezuje srednjenaponski rasplet na transformatoru sa postrojenjem srednjeg napona (35 kV, 20 kV i 10 kV), odnosno sa ćelijama u zgradi. Za izradu sabirnica na visokonaponskoj strani najčešće se koriste aluminijumski provodnici i njegove legure (AlMgSi0,5 F22), aluminijumsko-čelična užad najčešćeg preseka 240/40 mm^2. Za izradu sabirnica na niskonaponskoj strani koriste se neizolovani bakarni provodnici u obliku pravougaonih šina preseka od 30 x 5 mm do 120 x 10 mm, u zavisnosti od struje opterećenja. Na slici 1 prikazan je izgled srednjenaponskog raspleta u vidu šinskog razvoda.

Slika 1. *Izgled šinskog razvoda koji je na sekundarnoj strani energetskog transformatora.*

3. TEORIJSKA OSNOVA PRORAČUNA

3.1 Elektromagnetsko polje

Da bismo rešili problem analize elektromagnetskog polja, polazimo od Amperovog zakona u diferencijalnom obliku, koji predstavlja jednu od Maksvelovih jednačina:

$$\nabla \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t}. \quad (1)$$

Pri tome je:

- H – jačina magnetskog polja [A/m],
- J – gustina struje [A/m^2],
- D –vektor električnog pomeraja [C/m^2].

Osim toga, koristimo i konstitutivne relacije kako bi opisali svojstva prostora, odnosno sredine u kojoj se određuje magnetsko polje. One su date na sledeći način:

$$\vec{J} = \sigma \vec{E}, \quad (2)$$

$$\vec{B} = \mu \vec{H}\,, \mu = \mu_0 \cdot \mu_r, \quad (3)$$

$$\vec{D} = \varepsilon \vec{E}\,, \varepsilon = \varepsilon_0 \cdot \varepsilon_r. \quad (4)$$

Pri tome je:

- σ – specifična provodnost materijala [S/m]
- E – jačina električnog polja [V/m]
- μ – permeabilnost materijala [H/m]
- $\mu_0 = 4\pi \cdot 10^{-7}\mathrm{H/m}$ – permeabilnost vakuuma,
- μ_r – relativna permeabilnost
- B – vector magnetske indukcije [T]
- ε – permitivnost materijala [F/m]
- $\varepsilon_0 = 8{,}85 \cdot 10^{-12}$ [F/m] – dielektrična permitivnost vakuuma
- ε_r – relativna dielektrična permitivnost

Kada promene polja nisu tako brze, polje može da se posmatra kao kvazi-statičko. U tom slučaju Maksvelova jednačina (1) za kvazi-statičko polje glasi:

$$\nabla \times \vec{H} = \vec{J} + \vec{J}_e, \quad (5)$$

pri čemu je gustini struje J dodata i gustina eksternih struja J_e [A/m^2].

Određivanje magnetskog i električnog polja je moguće pomoću električnog skalar potencijala V i magnetskog vektor potencijala $\vec{A}$ i to na sledeći način:

$$\vec{B} = \nabla \times \vec{A}, \quad (6)$$

$$\vec{\mathrm{E}} = -\nabla \mathrm{V} - \frac{\partial \vec{A}}{\partial t}. \quad (7)$$

U radu je posmatrano samo indukovano električno polje ($\vec{E_i}$) predstavljeno drugim članom jednačine (7).

Uvrštavanjem izraza (2), (3), (6) i (7) u izraz (5), dobija se jednačina magnetskog vektor potencijala:

$$\sigma \frac{\partial \vec{A}}{\partial t} + \nabla \times \left(\frac{1}{\mu} \nabla \times \vec{\mathrm{A}} \right) = \vec{J}_e. \quad (8)$$

To je jednačina koju koristi programski paket COMSOL prilikom rešavanja problema kvazi-statičkog magnetskog polja. Iz magnetskog vektora potencijala se računa indukovano električno polje, a zatim i indukovana gustina struje. Ukupna gustina struja je J+J_e.

3.2 Prenos toplote

Prenos energije sa jednog na drugo telo vrši se na dva suštinski različita načina: u prvom slučaju energiju prenose materijalne čestice, a u drugom to čine elektromagnetski talasi. Prenos energije posredstvom materijalnih čestica odvija se takođe na dva znatno različita načina. Materijalne čestice, odnosno tela od tih čestica, mogu prenositi energiju ne krećući se vidljivo (telo se kao celina ne pomera). Tada se govori o *provođenju* toplote, a to se ostvaruje prenosom energije sa molekula na molekul tela (molekuli nikada ne stoje, nego se kod gasova haotično kreću po slobodnom prostoru, a kod čvrstih tela i mirnih tečnosti osciluju oko ravnotežnog položaja). Ako se telo kao celina kreće (misli se pre svega na strujanje fluida) i prenosi energiju sa jednog mesta na drugo govori se o *konvektivnom* (strujnom) prenosu toplote. Najzad, prenos energije posredstvom elektromagnetskih talasa se zove *zračenje.*

3.2.1 Provođenje toplote

Ovaj način prenosa energije vezan je pre svega za čvrsta tela kod kojih se samo po sebi obavlja oscilovanje molekula bez da neki deo čvrstog tela pokrene u odnosu na ostali (okolni) nepokretan deo. Toplota se provodi kroz čvrsto telo uvek u smeru opadanja temperature, dakle, od mesta više ka mestu niže temperature. Količina toplote koja se kroz telo provodi u jedinici vremena naziva se toplotni fluks. Ako se sa $\dot{Q}$ [W] označi toplotni fluks, sa $\dot{q}$ [W/m^2] gustina toplotnog fluksa i sa A površina kroz koju se toplota prenosi, tada važi da je:

$$\dot{q} = \frac{\dot{Q}}{A}. \quad (9)$$

Iskustvo pokazuje da mora postojati veza između gustine fluksa i temperaturnog polja, a da je ta veza različita za razne materijale. Maksimalna promena temperature po jedinici biće u pravcu normale, i ta se razlika naziva gradijent temperature na izabranom mestu izotermne površine (zamišljene površine na kojima u svakoj tački vlada ista temperatura).

$$\mathrm{grad}\, T = \vec{n}_o \frac{dt}{dn}. \quad (10)$$

gde je $\frac{dt}{dn}$ temperaturni gradijent u pravcu normale $\vec{n}_o$. Gradijent temperature ima pravac normale na izotermnu površinu, smer mu je (smer normale) ka izotermnoj površini više temperature. Veza između gustine toplotnog fluksa i temperature je izražena Furijeovim zakonom:

$$\vec{\dot{q}} = -\lambda\, \mathrm{grad}\, T\,. \quad (11)$$

Negativan znak je postavljen jer fluks ima smer ka izotermnoj površini niže temperature, a gradijent ka višoj temperature

3.2.2 Prenos toplote konvekcijom

Prenos toplote konvekcijom vezan je za kretanje fluidnih masa čiji je prostor oivičen čvrstim telima. Kada se fluid (gas, tečnost) kreće pored čvrstog zida više temperature nego u fluidu, toplota će se prenositi sa zida na fluidnu masu. Međutim, čim se pojavi razlika temperatura u dva susedna sloja fluida (tečnost ili gas) onda dolazi i do razmene toplote između njih. To izaziva sile potiska koje pokreću fluid i izazivaju takozvanu *prirodnu konvekciju.*

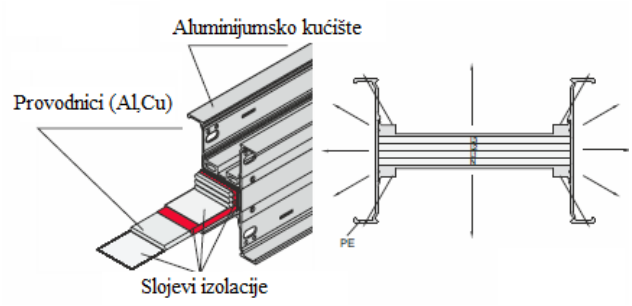
Daleko do zida fluidna masa ima skoro potpuno ujednačenu temperaturu t_f, dok na samom zidu ima istu temperaturu kao površina zida t_z. Tako se za prenos toplote konvekcijom piše sasvim prost izraz:

$$\dot{q} = \alpha \cdot \Delta t = \alpha \cdot (t_z - t_t) \left[\frac{J}{(m^2 s)} = \frac{W}{m^2}\right]. \qquad (12)$$

Tim izrazom definisan je *koeficijent prenosa* (konvekcije) toplote α $[W/(m^2K)]$.

4. MODEL

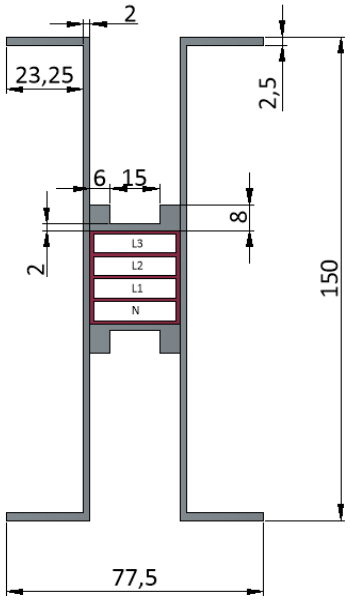
U ovom poglavlju je analiziran model sistema energetskih sabirnica, modularnog tipa, za prenos i distribuciju energije kreiran izolovanim aluminijumskim ili bakarnim provodnicima smeštenih u zatvorenom kućištu (Slika 2). Sistem sabirnica se koristi za prenos energije do tačke duž trase unutar objekta, počevši od tačke kao što je transformator, generator ili panel pa do krajnje tačke opterećenja. Model je analiziran u Comsol razvojnom okruženju. Primer takvog sabirnog šinskog razvoda je uzet iz kataloga firme *E-LINE-KX*.



Slika 2. *Izgled šinskog razvoda*

Na osnovu podataka iz kataloga proizvođača kreiran je model koji je korišćen u programskom paketu *COMSOL*.

Izgled modela koji je implementiran u samom programskom okruženju se nalazi na slici 3.



Slika 3. *Model trofaznog šinskog provodnika sa neutralnim provodnikom i aluminijumskim kućištem (dimenzije su predstavljene u [mm]).*

Osnovne karakteristike modela koje su neophodne za numerički proračun, kao što su permitivnost, provodnost ili permeabilnost, su uključene u samom programskom okruženju u okviru postojećih biblioteka. Samo za epoksidnu smolu to jest njen toplotni kapacitet i relativnu permitivnost su vrednosti uzete sa inženjerskog sajta za primenu materijala u tehničkim aplikacijama.

5. INSTRUKCIJE ZA MODELIRANJE

Za rešavanje konkretnog problema korišćen je *COMSOL Multiphysics* (5.5 verzija) programski paket za rešavanje parcijalnih diferencijalnih jednačina sa početnim i graničnim uslovima. Primenjuje se metod konačnih elemenata (*FEM – finite element method*), kojim se numeričkim putem dobijaju približna rešenja za rešavanje različitih fizičkih i inženjerskih problema. Suština ovog metoda konačnih elemenata je u diskretizaciji (podela) posmatranog domena na izabrane poddomene, odnosno konačne elemente, usvojenog oblika (u našem slučaju trouglove), koji su konačnih dimenzija sa čvornim tačkama na granicama. Pored trouglova, konačni elementi mogu biti: linijski segmenti, četvorouglovi, paralelopipedi (za 3D modele) i sl. Broj konačnih elemenata zavisi od složenosti modela i od toga da li se model može posmatrati kao dvodimenzionalni ili trodimenzionalni (*COMSOL* može da kreira i 2D i 3D modele).

Za modeliranje razmatranog problema koristimo dva modula *Heat Transfer* i *AC/DC*. *AC/DC* modul poseduje podmodul *Magnetic fields*, koji služi za izračunavanje raspodele gustine struje. *Heat transfer* modul poseduje dva podmodula: *Electromagnetic Heating* i *Induction Heating*, i on predstavlja spregu između magnetnog polja i prenosa toplota u čvrstim telima. U programskom paketu *COMSOL* to se deklariše kao multifizika.

U zavisnosti od vrste problema, bira se prostorna dimenzija. U našem slučaju prostorna dimenzija može biti i 2D i 3D. Zbog kompleksnosti i vremenski zahtevnog 3D modela kao i zbog dužine samog objekta kod kog nema promene parametara duž ose sistema (svaki poprečni presek je isti) izabran je 2D model.

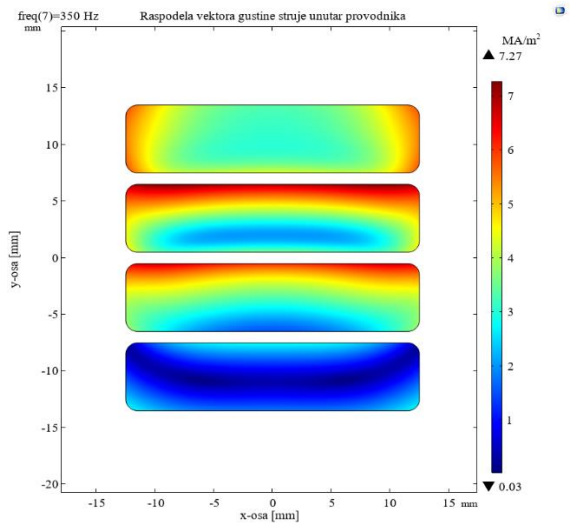
Detaljna uputstva i koraci prilikom izrade modela u programskom paketu *COMSOL* su opisana u samom master radu.

6. REZULTATI

U radu je analiziran slučaj trofaznog šinskog provodnika sa uvaženim svojstvima okoline. Domen od interesa po pitanju temperature su sami provodnici stoga je urađena detaljna analiza i pokazano da nema odstupanja veličina posle ukidanja okoline modela (vazduha).

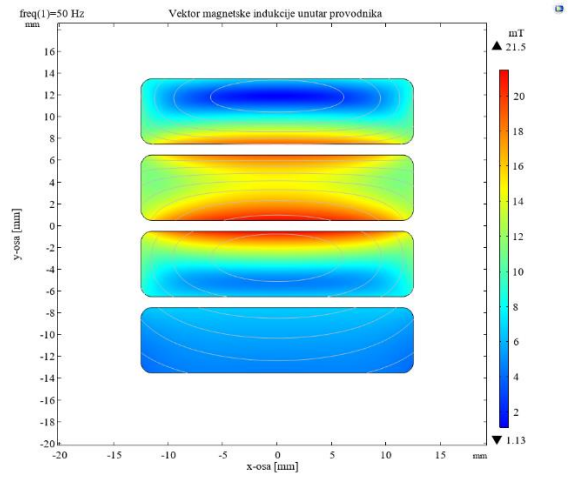
U master radu su grafički prikazani: vektor magnetske indukcije, vector gustina struje i temperatura, kao i njihove najmanje i najveće vrednosti u određenim tačkama modela.

Za slučaj trofaznog provodnika u simetričnom režimu, pri jačini struje od 550A. Maksimalni intenzitet vektora gustine struje iznosi $7{,}27 \cdot 10^6$ A/m^2 pri frekvenciji od 350Hz koji je prikazan na slici 4.



Slika 4. *Raspodela vektora gustine struje za 350Hz.*

Maksimalni intenzitet magnetske indukcije iznosi 21,mT pri frekvenciji od 50Hz i. prikazan je na slici 5.



Slika 5. *Raspodela vektora magnetske indukcije za 50Hz.*

Rađena je analiza uticaja magnetskog polja po zdravlje čoveka prema pravilniku o nejonizujućim zračenjima. Ustanovljeno je da magnetsko polje naglo opada sa udaljavanjem od šinskog razvoda i na manjim distancama (preko 20cm) od razvoda je bezbedno za boravak čoveka.

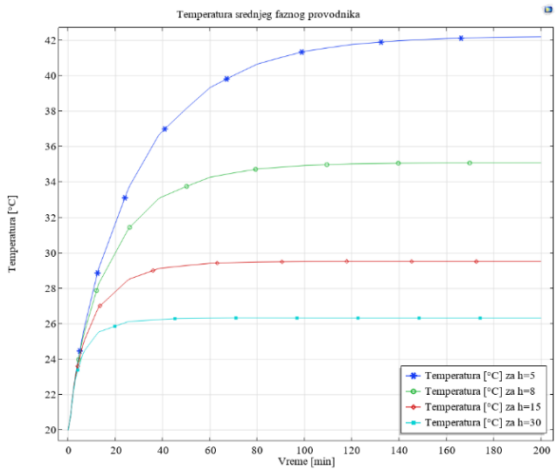
Za analizu temperature korišćen je frekvencijsko-tranzijentni domen primenom dve metode simulacija. Prva metoda je rađena za frekvenciju od 50Hz, gde se zadavao koeficijent odvođenja toplote (h) celog sistema.

U zavisnosti od njegove vrednosti, određen period je bio potreban kako bi temperature ušla u ustaljeni režim.

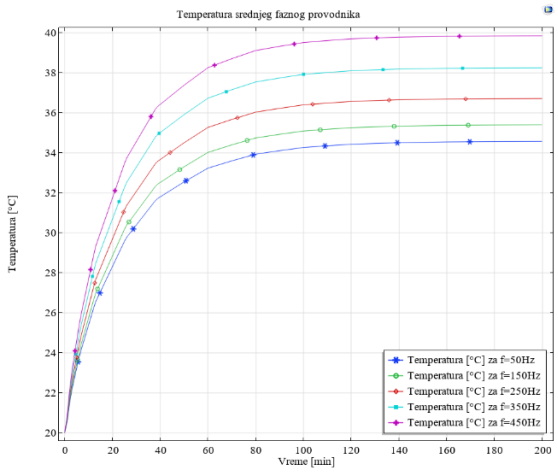
Najmanja vrednost koeficijenta prenosa toplote uzima u obzir i najveću temperaturu, pošto se model sporije hladi i sporije odvodi toplotu. Prikaz rezultata prve metode dat je na slici 6.

Drugi metod rađen je za opseg frekvencija od 50 do 500Hz koji je vezan za spoljašnje prirodno hlađenje.

Primenjuje se direktnim zadavanem vrednosti dimenzija elemenata šinskog razvoda. Prikaz rezultata druge metode dat je na slici 7.



Slika 6. *Grafik zavisnosti temperature drugog faznog provodnika (srednjeg) za različite vrednosti koeficijenta prenosa toplote (h).*



Slika 7. *Grafik zavisnosti temperature drugog faznog provodnika (srednjeg) za različite vrednosti frekvencija.*

7. ZAKLJUČAK

Sa povećanjem frekvencije kreće i povećanje vektora gustine struje i dolazi do povećanja raspodele ka površini provodnika. U drugom faznom provodniku se indukovala najveća struja. Jedan deo se indukovao i u kućištu pošto je u pitanju metalno aluminijumsko kućište, ali znatno manje nego unutar samih provodnika. Sa druge strane intenzitet vektora magnetske indukcije opada sa povećanjem frekvencije. Uz samo kućište je najveće polje i brzo opada sa udaljenjem od šinskih provodnika. A

nalizom je pokazano da na kratkim distancama od šinskog razvoda je sasvim bezbedan boravak po čoveka prema zakonu o nejonizujućim zračenjima. Uočava se da sa porastom koeficijenta prenosa tolpote opada temperature i koja posle određenog vremena ulazi u ustaljeno stanje. Temperatura je najveća unutar drugog faznog provodnika kao posledica najveće indukovane struje u njemu. Sa povećanjem frekvencije povećava se i temperature ali skoro zanemarljivo utiče na period gde se ona ustaljuje.

8. LITERATURA

[1] https://www.eaeelectric.com/catalogs/busbar-systems/e-line-kx-busbar.pdf
[2] Comsol AD/DC Module User's Guide, documentation for version 5.5
[3] Comsol Heat Transfer Module User's Guide, documentation for version 5.5
[4] Comsol Multiphysics Reference Manual, documentation for version 5.5
[5] Vladimir C. Strezoski, *Osnovi Elektroenergetike*, FTN Izdavaštvo, Novi Sad, 1996..
[6] Sreten Škuletić i Nikola Kaljević, *Visokonaponska Razvodna Postrojenja*, udžbenik za treći razred srednje stručne škole, Zavod za udžbenike i nastavna sredstva, Podgorica, 2019.
[7] https://www.elvod.net/wp-content/uploads/2018/07/01-RAZVOD-ZA-VELIKE-STRUJE-VR.pdf
[8] Branko D. Popović, *Elektromagnetika*, Građevinska knjiga, Beograd, 1990.
[9] Mića Marić, *Nauka o toploti, Termodinamika, Prenos toplote, Sagorevanje*, FTN Izdavaštvo, Novi Sad, 2006.
[10] Mića Marić, *Nauka o toploti, Kratki kurs*, FTN Izdavaštvo, Novi Sad, 2009.
[11] M. Sekulić, *Metoda konačnih elemenata*, Građevinska knjiga, Beograd, 1988.
[12] *Pravilnik o granicama izlaganja nejonizujućim zračenjima*, Sl.Glasnik RS, br.36/09, str.8.

Kratka biografija:

Aleksa Lazić rođen je u Vršću 1995. god. Osnovne akademske studije na Fakultetu tehničkih nauka upisuje školske 2014/2015. godine, smer energetike, elektronike i telekomunikacija. Diplomirao je 2019.god. i iste godine upisuje master studije na Fakultetu tehničkih nauka, smer elektroenergetika-elektroenergetski sistemi.

kontakt: aleksalazic@live.com

Stevan Cvetićanin rođen je 4.10.1986. u Bačkoj Palanci. Doktorirao je 2017. godine na studijskom programu Energetika, elektronika i telekomunikacije Fakulteta tehničkih nauka, Univerziteta u Novom Sadu. Trenutno radi kao docent na Katedri za elektroenergetiku i primenjeno softversko inženjerstvo.

kontakt: stevan.cveticanin@uns.ac.rs

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39
DOI: https://doi.org/10.24867/23BE09Lalovic

OPTIMIZACIJA APLIKACIJE QRTV REMINDER ZA RAZLIČITE OPERATIVNE SISTEME

QRTV REMINDER APPLICATION OPTIMIZATION FOR VARIOUS OPERATING SYSTEMS

Vladimir Lalović, Željen Trpovski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Razvijanjem i napretkom tehnologija i povećanjem baze korisnika, QR kodovi su postali pogodan posrednik u prenosu male količine podataka. U radu su predočene karakteristike QR kodova i potencijalne mogućnosti primene za različite namene. Predstavljena je pre svega primena QR koda kao multimedijalnog posrednika kao i praktičan deo razvoja mobilne aplikacije QRTV Reminder za Android i iOS operativne sisteme uz upotrebu Dart programskog jezika u Flutter multi-platform okruženju.*

Ključne reči: *QR kod, Dart, Flutter, Android, iOS*

Abstract – *With the development and progress of technologies and the increase of the user base, QR codes have become a suitable intermediary in the transfer of small amounts of data. The paper presents the characteristics of QR codes and potential possibilities for various purposes. First of all, the application of the QR code as a multimedia mediator was presented, as well as the practical part of the development of the QRTV Reminder mobile application for Android and iOS operating systems using the Dart programming language in the Flutter multi-platform framework.*

Keywords: *QR code, Dart, Flutter, Android, iOS*

1. UVOD

QR (engl. Quick Response) kod izabran je kao izuzetno dobro rešenje za prenos malih količina podataka umesto klasičnih bar kodova. Shodno tome QR kod je postao vitalni deo i prilikom kreiranja QRTV Remainder aplikacije koja je najbitniji segment ovog rada.

Takođe su opisane i karakteristike Quick Response koda i njegova primena, pogotovo kao multimedijalnog posrednika između TV programa i njihovih gledalaca, sa posebnim osvrtom na primenu u Flutter razvojnom okruženju (engl. Framework) u Dart programskom jeziku (engl. Dart Programming Language). Predočen je i razvoj i istorijat QR koda, njegova struktura a poseban osvrt je dat na primenu QR koda unutar QRTV Reminder aplikacije.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željen Trpovski, red. prof.

2. RAZVOJ I ISTORIJAT QR KODA

Prema tipu predstavljanja informacija razlikujemo nekoliko desetina standarda za generisanje linijskih (jednodimenzionalnih) i matričnih (dvodimenzionalnih) bar-kodova. Među njima se posebno ističe jedna vrsta matričnog koda pod nazivom QR (engl. Quick Response) kod za čiji početni razvoj zadužen je istraživački tim u Denso Wave Incorporated, kompaniji koja se u to vreme bavila razvojem bar-kod čitača, na čelu sa Masahiro Hara (Slika 2.1) koji je rad započeo sa još jednim kolegom.
Kompanija Denso Wave, ogranak Denso Corporation, objavila je 1994. godine i pustila u promet QR kod. Naziv QR kod je dobijen kao skraćenica od quick response (brzi odziv), što ističe glavnu osobinu ovog tipa bar-koda a to je brzo očitavanje podataka.

Slika 2.1. *Jedan od autora standarda QR koda, Masahiro Hara*

3. STRUKTURA QR KODA

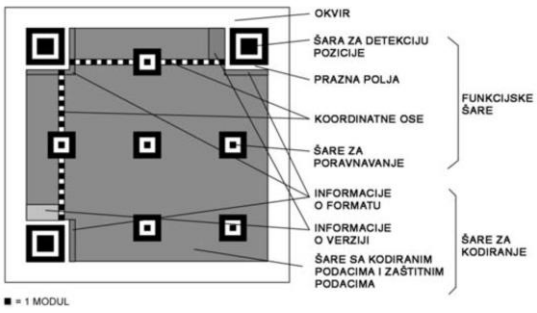
Detalji o osobinama QR koda u daljem tekstu dati su na osnovu ISO/ IEC18004 standarda [1].

Osnovni sastavni delovi QR koda su moduli, odnosno crni i beli kvadratići koji čine kvadratnu mrežu. Crni kvadratići predstavljaju logičku "1" a beli predstavljaju logičku "0". Po ISO/ IEC18004 standardu, minimalna veličina jednog modula je 4x4 tačke (eng. pixels) pri rezoluciji štampe od 300 dpi (engl. dot per inch).

Ako pogledamo strukturu QR koda prikazanog na slici 3.1. uočavamo dve različite grupe elemenata po funkciji koju vrše:

1. Funkcijske šare;
2. Šare za kodiranje.

Funkcijske šare su obavezni elementi QR koda. One su namenjene očitavanju koda i ne sadrže podatke.



Slika 3.1. *Struktura slike QR koda*

4. QRTV REMINDER APLIKACIJA UNUTAR FLUTTER RAZVOJNOG OKRUŽENJA

U današnje vreme mobilne aplikacije beleže ogromnu popularnost. Trenutno na tržištu mobilnih uređaja dominiraju dva operativna sistema a to su Android i iOS te ako težimo uspešnosti mobilne aplikacije ona mora biti dostupna na ova dva navedena operativna sistema. Zbog toga je za klasičan razvoj aplikacija za Android i iOS potrebno napraviti dve različite aplikacije koje rade istu stvar ali na specifičan način koji zavisi od operativnog sistema. To dovodi do značajnog povećavanja napora i troškova koji su neizostavni za razvoj jedne takve aplikacije.

Da bi se rešio problem da se pišu dva različita koda za istu namenu, tokom godina je razvijen veliki broj razvojnih okruženja (engl. Framework) koji pokušavaju da ublaže ili reše taj problem. Dakle, problem se rešava pomoću višeplatformskih radnih okruženja koja koriste jedan kod i za Android i za iOS. Jedan od modernijih i jako dobrih višeplatformskih mobilnih razvojnih okruženja jeste Flutter koji ima odlične alate za razvoj i razvojnu dokumentaciju. Takođe, Flutter utiče i na poboljšanje performansi aplikacija kao i na lepši izgled mobilnih aplikacija.

Slika 4.1. *Izgled Flutter logo-a*

Flutter [2] je Google-ovo višeplatformsko razvojno okruženje (engl. Framework) koje ima softver otvorenog koda (engl. Open Source Software) a to je softver čiji je izvorni kod dostupan svim korisnicima za menjanje, prepravljanje i poboljšavanje njegovog sadržaja. Ovaj Framework je objavljen u maju 2017. godine, a prva stabilna distribucija je objavljena u decembru 2018. godine. Iako je sam Flutter prvobitno namenjen za razvoj iOS i Android mobilnih aplikacija uporedo je obezbeđena podrška i za web i desktop platforme a ima indikacija i da će se na Flutteru temeljiti razvoj aplikacija za Google-ov novi nadolazeći operativni sistem Fuchsia [3].

Višeplatformsko razvojno okruženje Flutter sadrži niz alata koji pomažu programeru u velikoj meri i ubrzavaju razvoj aplikacija. Najznačajniji od tih alata je alat "vrućeg ponovnog učitavanja" (engl. hot reload). Ovaj alat daje mogućnost programeru da nakon promene programskog koda aplikacije, u vremenu koje je manje od jedne sekunde, ponovo učita aplikaciju i primeni svoje promene bez gubitka stanja aplikacije, u odnosu na neke druge radne okvire kod kojih to može trajati i nekoliko minuta. Jedna od glavnih stvari koja tako nešto čini mogućim je upotreba programskog jezika Dart [4] (engl. Dart programming language) u Flutter razvojnom okruženju.

Slika 4.2. Logo Dart programskog jezika

5. OPTIMIZACIJA APLIKACIJE QRTV REMINDER

5.1. Uvod u korišćenje aplikacije

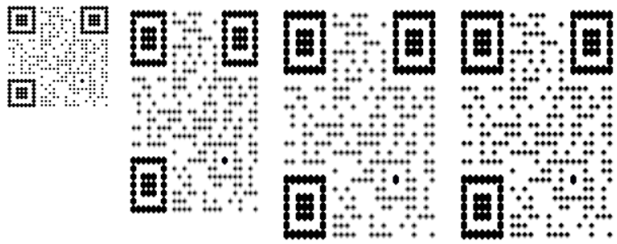
Kao primer primene QR kodova na Android i iOS platformi pomoću Flutter razvojnog okruženja razvijamo aplikaciju za skeniranje QR kodova koja najčešće sa TV ekrana (takođe mogu da se očitaju i drugi QR kodovi, recimo sa papira, zida, flajera, postera i td.) iščitava kod koji je povezan sa trenutnim programom (serija, film, reklama ili najava za neku emisiju), omogućavajući korisniku pregled informacija i određenu akciju, u ovom slučaju pravljenje unosa u kalendar koji će služiti kao podsetnik (engl. Reminder). Dakle, glavna namena naše aplikacije QRTV Reminder jeste kreiranje podsetnika tj. Reminder-a skeniranjem namenski generisanih QR kodova sa Internet sajta https://qrtvreminder.com gde pored toga, imamo i mogućnost slanja SMS poruka, pozivanja bilo kog broja ili otvaranja odabranog linka prostim skeniranjem QR koda sa navedenog sajta. Pored namenski generisanih QR kodova sa sajta https://qrtvreminder.com naša aplikacija "čita" i bilo koji QR kod koji sadrži link ka određenoj adresi.
Da bi korisničko iskustvo bilo što bolje, potrebno je prilagoditi QR kodove prikazivanju na TV ekranima. Vreme trajanja prikazivanja koda na ekranu treba da traje minimalno 30 sekundi. Sami QR kodovi ne bi trebalo da budu sitni, da bi sa normalne udaljenosti (npr. sa kauča) korisnik nesmetano mogao da očita kod, pogotovo ako je ekran televizora mali, a kamera telefona relativno niske rezolucije (ovo je sve ređi slučaj kod pametnih telefona današnje generacije). Ipak, ako se desi da očitavamo QR kod sa veoma velikog rastojanja ili ako je QR kod veoma sitan u odnosu na mesto sa koga ga učitavamo, možemo da upotrebimo takozvano "zumiranje" (engl. zooming) unutar aplikacije gde bez obzira na udaljenost od QR koda koji treba da se skenira, njegovo očitavanje biva uspešno obavljeno.
QR kod, takođe, ne bi trebalo da bude previše upadljiv, centralno pozicioniran i previše kontrastan na TV ekranima kako ne bi narušavao kontinuitet slike i ometao korisnike da prate ono što je za njih najbitnije na ekranu. Zbog toga se pojavila potreba za QR kodovima koji su providni ili koji su pravougaoni (vrši se prelazak iz kvadratnog u pravougaoni oblik). Treba istaći da su sami QR kodovi najčešće pozicionirani u neki od uglova ekrana, u zavisnosti od celokupnog dizajna programa.
Ideja procesa očitavanja koda jeste sledeća: pokrene se QR čitač ugrađen u QRTV Reminder aplikaciju, očita se kod. Na osnovu informacija dobijenih iz koda poziva se

online servis koji vraća podatke o programu, kao što su naziv, opis, vreme početka i trajanje. Ti podaci se prikazuju u aplikaciji i korisniku se daje opcija da sačuva podsetnik za taj televizijski sadržaj.
Neke od platformi na kojima se ovako nešto može realizovati su uređaji sa Android i iOS operativnim sistemima.

5.2. Izazovi u toku razvoja aplikacije

Pravougaoni i kvadratni QR kodovi

Prilikom razvoja QRTV Reminder aplikacije bilo je vidljivo različito reagovanje Flutter biblioteka na pravougaone QR kodove dok sa kvadratnim QR kodovima navedeni problemi nisu bili toliko prisutni. Slika 5.1. prikazuje navedene vrste QR kodova.

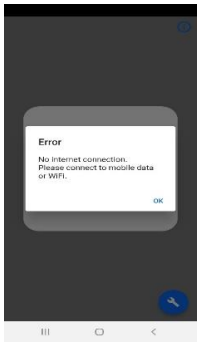


Slika 5.1. *Kvadratni QR kod (skroz levo) i pravougaoni QR kodovi (svi osim levog)*

Error handling (exceptions)

Aplikacija mora da se izbori sa određenim greškama ukoliko se one dese. Neke od njih je teško predvideti ali određene vrste i nije pa je taj problem "predvidivih" grešaka rešavan.

U QRTV Reminder aplikaciji predupređena je situacija u kojoj prilikom skeniranja QR koda dolazi do gubitka pristupa Internetu ili pristupa nema uopšte. Na slici 5.2. prikazan je izgled ekrana Android uređaja ukoliko dođe do navedenog slučaja tj. prikaz povratne poruke korisniku ukoliko dođe to "gubitka Interneta".



Slika 5.2. *Izgled ekrana aplikacije ukoliko ne postoji Internet konekcija*

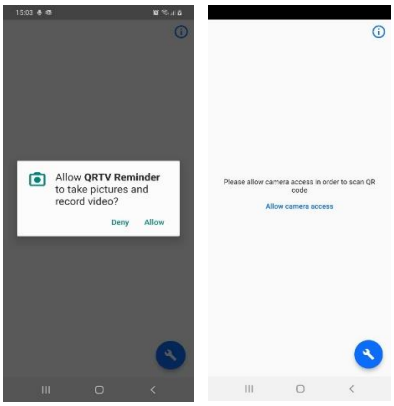
Permissions (Allow/Deny)

Permisije ili dozvole predstavljaju veoma bitan segment razvoja svake aplikacije pa samim tim i ove. Veoma je bitno obratiti pažnju na rešenja onih situacija u kojima sam korisnik aplikacije na neki način "ne sarađuje" sa samom aplikacijom koju koristi. Više je razloga takvog korisnikovog ponašanja a jedan od njih jeste i nedovoljno razumevanje tj. nemanje dovoljne informatičke pismenosti, zatim nedostatak strpljenja ili bilo koji drugi razlog.

Dakle, QRTV Reminder aplikacija zahteva upotrebu kamere telefona i ukoliko sam korisnik aplikacije, prilikom prve instalacije i pokretanja QRTV Reminder-a, ne želi da nam dozvoli korišćenje kamere sama aplikacija gubi suštinsku funkcionalnost a ta funkcionalnost jeste skeniranje QR kodova upotrebom kamere telefona. Ovde, iako je identičan kod koji je pisan u programskom jeziku Dart u Flutter razvojnom okruženju, postoje razlike kada taj kod koriste Android uređaji u odnosu na to kada kod koriste iOS uređaji.

Na Androidu korisnik nekoliko puta može da ne da dozvolu (engl. Deny) za korišćenje kamere QRTV Reminder aplikaciji i tek posle određenog perioda (višestrukih onemogućavanja pristupa kameri) pojavljuje se info dijalog koji korisnika obaveštava da mora da ode u podešavanja (engl. Settings) samog Android operativnog sistema i ovaj put sam dozvoli pristup kameri i na taj način omogući rad QRTV Reminder aplikacije.

Na iOS uređajima se odmah nakon prvog odbijanja pristupa kameri telefona mora ići u podešavanja iOS operativnog sistema kako bi se omogućio nesmetan rad QRTV Reminder aplikacije tj. upotreba kamere. Izgled info dijaloga kod Android telefona je prikazan na slici 5.3.



Slika 5.3. *Izgled info dijaloga za omogućavanje pristupa kameri*

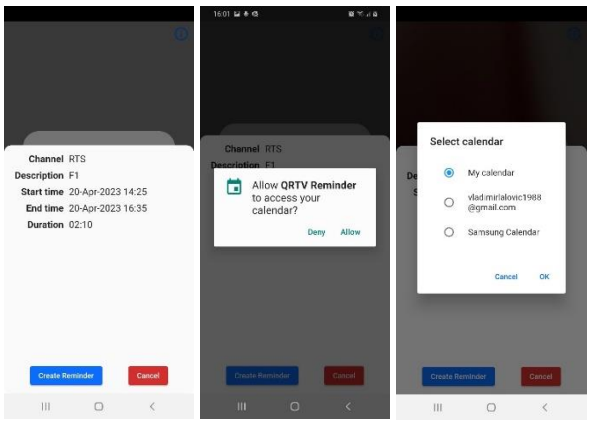
Calendar (kreiranje podsetnika unutar kalendara)

Prilikom dodavanja podsetnika (engl. Reminder) moramo imati kalendar (engl. Calendar) gde ćemo "smeštati" podsetnik. I na Android-u i na iOS-u pravljenje podsetnika u kalendaru je podešeno tako da, prilikom kreiranja događaja (engl. Event) koji treba da nas podseti na nešto, sam QRTV Reminder vrši pozadinsku proveru. Pošto svaki i Android i iOS uređaj u sebi ima određene kalendare (Android ih često ima više a iOS uglavnom samo jedan) QRTV Reminder vrši proveru broja postojećih kalendara u uređaju.

Koraci kreiranja podsetnika unutar same QRTV Reminder aplikacije su prikazani na slici 5.4. Sva tri izgleda ekrana telefona su takođe sa Android uređaja.

Različite biblioteke za skeniranje QR kodova

Flutter kao razvojno okruženje (engl. Framework) ima u pozadini takozvane "obmotavajuće" (engl. wrap) elemente koji se nazivaju "package", unutar kojih mogu da se nalaze, za našu aplikaciju neophodne, biblioteke za skeniranje QR kodova. Svi detalji u vezi sa korišćenim bibliotekama mogu se pronaći u master radu [5].



Slika 5.4. *Kreiranje Reminder-a na Android telefonu*

5.3. Izgled same QRTV Reminder aplikacije

Aplikacija se sastoji od takozvanog "prozora za skeniranje" (engl. Scanner Window), "dugmeta za prikaz informacija o aplikaciji" koje u sebi sadrži slovo i (engl. Info Button), i dugmeta koje ima više funkcionalnosti i koje ima izgled ključa koji je veoma često viđen na podešavanjima raznih opcija na raznim uređajima (engl. Settings Button). Sve slike u ovom poglavlju predstavljaju izgled ekrana sa Android uređaja.

Scanner Window

Prozor za skeniranje zauzima centralni dio ekrana i kao pojedinačni segment zauzima najveći dio površine ekrana kada se uđe u aplikaciju. Oivičen je, zaobljen na krajevima i jasno ukazuje na mesto gde treba da se "obuhvati" QR kod koji želimo skenirati. Izgled Scanner Window-a prikazan je na slici 5.5.

Info Button

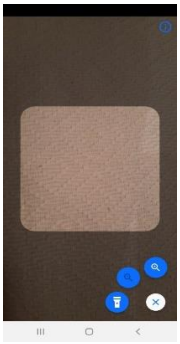
Dugme za prikaz informacija o aplikaciji zauzima desni gornji deo ekrana i ima izgled manjeg kružića unutar koga se nalazi malo latinično slovo i. Klikom na i dugme, sklanja se Scanner Window i sada centralni dio ekrana zauzima malo veći informacioni prozor u kome se nalaze informacije Website, Email, Telephone, Skype i Share QRTV Reminder opcija koja je veoma bitna za distribuciju informacija o aplikaciji putem društvenih platformi za dopisivanje kao što su Viber, WhatsApp i mnoge druge. Izgled prozora za prikaz informacija o aplikaciji dat je na slici 5.6.

Slika 5.5 *Scanner Window*

Slika 5.6. *Info Button*

Settings Button

Dugme za podešavanja mogućnosti nalazi se u donjem desnom uglu ekrana i ima izgled plavog kruga u kome se nalazi beli ključ. Pritiskom na njega otvara se lepeza sa četiri moguće aktivnosti.



Slika 5.7. *Aktiviran Settings Button*

6. ZAKLJUČAK

Upotreba Flutter-a kao multiplatformskog radnog okruženja i Dart-a kao programskog jezika donosi velika poboljšanja jer se piše jedan te isti kod koji se "spušta" i na Android i na iOS mobilne operativne sisteme. Postoje različita reagovanja Android i iOS uređaja na dostupne open-source biblioteke za skeniranje QR kodova unutar QRTV Reminder aplikacije.

7. LITERATURA

[1] https://www.qrcode.com/about/standards.html DENSO WAWE INCORPORATED (pristupljeno u decembru 2022.)
[2] https://flutter.dev/multi-platform (pristupljeno u decembru 2022.)
[3] https://fuchsia.dev/fuchsia-src/get-started/learn-fuchsia (pristupljeno u decembru 2022.)
[4] https://dart.dev/overview (pristupljeno u decembru 2022.)
[5] Vladimir Lalović, Optimizacija aplikacije QRTV Reminder za različite operativne sisteme, master rad, FTN, Novi Sad, 2022.

Kratka biografija:

Vladimir Lalović rođen je u Sarajevu 1988. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Komunikacione tehnologije i obrada signala odbranio je 2022.god. kontakt: vladimirlalovic1988@gmail.com

Dr Željen Trpovski, redovni profesor, rođen je u Rijeci 1957. godine. Doktorirao je na Fakultetu tehničkih nauka 1998. god. Oblast interesovanja su telekomunikacije i obrada signala.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/23BE10Petakovic

JEZIK ZA OPIS JEDINIČNIH TESTOVA BESERVERSKIH APLIKACIJA

DOMAIN-SPECIFIC LANGUAGE FOR DESCRIBING SERVERLESS UNIT TESTS

Aleksandar Petaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad predstavlja i opisuje jezik za specifikaciju testova za beserverske aplikacije. Opisana je i implementacija sistema koji interpretira dati jezik i generiše testove na drugim programskim jezicima na osnovu šablona. U sklopu sistema, razvijena je i nadogradnja za Visual Studio Code za naglašavanje sintakse prilikom opisa testova.*

Ključne reči: *DSL, beserverske tehnologije, jedinični testovi, textX, Visual Studio Code*

Abstract – *This paper presents and introduces a language for specifying unit tests for serverless applications. Also described is the implementation process of the system that interprets this language and generates tests for other programming languages based on templates. As a part of the system, a Visual Studio Code extension for syntax highlighting was also developed.*

Keywords: *DSL, serverless, unit tests, textX, Visual Studio Code*

1. UVOD

Jezici specifični za domen (skraćeno JSD, eng. *Domain-Specific Languages*) su počeli da se razvijaju kad i sami programski jezici, a razvoj jezika specifičnih za domen najčešće obuhvata proučavanje domena, konsultacije sa stručnjacima i velike količine planiranja i projektovanja. Sa druge strane spektra, u prethodnih deset godina razvijale su se tehnologije koje su danas nezaobilazne kada se priča o razvoju internet aplikacija i koje omogućavaju brz i lak pristup tim aplikacijama širom sveta, od strane miliona korisnika. Te tehnologije se mogu jednim imenom nazvati beserverskim (eng. *serverless*) tehnologijama.

Testiranje beserverskih aplikacija, kao i testiranje svih drugih tipova softvera, najčešće obuhvata pisanje programskog koda koji će iskoristiti delove softvera i proveriti da li su rezultati upotrebe tih delova zadovoljavajući. Obim testova često ume da bude približan, a nekada i da prevazilazi obim softvera koji se testira, pa se može zaključiti da je pisanje testova, odnosno samo testiranje softvera jedan zahtevan posao. U ovom radu biće predstavljeno rešenje čija je osnovna ideja ubrzanje i olakšanje pisanja testova za beserverske aplikacije.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Igor Dejanović, red. prof.

Sistem će se bazirati na pogodnostima i prednostima razvoja jezika specifičnog za domen, za rešavanje problema ovog uskog domena.

2. TEORIJSKA OSNOVA

Tri glavne oblasti na kojima se bazira čitav rad i koje će biti predstavljene u najviše detalja u nastavku jesu: jezici specifični za domen, beserverske aplikacije, i testiranje softvera.

2.1. Jezici specifični za domen

Programski jezik je veštački jezik čija je svrha upravljanje računarom. Programski jezici su, slično ljudskim jezicima, definisani kroz sintaksna i semantička pravila, kako bi se odredila njihova struktura i njihovo značenje [1]. Ako posmatramo tip problema kao oblast znanja, odnosno sferu interesa, primetićemo da se pojedini programski jezici mogu protumačiti i kao jezici prilagođeni tipu problema, odnosno jezici specifični za domen (problema).

Radi utemeljivanja osnovih koncepata, baziraćemo se na definiciji jezika specifičnih za domen koju je postavio Martin Fauler (eng. *Martin Fowler*) u svojoj knjizi o jezicima specifičnim za domen: „*Jezici specifični za domen su programski jezici ograničene ekspresivnosti, fokusirani na određeni domen*“ [2].

2.2. Beserverske aplikacije

Beserverske tehnologije predstavljaju vid razvoja softvera u oblaku koji omogućava razvojnim timovima da razvijaju i pokreću svoja softverska rešenja bez upravljanja serverima. Kao početak ere beserverskog razvoja softvera, najčešće se uzima početak upotrebe servisa Lambda razvijenog od strane američke kompanije Amazon, koji je pušten u javnost krajem 2014. godine [3]. Razvojni timovi su ubrzo osetili prednosti ovog pristupa koji im je omogućio da se fokusiraju na kod, odnosno srž svakog softvera, bez da se brinu i troše vreme na kupovinu i podešavanje hardvera, operativnog sistema i ostalog pratećeg softvera na kome će se kod izvršavati.

2.3. Testiranje softvera

Pri razvoju softvera, potrebno je utvrditi da li softver zadovoljava određene zahteve, a stepen usklađenosti sa zahtevima direktno određuje kvalitet softvera. Kao definiciju testiranja softvera, možemo iskoristiti sumu glavnih teza o testiranju softvera: Testiranje je proces koji se sastoji od svih aktivnosti u životnom ciklusu softvera vezanih za planiranje, pripremu i izvršavanje zadataka koji treba da pokažu da li softver zadovoljava specificirane zahteve i ispunjava namenu, kao i da ukažu na greške u softveru [4].

3. SPECIFIKACIJA SISTEMA

Sistem, i jezik na kome se bazira čitav sistem, nose naziv „ServerlessTestS“, ili skraćeno STS.

3.1 Motivacija i cilj rešenja

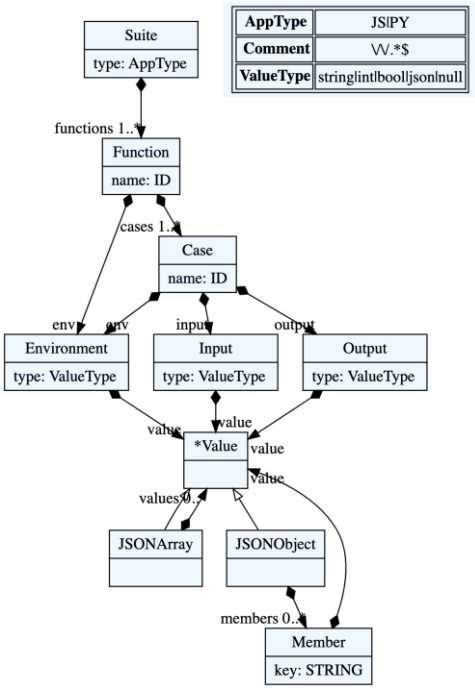
U svetu razvoja beserverskih aplikacija, kao i u ostalim sferama razvoja softvera, potrebno je testirati svako rešenje koje je namenjeno za upotrebu od strane drugih korisnika. Postoji više pristupa u testiranju beserverskih aplikacija, međutim, kada je u pitanju sama implementacija, ovi pristupi nisu zahvalni iz više razloga. Cilj ovog jezika, odnosno sistema, jeste da prevaziđe ove nedostatke i ponudi rešenje koje bi bilo lakše za savladati od ostalih programskih jezika. Zatim, koje bi nudilo dovoljno opcija za većinu slučajeva koji se testiraju, i koje ne bi zahtevalo pisanje velike količine šablonskog koda. Ovakav pristup je moguć uz korišćenje jezika specifičnog za domen kao alata za modelovanje testova, a zatim generisanje koda testova na određenom programskom jeziku.

3.2 Model sistema

Rešenje koje ovaj rad predstavlja, *ServerlessTestS*, sastoji se, pre svega iz istoimenog jezika specifičnog za domen. *ServerlessTestS* jezik razvijen je kao eksterni jezik specifičan za domen, sa ciljem da bude pregledan, ali i funkcionalan - i omogućava modelovanje testova za koje želimo da generišemo programski kod na nekom od podržanih jezika. Modelovanje se vrši upotrebom elemenata jezika uz specificiranje proizvoljnih podataka gde je to neophodno. Zatim, na osnovu gramatike ovog jezika, parser STS jezika parsira specifikaciju testova koju mu prosleđujemo. Parser vrši inicijalnu validaciju modela, a kao rezultat parsiranja dobijamo strukturu objekata. Ova struktura objekata je povoljan oblik podataka za dalju obradu i generisanje testova na osnovu šablona, a pre svega i validaciju.

3.3 ServerlessTestS jezik

Prikaz elemenata *ServerlessTestS* jezika i veze između istih, prikazan je na slici 1.



Slika 1: *Model ServerlessTestS jezika*

Kako su podržani i *JavaScript* i *Python* jezici prilikom generisanja programskog koda testova, teško je bilo napraviti paralelu između sintakse ova dva jezika. Međutim, beserverske aplikacije su najčešće veb-bazirane aplikacije, pa možemo pretpostaviti da se većina razvojnih timova koji se bave izradom veb- baziranih aplikacija susretala sa *markup* jezicima, naime XML-om, ili HTML-om. Stoga je STS sintaksa i gramatika razvijana tako da podseća na *markup* jezike.

4. IMPLEMENTACIJA

Svrha ovog poglavlja jeste dublje upoznavanje sa ServerlessTestS jezikom i sistemom, kao i predstavljanje procesa razvoja sistema.

4.1 Tehnologije

Alat upotrebljen za izradu i implementaciju STS jezika jeste *textX*, dok su za uspešno izvršavanje generisanih testova upotrebljeni elementi radnh okvira za testiranje, odnosno – *Jest* kada je u pitanju *JavaScript*, i *pytest*, kada je u pitanju *Python*. A za generisanje programskog koda upotrebljen je *jinja* obrađivač šablona.

TextX je meta-jezik, odnosno jezik za definisanje jezika, čija svrha je specifikacija jezika specifičnih za domen kroz *Python* programski jezik.

Jinja je obrađivač šablona (eng. *Template engine*) implementiran u *Python* programskom jeziku.

4.2 ServerlessTestS jezik

Suite je osnovni element *ServerlessTestS* jezika koji sadrži sve druge elemente. Jezik je zamišljen tako da jedna „sts“ datoteka predstavlja jednu instancu *suite* elementa, što bi u svetu testiranja softvera grubo odgovaralo nizu povezanih, odnosno sličnih testova.

Function element je jedan od glavnih elemenata *ServerlessTestS* jezika. Ideja ovog elementa jeste da sadrži sve testove vezane za jednu beserversku funkciju. Takođe može sadržati i element za definisanje promenljivih okruženja.

Environment element je zamišljen tako da se kroz njega mogu definisati sve neophodne promenljive okruženja koje su potrebne za izvršavanje testova.

Case element služi za definisanje test slučajeva vezanih za jednu funkciju. Slično *Function* elementu – i *Case* elementu je potrebno dodeliti ime kao identifikator elementa i to ime mora biti jedinstveno na nivou jednog *Function* elementa.

Input i *Output* elementi služe za navođenje ulaznih, odnosno izlaznih podataka prilikom modelovanja testova.

4.3 Interpretacija parsiranih modela

U ovom sistemu, pre generisanja koda, vrši se neophodna provera parsiranih modela, gde se proveravaju pretežno semantička ograničenja.

Za primenu ovih ograničenja, iskorišćena je funkcionalnost procesora koju nudi *textX* alat. Upotrebljene su dve vrste procesora, procesori modela, i procesori objekata.

4.4 Generisanje koda

Šablon za generisanje *JavaScript* testova je implementiran tako da koristi funkcionalnosti *Jest* radnog okvira za testiranje, dok šablon za generisanje *Python* testova koristi funkcionalnosti *pytest* radnog okvira za testiranje.

Glavni izazov u oba šablona bio je pravilno formatiranje JSON objekata. Za rešavanje ovog problema, upotrebljena je funkcionalnost *jinja*-e za formiranje makroa.

4.5 Komanda za generisanje testova

U cilju lakog pokretanja sistema, formirana je komanda pri čijem izvršavanju će biti izvršeno generisanje testova, a kojoj je moguće proslediti putanje do željenog ulaza i izlaza.

4.6 Nadogradnja za sintaksno naglašavanje

Kako bi korisnici imali što bolje iskustvo pri korišćenju STS jezika i formiranju modela testova. Kako je Visual Studio Code (skraćeno VSCode), razvojno okruženje i editor teksta razvijen od strane *Microsoft*-a - jedno od trenutno najpopularnijih razvojnih okruženja, u kome je i sam STS jezik razvijan – implementirana je nadogradnja upravo za ovaj editor.

5. KONFIGURISANJE I IZVRŠAVANJE SISTEMA

Ovo poglavlje biće posvećeno konfigurisanju i izvršavanju samog sistema i prikazu rezultata izvršavanja sistema kroz primer modelovanja testova za dve beserverske aplikacije.

5.1 Konfigurisanje sistema

Čitav sistem je napisan u *Python* programskom jeziku. A kako bi sistem bio prenosiv, i instalacija i konfiguracija tekla lakše na ostalim računarima, sistem je konfigurisan tako, da se može preneti i instalirati na željenom računaru uz *pip*, upravljač *Python* paketima. Nakon uvezivanja, odnosno instalacije uz pomoć *pip* upravljača paketima, funkcionalnosti sistema će biti dostupne kroz *textX* CLI (eng. *Command Line Interface* – interfejs komandne linije).

5.2 Konfigurisanje nadogradnje za VSCode

VSCode nudi šablon za kreiranje nadogradnji i taj šablon je iskorišćen za kreiranje ove nadogradnje. Nadogradnje za VSCode kreirane na ovaj način, koje takođe nisu objavljene na zvanični repozitorijum nadogradnji, mogu lokalno da se uvežu premeštanjem direktorijuma nadogradnje u direktorijum rezervisan za VSCode nadogradnje.

5.3 Generisanje i izvršavanje testova

U cilju da se što bolje predstavi prenosivost STS koda između dva jezika, *JavaScript* i *Python* beserverska aplikacija implementiraju iste funkcionalnosti. Funkcije specificirane u aplikacijama za koju će biti generisani testovi, prikazane su na slici 2.

Specificirana su dva test slučaja za testiranje funkcije *hello*. Prvi slučaj testira pozitivan ishod funkcije, gde je promenljiva okruženja definisana i gde su *header*-i zahteva ispravni. A u drugom slučaju očekujemo negativan ishod izvršavanja zbog prisustva simbolično nazvanog „lošeg“ (*Bad* u prevodu na engleski jezik) *header*-a.

Kada je u pitanju funkcija *activity*, u mogućnosti smo samo da testom potvrdimo da će izvršavanje funkcije kao rezultat vratiti HTTP odgovor sa status kodom 200.

```
functions:
  hello:
    handler: hello.handler
    events:
      - http:
          path: hello
          method: get

  activity:
    handler: activity.handler
    events:
      - http:
          path: activity
          method: get

  calculate:
    handler: calculate.handler
    events:
      - http:
          path: activity
          method: get

```

Slika 2: *Funkcije specificirane u serverless.yml*

Za funkciju *calculate* ponovo imamo specificirana dva slučaja. U prvom slučaju vršimo operaciju množenja nad brojevima dva i tri, a u drugom slučaju sabiramo ta dva broja. Kao rezultat izvršavanja prvog slučaja, očekujemo broj šest, dok kao rezultat izvršavanja drugog slučaja, očekujemo broj pet. Željene operacije prosleđujemo kroz promenljivu okruženja „OPERATION“. Specifikacija prvog slučaja prikazana je na slici 3.

```
< function = calculate
  < case = calculateCase1
    < env = json
      {
        "OPERATION": "multiply"
      }
    >
    < input = json
      {
        "a": 2,
        "b": 3
      }
    >
    < output = int
      6
    >
  >
```

Slika 3: *Funkcija calculate, prvi test slučaj*

Generisanje testova moguće je izvršiti pokretanjem komande „textx sts-generate” kroz interfejs komandne linije *textX*-a. Rezultat generisanja su tri datoteke koje nose nazive funkcija, uz nastavke specifične za okruženje, odnosno jezik za koje su generisani.

5.4 Pregled rezultata izvršavanja sistema

Otvaranjem bilo koje generisane datoteke, primetićemo običan *JavaScript*, odnosno *Python* kod, međutim, ono što možda nije očigledno na prvi pogled, jeste odnos broja karaktera specifikacije i samih testova. Ako saberemo broj karaktera u generisanim datotekama, i uporedimo ga sa brojem karaktera u STS specifikaciji, primetićemo brojeve navedene u tabeli 1.

Tabela 1: *Broj karaktera potreban za specifikaciju testova*

Jezik	JavaScript	Python	ServerlessTestS
Broj karaktera	1823	2054	1440

Broj karaktera potreban za specifikaciju testova na *JavaScript* jeziku je 27% veći, dok je broj karaktera potreban za specifikaciju testova na *Python* jeziku čak 43% veći.

5.5 Primena

Primarno se sistem može upotrebiti za specificiranje i generisanje testova tokom razvoja beserverske aplikacije.

Upotrebom ovog sistema, bilo bi moguće skladištiti samo specifikaciju testova na repozitorijumu, a generisanje testova bi moglo da se odvije tek kada je to neophodno da bi se izvršili testovi, odnosno, u CI (eng. *Continuous Integration*) okruženju.

Direktna posledica čuvanja samo specifikacije testova umesto svih datoteka koje se mogu generisati jeste i bolja, odnosno veća prenosivost koda između dva uređaja, ili uređaja i repozitorijuma.

5.6 Smernice za dalji razvoj sistema

Glavne smernice za unapređenje ovog sistema, bile bi unapređivanje nadogradnje za VSCode i dodavanje podrške za više editora, zatim podrška za sekcije ručno unesenog koda. Sledeće, podrška za više kriterijuma za prolaznost testova, takođe i integracija sa beserverskim radnim okvirom i podrška za više jezika.

6. ZAKLJUČAK

Priloženo rešenje opisano kroz ovaj rad nudi mogućnost efikasnijeg specificiranja testova za beserverske aplikacije. Ovo rešenje takođe demonstrira moć koju imaju jezici specifični za domen, na kojima se zasniva, i pokazuje da i dalje u nekim domenima postoji prilika za kreiranje jezika koji bi približio domen korisnicima. Takođe, u situacijama u kojima se pojavljuju linije koda koje se ponavljaju, treba uvideti i kreirati šablon, pre svega kako bi se izbegle greške prilikom kopiranja i lepljenja delova koda, ali i kako bi se moglo automatizovati popunjavanje šablona, kao što je slučaj u ovom sistemu. Ukupan rezultat ovih rešenja je efikasniji proces izrade novih softverskih rešenja i manje prostora za greške. A to je ujedno i budućnost izrade softvera kakvoj treba težiti.

7. LITERATURA

[1] Programming language - https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/p/Programming_language.htm (pristupljeno u julu 2022.)

[2] Martin Fowler, Rebecca Parsons, „Domain Specific Languages“, Addison-Wesley Professional, 2010.

[3] AWS Lambda Wikipedia - https://en.wikipedia.org/wiki/AWS_Lambda (pristupljeno u novembru 2022.)

[4] Andreas Spillner, Hans Schaefer, Tilo Linz, „Software Testing Foundations: A Study Guide for the Certified Tester Exam“, Rocky Nook, 2021.

Kratka biografija:

Aleksandar Petaković rođen je u Sremskoj Mitrovici 1997 god. Po završetku srednje škole, 2016. godine, upisuje smer računarstvo i automatika na Fakultetu Tehničkih Nauka u Novom Sadu. Diplomirao je 2020. godine, nakon čega je iste godine upisao master studije.

kontakt:
aleksandar.petakovic@outlook.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE11Maksimovic

РАЗВОЈ ДЕЦЕНТРАЛИЗОВАНОГ ИНФОРМАЦИОНОГ СИСТЕМА ЗА УПРАВЉАЊЕ ТРАНСАКЦИЈАМА НЕЗАМЕЊИВИХ ТОКЕНА

DEVELOPMENT OF A DECENTRALIZED INFORMATION SYSTEM FOR THE NFT TRANSACTION MANAGEMENT

Владислав Максимовић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду описан је информациони систем који омогућава размену дигиталних средстава у виду незамењивих токена. Као платформа за подршку овој апликацији је изабрана Ethereum блокчејн платформа. Приказано решење користи блокчејн као део свог система како би омогућио децентрализовану размену дигиталних средстава. У раду су ради бољег разумевања самог проблема и његовог решења приказани основи децентрализације, блокчејна као и основи технологија коришћених за израду информационог система.*

Кључне речи: *Базе података, дистрибуирани информациони системи, блокчејн, незамењиви токени*

Abstract – *This thesis describes an information system which implements the exchange of digital assets in the form of NFTs. The Ethereum blockchain has been chosen as the platform to support this application. The presented solution uses blockchain as part of its system to enable the decentralized exchange of digital assets. In order to better understand the problem itself and its solution, the thesis presents the basics of decentralization, blockchain and basics of technology used to create an information system.*

Keywords: *Databases, distributed information systems, blockchain, NFT*

1. УВОД

У овом раду описан је информациони систем са фокусом решавања проблема децентрализоване размене незамењивих токена (енгл. NFT). Другим речима, циљ платформе јесте то да се омогући децентрализовано место на коме корисници, односно власници незамењивих токена, могу да размењују своје токене са другим учесницима платформе кроз интерактивни интерфејс клијентске апликације самог информационог система.

Тема која је обрађена у овом раду покушава да реши проблем једноставне размене дигиталних средстава у виду незамењивих токена кроз једну платформу. Намера је да се уз што једноставније кораке приступи платформи и обави размена средстава.

Постоји неколико мотива за изградњу децентрализованог информационог система за размену незамењивих токена. Први од њих јесте безбедност која се огледа у томе да децентрализовани системи могу пружити већу сигурност од централизованих система јер се не ослањају на једну тачку квара за разлику од централизованих система. То је изразито битно за незамењиве токене који су често вредни и могу бити мета нападача.

Још један од мотива јесте и отпор цензури. Децентрализовани системи су отпорни на цензуру јер немају један ентитет који их контролише. То значи да се незамењиви токени не могу произвољно уклонити или не могу бити блокирани за трговину на платформи. Поред отпора цензури још један од мотива јесте и истинско власништво средстава. То се односи на то да децентрализовани системи омогућавају право власништво над незамењивим токенима јер се чувају у децентрализованој књизи коју не контролише ниједан ентитет. Другим речима, то значи да власнику токена не може бити одузето власништво над токенем без његовог првобитног пристанка на то.

Још неки од мотива за изградњу децентрализованог информационог система за размену незамењивих токена су и интероперабилност и транспарентност. Интероперабилност се односи на то да децентрализовани системи могу олакшати размену токена између различитих платформи и протокола, што олакшава куповину и продају токена на широком спектру платформи. С друге стране, транспарентност се односи на то да децентрализовани системи пружају транспарентност јер се све трансакције евидентирају у јавној књизи. На тај начин су омогућене транспарентност и одговорност на тржишту незамењивих токена.

Све у свему, децентрализовани информациони системи за размену незамењивих токена нуде бројне предности, укључујући повећану безбедност, отпорност на цензуру, право власништво, интероперабилност и транспарентност што представља довољан број мотива и разлога за изградњу једног таквог информационог система [1].

2. БЛОКЧЕЈН, ПАМЕТНИ УГОВОРИ И НЕЗАМЕЊИВИ ТОКЕНИ

Пре неколико година, термин блокчејна био је само израз који се користио у компјутерским наукама и односио се на чување и структурирање података. Данас, блокчејн се сматра револуцијом, не само у крипто индустрији или индустрији новца, већ и у будућности технологије, бизниса и света уопште.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Душан Гајић, ванр. проф.

Принципи блокчејна, иако делују компликовано, почивају на неким једноставним основама које није толико тешко разумети. Блокчејн је заправо врста базе података. Прва кључна разлика између типичне базе података и блокчејна је начин на који су подаци структурирани. Блокчејн не групише информације у табеле, већ у блокове. Дакле, блок се може посматрати као једна јединица базе података која садржи одређене информације.

Ако се узме у посматрање тренутно један од најпопуларнијих блокчејнова или ти Биткоинов блокчејн, може се приметити да је он база података која чува све трансакције Биткоина икада направљених и да му је то једина функција. Ултимативни циљ овог блокчејна јесте да свака трансакција буде верификована, исправна и забележена. Блок можете једноставно замислити као виртуелно парче папира које садржи информације о трансакцијама на мрежи. Један блок у ланцу Биткоина садржи информације о више од 500 трансакција. Информације се односе на то ко је послао биткоин, коме, колико, хеш блока (који се може сматрати јединственим идентификатором, отиском прста који је различит за сваки блок), као и хеш претходног блока. Баш зато што сваки блок садржи свој хеш, али и хеш претходног блока, сви блокови се надовезују један на други и креирају тако ланац или ти популарно блокчејн (енгл. Blockchain) [2] [3] [7].

Оно што гарантује безбедност мрежи је то што је блокчејн мрежа равноправних корисника (енгл. peer-to-peer) или популарно П2П (енгл. P2P). Блокчејн користи мрежу равноправних корисника како би вршио верификације на безбеднији начин. За разлику од сервера који су суштински гомила моћних компјутера у власништву једне компаније, на једном месту, П2П мрежа коју користи блокчејн састављена је од независних рачунара који се налазе широм света. У серверској архитектури, сви подаци се налазе на једном месту односно на серверу. Рачунари морају да приступе серверу како би приступили подацима. У П2П архитектури, сервер не постоји, већ су сви подаци дељени, дистрибуирани међу независним рачунарима који својим постојањем креирају мрежу.

Иако можда делује да је Виталик Бутерн као оснивач Етхереума оснивач и проналазач паметних уговора, то заправо није тако. Паметне уговоре и њихову дефиницију и појам је увео Ник Шабо још 1994. године. Тада је Ник утврдио да је паметни уговор (енгл. Smart contract) рачунарски програм или протокол трансакције који је намењен за аутоматско извршавање, контролу или документовање догађаја и радњи у складу са условима уговора или споразума. Циљеви паметних уговора су смањење потребе за посредницима од поверења, трошкова арбитраже и губитака од преваре, као и смањење злонамерних и случајних изузетака. Паметни уговори се обично повезују са криптовалутама, а паметни уговори које је увео Етхереум генерално се сматрају основним блоком за децентрализоване финансије (енгл. DeFi) и NFT апликације. Другим речима, паметни уговори су програми који се налазе у оквиру децентрализованих блокова и извршавају се у складу са покренутим упутствима. Паметни уговор делује на сличан начин као традиционални споразум, али негира потребу за учешћем треће стране. Паметни уговори су способни да аутоматски иницирају своје команде, чиме се елиминише учешће регулаторног тела. Као последица непроменљиве карактеристике блокчејна, паметни уговори се развијају на начин који се разликује од традиционалног софтвера. Када се једном примени на блокчејн, паметни уговор се не може модификовати или ажурирати за безбедносне пропусте, чиме се охрабрују програмери да примене јаке безбедносне стратегије пре примене како би избегли потенцијалну експлоатацију у каснијем тренутку.

Постоји неколико корака које програмери могу предузети како би осигурали безбедност и сигурност њихових паметних уговора. Прво, важно је пажљиво планирати и дизајнирати уговор пре него што се напише било који код. Ово укључује јасно прецизирање захтева и циљева уговора и идентификацију свих потенцијалних ризика или рањивости. Затим је од суштинског значаја темељно тестирање уговора пре објављивања на блокчејн. Ово може укључивати писање јединичних тестова како би се осигурало да појединачне компоненте уговора функционишу исправно, као и извођење ручног тестирања како би се проверило целокупно понашање уговора. Такође је важно користити безбедне праксе кодирања приликом писања уговора [4]. Ово може укључивати праћење утврђених најбољих пракси и коришћење сигурносних библиотека и оквира, као и избегавање уобичајених безбедносних замки као што су несигурно генерисање случајних бројева и непроверени унос.

У економији, замењивост се односи на својство робе и добара. На пример злато, оно је замењиво. Кило злата у златним полугама исто је вредно као и кило злата у златним новчићима истоветне чистоће. Десет долара је десет долара, без обзира да ли долази у облику једне новчанице или у четири кованице по два и по долара. Незамењивост се односи на чињеницу да је посреди податак који „живи” на блокчејну и јединствен је. Ова јединственост се изражава тиме што постоји „оригинал” и самим тим је погодан за бележење разних облика власништва, идентитета, права. Незамењиви токен је могуће купити или продати, „заменити” га (у значењу замене добара) за нешто друго. Али два незамењива токена нису међусобно замењива, као два биткоина, на пример. Ако вам неко да једну новчаницу од десет динара, а ви неком трећем дате једну новчаницу од десет динара, то је исто, осим ако нека од тих новчаница није поцепана или уништена. Али ако вам неко да токен који доказује да је тај неко власник тог токена, а ви њему дате токен који доказује да сте ви власник неког трећег токена, размена се догодила, али не и замена, јер свако одлази са својим јединственим токеном, на коме је паметни уговор. Незамењиви токен је ништа више од паметног уговора на ERC-721 токену [8].

3. КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ И АЛАТИ

Један од основа за изградњу једног оваквог информационог система је Хардхат који представља вредан алат за Етхереум програмере који желе да поједноставе свој развојни процес и осигурају квалитет својих паметних уговора [2] [5]. Он је пројекат отвореног кода и може се лако инсталирати и конфигурисати на било ком систему који подржава Node.JS.

Са својим уграђеним функцијама за тестирање и примену, модуларном архитектуром и снажном подршком заједнице, Хардхат је моћно развојно окружење за изградњу и примену Етхереум паметних уговора [6]. Да би се ти паметни уговорили написали на Етхереуму, коришћен је програмски језик Солидити. Солидити је програмски језик за писање паметних уговора на Етхереум мрежи. Он је објектно оријентисан језик високог нивоа са синтаксом сличном оној у Јава Скрипту. Солидити је дизајниран да буде лак за писање и читање, а истовремено је довољно изражајан и моћан да имплементира сложену логику и уговоре. Како би се све то приказало и како би се омогућила интеракција преко корисничког интерфејса, коришћен је оквир за развој клијентских апликација под називом Next.JS. Оно што је спојило паметни уговор са клијентском апликацијом јесте TheGraph који представља софтвер отвореног кода који се користи за прикупљање, обраду и складиштење података из различитих блокчејн апликација како би се олакшало проналажење информација [6]. Поред тога, један од алата преко ког се поједностављује коришћене апликације јесте и Metamask који представља бестплатну екстензију за веб претраживач и мобилну апликацију која омогућава корисницима да чувају и замењују криптовалуте, комуницирају са Етхереумом и хостују децентрализоване апликације.

4. РЕШЕЊЕ

Архитектура система је декомпонована на три модула. Сваки од модула има своју сврху и намену у читавом систему. Први модул је клијентска апликација изграђена на оквиру за развој клијентских апликација са називом Нехт (енгл. Next.JS). Корисник платформе користи клијентску апликацију кроз интерактивни интерфејс, након чега се клијентска апликација обраћа другом модулу, односно Граф (енгл. TheGraph) пројекту који је задужен да чува и складишти догађаје (енгл. Events) које трећи модул, односно Хардхат пројекат објављује, у оквиру њега је и изграђен паметни уговор система који се налази на блокчејну.

5. ЗАКЉУЧАК

Децентрализовани информациони систем за размену незамењивих токена описан у овом раду је развијен и успешно извршаван на Етхереум платформи. Имплементацијом овог система омогућена је безбеднија, отпорнија на цензуру, са правим правом на власништво размена незамењивих токена при чему је омогућена и интероперабилност, као и транспарентност кроз блокчејн. Потенцијална унапређења могу бити вођена у више смерова, први и конкретан би могао да буде повећање ефикасности потрошње гаса. Неопходно је анализирати један део система, односно конкретно паметни уговор и видети сваку функцију и измерити колико гаса троши свака од њих, те потом идентификовати проблематичне и онда пробати извршити оптимизацију потрошње. Наредно унапређење може бити оријентисано ка анализи безбедности и сигурности паметног уговора, где је потребно савладати принципе и оквире добрих пракси у Солидитију са којима сте имали прилику да се упознате у претходним поглављима, а након тога извршити детаљну проверу тренутног паметног уговора и покушати наћи рањивости и одмах након тога решења која спречавају те рањивости, чиме би се повећала безбедност и сигурност паметног уговора, а тако и самог информационог система. Наравно, једно од унапређења може бити и повећање броја функционалности које пружа сама платформа.

6. ЛИТЕРАТУРА

[1] Душан Гајић, *Материјали са предмета Паралелни и дистрибуирани алгоритми и структуре података*, доступно на: http://www.acs.uns.ac.rs/sr/node/237/4468699, последњи приступ јул 2022.

[2] *Ethereum Documentation*, доступно на https://ethereum.org/en/developers/docs/, последњи приступ децембар 2022.

[3] Виталик Бутерин, *The Meaning of Decentralization*, доступно на: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274, последњи приступ новембар 2022.

[4] *Solidity Patterns*, доступно на: https://fravoll.github.io/solidity-patterns/, последњи приступ децембар 2022.

[5] *Hardhat Documentation*, доступно на: https://hardhat.org/docs, последњи приступ септембар 2022.

[6] *TheGraph*, доступно на: https://thegraph.com/, последњи приступ новембар 2022.

[7] *Ecd*, доступно на: https://ecd.rs/, последњи приступ август 2022

[8] *Ethereum Improvement Proposals*, доступно на: https://eips.ethereum.org/EIPS/eip-721, последњи приступ јануар 2023

Кратка биографија:

Владислав Максимовић, рођен је 3. јула 1998. Факултет техничких наука у Новом Саду, студијски програм Рачунарство и аутоматика, уписао је 2017. год. Дипломирао је 2021. год., а потом уписао мастер академске студије из исте области.

контакт: maksimovic98vladislav@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/23BE12Kalocanj

ПЛАТФОРМА ЗА КУПОПРОДАЈУ НЕЗАМЕНЉИВИХ ТОКЕНА

PLATFORM FOR BUYING AND SELLING NON-FUNGIBLE TOKENS

Андреј Калочањ Мохачи, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Популарност blockchain технологија и незаменљивих токена, односно NFT, је током претходних неколико година знатно порасла. Услед тога су се појавили нови облици трговине и нова тржишта која се називају продавнице незаменљивих токена и омогућавају власницима да понуде своје токене под одређеним условима, а заинтересованим лицима да под тим условима купе токене. Примена паметних уговора омогућава кодификацију услова под којима се одвија купопродаја, а аутоматско извршавање тих уговора омогућава реализацију уговора односно уговорених одредби. У овом раду је представљено софтверско решење за купопродају незаменљивих токена под називом eArtRegister. Корисницима платформе дат је увид у одредбе паметних уговора на природном и програмском језику како би се боље информисали о правима и обавезама дефинисаним тим уговорима. У раду је приказана и евалуација ове платформе у односу на друга слична решења и дате су смернице за њено даље усавршавање.*

Кључне речи: *blockchain, NFT, паметни уговори*

Abstract – *The popularity of blockchain technologies and non-fungible tokens, NFTs, has increased significantly over the past few years. As a result, new forms of trade and new markets have emerged, which are called NFT marketplaces. These stores allow owners of NFTs to offer their tokens under certain conditions, and interested parties to buy tokens under those conditions. The application of smart contracts enables the codification of the conditions under which the purchase takes place, and the automatic execution of those contracts enables the realization of the contract, i.e. agreed provisions. This paper presents a software solution for buying and selling NFTs called eArtRegister. Users of this platform are able to preview smart contract provisions in natural language and programming language to better understand the rights and obligations defined by those contracts. The paper also shows the evaluation of this platform in relation to other similar solutions and provides guidelines for its further improvement.*

Keywords: *blockchain, NFT, smart contracts*

1. УВОД

Убрзани развој информационих технологија омогућио је да се помоћу сложених криптографских механизама створе услови за безбедно чување и размену података у небезбедном окружењу какав је Интернет. На тај начин је настао blockchain односно ланац блокова са подацима који истовремено пружају анонимност учесницима у трансакцијама, обезбеђују аутентичност и транспарентност података у тим трансакцијама.

Трансакцијама је најпре подржан промет новчаних средстава у такозваним криптовалутама, а на неким од blockchain платформи је убрзо затим подржано и коришћење паметних уговора чиме је омогућено да предмет трансакција буде извршавање програмског кода. Паметним уговорима је тако омогућено аутоматизовано управљање подацима или прометом новчаних средстава у складу са трансакцијама учесника који врше интеракцију са тим уговором.

Када се паметним уговорима представљају одредбе уговора у смислу облигационих односа, на тај начин се обезбеђује аутоматска примена и извршавање права и обавеза уговорних страна које из тог уговора произилазе. То је од посебног значаја за примену паметних уговора у купопродаји.

Помоћу ових уговора је могуће управљати подацима о власништву над такозваним токенима који могу бити заменљиви или незаменљиви [1]. Пример заменљивих токена су криптовалуте с обзиром на то је неку количину ових токена могуће заменити једнаком количином токена исте врсте. Незаменљиви токени представљају ствари које су јединствене и самим тим нису међусобно заменљиве.

У овом раду је приказано коришћење blockchain технологија за подршку купопродаји незаменљивих токена. Овим софтверским решењем се омогућава трговина дигиталним сликама, уз могућност увида у садржину паметних уговора у форми природног језика и програмског кода како би се олакшало разумевање права и обавеза купца и продавца.

Остатак овог рада је организован на следећи начин: у наредном одељку су анализирана сродна решења и технологије за купопродају незаменљивих токена, трећи одељак приказује метод израде платформе за купопродају незаменљивих токена eArtRegister, у четвртом одељку је представљен прототип платформе, пети одељак евалуира ефикасност ове платформе, а у шестом одељку су изнети закључци и дате су смернице за даље усавршавање прототипа.

НАПОМЕНА:

Овај рад је проистекао из мастер рада чији ментор је био др Марко Марковић, доцент.

2. СРОДНА ИСТРАЖИВАЊА

У овом одељку су анализирана сродна софтверска решења и технологије које омогућавају трговину незаменљивим токенима

OpenSea [2] је једна од највећих платформи за трговину незаменљивим токенима креираних по ERC-721 стандарду [3]. Платформа подржава више криптовалута од којих је Ethereum [4] подразумевана криптовалута. Безбедност трансакција на платформи је такође на високом нивоу, јер се при трговини власништво над токенима не преноси све док се продаја не реализује. Платформа задржава провизију у износу од 2,5% од сваке продаје.

Rarible [5] је такође једна од најпопуларнијих платформи за трговину незаменљивим токенима. За разлику од OpenSea платформе, Rarible подржава уплате у државним валутама, односно ако корисник поседује платну картицу омогућена му је куповина на овој платформи. Rarible је основао фондацију RARI [6] и креирао истоимену криптовалуту и дозволио њеним власницима право гласа у вези са будућим изменама на платформи. Платформа је испратила стандард за провизију према стваоцу NFT EIP-2981 [7].

Binance [8] је једна од највећих крипто-мењачница као и једна од највећих светских централизованих NFT тржишта, где је могуће прегледати понуду и трговати широким спектром артикала у играма, виртуелним земљиштем, уметничким делима и другим категоријама.

Crypto.com [9] је продавница незаменљивих токена и у потпуности је заснована на Crypto.org мрежи [10] која је потпуно децентрализована јавна blockchain [11] мрежа. Ова blockchain мрежа је дизајнирана да буде јавно добро које помаже у масовном усвајању blockchain технологије кроз разне случајеве коришћења, међу којима је и управљање незаменљивим токенима.

За идентификацију корисника и потписивање трансакција на blockchain мрежи користе се такозвани дигитални новчаници. Дигитални новчаник MetaMask [12] поседује релативно једноставан кориснички интерфејс и може се користити као додатак у веб прегледачима. MetaMask подржава Ethereum трансакције, док трансакције на другим платформама као што је Bitcoin [13] нису подржане.

За интеракцију *front-end* технологија са blockchain мрежом користе се посебне библиотеке које укључују и подршку за повезивање са дигиталним новчаницима. Тако се при изради корисничке апликације развијене у Angular радном оквиру [14] интеракција са MetaMask корисничким дигиталним новчаником може постићи употребом библиотеке web3.js [15] која уједно обезбеђује и комуникацију са blockchain мрежом.

Подршка за компајлирање паметних уговора за Node.js [16] окружење је доступна у оквиру пакета solc [17]. Овај сервис генерише bytecode и application binary interface (ABI) [18] на основу изворног кода паметног уговора. Поменути bytecode представља извршни кôд за Ethereum виртуелну машину EVM [19], а ABI је интерфејс преко којег је могућа интеракција са паметним уговором.

IPFS протокол [20] омогућава дистрибуирано складиштење датотека и пратећих метаподатака. С обзиром на то да blockchain мреже нису погодне за чување већих количина података, IPFS решава овај проблем на тај начин што се незаменљиви токени само референцирају на идентификаторе додељене датотекама при њиховом постављању на IPFS.

Програмски језик Solidity [21] омогућава имплементацију паметних уговора. Постављањем ових уговора на blockchain мрежу се омогућава аутоматско извршавање програмског кода који је у њима дефинисан. Solidity језик поседује доста сличности са савременим програмским језицима и представља најчешћи избор при изради паметних уговора за Ethereum платформу.

За потребе тестирања паметних уговора од великог значаја су тестне платформе на којима је могуће извршавати паметне уговоре без потребе за плаћањем провизија и без ризика од евентуалних финансијских губитака у случају грешака у програмском коду. Goerli Ethereum Testnet [22] је једна од највећих и најчешће коришћених тестних мрежа за Ethereum.

Паметни уговори су рачунарски програми који су смештени на blockchain мрежу и извршавају се на чворовима те мреже. Методе паметних уговора је могуће позивати путем трансакција чиме се врше промене стања уговора уз евентуалне новчане трансакције. Познавање интерфејса паметног уговора (ABI) је неопходан предуслов за позивање метода тог уговора [23]. Како су подаци на blockchain мрежи јавно доступни, тако су и информације о трансакцијама које су извршене над паметним уговорима транспарентне.

Стандард ERC-721 дефинише методе помоћу којих се обезбеђује управљање незаменљивим токенима. Овај стандард је заснован на оригиналном стандарду ERC-20 [24] који служи за креирање заменљивих токена, односно криптовалута.

3. МЕТОД

У овом одељку је описана имплементација платформе за купопродају незаменљивих токена. Наведени су функционални захтеви платформе, објашњена је структура паметних уговора које ова платформа користи и представљен је дизајн софтверског решења.

У систему постоје два типа учесника, нерегистровани корисник који има могућност претраживања и прегледања свих колекција незаменљивих токена, док регистровани корисници имају могућност креирања депозит уговора, уплате и исплате средстава са депозит уговора, креирања колекције токена, креирање токена за дигиталне слике, стављање токена на продају уз креирање купопродајног уговора (за продају у пуном износу, на рате или путем аукције) и куповине незаменљивих токена.

Депозит уговором се омогућава корисницима уплата средстава којима ће располагати на платформи eArtRegister и пријем средстава од продаје токена. Уплате на депозит уговор и повлачење средстава са овог уговора се врше уз помоћ MetaMask дигиталног новчаника.

Управљање незаменљивим токенима је реализовано по ERC-721 стандарду. Паметан уговор који обезбеђује управљање токенима имплементира интерфејс ERC721 који је дефинисан овим стандардом.

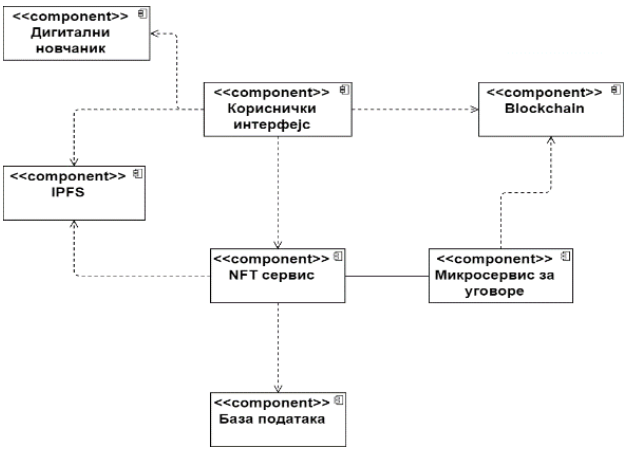
Овај стандард омогућава продавцу да купопродајном уговору додели права над токеном како би се у потпуности аутоматизовао пренос власништва у складу са уговореним одредбама.

На слици 1 приказан је дијаграм компоненти платформе eArtRegister. Компонента „Дигитални новчаник“ репрезентује кориснички дигитални новчаник који је неопходан да би корисник могао куповати, креирати или продавати незаменљиве токене путем ове платформе. „Кориснички интерфејс“ је *front-end* апликација преко које корисник врши интеракцију са платформом.

Компонента „NFT сервис“ обрађује захтеве са *front-end* апликације, стара се о перзистенцији података и комуницира са микросервисом за паметне уговоре. „IPFS“ је компонента за складиштење дигиталних слика, односно графичких датотека и пратећих датотека са метаподацима.

Компонента „Микросервис за уговоре“ генерише паметне уговоре и поставља их на blockchain. „База података“ представља складиште за податке о корисничким колекцијама токена и самим токенима који су предмет трговине на платформи.

Компонента „Blockchain“ репрезентује blockchain мрежу на којој се налазе креирани уговори и на којој се извршавају неопходне трансакције.

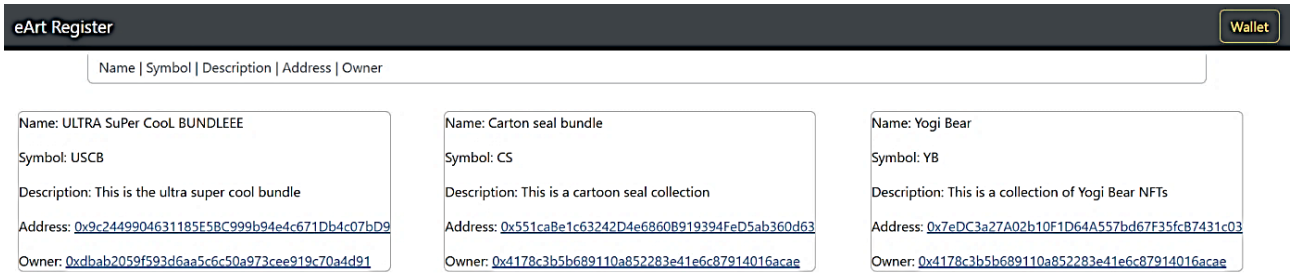


Слика 1. *Дијаграм компоненти*

4. РЕЗУЛТАТ

Према описаном методу развијено је прототипско решење за купопродају незаменљивих токена, односно платформа eArtRegister. Ово решење је имплементирано као веб апликација у којој је корисницима омогућено креирање сопствених колекција незаменљивих токена, стављање на продају и куповина токена. Депозит уговор и купопродајни уговор, који се користе у раду платформе, су доступни у текстуалном формату и у облику програмског кода, а њихова садржина се динамички генерише према одабраним условима купопродаје.

На слици 2 приказан је кориснички интерфејс платформе eArtRegister.



Слика 2. *Изглед корисничког интерфејса платформе eArtRegister*

5. ЕВАЛУАЦИЈА

У овом одељку је приказана евалуација платформе за купопродају незаменљивих токена. Анализирана је потрошња такозваног горива (енг. gas) потребног за извршавање трансакција у којима се власништво над незаменљивим токеном преноси са једног власника на другог. Количина утрошеног горива одговара ангажованим рачунарским ресурсима, па зато она представља значајан индикатор сложености и ефикасности имплементације паметног уговора. Поред тога, потрошња горива при извршавању трансакција директно утиче на накнаду која се обрачунава за иницирање ових трансакција.

Висина накнаде се израчунава као производ количине утрошеног горива и цене горива, при чему избор цене горива утиче на брзину извршавања трансакције [25].

Са сваке од анализираних платформи је прикупљен узорак од по пет трансакција у којима је извршен пренос власништва над незаменљивим токенима и упоређена је потрошња горива у тим трансакцијама.

Табела 1 даје преглед потрошње горива при преносу власништва над незаменљивим токенима за платформу eArtRegister и друге сродне платформе.

Табела 1. *Потрошња горива по платформама*

Платформа	Утрошено гориво		
	најнижа вредност	средња вредност	највиша вредност
eArtRegister	96443	108315,0	114852
OpenSea	189150	213172,0	617330
Rarible	108545	110174,6	110619
Binance	1617326	2841146,0	4280228
Crypto.com	120959	225572,0	360680

Из приказаних података се може закључити да је најнижа просечна потрошња горива на платформи eArtRegister и износи 108315 гаса. Највиша потрошња горива је забележена на Binance платформи и њена просечна вредност износи 2841146 гаса.

Без познавања тачне имплементације паметних уговора над којима су извршене ове трансакције није могуће дати прецизно тумачење ових резултата. С обзиром на то да потрошња горива зависи од комплексности операција које је потребно извршити овим трансакцијама, објашњење се може тражити како у сложености имплементације паметних уговора, тако и у оптимизованости њиховог програмског кода.

Платформа eArtRegister представља прототипску апликацију, те се може претпоставити да друге анализиране платформе, које постоје и развијају се дужи временски период, у својој имплементацији поседују захтевније операције како би се одговорило на изазове које поставља тржиште незаменљивих токена.

6. ЗАКЉУЧАК

У овом раду је приказано софтверско решење за купопродају незаменљивих токена. Објашњена је имплементација ове платформе и представљена је прототипска апликација eArtRegister.

Предност eArtRegister платформе је томе што она пружа корисницима могућност увида у одредбе паметних уговора на природном језику и на програмском језику, како би им се олакшало разумевање њихових права и обавеза.

Рад платформе је евалуиран уз поређење са другим сродним платформама у смислу потрошње горива. Платформа eArtRegister је показала најнижу просечну потрошњу горива у односу на остале платформе.

Даљим развојем ове платформе би требало обухватити подршку за друге типове дигиталних новчаника и друге blockchain платформе. Такође, корисницима би се могло омогућити да на платформи користе више од једног дигиталног новчаника.

7. ЛИТЕРАТУРА

[1] Knowledgehut, "NFT vs Cryptocurrency [Head-to-head Comparison]" https://www.knowledgehut.com/blog/blockchain/nft-vs-crypto (приступљено у фебруару 2023.)

[2] OpenSea, https://opensea.io/ (приступљено у фебруару 2023.)

[3] Ethereum, "ERC-721 Non-Fungible Token Standard" https://ethereum.org/en/developers/docs/standards/tokens/erc-721/ (приступљено у фебруару 2023.)

[4] Ethereum, "What is ether (ETH)?" https://ethereum.org/en/eth/ (приступљено у фебруару 2023.)

[5] Rarible, https://rarible.com/ (приступљено у фебруару 2023.)

[6] RARI Foundation, https://rari.foundation/ (приступљено у фебруару 2023.)

[7] Medium, "Create NFT with Royalty (EIP-2981)" https://medium.com/@nufailismath15/create-nft-with-royalty-eip-2981-bf201105ab96 (приступљено у фебруару 2023.)

[8] Binance NFT, https://www.binance.com/en/nft/home/ (приступљено у фебруару 2023.)

[9] Crypto.com NFT, https://crypto.com/nft/ (приступљено у фебруару 2023.)

[10] Cryto.org, https://crypto.org/ (приступљено у фебруару 2023.)

[11] Investopedia, "Blockchain Facts: What Is It, How It Works, and How It Can Be Used", https://www.investopedia.com/terms/b/blockchain.asp (приступљено у фебруару 2023.)

[12] Metamask, https://metamask.io/ (приступљено у фебруару 2023.)

[13] Bitcoint, https://bitcoin.org/en/ (приступљено у фебруару 2023.)

[14] Angular, https://angular.io/ (приступљено у фебруару 2023.)

[15] Web3.js, https://web3js.readthedocs.io/en/v1.8.1/ (приступљено у фебруару 2023.)

[16] Nodejs, https://nodejs.org/en/ (приступљено у фебруару 2023.)

[17] https://www.npmjs.com/package/solc-js (приступљено у фебруару 2023.)

[18] Chainlink, "What Are ABI and Bytecode in Solidity?", https://blog.chain.link/what-are-abi-and-bytecode-in-solidity/ (приступљено у фебруару 2023.)

[19] Ethereum, "Ethereum Virtual Machine (EVM)", https://ethereum.org/en/developers/docs/evm/ (приступљено у фебруару 2023.)

[20] IPFS, https://ipfs.tech/ (приступљено у фебруару 2023.)

[21] Solidity, https://docs.soliditylang.org/en/v0.8.17/ (приступљено у фебруару 2023.)

[22] Goerli Testnet, https://goerli.net/ (приступљено у фебруару 2023.)

[23] QuickNode, "How to call another smart contract from your solidity code", https://www.quicknode.com/guides/smart-contract-development/how-to-call-another-smart-contract-from-your-solidity-code (приступљено у фебруару 2023.)

[24] Investopedia, "What Are ERC-20 Tokens on the Ethereum Network?", https://www.investopedia.com/news/what-erc20-and-what-does-it-mean-ethereum/ (приступљено у фебруару 2023.)

[25] Binance academy, "What Are Blockchain Transaction Fees", https://academy.binance.com/en/articles/what-are-blockchain-transaction-fees (приступљено у фебруару 2023.)

Кратка биографија:

Андреј Калочањ Мохачи је рођен 14.04.1997. у Кикинди где је стекао своје основно и средње образовање. Школске 2016/17 уписује основне академске студије на Факултету техничких наука, студијски програм Примењено софтверско инжењерство. На мастер академске студије Факултета техничких наука се уписује школске 2021/22. године на студијском програму Рачунарство и аутоматика.

контакт: andrej.km997@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38:620.9
DOI: https://doi.org/10.24867/23BE13Vracaric

IEEE TEST MREŽE I SAVREMENE METODE REAL-TIME SIMULACIJE

IEEE TEST GRID AND METHODS OF REAL-TIME SIMULATION

Dejan Vračarić, Aleksandar Stanisavljević, *Fakultet tehničkih nauka, Novi Sad, Srbija*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su razmatrani multi-time analize elektroenergetskih sistema, osvćući se na dugoročno planiranje, kratoročno (operativno) planiranje, operativno stanje i kontrolu, kao i na klasifikaciju elektroenergetskih sistema. Zatim savremene test mreže, njihove mane i prednosti. Naredni segment savremene metode real-time simulacije. Poslednji deo čini detaljan opis IEEE 34-bus test mreže, kao i izrada simulacionog modela u programskom okruženjeu Matlab (Simulink) navedene test mreže. Ovaj deo čini i na testiranje simulacionog modela i potvrda da verno reprezentuje stvarnu mrežu.*

Ključne reči: *Real time simulacije, IEEE test mreže, IEEE 34-bus, Modelovanje i proračun*

Abstract – *The paper discusses multi-time analyzes of power systems, focusing on long-term planning, short-term (operational) planning, operational status and control, as well as the classification of power systems. Then modern test networks, their disadvantages and advantages. The next segment of the modern method of real-time simulation. The last part is a detailed description of the IEEE 34 test network, as well as the creation of a simulation model in the Matlab (Simulink) programming environment of the said test network. This part also involves testing the simulation model and confirming that it faithfully represents the real network.*

Keywords: *Real time simulations, IEEE test grids, IEEE 34-bus, Modeling and calculation*

1. UVOD

Pouzdan dizajn, planiranje i rad elektroenergetskih sistema su od od krucijalne važnosti za obezbeđivanje pouzdane usluge prema kupcima. Ovaj rad razmatra različite aspekte pouzdanosti elektroenergetskog sistema, raspon od planiranja do rada. Prikazani pregled pruža prednosti i nedostatke postojećih sistema za testiranje, uključujući obnovljive resurse i savremene tehnologije. Štaviše, daju se i zahtevi za unapređenje i izmenu merila za analizu savremenih elektroenergetskih sistema.

Savremene tehnologije, posebno, obnovljivi (promenljivi) izvori energije i distribuirani izvori, kao što su fotonaponski (PV) i vetroelektrane, e-mobilnost, a u najskorije vreme i distribuirana skladišta zasnovana na jednosmernoj (DC) struji i naponu, zajedno sa konceptom pametne mreže, menjaju elektroenergetske sisteme (EES) širom sveta. Oni postaju distribuirani u velikoj meri, kako u fizičkom, tako i u sajber sloju. U fizičkom sloju, energetska elektronika igra značajnu ulogu u konverziji energije za proces proizvodnje, prenos, distribuciju i nivoe potrošnje. Promena paradigme u načinu organizacije EES-a, od vertikalnog (odozgo prema dole) na distribuirani način, naglašava važnost komunikacionih sistema u planiranju i radu savremenih EES-a. Nasuprot tome, električne mreže su jedna od najkritičnijih infrastruktura i jedan od najsloženijih sistema, gde bilo kakva nezgoda ili ispad može da dovede do nena-doknadivih socijalno-ekonomskih poseledica. Napori modernizacije i liberalizacije nastoje da poboljšaju efikasnost i performanse, što je dovodi do daljeg usložnjavanja ovog sistema, ali i ranjivosti na pitanjama pouzdanosti, bezbednosti i sajber bezbednosti.

U radu će biti pregledani sistemi za testiranje za ocenu pouzdanosti, koji su predstavljeni u više od 240 radova koji su objavljeni u naučnim časopisima. Taj pregled je fokusiran pre svega na glavne koncepte pouzdanosti, odnosno adekvatnosti i bezbednosti. Međutim, na pouzdanost elektroenergetskog sistema može uticati širok opseg pojava, koje mogu ugroziti njegovu pouzdanost i prebaciti ga u nesigurno područje.

2. MULTI-TIME ANALIZA ELEKTROENERGETSKIH SISTEMA

Planiranje i rad elektroenergetskog sistema su procesi idealnog, ekonomičnog dizajna, proširenja, praćenja, upravljanja, zaštite i kontrole električne mreže koje snabdevaju krajnje korisnike sa željenim nivoom pouzdanosti. Zahtevaju različite studije u različitim vremenskim skalama u rasponu od mikrosekundi do nekoliko godina. Analaza elektroenergetskih sistema se generalno može proučavati u tri glavna vremenska okvira, uključujući dugoročno planiranje objekata (tzv. planiranje proširenja), kratkoročno operativno planiranje i rad u realnom vremenu.

2.1. Dugoročno planiranje

Planiranje objekta uzima u obzir opterećenje i tehnološki rast za proširenje i izgradnju novih objekata u narednih 5 do 30 godina. Glavni cilj dugoročnog planiranja je da se obezbedi ekonomično proširenje elektroenergetskih sistema za obezbeđivanje adekvatne i bezbedne isporuke energije [1].

2.2. Kratkoročno planiranje (Operativno planiranje)

Operativno planiranje podrazumeva marketing i održavanje u vremenskom okviru od nekoliko minuta do jedne godine. Upravljanje održavanjem objekata kako bi se osigurala pouzdana isporuka električne energije je od najveće važnosti za planiranje elektroenergetskog sistema. Teoretski, održavanje može biti preventivno ili korek-

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Stanisavljević, docent.

tivno. Preventivno održavanje se vrši periodično kako bi se smanjila verovatnoća kvara, a korektivno održavanje se sprovodi nakon što dođe do kvara. Kao rezultat toga, upravljanje održavanjem može značajno uticati na dostupnost sistema i operativne troškove [1].

2.3. Operativno stanje i kontrola

Tokom faze operativnog planiranja, ISO je zadužen za održavanje, marketing, kontrlu i optimalno planiranje proizvodnih jedinica. Sledeća faza je rad u realnom vremenu, kada je predviđeno da generatori proizvedu svoju unapred definisanu izlaznu snagu i primarni rezervni provajderi obračunavaju im bilanse. Sekundarne i tercijalne rezerve dobavljači onda kompenzuju ove neravnoteže izazvane neizvesnošću prognoza opterećenja, prognoze obnovljive proizvodnje i nenamerni prekidi (ovi termini su povezani sa adekvatnošću sistema). U među-vremenu, tokom rada u realnom vremenu, operateri sistema su zabrinuti za sigurnost sistema; naročito, moraju znati koliko je sistem siguran u svom sadašnjem stanju i koliko je bezbedan u narednih nekoliko minuta. Dakle, bezbednost sistema postaje najvažnija u okviru pogona sistema.

3. SAVREMENE METODE SIMULACIJE U REALNOM VREMENU

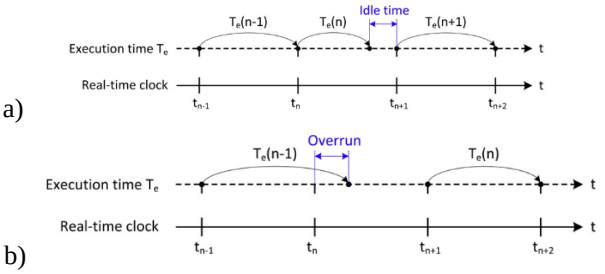
Digitalna simulacija u realnom vremenu (eng. DRTS) električnih elektroenergetski sistema je reprodukcija izlaza talasnih oblika (napona/struje) sa željenom tačnošću, koji su reprezentativni sa stvarnim ponašanjem u sistemu koji se modeluje. Da bi se postigao takav cilj, digitalni simulator u realnom vremenu treba da reši jednačine modela za jedan vremenski korak unutar istog vremena, kao i u realnom sistemu. Stoga daje izlaze u diskretnim vremenskim intervalima, gde se stanja sistema računaju u određenim diskretnim vremenima koristeći fiksni vremenski korak.

DRTS je tehnika za prolaznu simulaciju elektroenergetskih sistema koji koriste digitalno-računarski vremenski domen rešenja. Sistemi su predstavljeni korišćenjem komponenti koje su dostupne u biblioteci softverskih alata, koji koristi grafički interfejs i simulira na hardverskoj platformi koja koristi paralelno računanje.

U zavisnosti od potrebnog vremena mogu se pojaviti dve situacije pomoću platforme za simulaciju da se izvrši izračunavanje stanja izlaza za svaki vremenski korak (slika 1.a), ako izvršenje vremena Te za simulaciju sistema je kraće ili jednako sa vremenom izabranog vremenskog koraka, smatra se da je simulacija u realnom vremenu (slika 1.b). Ako je vreme Te veće od svoje veličine vremenskog koraka za jedan ili više vremenskih koraka, dolazi do prekoračenja i simulacija se smatra da nije u realnom. U poslednjem slučaju, može se povećati vremenski korak ili se model sistema može pojednostaviti za pokretanje u realnom vremenu [2].

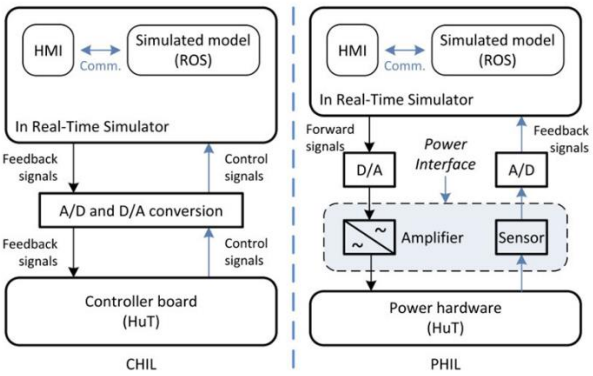
3.1. Kategorije digitalne simulacije u realnom vremenu

DRTS primenjen u domenu elektroenergetskih sistema može se klasifikovati u dve kategorije: 1) potpuno digitalna simulacija u realnom vremenu (npr. model-u-petlji, softver-u-petlji ili procesor u-petlji) i 2) hardver-u-petlji (HIL) simulacija u realnom vremenu. Za prvi tip potrebna je potpuno digitalna simulacija u realnom vremenu celog sistema (uključujući kontrolu, zaštitu i ostale delove) da se modeluju unutar simulatora i da se ne uključuje spoljno povezivanje ili ulazi/izlazi (I/O). S druge strane, HIL simulacija se odnosi na stanje gde delovi potpuno digitalnih simulacija u realnom vremenu bivaju zamenjeni stvarnim fizičkim komponentama.



Sl.1. *(a) Simulacija u realnom vremenu, (b) Simulacija koja nije u realnom vremenu [2]*

Ako HIL sistem uključuje pravi hardver kontrolera koji deluje sa ostatkom simuliranog sistema, tzv. hardver u petlji kontrolera (CHIL). Takođe se koristi za brzu izradu prototipa kontrolera. U ovoj metodi nema stvarnog prenosa snage i elektroenergetski sistem je modelovan kao virtuelni sistem unutar simulatora i spoljni kontroler hardver razmenjuje I/O kontrolera sa sistemom unutar simulatora. Generalno, novo dizajniran/razvijen kontroler se testira pomoću ove metode, gde kontroler uzima povratne signale sa simulatora i obrađuje ih da bi proizveo potrebne izlazne signale, koji se zatim šalju nazad u sistem (unutar simulatora). Takva postavka kontrolera prototipa ili CHIL je prikazan na slici 2., gde je energetski elektronski pretvarač modelovan unutar simulatora, a realni kontroler je povezan sa njim preko I/O [2].



Sl.2. *Osnovni HIL koncpet simulacije za CHIL i PHIL [2]*

Bilo koja HIL simulacija koja uključuje prenos snage na ili sa HuT-a je poznata kao hardver za napajanje u petlji (PHIL) (slika 2.). U ovom slučaju, deo elektroenergetskog sistema je interno simuliran, a drugi deo je realano hardverski spojen spolja. Izvor napajanja ili potrošač (povezan preko PHIL interfejsa) je potreban za ovo podešavanje, koje će ili proizvoditi ili apsorbovati snagu. Referentni signali se generišu na osnovu rešenja virtuelnog sistema unutar simulatora u realnom vremenu i šalju se na pojačavač snage koji proizvodi potrebne napone ili struje za primenu na HuT-u. Povratni signali dobijeni merenjem napona/struje sa HuT-a su na odgovarajući način skalirani i vraćeni u simulator da bi se završila simulaciona petlja.

Primer takve simulacije u realnom vremenu može biti ispitivanje mašina, pretvarača, ograničavača struje kvara, odnosno bilo koja druga električna oprema. Ispitivanje zaštitnih uređaja, kao što su releji, mogu zahtevati pojačavače napona ili struje za HIL testiranje, međutim u tome se ne razmenjuje nikakva snaga [2].

4. STANDARDNI TEST SISTEMI

Dva glavna test sistema obuhvaćena u ovom radu su standardi IEEE i CIGRE [1,3]. Najbitnije karakteristkike stan-dardnih sistema za ispitivanje su predstavljeni u tabeli 1, a odgovarajuće jednopolne šeme su prikazane u radu. Sve IEEE test mreže su pogodne za konvencionalne sisteme napajanja naizmeničnom strujom. U međuvremenu, CIGRE test mreže koje su deo mreža evropskih zemalja, uključujući sisteme napajanja naizmeničnom i/ili jedno-smernom strujom [1, 3].

Tabela 1. *Standardne test mreže [1,3]*

Test sistem	Napon [kV]	Broj sabirnica	Namena
IEEE 9	13,8; 16,5; 18 i 230	9	Stabi-lnost
IEEE 14	13,8; 18 i 69	14	Procena stanja
IEEE 30	33; 132	30	Planiranje
IEEE 57	345	39	Stabilnost
IEEE 118	138; 345	57	Procena stanja
IEEE 300	138; 230 i 345	118	Planiranje
CIGRE B4 DC	±400; ± 200; 380 i 450	300	Nove tehnologije

4.1. Izazovi savremenih test mreža i zahtevi koji se postavljaju

IEEE test mreže se široko koriste za različite studije pojava u elektroenergetskom sistemu [5-8]. Mogu se modifikovati da uključe u razmatranje i promenljive energetske resurse i nove tehnologije, odnosno prenosne i distributivne energetske sisteme, mada te promene nisu usvojene iz realnih sistema napajanja.

Međutim, podaci koji se koriste za moderne postojeće konvencionalne mreže, mogu neki put izazvati pogrešne procene. Pored toga, uprkos naporima koji su posvećeni ažuriranju postojeće test mreže (na pr. IEEE PTC-96) i koji omogućuvaju da se ova mreža koristi za savremene sistemske studije, potrebno je više nadogradnje u budućim verzijama. Razlog za to je činjenica da ne obuhvataju detaljne informacije o vetroturbinama, vetroparkovima, solarnim elektranama i skladištima energije.

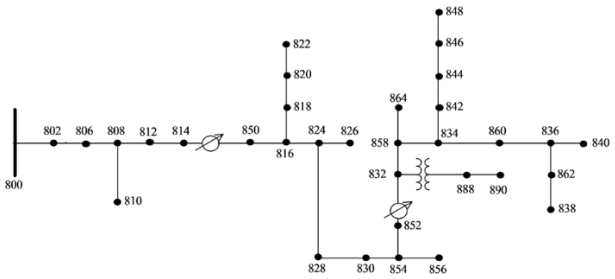
Ovakav koncept, više je primenljiv na studije planiranja, npr. upravljanje energijom i procene adekvatnosti. U međuvremenu, u predstavljenom ažuriranju PTC GLMC, podaci o pouzdanosti za promenljive izvore napajanja nisu obezbeđeni energetskim resursima, posebno za tehnologiju baziranu na energetskoj elektronici. Ove tehnologije su jedne od čestih neuspeha izvora u vetroelektranama. Nasuprot tome, struktura vetroparka može uticati ne njegovu pouzdanost, dakle detaljnije tehničke informacije u vezi sa vetrom i solarne elektrane trebaju biti obezbeđene u budućim ažuriranjima.

Štaviše, IEEE sistemi za testiranje koji su namenjeni za analize stabilnosti nisu prikladni za pitanja stabilnosti u vezi sa kontrolom, kao što je harmonijska stabilnost. Čak i test mreža IEEE 39, koja je namenjena posebno za analizu stabilnosti, zahteva odgovarajuće modifikacije da bi se olakšalo uključivanje novih tehnologija.

Test mreže CIGRE su više pogodne za savremene studije elektroenergetskog sistema, koji obuhvataju nove tehnologije. CIGRE B4DC se može koristiti za analizu stabilnosti, zaštite, protoka energije, planiranje, kontrolu kvaliteta električne energije, koje odgovoraju u DC mrežama sa više terminala. Međutim, nije primenljivo za analize orijentisane na pouzdanost, tj. procene adekvatnosti zbog nedostatka podataka o pouzdanosti. Štaviše, farme vetroparkova nisu bile obuhvaćene ovom test mrežom. CIRGE LV/MV mreže su pogodne za proučavanje mikromreža naizmenične struje, npr. kontrolu i planiranje, međutim zahtevaju neke modifi-kacije za analizu pouzdanosti i sajber bezbednosti [1].

4.2. IEEE 34-bus

IEEE 34-bus test mreža je mreža koja se nalazi u Arizoni, sa nominalnim naponom od 24,9 kV. Odlikuje se dugim i lagano opterećenim nadzemnim dalekovodima, dva linijska regulacina transformatora, jedan linijski transformator za kratku deonicu naponskog nivoa 4,16 kV, 24 nebalansirana opterećenja i dva šant kondenzatora. Na slici 3 data je topologija same mreže, sa svim svojim deonicama, opterećenjima i sabirnicima, kao i transformatorima.

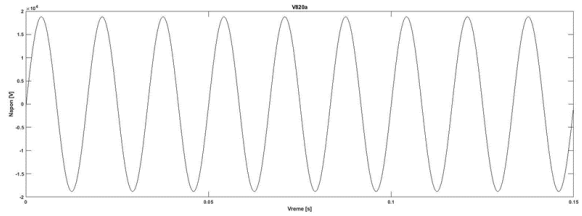


Sl. 3. *Šematski prikaz IEEE 34 test mreže [3,4]*

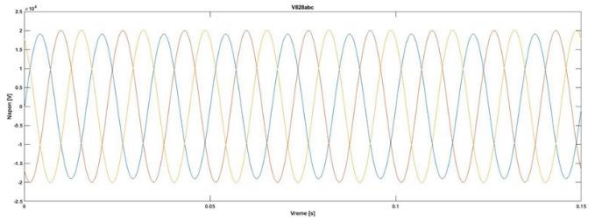
Iz dokumentacije koja je dostupna, bilo je potrebno iščitati podatke za datu mreži i za sve njene segmente. Kao što su podaci nadzemnih vodova, dužina trase, konfiguracija, zatim sve podatke vezane za potrošače koji su priključeni u određene čvorove bilo da su distribuirani ili tačkasto orijentisani. Potreblo je i prilagoditi navedene podatke programskom okruženju Matlab – Simulink, kao što su merne jedinic i odgovarajuće fizičke veličine. Iz razloga što je model većih dimenzija neće biti prikazan ovom članku, dok se model može pogledati u master radu.

4.3. Rezultati simulacije

U cilju testiranja modela izvršeno je niz simulacija. Mereni su fazni naponi (između faze i zemlje), međufazni naponi, kao i viši harmonici (2, 5, 7) u pojedinim čvorovima. Za potrebe simulacije izvršena je modifikacija modela, dodati su blokovi za merenje. Neki rezulztati prikazani su na slikama 4 i 5. Vidi se da model dobro funkcioniše i da su dobijeni očekivani rezultati.

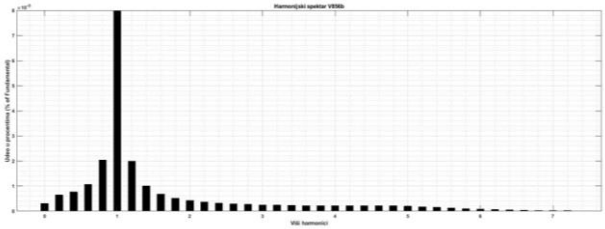


Sl. 4. *Fazni napon u fazi A čvora 820*



Sl. 5. *Međufazni napon u čvoru 828*

U nastavku, razmatran je uticaj viših harmonika na mrežu. Ukoliko je napon izobličen u odnosu na idealni sinusni talasni oblik, kaže se da je „zagađen“ višim harmonicima. Takav talsni oblik može se predstaviti kao zbir osnovnog harmonika (u ovom slučaju to je na 60 Hz) i viših harmonika koji su umnožak osnovog (2, 5, 7...). Na osnovu simulacionih rezultata, datih na slici 6 može se zaključiti da je dominantan osnovni harmonik na 60 Hz i da mreža nije zaprljana višim harmonicima.



Sl. 6. *Harmonijski spektar u fazi B čvora 856*

5. ZAKLJUČAK

Ovaj rad je u prvom delu pregledao standardne test sisteme, uključujući IEEE i CIGRE test mreže, kao i njihovu primenljivost na proučavanje elektroenergetskog sistema. Krajnji cilj studija elektroenergetskog sistema je da efikasno snabdeva krajnje korisnike sa prihvatljivim nivoom pouzdanosti. Postizanje takvog cilja za velike, složene sisteme zahteva dugoročna istraživanja, projektovanja i planiranja, kratkoročne studije za operativno planiranje, kao i rad u realnom vremenu.

Iako je izvestan broj simulatora u realnom vremenu, samo nekoliko njih je u stanju da simulira velike sisteme. Preostali su ili pogodni za male sistema ili da služe kao kontrolor u realnom vremenu. Stoga karakteristike tri glavna simulatora u realnom vremenu su detaljno razmotreni, dok su neki od ostalih simulatora samo spomenuti. Većina simulatora u realnom vremenu je sposobna sa interfejsom eksternog hardvera za izvođenje HIL testova i eksperimenata.

Iako su dizajnirani za aplikacije elektroenergetskih sistema, mnogi ovih simulatora u realnom vremenu su pogodni za izvođenje simulacija korišćenjem multirate/ multiphysics simulacije. Stalna korisnička podrška, omogućava ovim simulatorima da se mogu koristiti da udovolje svim posebnim potrebama potrošača.

Poslednji segment rada je detaljan opis test mreže IEEE 34, kao i izrada simulacionog modela i prikaz rezultata simulacije, čime potvrđujemo verodostojnost razvijenog modela.

6. LITERATURA

[1] S. Peyghami, P. Davari, M. Fotuhi-Firuzabad, and F. Blaabjerg, „Standard Test Systems for Modern Power System Analysis”, IEEE Industrial Electronics Magazine, Vol.13, No.4, Dec. 2019, pp.86-105.

[2] T. Strasser, „Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis”, IEEE Power and Energy Technology Systems Journal, Vol.2, No.2, June 2015, pp.63-73.

[3] A.M. Stanisavljević , V.A. Katić, B.P. Dumnić, B.P. Popadić, „A Brief Overview of the Distribution Test Grids with a Distributed Generation Inclusion Case Study“, *Serbian Journal of Electrical Engineering*, Vol.15, No.1, Feb.2018, pp. 115 – 129.

[4] IEEE PES, „IEEE 34 Node Test Feeder”, 2004

[5] D. Stojišić, V.A. Katić, A.M. Stanisavljević, „Modelovanje i analiza kvaliteta napona u realnoj i test mreži”, Zbornik radova FTN, Vol. 37, No.11, 2022, pp.1934-1937.

[6] N. Lučić, V.A. Katić, A.M. Stanisavljević, „Simulacija propada napona u distributivnoj test mreži sa obnovljivim izvorom energije”, Zbornik radova FTN, Vol. 37, No.2, 2022, pp.262-265.

[7] V. Vidačić, V.A. Katić, “Uticaj obnovljivih izvora energije na propade napona u distributivnim mrežama”, Zbornik radova FTN, Vol. 36, No.7, 2021, pp.1287-1290.

[8] J. Toholj, V.A. Katić, A.M. Stanisavljević, „Modelovanje i analiza uticaja propada napona primjenom test mreža sa distribuiranim generatorima”, Zbornik radova FTN, Vol.36, No.1, 2021, pp.123-126.

Kratka biografija:

Dejan Vračarić rođen je u Vrbasu 1997. god. Srednju školu ETŠ Mihajlo Pupin, završio je u Novom Sadu, 2016 god. Fakultet tehničkih nauka, studijski program Energetika, elektronika i telekomunikacije upisao je školske 2016/2017. Na studijama se opredelio za smer Elektroenergetika - energetska elek-tronika i mašine i diplomirao 22.09. 2021. god. Master studije upisuje školske 2021/2022. na smeru Elektroener-getika - energetska elektronika i mašine i odbranio master rad 2023.

Aleksandar Stanisavljević rođen je 1988. godine u Beogradu. Doktorirao je na Fakultetu tehničkih nauka u Novom Sadu 2019. god. Kao asistent na Katedri za energetsku elektroniku i pretvarače radi od 2016. god., a kao docent od 2019. god. Učestvovao je na brojnim međunarodnim konferencijama. Autor ili koautor je više od 20 naučnih radova, od kojih su tri publikacije u vrhunskim međunarodnim časopisima.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/23BE14Cuk

PRIMENA AMI, DR I DERMS TEHNOLOGIJA PAMETNIH DISTRIBUTIVNIH MREŽA U REŠAVANJU PROBLEMA NEDOSTATKA SNAGE I ENERGIJE U USLOVIMA ENERGETSKE KRIZE

APPLICATION OF AMI, DR AND DERMS TECHNOLOGIES OF SMART GRIDS, IN FIXING THE LACK OF POWER AND ENERGY CAUSED BY ENERGY CRISIS

Ivana Ćuk, Vladan Krsman; *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je dat primer rešenja problema nedostatka energije i snage u elektroenergetskom sistemu. Predložene su tri tehnologije pametne mreže čijom implementacijom se može poboljšati trenutno stanje. Dat je kratak osvrt na prednosti koje se mogu ostvariti upotrebom ovih tehnologija.*

Ključne reči: *Elektroenergetski sistem, nedostatak energije, tehnologije pametne mreže*

Abstract – *This paper presents a problem of lack of power resources in traditional electric power system. Three smart grid technologies that can improve current situation are proposed. A review of the advantages that can be realized by their implementation is given.*

Keywords: *Electric power system, lack of resources, smart grid technologies*

1. UVOD

Energetika je privredna delatnost koja se bavi proučavanjem i iskorištavanjem različitih izvora energije, te proizvodnjom, prenosom i distribucijom električne energije. Bitna je za razvoj čitavog društva.

Smanjenje gubitaka i ekonomična proizvodnja, efikasan raspored proizvođača i izbor optimalnog načina transporta predstavljaju jedan od zadataka energetike. Pored toga, teži se što manjem zagađenju životne sredine, pre svega se misli na smanjenje emisije gasova (ugljen-dioksida) koji predstavljaju uzrok efekta staklene bašte, kao i drugih štetnih proizvoda nastalih pri procesu proizvodnje energije.

Od velikog značaja za svaku državu je postizanje energetske stabilnosti i nezavisnosti, za postizanje toga potrebno je ulagati u obnovljive izvore energije kao što su energija Sunca i vetra.

Tradicionalna mreža je napravljena tako da izdrži ekstremne tokove snaga koji se retko dešavaju. Ovakvo dimenzionisanje je skupo ali je u skladu sa trenutnim tarifama koje postavlja Evropska regulativa.

Uvodi se koncept pametne mreže koji otvara nove mogućnosti upravljanja, monitoringa, dvosmerne komunikacije. U pametnoj mreži uloge i odgovornosti učesnika se menjaju kako bi se prilagodila integracija distribuirane proizvodnje, energetski efikasne usluge, električna vozila i punjači, lokalno balansiranje snage, fleksibilnost nabavke i velike količine podataka.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladan Krsman, docent.

2. TRADICIONALNI ELEKTROENERGETSKI SISTEM

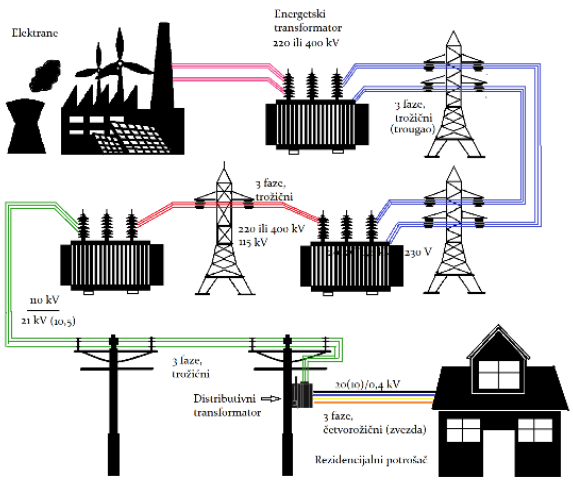
Elektroenergetski sistem (EES) sastoji se od četiri podsistema (prikazano na slici 2.1); proizvodnja, prenos, distribucija i potrošnja.

Podsistem proizvodnje čine elektrane u kojima se vrši proizvodnja električne energije. Cilj je zadovoljenje trenutne potražnje u sistemu uz uvažavanje gubitaka koji se javljaju u prenosnim i distributivnim mrežama. Usled raznih neplaniranih situacija potrebno je obezbediti i odgovarajuću rezervu kapaciteta.

Podsistem prenosa vrši transport električne energije na velike udaljenosti. Prenosna mreža se sastoji od elektroenergetskih vodova i elektroenergetskih razvodnih postrojenja. Za prenos električne energije se koriste naponi reda 110 kV, 220 kV i 400 kV, dok se u razvijenijim zemljama koriste naponi višeg reda.

Podsistem distribucije obuhvata distributivnu mrežu i distributivne transformatore. Njen zadatak jeste prenos električne energije do srednjih i malih potrošača. Naponi koji se koriste u distributivnoj mreži su reda 110 kV, 35 kV, 20 kV i 10 kV.

U podsistemu potrošnje električna energija ima široku primenu. Varijacije u potrošnji koje sa javljaju su rezultat različitih faktora kao što su period dana, nedelje, meseca i godine. Osnovni pokazatelji potrošnje su maksimalno opterećenje P_{pM}, minimalno opterećenje P_{pm}, ukupno utrošena energija W_p.

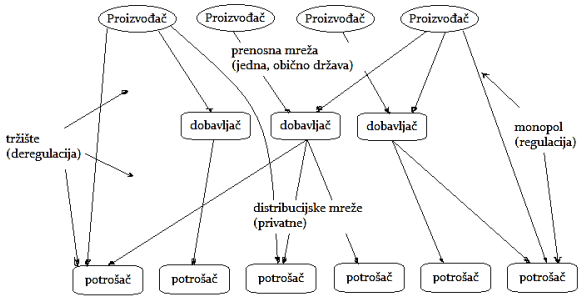


Slika 2.1 – *Tradicionalni elektroenergetski sistem*

3. ORGANIZACIJA ELEKTROPRIVREDE

Elektroprivreda predstavlja granu privrede koja se bavi eksploatacijom elektroenergetskog sistema. Organizacija elektroprivrede zavisi od tržišta električne energije, a time i od vlasništva nad elektroenergetskim sistemom.

Savremena organizacija elektroprivrede je prikazana na slici 3.1.



Slika 3.1 – *Savremena organizacija elektroenergetskog sistema*

Iz JP Elektroprivrede Srbije formirana su dva preduzeća: Elektroprivreda Srbije (EPS) i Elektromreža Srbije (EMS).

Cilj EPS-a je snabdevanje kupaca električnom energijom pod najpovoljnijim uslovima tržišta, teže ka stalnom podizanju kvaliteta usluga uz brigu o životnoj sredini. EPS je vertikalno organizovano preduzeće.

Osnovni cilj EMS-a je zadovoljenje potreba korisnika što se ima pri sigurnom i pouzdanom prenosu električne energije, teži se ka efikasnom upravljanju prenosnim sistemom, njegovom optimalnom razvoju i održivosti.

4. PROBLEMI TRADICIONALNOG ELEKTROENERGETSKOG SISTEMA

Problem koji postoji u trenutnom stanju elektroenergetskog sistema u većini država je zastarela infrastruktura napravljena tokom perioda XX veka, osim toga postoji velika zavisnost od fosilnih goriva što ima negativan uticaj na životnu okolinu usled emisije ugljen-dioksida. Zbog ekoloških potreba, finansijskih i tehničkih izazova ali i manje cene tehnologija obnovljivih izvora, postoji težnja ka modernizaciji elektroenergetskog sistema [1]. Ovo zahteva radikalne promene kako bi se postigla tranzicija ka dekarbonizaciji, pouzdanom i pristupačnom elektroenergetskom sistemu.

Trenutno stanje energetskog sektora je ugroženo, ravnoteža između sigurnosti, pristupačnosti i održivosti je došla pod opterećenje, postoji nestabilnost energetskog tržišta. Ako se ne upravlja efikasno, kriza može imati negativan uticaj na postizanje cilja nulte emisije štetnih gasova kao i sabotirati tranziciju ka zelenoj agendi (European Green Deal – EGD).

U Srbiji se oko 40 odsto energije proizvodi u termoelektranama, oko 56 odsto u HE, dok elektrane na gas i vetar ostvaruju oko 4 odsto energije [2]. Kod nas TE koriste ugalj niske energetske vrednosti (lignit), oko 50 % čine ga vlaga i pepeo [2]. U periodu prošle godine zabeležene je niz havarija u termoelektranama i rudnicima uglja Elektroprivrede Srbije, tokom leta 2022. godine sušni period je spustio nivo vode u akumulacijama za HE u Srbiji te je država morala da se okrene ka interventnom uvozu struje. Predviđa se godišnji pad proizvodnje HE Đerdap od najmanje 30 odsto. Slična situacija sa hidroelektranama je i u ostatku Evrope.

Usled nestašice energenata isčekuje se neizvesna zima za čitavu Evropu. Očekivana su poskupljenja struje jer se očekuje i potreba za uvozom određene količine energije. Moguće su i redukcije struje, i problem sa kvalitetom vazduha. Kako bi se radilo na prevazilaženju ovih problema neophodno je raditi na diverzifikaciji snabdevanja i povećanju energetske efikasnosti. Srbija ima mnoge mogućnosti upotrebe energije vetra i Sunca te je potrebno ulagati u njihov razvoj. Do 2025. godine planira se povećanje korišćenja energije iz obnovljivih izvora na 27 odsto [2].

5. KONCEPT PAMETNE MREŽE

Pametna mreža se odnosi na elektroenergetski sistem u kome se primenjuje moderna informaciona i komunikaciona tehnologija radi postizanja efikasne proizvodnje, prenosa, distribucije i korišćenja električne energije.

Podaci se prikupljaju od korisnika i proizvođača kako bi se obezbedio stalan i pouzdan dotok električne energije. One omogućavaju decentralizaciju proizvodnje energije, individualnim potrošačima može obezbediti dvosmeran protok električne energije tako da se višak energije koji proizvedu vraća u mrežu. Pametna mreža doprinosi i da državna mreža najefikasnije koristi svoje energetske resurse, pametnim merenjima se može obezbediti da se u svakom trenutku koristi najjeftiniji i najefikasniji izvor energije. Radi ograničenja potrošnje tokom perioda najvećeg opterećenja elektromreže, pametna mreža može obavljati ulogu posrednika između sistema distribucije i krajnjih korisnika [3].

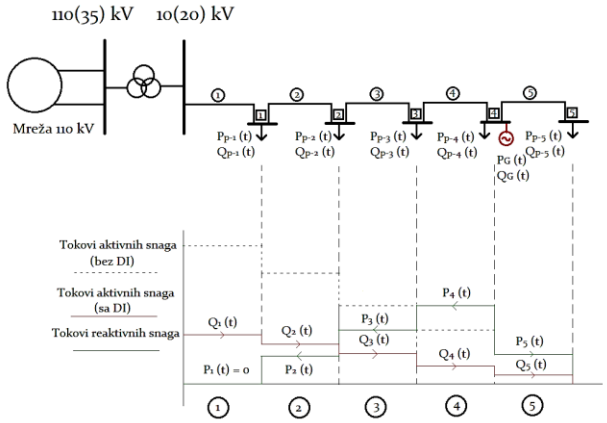
Pametna mreža daje brojna nova rešenja i novu generaciju tehnologija, ona mora obezbediti: potrebno je da postoji potpuno automatizovana i integrisana dvosmerna komunikacija između svih komponenti mreže; automatska kontrola snage svake tačke električne mreže; automatska detekcija i korekcija kvarova; treba da poveća ukupnu efikasnost i sigurnost sistema; unapređenje celog EES-a; efikasniju kontrolu infrastrukture EES-a; napredni upravljački panel i odgovarajući softveri; precizna tehnologija merenja i senzora.

Neke od tehnologija pametne mreže su: napredna merna infrastruktura, sistem za upravljanje energijom (*Energy management system* – EMS), sistem za upravljanje opterećenjem (*Demand response* – DR), električna vozila (electric vehicles – EV), sistem za upravljanje distributivnom mrežom (*Distribution management system* – DMS), sistem za upravljanje ispadima (*outage management system* – OMS), pametni prekidački uređaji i multifunkcionalni senzori (*Inteligent electronic device* – IED), automatizovani fideri, automatizovane transformatorske stanice.

6. PREDLOG REŠENJA PROBLEMA TRADICIONALNOG EES UPOTREBOM TEHNOLOGIJA PAMETNE MREŽE

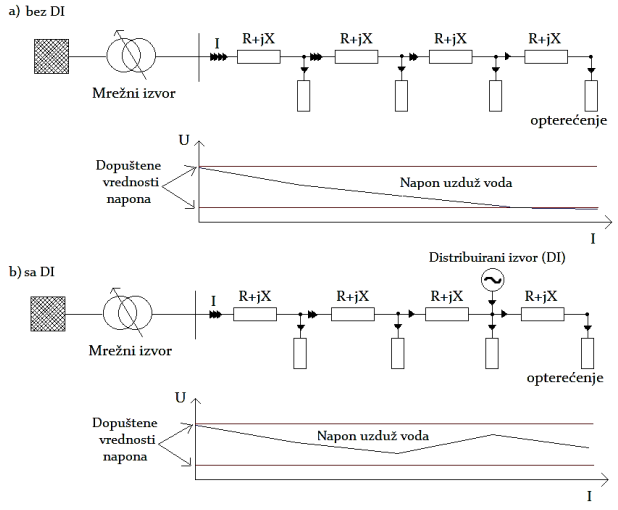
Trenutni elektroenergetski sistem je nestabilan sa značajnim gubicima u prenosu, niskim kvalitetom električne energije, visokim rizikom od prekida napajanja, nedostatkom energije i snage. Jedno od potencijalnih rešenja za prevazilaženje problema nedostatka energije i snage u sistemu se može naći pri upotrebi tehnologija pametne mreže. Potrebno je ulagati u distribuirane energetske izvore (*Distributed energy resources* – DER), oni predstavljaju proizvodnju i skladištenje u blizini potrošačkih konzuma. Na ovaj način se ostvaruje decentralizovana proizvodnja, pri tome treba voditi računa o promenama pogonskih parametara mreže koje se javljaju. Ovi novi fleksibilni resursi će omogućiti balansiranje energije u sistemu, kako bi se to postiglo potrebno je ostvariti njihovu vidljivost i omogućiti njihovo upravljanje.

Na slici 6.1 je prikazan deo distributivne mreže, 10(20) kV-ni izvod sa 5 čvorova i 5 grana vodova. Sa P_{p-1} i Q_{p-1} su označene aktivne i reaktivne snage i-tog čvora [4]. Mreža sada postaje aktivna, vrednosti i smerovi tokova snaga se menjaju u svim granama. Aktivna snaga koju proizvodi mala elektrana u trenutku t je označena sa $P_G(t)$, dok je reaktivna snaga označena sa $Q_G(t)$.



Slika 6.1 – *Aktivna distributivna mreža*

Uticaj priključenja male elektrane na naponske prilike je prikazan na slici 6.2.



Slika 6.2 – *Naponske prilike u a) pasivnoj i b) aktivnoj distributivnoj mreži*

Priključenjem distribuiranog izvora u čvoru 4 doprinosi smanjenju gubitaka u prenosnom sistemu, proizvodnja se približava potrošnji čime se smanjuje put od proizvodnje do potrošačkog konzuma, povećava se sigurnost snabdevanja, smanjuje se uticaj na životnu sredinu, doprinosi se poboljšanju naponskih prilika. Napon je potrebno održavati u propisanim granicama, pri niskom ili povišenom naponu se javljaju problemi u radu električnih uređaja ili njihovo oštećenje.

Od obnovljivih izvora danas se najviše teži ka razvoju energije vetra, Sunca i biomase. Upotreba solaranih panela (photovoltaic – PV) ima sve veći značaj, jednostavni su za postavljanje, ne zahtevaju mnogo održavanja ni prostora, pristupačnih su cena. Potrebno je zameniti postojeća brojila, upotreba napredne merne infrastructure (advanced metering infrastructure – AMI) koja obuhvata pametna brojila će omogućiti uvid u razliku između proizvedene i potrošene električne energije.

AMI predstavlja instegrisani sistem pametnih brojila, komunikacione mreže i sistema za upravljanje podacima koji omogućava dvosmernu komunikaciju između potrošača i operatora distributivne mreže. Glavne komponente AMI tehnologije su: pametna brojila, komunikaciona infrastruktura, prijemnik podataka, kućna mreža. Neke od prednosti koje će omogućiti upotreba AMI tehnologija su: automatsko uključivanje/isključivanje potrošača, češća merenja (u intervalima od 5-, 15-, 30 ili 60-minuta, smanjenje računa i troškova merenja, veći uvid potrošačima o sopstvenoj potrošnji, veće iskorišćenje i održavanje asseta-a (dobro, imovina), lakše upravljanje kvarovima i prekidima napajanja, monitoring napona, implementacija pre-paid koncepta.

Sistem za upravljanje opterećenjem (DR) predstavlja jedno od rešenja pametne mreže čija upotreba obezbeđuje efikasnu interakciju između lokalne proizvodnje, skladišta baterija i potrošnje. Postoje različite kategorije DR programa, osnovna podela je na DR zasnovan na cenama (vremenu korišćenja) i programi zasnovani na podsticajima. Njihovom primenom se daje mogućnost potrošačima da dobrovoljno pomere ili smanje sopstvenu potrošnju. Neke od tehnologija koje mogu obezbediti takvu fleksibilnost su toplotne pumpe, klima uređaji, električni sistemi grejanja, električni bojleri, mašine za pranje sudova, veš mašine, frižideri/zamrzivači, i sl.

U slučaju posedovanja DER-ova kao što su solarni paneli, omogućava se prodaja viškova proizvedene električne energije. Primena DR-a ima veliki značaj kao odgovor na stalno rastuću potražnju, takođe utiče na smanjenje troškova optimizovanjem gubitaka čime se smanjuju investicije na mrežu.

Upotreba DR programa može pomoći u prevazilaženju problema koji nastaje pri intermitentnoj prirodi DER-ova, omogućavajući balansiranje proizvodnje i potrošnje nudeći razne programe potrošačima. Za cilj se ima maksimizacija usluga, ušteda energije i smanjenje operativnih troškova. Pomeranjem potrošnje u vršnim periodima dana putem DR programa smanjuje potrošnju električne energije, račune, obezbeđuje veću sigurnost distributivnih mreža. Industrijski potrošači predstavljaju veliki teret energetskom sistemu te im njihovo učešće u DR programima može doprineti veliku finansijsku korist i smanjiti teret sistemu. Implementacijom DR programa

kao i upotrebom PV sistema u rudarskoj industriji došlo bi do promene upravljanja u rudnicima, ublažio bi se efekat zagađenja, dobile bi se jeftinije rudarske operacije, povećala bi se energetska efikasnost na lokacijama rudnika, poboljšala bi se sigurnost i pouzdanost. Kako bi se to omogućilo potrebno je postepeno menjati poslovne modele i odrediti optimalan miks energije koji bi ostvario rad sa maksimalnim kapacitetom.

Pri implementaciji DR programa potrebno je obezbediti jasnu komunikaciju, voditi računa o sajber bezbednosti i obezbeđenju pravičnosti i pristupačnost. Primenom DR-a omogućava se da potražnja za električnom energijom postane fleksibilna, prilagođena prema proizvodnji čime se smanjuje stres na elektroenergetskoj infrastrukturi. Efikasni DR programi su ključ ka implementaciji pametne mreže kao i promovisanju zelenih i čistih izvora energije. Smanjenje potrebnih kapaciteta dovodi do smanjenja u emisiji štetnih gasova što doprinosi zaštiti životne okoline.

Pri velikom prodoru DER-ova u distributivnoj mreži mogu se javiti problemi kao što su obrnuti tok snage (viškovi proizvedene energije se vraćaju iz distributivne mreže u prenosnu mrežu), tradicionlani zaštitni uređaji nemaju mogućnost zaštite i može doći do njihovog nepravilnog rada. Kako bi se nosili sa tranzicijama koje uvodi upotreba DER-ova potrebno je obezbediti nove pristupe, nova rešenja u njihovom upravljanju kao i upravljanju mrežom. Pametno korišćenje DER-ova može doprineti odlaganju projekata ojačanja mreže kao i umanjiti uticaj planiranih i neplaniranih prekida na potrošače. Pri udelu od 20-30 % DER-ova u režimu srednjenaponskog fidera se javlja potreba za njihovom aktivnom kontrolom [5]. Jedno od softverskih rešenja za monitoring, predviđanje i kontrolu DER-ova predstavlja distribuirani sistem za upravljanje energetskim resursima (*Distributed energy management system* – DERMS).

DERMS se odlikuje raznim funkcionalnostima, od toga pet osnovnih su: fleksibilnost, grupisanje, prognoza, monitoring i optimizacija. Ovaj sistem je moguće razvijati, povećavati njegovu složenost uz testiranje svakog koraka, na taj način se može omogućiti napredovanje DERMS-a u okviru organizacije. Pri implementaciji treba voditi računa o sajber sigurnosti, potencijalnim slabim tačkama, standardima i regulativama. DERMS nudi brže, jeftinije i ekonomski efikasno rešenje za izazov povezivanja DER-ova na mrežu. DERMS će omogućiti mrežnim operatorima nadgledanje u realnom vremenu i prognozirano ponašanje DER-ova, dati na uvid njihovu fleksibilnost i omogućiti optimalno angažovanje grupe DER-ova.

Pregled nekih od razloga za upotrebu DERMS-a: kapitalne uštede/odlaganje ulaganja; uštede usled obezbeđenja operativne efikasnosti; poboljšanje pouzdanosti mreže; rešavanje nastalih nedostataka u snabdevanju; smanjenje emisije štetnih gasova, doprinos zaštiti životne sredine; poboljšanje saradničkih odnosa sa kupcima.

7. ZAKLJUČAK

Problema tradicionalnog elektroenergetskog sistema se može rešiti ulaganjem u distribuirane izvore energije koji će omogućiti decentralizovanu proizvodnju.

Kako bi se to ostvarilo potrebno je ulagati u razvoj i unapređenje mreže upotrebom tehnologija pametne mreže.

Upotreba AMI tehnologije predstavlja prvi korak ka transformisanju postojeće mreže. Ona obezbeđuje monitoring i kontrolu mreže, na taj način pomaže elektroprivrednim preduzećima da odgovore na fluktuacije potražnje za električnom energijom, omogućava lakšu implementaciju DR sistema. U kombinaciji sa pametnim brojilom i digitalnim sistemima upravljanja povezani uređaji i DER-ovi mogu značajno doprineti DR-u.

S jedne strane, pomeranjem potrošnje u vršnim periodima dana putem DR programa smanjuje se potrošnja električne energije, računi, obezbeđuje se veća sigurnost distributivne mreže. S druge strane, raspolažući velikim brojem DER-ova potrebno je obezbediti dobru kontrolu čime se može omogućiti veća fleksibilnost na strani potražnje i pun potencijal DR programa. To se može obezbediti upotrebom DERMS-a koji ima mogućnost identifikovanja problema u sistemu kao i proaktivnog predviđanja mogućnosti DER-ova u cilju ublažavanja problema.

Upotreba tehnologija pametne mreže doprinosi pouzdanijem i sigurnijem radu mreže, ostvaraju se mnogobrojne mogućnosti i benefiti kako za elektroenergetski sistem tako i za potrošače električne energije.

8. LITERATURA

[1] Kristina Hojckova, „Emerging networks of power – Exploring sociotechnical pathways towards future electricity systems based on renewable energy technologies“, phD thesis, Gothenburg, 2020.

[2] Jovana Georgievski, „Srbija, energija i TENT: Od prvog kilovat-sata do struje za pola Srbije – kako radi najveća termoelektrana na Balkanu”, 2022.

[3] „Obnovljivi izvori energije vodič za parlamentarce“, dostupno na: https://www.rs.undp.org/content/serbia/sr/home/library/democratic_governance/obnovljivi-izvori-energije---vodi-za-parlamentarce.html

[4] „Distribuirani izori električne energije,” 2018. Dostupno na: https://www.ucg.ac.me/skladiste/blog_9437/objava_23415/fajlovi/!!PrDIEE18.pdf

[5] Stuart Borlase, „Smart Grids – Advanced Technologies and Solutions,“ Taylor&Francis Group, 2018.

Kratka biografija:

Ivana Ćuk rođena je u Zrenjaninu 1997. godine. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektroenergetski sistemi odbranila je 2021. godine.

Vladan Krsman rođen je u Sarajevu 1985. godine. Doktorsku disertaciju odbranio je 2017. godine na Fakultetu tehničkih nauka iz oblasti Elektroenergetski sistemi.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/23BE15Blesic

SERVIS ZA DETEKCIJU PLAGIJARIZAMA U NAUČNIM RADOVIMA

SERVICE FOR DETECTION OF PLAGIARISM IN SCIENCE WORKS

Nikola Blesić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je predstavljena arhitektura i implementacija servisa za detekciju plagijata u naučnim radovima. Servis upoređuje sumnjiv dokument sa referentnom kolekcijom koja predstavlja skup dokumenata za koje se pretpostavlja da su originali. Takođe, u radu je predstavljeno prethodno istraživanje koje se odnosi na višeslojnu veb aplikaciju za prijavu i evidenciju radova, kao i potrebne izmene kako bi se izvršila integracija ova dva servisa. Na kraju rada predstavljeni su rezultati rada servisa.*

Ključne reči:*Elasticsearch, plagijarizam, naučni radovi*

Abstract – *This paper presents the architecture and implementation of the service for plagiarism detection in scientific papers. The service compares a suspect document with a reference collection which represents a set of documents that are assumed to be originals. Also, the paper presents previous research related to n-tier web application for thesis submission and assessment, as well as the necessary changes in order to integrate these two services. At the end of the work, the results of the service are presented.*

Keywords:*Elasticsearch, plagiarism, scientific papers*

1. UVOD

Plagijat u visokom obrazovanju bio je čest razlog za uzbunu u institucijama širom sveta u poslednjih nekoliko decenija [1]. Kada se provera plagijata radi ručno, to nije efikasno i veoma je zamorno za bilo kog pojedinca da bira datoteke jednu po jednu i upoređuje ih sa drugim datotekama kako bi utvrdio pojavu plagijata. Iz tog razloga, razvijaju se sistemi za detekciju potencijalnih plagijata koji sa velikom uspešnošću pronalaze podudaranja između dokumenata. Dakle, njihova glavna namena je ušteda vremena akademskom osoblju koje pokušava da otkrije plagijat u studentskom projektu, ali i da smanji prisutnost plagijata unutar institucije [2].

2. TEORIJSKE OSNOVE

Plagijat podrazumeva predstavljanje tuđeg rada ili ideja kao svojih, sa ili bez njihovog pristanka, tako što se ugrađuju u rad bez punog priznanja [3].

Sav objavljeni i neobjavljeni materijal, bilo u rukopisu, štampanom ili elektronskom obliku, obuhvaćen je ovom definicijom. Plagijat može biti nameran ili nepromišljen, ili nenameran [3]. Generalno, tehnike plagijata, po načinu nastanka, možemo klasifikovati na tri grupe: leksička, sintaksna i semantička [4].

Softver za detekciju potencijalnih plagijata, odnosno softver za otkrivanje sličnosti teksta, postao je širokodostupan proizvod, kako u obliku komercijalno dostupnih proizvoda, tako i u obliku otvorenog koda [5].

Softveri za otkrivanje sličnosti teksta implementiraju jedan od dva generička pristupa otkrivanju, jedan je spoljašnji, a drugi unutrašnji [5]. Predstavljeni servis za detekciju potencijalnih plagijata zasnovan je na spoljašnjem pristupu.

Spoljašnji pristup za otkrivanje upoređuje sumnjiv dokument sa referentnom kolekcijom koja predstavlja skup dokumenata za koje se pretpostavlja da su originali. Na osnovu izabranog modela dokumenta i unapred definisanih kriterijuma sličnosti, zadatak otkrivanja je da se pronađu svi dokumenti koji sadrže tekst čiji je stepen sličnosti iznad izabranog praga tekstu u sumnjivom dokumentu [5].

Softveri za detekciju plagijata oslanjaju se na oblast pronalaženja informacija. Pronalaženje informacija je oblast koja se bavi tehnikama za reprezentaciju, skladištenje, organizaciju, pristup i pronalaženje informacija. Sistemi za pronalaženje informacija najčešće imaju odvojene procese indeksiranja i pretraživanja [6].

Indeksiranje teksta je korak pretprocesiranja za pretraživanje teksta. Tokom procesa indeksiranja teksta, informacije se prikupljaju, parsiraju (raščlanjuju) i čuvaju, kako bi se omogućila brza i tačna pretraga [7].

Pretraživanje teksta definiše se kao podudaranje nekih navedenih korisničkih upita prema skupu tekstova. Kao rezultat pretrage teksta, tekstovi se rangiraju i prezentuju korisniku prema njihovoj relevantnosti u odnosu na upit (tj. informacionu potrebu) korisnika [7].

Pretprocesiranje igra važnu ulogu u pronalaženju informacija i predstavlja primenu određenih pravila na tekstualne sadržaje koji obezbeđuju fleksibilnost pretrage. Isto pretprocesiranje teksta se primenjuje na tekstove u dokumentima prilikom indeksiranja, kao i na reči u upitu prilikom pretraživanja. [6]

Postoje različite tehnike pretprocesiranja tekstualnih dokumenata. To su: pretvaranje velikih u mala slova, normalizacija, lematizacija, stemovanje, uklanjanje stop reči, uklanjanje akcenta, filtracija, itd. [6].

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

3. TEORIJSKE OSNOVE

Java je objektno orijentisani programski jezik visokog nivoa, zasnovan na klasama, koji je dizajniran da ima što manje zavisnosti od implementacije. To je programski jezik opšte namene, namenjen da dozvoli programerima da program pišu jednom i pokreću ga bilo gde, što znači da kompajlirani Java kod može da radi na svim platformama koje podržavaju Javu, bez potrebe za ponovnim kompajliranjem. Java aplikacije se obično kompajliraju u bajt kod koji može da radi na bilo kojoj Java virtuelnoj mašini (JVM) bez obzira na arhitekturu računara [8].

Spring je najpopularniji okvir za razvoj aplikacija za Java platformu. Spring okvir je otvorenog koda. Spring je lagan (eng. *lightweight*), kada je u pitanju veličina i transparentnost. Osnovna verzija Spring okvira je oko 2MB. Osnovne karakteristike okvira može da koristi bilo koja Java aplikacija, ali postoje proširenja za pravljenje veb aplikacija na vrhu Java EE (Enterprise Edition) platforme [9].

Elasticsearch je distribuiran, besplatan i otvoren server za pretragu i analizu svih tipova podataka, uključujući tekstualne, numeričke, geoprostorne, strukturirane i nestrukturirane podatke. Napisan je u Java programskom jeziku što omogućava pokretanje na svim platformama. Baziran je na *Lucine* indeksima i omogućava korisnicima da pretraže veliku količinu podataka vrlo brzo.

Može se koristiti i za čuvanje podataka, ali je njegova glavna uloga indeksiranje i pretraga podataka u realnom vremenu. Poznat je po svom jednostavnom REST API-ju, distribuiranoj prirodi, brzini i skalabilnosti [10, 11].

4. SPECIFIKACIJA

Servis za detekciju plagijata je veb servis čija je glavna namena provera prisutnosti plagijata u naučnim radovima, a pored toga on omogućava proveru sličnosti teme, prilikom njenog unosa sa već postojećim temama. Korisnički zahtevi koje sistem za detekciju plagijata treba da ispuni su:

- unos nove teme,
- provera sličnosti teme u odnosu na već postojeće teme,
- unos novog rada i
- provera plagijarizama.

Ideja rada je da prikaže primenu ovog servisa na već postojećoj veb aplikaciji, koja je predstavljena u okviru diplomskog rada „Višeslojna veb aplikacija za prijavu i evidenciju diplomskih radova”. Shodno tome, u ovom radu biće navedene izmene veb aplikacije, odnosno slučajevi krorišćenja na koje se izmene odnose. To su sledeći slučajevi:

- slanje diplomskog rada na proveru kod profesora,
- evaluacija diplomskih radova,
- predlaganje teme,
- odobravanje/odbijanje predloženih tema i
- objavljivanje novih tema.

4.1. Opis arhitekture

Servis za detekciju plagijata je serverska aplikacija koja je napravljena prateći višeslojnu arhitekturu aplikacija i sastoji se iz 4 sloja. Slojevi su sledeći:

- API sloj,
- sloj poslovne logike,
- sloj za pristup bazi podataka i
- baza podataka.

API sloj se sastoji iz programskih interfejsa API-ja koji su napravljeni po principu REST arhitekture. Zahtevi koji se šalju na servis namapiraju se na odgovrajući REST kontroler [12].

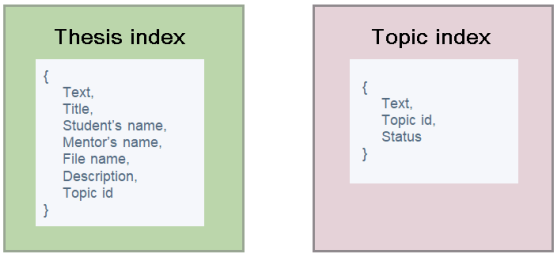
Sloj poslovne logike se sastoji pre svega iz modela. Model čini skup entiteta odnosno klasa koje se mapiraju u tabele baze podataka. Servisi su skup klasa sa zadatkom da izvršavaju neku konkretnu poslovnu logiku, odnosno funkcionalnost koja je srž aplikacije [12].

Sloj za pristup bazi podataka napravljen je po *repository* obrascu i sadrži klase koje izvršavaju upit ili komande za upis i brisanje nad bazom podataka.

Na kraju, tu i baza podataka koja služi za trajno čuvanje podataka koji su od važnosti za rad ovog servisa. Za bazu podataka korišćen je Elasticsearch, koji je NoSQL baza podataka.

4.2. Model podataka

Servis za detekciju plagijata sastoji se iz dva indeksa. Jedan indeks odnosi se na odbranjene radove, dok se drugi odnosi na teme radova. Na ovaj način obezbeđuje se brza i fleksibilna pretraga odbranjenih radova, kao i pretraga tema u bilo kom statusu (slobodna, zauzeta ili odbijena). Na slici 1. prikazani su indeksi sa pripadajućim poljima.



Slika 1. *Indeksi servisa za detekciju plagijata*

Jedna pokrenuta instanca Elasticsearch-a naziva se čvor (eng. *node*). *Shard* predstavlja mesto gde se podaci čuvaju i odakle se upiti izvršavaju. S obzirom da je trenutno količina podataka koju ova aplikacija obrađuje jako mala, reda veličina od nekoliko desetina radova, broj čvorova podešen je na 1. Takođe, broj *shard*-ova po jednom indeksu postavljen je na 1.

Vremenom broj podataka koje aplikacija obrađuje postaće sve veći. Imajući u vidu da broj dokumenata koje *shard* može da sačuva zavisi od kapaciteta čvora, zaključuje se da gore pomenuti jedan čvor i jedan *shard* neće moći da opsluže takve količine podataka. Rešenje ovog problema

je *sharding* i on podrazumeva dodavanje novih *shard*-ova i čvorova.

5. IMPLEMENTACIJA

Dve ključne funkcionalnosti na koje se servis za detekciju plagijata oslanja su: indeksiranje i pretraživanje.

Prilikom dodavanja novog rada vrši se otpremanje dokumenta koji je prosleđen. Potom, potrebno je izvršiti indeksiranje pomenutog dokumenta, tako što se vrši mapiranje parametara na dokument koji se indeksira (eng. *indexing unit*). Na listingu 1. prikazan je isečak koda koji prikazuje način na koji je implementiran *indexing unit*

```
@Document(indexName =
ThesisIndexUnit.INDEX_NAME,
replicas = 0)
public class ThesisIndexUnit {

	public static final String
INDEX_NAME = "thesislibrary";

	@Field(type =
	FieldType.Text, analyzer =
	"serbian", store = true)
	private String text;
	@Field(type =
FieldType.Text, store = true)
	private String title;
	@Field(type =
FieldType.Text, store = true)
	private String studentsName;
	@Field(type =
FieldType.Text, store = true)
	private String mentorsName;
	@Id
	@Field(type =
FieldType.Keyword, store = true)
	private String filename;
	@Field(type =
FieldType.Text, store = true)
	private ThesisStatus
thesisStatus;
	@Field(type =
FieldType.Text, index = false,
store = true)
	private String description;
	@Field(type =
FieldType.Keyword, store = true)
	private String topicId;
	...
}
```

Listing1. *Klasa ThesisIndexUnit*

koji se odnosi na odbranjene radove.

Prilikom kreiranja *indexing unit*-a, obavezno je definisati ime indeksa. Podrazumevani broj *shard*-ova i replika je 1. U ovom slučaju, zbog jednostavnosti projekta, broj replika postavljen je na 0.

Analizator koji je korišćen za indeksiranje i pretragu je *SerbianAnalyzer*. Koristi standardni tokenizator (eng. *standard tokenizer*) kako bi formirao tokene. Nakon tokenizacije, tokeni prolaze kroz sledećih pet filtera:

- *LowerCaseFilter* – transformiše znakove iz tokena u mala slova,
- *LatCyrFilter* – pretvara reči napisane malim slovima iz ćirilice u latinicu,
- *StopFilter* – uklanja stop reči/tokene,
- *SnowballFilter* – vrši stemovanje tokena, uzimajući koren reči i
- *RemoveAccentsFilter* – zamenjuje 'Dj' u 'D' [24].

Nakon što je *indexing unit* kreiran, potrebno je izvršiti njegov upis u *Elasticsearch*. *ElasticsearchRepository* proširuje *Spring* interfejs za repozitorijume tako da se može koristiti kao jedno od njih. Sve što je potrebno uraditi da bismo mogli da indeksiramo objekte u *Elasticsearch* je da na klasu koja se indeksira dodamo anotaciju *@Document* i da kreiramo interfejs repozitorijuma koji proširuje *ElasticsearchRepository*.

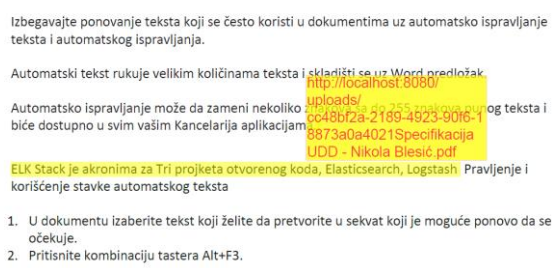
Algoritam provere plagijata, zasnovan je na pretraživanju prethodno indeksiranog sadržaja i podrazumeva sledeće korake:

- Na osnovu prosleđenog upita (napravljenog od dela teksta – maks. 500 termova), izvršiti pretragu u *Elasticsearch*-u i sačuvati dobijene rezultate.
- Prethodno dobijeni rezultati predstavljaju kolekciju potencijalnih izvora plagijata, odnosno delova teksta iz drugih dokumenata odakle je plagijat nastao. Potrebno je izvršiti filtriranje, te iz date kolekcije izdvojiti samo one elemente čija dužina je veća od unapred definisane dužine.
- U ovom trenutku imamo kolekciju delova dokumenata za koje se pretpostavlja da su izvor plagijata i čija dužina ne prelazi definisani minimum. Za svaki od pronađenih izvora potrebno je pronaći plagijat (sadržan u dokumentu koji se ispituje) na koji se dati izvor odnosi. Ovo se izvodi tako što se od svakog pronađenog izvora plagijata kreira upit i vrši se pretraga nad dokumentom koji se ispituje.
- Plagijati, zajedno sa izvorom odakle su nastali, čuvaju se u listu i u daljem procesiranju koriste za obeležavanje plagijata i prikaz rezultata provere plagijata.

Nakon završene provere rada na plagijate, ukoliko oni postoje, potrebno je napraviti izveštaj o njihovom prisustvu. Izveštaj predstavlja PDF dokument koji rad, za koji je vršena provera, sadrži u celosti, pri čemu delove teksta za koje je utvrđeno da su plagijati markira i u komentaru vezanom za markirani deo sadržaja navodi izvor plagijata.

6. PRIKAZ REZULTATA

Kada pristigne zahtev za proveru rada na plagijate, rad se preuzima i servis za detekciju plagijata vrši proveru. U slučaju da je plagijat detektovan, vrši se njegovo obeležavanje i pridružuje se komentar čiji sadržaj predstavlja listu referenci ka dokumentima u odnosu na koje je utvrđeno podudaranje. Na slici 2. prikazan je isečak PDF dokumenta, u kojem je pronađen i markiran potencijalni plagijat.



Slika 2. *Potencijalni plagijat*

6. ZAKLJUČAK

U ovom radu predstavljena je implementacija servisa za detekciju plagijata baziranog na tehnikama za pronalaženje informacija. Objašenjeni su pojmovi koji su neophodni za razumevanje funkcionisanja servisa. Fokus datih teorijskih osnova je razumevanje samog pojma plagijata, kao i mogućih načina na osnovu kojih može biti otkriven. Da bi se pokazala stvarna primena servisa za detekciju plagijata, izvršena je integracija pomenutog servisa sa veb aplikacijom koja je predstavljena u okviru diplomskog rada „Višeslojna veb aplikacija za prijavu i evidenciju diplomskih radova“. U radu su navedeni slučajevi korišćenja za koje potrebno izvršiti izmene, kako bi integacija bila uspešna.

Namena ovog servisa je da na brz i lak način uoči potencijalne plagijate u radovima studenata. Takođe, sistem pored pronađenih potencijalnih plagijata, pronalazi i delove teksta za koje je potrebno proveriti da li su dobro referencirani.

Ono što bi u velikoj meri olakšalo proces pregledanja rada od strane mentora, je automatska provera referenci. Zatim, u trenutnom rešenju, za svaki rad vrši se detekcija plagijata nezavisno u odnosu na prethodne verzije. U tom smislu, postoji prostor za kreiranje mehanizma koji bi prepoznao potencijalne plagijate iz prethodnih radova, a za koje je zaključeno da nisu plagijati i njih ne bi markirao. Takođe, trebalo bi uzeti u obzir da sistem može da greši i da tekst koji nije plagijat proglasi plagijatom i na taj način zaustavi dalji postupak odbrane rada. Dalji razvoj podrazumevao bi nadogradnju predstavljenog rešenja u cilju prevazilaženja navedenih nedostataka.

7. LITERATURA

[1] Kara Ronai, „Plagiarism Defined? A multiple case study analysis of institutional definitions.“, 2020.
[2] Ugo Chidera Chinedu, Dr. Charles Ikerionwu, Engr. Obi Nwokonkwo, „Plagiarism Detection Systems“ , 2021.
[3] Plagiarism, url: https://www.ox.ac.uk/students/academic/guidance/skills/plagiarism [Poslednji pristup: 30.1.2023.]
[4] El Mostafa Hambi, Faouzia Benabbou, „A systems for Ideas Plagiarism Detection: State of art and proposed approach“, 2022.
[5] Content similarity detection, url: https://en.wikipedia.org/wiki/Content_similarity_detection [Poslednji pristup: 30.1.2023.]
[6] Dragan Ivanović, Branko Milosavljević, „Upravljanje digitalnim dokumentima“, 2015.
[7] Haoda Huang, Benyu Zhang, „Text indexing and Retrieval. Encyclopedia of Database Systems“, 2009.
[8] Java (programming language), url: https://en.wikipedia.org/wiki/Java_(programming_language) [Poslednji pristup: 30.1.2023.]
[9] Spring Framework, url: https://en.wikipedia.org/wiki/Spring_Framework [Poslednji pristup: 30.1.2023.]
[10] What is Elasticsearch?, url: https://www.elastic.co/what-is/elasticsearch [Poslednji pristup: 30.1.2023.]
[11] Elasticsearch, url: https://blog.imi.pmf.kg.ac.rs/elasticsearch/ [Poslednji pristup: 30.1.2023.]
[12] Nikola Blesić, „Višeslojna veb aplikacija za prijavu i evidenciju diplomskih radova“, 2021.

Kratka biografija:

Nikola Blesić rođen je u Subotici 1998. godine. 2017. Godine završio je Srednju tehničku školu „Ivan Sarić“ u Subotici, smer Elektrotehničar informacionih tehnologija. Iste godine, upisuje Fakultet Tehničkih Nauka u Novom Sadu, smer Računarstvo i automatika. Osnovne studije završava 2021. godine, odbranom diplomskog rada na temu „Višeslojna veb aplikacija za prijavu i evidenciju diplomskih radova“. Nakon završenih osnovnih studija upisuje master akademske studije na istom fakultetu i završava ih 2023. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/23BE19Stojanovic

ENERGETSKA KRIZA KAO POVOD DECENTRALIZACIJE ELEKTROENERGETSKOG SISTEMA

ENERGY CRISIS AS A MAIN REASON FOR POWER GRID DECENTRALISATION

Milan Stojanović, Vladan Krsman; *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su opisane posledice energetske krize i dat je predlog rešenja za veću energetsku nezavisnost. Detaljno su opisane tehnologije pametnih mreža, posebno mikromreže, tehnologija razmene energije između baterije električnog vozila i mreže, kao i pametne kuće i zgrade. Pomoću ovih tehnologija dat je predlog rešenja koji vodi ka većoj decentralizaciji elektroenergetskog sistema i ka većoj energetskoj nezavisnosti.*

Ključne reči: *Mikromreža, pametna kuća, pametna zgrada, stanice za punjenje električnih vozila*

Abstract – *This paper describes global energy crisis and gives a potential solution for more energy independence. By using smart grid technologies like microgrids, vehicle to grid technology and smart homes and buildings, we will provide a solution for more decentralised power grid.*

Keywords: *Microgrids, smart home, smart building, electric vehicle charging station*

1. UVOD

Povećanje ljudske populacije i rast životnog standarda neminovno dovode do globalnog rasta potrebe za električnom energijom, odnosno veće potrošnje električne energije. Električna energija je zastupljena u svim sferama savremenog društva i predstavlja nezaobilazan faktor u svim privrednim i društvenim delatnostima. Usled trenutne energetske krize, države koje nemaju prirodne izvore gasa i nafte rešenje traže u elektrifikaciji i energetskoj nezavisnosti, odnosno proizvodnju električne energije žele da prebace na obnovljive izvore. Taj prelazak jeste komplikovan, ali je neizbežan i usled trenutne energetske situacije je veoma ubrzan. Cilj države treba da bude što veća energetska nezavisnost i bezbednost, kako bi se energetske potrebe zemlje mogle zadovoljiti bez obzira na različite političke nestabilnosti u svetu.

Povećanje potrošnje električne energije može da ima negativne posledice na nivou distributivne mreže, kao što su preopterećenje elemenata mreže, veći troškovi energije, veći padovi napona i lošiji kvalitet napona.

Rešenje postojećih problema potrebno je tražiti u konceptu pametnih mreža, koje predstavljaju najveći napredak u elektroenergetici još od prelaska sa jednosmerne na naizmeničnu električnu energiju.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladan Krsman, docent.

Prelazak sa tradicionalnih na pametne mreže predstavlja samo prvi korak ka daljem razvoju i u ovom radu će detaljno biti opisane tehnologije pametnih mreža, plan ka većoj energetskoj nezavisnosti i decentralizacija elektroenergetskog sistema. Cilj je da se pomoću primarno tehnologija mikromreža, pametnih kuća i zgrada i tehnologije razmene energije od vozila ka mreži, ali i drugih tehnologija pametnih mreža izgradi sistem koji je skoro u potpunosti nezavisan od glavne mreže i koji može samostalno da radi duži vremenski period.

2. ENERGETSKA KRIZA

Energetska kriza ili nestašica energije predstavlja značajno sputavanje u snabdevanju privrede energetskim resursima. Kroz istoriju je većina energetskih kriza bila uzrokovana ratovima, lokalnim nestašicama i manipulacijama na tržištu. U 20. i 21. veku više puta je dolazilo do ovakvih situacija, ali sa njima su se pojavljivala i rešenja za nastale probleme. Najveća energetska kriza do sad desila se sedamdesetih godina 20. veka, ali se smatra da će trenutna kriza biti većih razmera.

Trenutna energetska kriza počela je nakon pandemije Covid-19 2021. godine, usled izuzetno ubrzanog ekonomskog oporavka. Veći deo sveta se suočio sa nestašicom i povećanjem cena na tržištu nafte, prirodnog gasa i električne energije. Međutim, situacija se dodatno pogoršala i prerasla u potpunu globalnu energetsku krizu nakon ruske invazije na Ukrajinu u februaru 2022. godine. Cena prirodnog gasa dostigla je rekordne vrednosti, a kao rezultat toga porasla je i cena električne energije, nafte i uglja.

Usled trenutne situacije, ali i gledajući zadnjih 50 godina, može se primetiti da nestašica energije može da izazove ozbiljne ekonomske probleme. Jedan od načina da se ostvari energetska nezavisnost je svakako elektrifikacija, a to znači da za motorni, železnički, vazdušni i vodni saobraćaj, kao i za grejanje, umesto fosilnih goriva koristi se električna energija. Da bi se ostvarila energetska nezavisnost, ova električna energija mora biti produkt obnovljivih izvora poput sunčeve energije, vetra, hidroenergije, biomase, geotermalne toplote, talasa i dr. Elektrifikacija bi dovela do velikog povećanja potrošnje električne energije, što bi izazvalo probleme u prenosnoj, a posebno distributivnoj mreži. Upravo nedostatak poverenja u sigurnost elektroenergetskog sistema navodi na ideje o potencijalnoj decentralizaciji. Pomoću decentralizacije primarni benefit bi bio veća sigurnost i bezbednost mikrosistema, a implicitno bi se rasteretio elektroenergetski sistem.

3. PAMETNE MREŽE

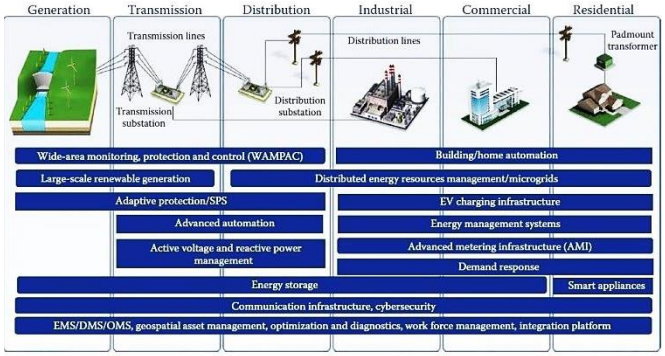
U poslednjoj deceniji se dešava novi svetski talas promena u elektroenergetskom sektoru koji je poznat pod nazivom pametne mreže (Smart Grids), a koji ima značajan uticaj na rad elektrodistributivnih sistema. Prema jednoj od definicija pametna distributivna mreža predstavlja arhitekturu budućnosti električne infrastrukture koja uključuje sve od tačke proizvodnje do tačke potrošnje. To uključuje kompleksnu mrežu tehnologija, sistema, hardvera, softvera, komunikacione infrastrukture, koja obezbeđuje visok nivo vidljivosti i kontrole sledećim učesnicima: proizvođačima, potrošačima, operatorima prenosne i distributivne mreže. Transformacija u pametnu mrežu je diktirana raznim faktorima društva, države, lokalnih propisa, lokalnih polisa i dostupnosti tehnologije.

Elektroenergetski sistemi budućnosti, a time i distributivni sistemi, će morati da omoguće efikasnu integraciju velikog broja interminentnih obnovljivih izvora, odnosno integraciju distribuiranih izvora različitih veličina i tehnologija. Takođe, moraće da omoguće promene u distributivnoj mreži koje će je prevesti iz domena pasivne mreže, zavisne od intervencija čoveka, u domen aktivne mreže. Ovo je neophodno zbog sve veće kompleksnosti operacija koje je potrebno sprovesti u mreži, široke primene distribuiranih izvora kao i povećanih zahteva za sigurnošću i kvalitetom napajanja [1].

Razlika između tradicionalnih i pametnih mreža je u tome što su tradicionalne mreže dominantno radijalne, one su realizovane bez distribuiranih izvora pa je tok snage jednosmeran. Kriva potrošnje i ponašanje potrošača su predvidivi, dok se za očitavanje struje koriste klasična daljinska brojila. U tradicionalnoj distributivnoj mreži broj prekidačkih uređaja je ograničen, ona je dominantno netelemetrisana izuzev kritičnih prekidačkih uređaja. Mreža je predvidiva u smislu režima i događaja, a sa njom se upravlja pomoću statičkih mapa.

Pametna distributivna mreža je radijalna sa sve više potencionalnih ili konstantnih upetljavanja, a zbog značajnog prisustva distribuiranih izvora, tok snage je dvosmeran. Potrošač je postao potrošač/proizvođač. Mreža postaje sve više telemetrisana i ugrađuje se sve više prekidačkih uređaja, inteligentnih uređaja i senzora. Pametna brojila zamenila su klasična brojila. Režim u pametnoj distributivnoj mreži je neizvestan usled prisustva stanica za punjenje električnih vozila i distribuiranih generatora, a potrošnja je drugačija usled mnoštva pametnih kuća i zgrada. Mrežom se upravlja daljinski pomoću online integrisanih informacionih sistema u realnom vremenu. Sedam glavnih karakteristika koje definišu funkcije pametne mreže su:

- Obezbeđuje potpune informacije o ceni električne energije u realnom vremenu.
- Visok nivo prisustva distribuiranih i obnovljivih izvora električne energije.
- Zreo, dobro integrisan market tj. tržište sa konstantnim rastom.
- Kvalitet električne energije je prioritet svima, ali se dozvoljava blaga optimizacija sa cenom električne energije.
- Visok nivo inteligentnih uređaja.
- Preventivno upravljanje poremećajima u mreži, minimizovanje štete i brza restauracija.
- Otporna na vremenske nepogode i hakerske napade, prediktivno upravljanje, izbegavanje i umanjivanje posledica, efikasna restauracija.



Slika 1. *Ključne tehnologije pametnih mreža [2]*

4. MIKROMREŽE

Kao osnovni koncept za decentralizaciju elektroenergetskog sistema izdvaja se tehnologija mikromreža. Pojam mikromreže se može definisati kao integrisani energetski sistem, koji se sastoji od međusobno povezanih distribuiranih energetskih resursa, potrošača i skladišta energije, koji zajedno mogu da rade u sklopu sa distributivnom mrežom tj. povezani na glavni sistem, ali i u autonomnom ostrvskom režimu.

Mikromreža predstavlja platformu za sjedinjavanje distribuirane proizvodnje, jedinica za skladištenje električne energije i potrošača koji se nalaze u lokalnoj distributivnoj mreži. Ona mora da bude osposobljena da radi i u nominalnom režimu, gde je povezana na glavnu mrežu, ali i u ostrvskom režimu, koji može da se izazove planski ili usled havarije. Ideja je da mikromreže u budućnosti veći deo vremena provode u povezanom režimu, iz koga bi se dobile najveće koristi samog koncepta mikromreža. Takođe, ona mora da zadovoljava i visoke zahteve po pitanju kapaciteta u slučaju povećanih zahteva za potrošnjom u ostrvskom pogonu. Pošto najveća korist koncepta mikromreža proizilazi iz povezanosti sa glavnom mrežom, predlog rešenja će se bazirati na tome, tako da se neće težiti ka potpunoj decentralizaciji odnosno autonomiji.

Razlika između mikromreže i pasivne mreže na koju su priključeni distribuirani izvori je u obimu regulacije i koordinacije koje poseduje mikromreža nad raspoloživim resursima. Takođe, mikromreže mogu ali ne moraju da budu sačinjene isključivo od uređaja električne energije, već one mogu da budu i više-energetske. Gas, toplotna energija ili vodovod mogu da se integrišu u mikromrežu i na taj način ona postaje još više nezavisna.

Zbog mogućnosti da funkcioniše zasebno, koncept mikromreže je sve popularniji, posebno zbog povećane potražnje za sigurnošću i stabilnošću sistema, ali i zbog lakše implementacije obnovljivih izvora.

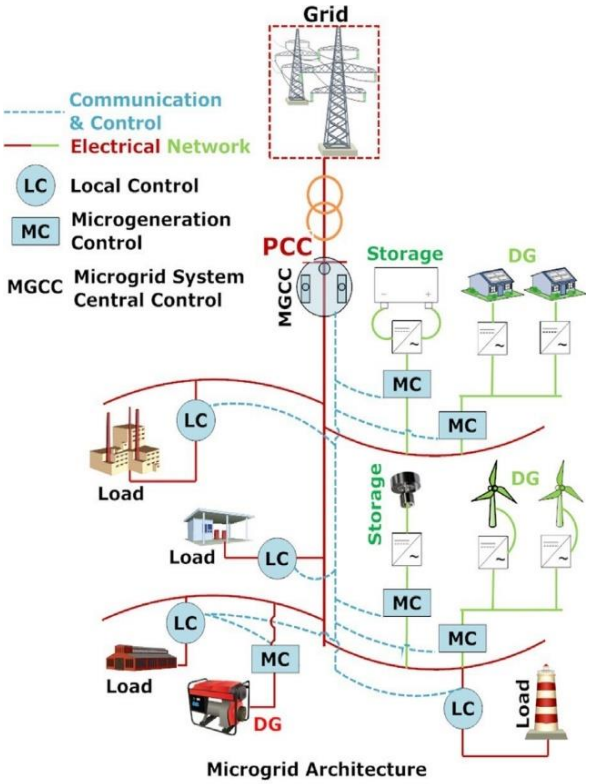
Karakteristike mikromreža su sledeće:

- Obezbeđuju dovoljnu i konstantnu električnu energiju za većinski deo potreba mikromreže.
- Ima svoju posebnu kontrolnu i optimizacionu strategiju.
- Mogu da pređu u ostrvski režim i da se ponovo povežu na glavnu mrežu veoma jednostavno,

preko tačke zajedničkog spajanja (Point of Common Coupling – PCC).

- Mogu da služe kao fleksibilna celina za optimizaciju mreže.
- Imaju kapacitet da skladište energiju.
- Primenljive su za različite naponske nivoe, obično manje od 20kV.

Tipičan primer mikromreže može da se vidi na slici 2.



Slika 2. *Tipičan primer mikromreže [3]*

Implementacija distribuiranih generatora u elektroenergetski sistem utiče na bilans snaga i učestanost u sistemu, pa upravljanje postaje veoma važno. Prednosti integracije distribuiranih generatora blizu potrošača je u smanjenju tokova snaga u distributivnim vodovima, što dovodi do smanjenja snaga gubitaka i poboljšanja kvaliteta napajanja krajnjih potrošača.

Postojanje dvosmernog napajanja rasterećuje vodove u periodima visokih opterećenja i pruža mogućnost bržeg ponovnog uspostavljanja napajanja posle kvarova. Najbolji način za iskorišćenje potencijala distribuiranih izvora jeste da se delovi mreže sa distribuiranim generatorima i potrošačima u njihovoj neposrednoj blizini posmatraju kao jedna podcelina sistema koja ustvari čini mikromrežu. Usvajanje mikromreža kao obrasca za masovnu integraciju distribuiranih generatora će omogućiti rešavanje tehničkih problema na decentralizovan način, čime će se smanjiti potreba za kompleksnom centralnom koordinacijom.

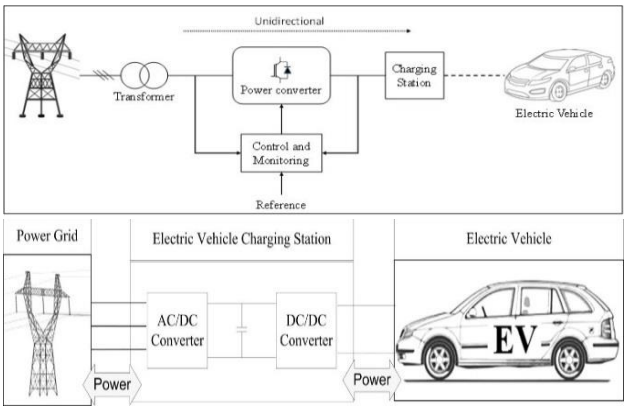
Očigledno je da koncept mikromreže može da donese brojne benefite elektroenergetskom sistemu, posebno kada se troši električna energija koja je proizvedena iz lokalnih obnovljivih izvora. Mikromreže mogu da utiču na smanjenje gubitaka električne energije, povećanje stabilnosti i sigurnosti mreže, kao i na smanjenje cene električne energije za potrošače i smanjenje troškova rada elektrodistributivnih preduzeća.

5. STANICE ZA PUNJENJE ELEKTRIČNIH VOZILA

Električna vozila predstavljaju alternativu saobraćajnog prevoza čija je glavna prednost nulta emisija štetnih gasova. Električna vozila za pogon koriste elektromotor i bateriju čime se postiže veća efikasnost i manji operativni troškovi u odnosu na konvencionalna vozila sa motorom sa unutrašnjim sagorevanjem. Usled brzog razvoja litijum-jonskih baterija i stanica za brzo punjenje, električna vozila postaju sve traženija, međutim industrija elektromobila se susreće i sa mnogim tehničkim ograničenjima poput visokih početnih cena, ograničenih mogućnosti punjenja, smanjeni domet vožnje i dugo vreme punjenja baterije. Takođe, konekcija velikog broja električnih vozila na mrežu može da izazove negativne uticaje na rad električne mreže.

Pojava koncepta pametnih mreža omogućila je modernizaciju elektroenergetskih sistema pre svega sa dodatnim komunikacionim karakteristikama. Koncepti mreža ka vozilu (grid to vehicle – G2V) i vozilo ka mreži (vehicle to grid – V2G) predstavljaju jednu od tehnologija pametnih mreža čiji je cilj poboljšanje rada elektroenergetskog sistema korišćenjem električnih vozila. Razmena energije između baterija električnih vozila i električne mreže doprinosi dodatne usluge mreži, ali i vlasnicima vozila.

Tehnologija razmene električne energije između baterija električnih vozila i mreže može da se kategoriše na dva načina: jednosmeran tok snage odnosno tok od mreže ka vozilu i bidirekcioni tok snage koji omogućava razmenu energije između baterije i mreže u oba pravca. Jednosmerni tok snage može da zaštiti mrežu od preopterećenja, pada napona i da utiče na stabilnost sistema. Iz perspektive mreže, baterija električnog vozila je pre svega potrošač, ali ukoliko bi postojala mogućnost dvosmernog toka snage, baterija bi se posmatrala i kao skladište električne energije. Bidirekcioni tokovi snaga između baterije i mreže omogućavaju dopunu baterije električnog vozila, ali i podršku elektroenergetskom sistemu.



Slika 3. *Jednosmerna i dvosmerna razmena energije [4]*

6. PAMETNE KUĆE I ZGRADE

Pametne kuće svojim stanarima pružaju udoban, potpuno kontrolisan i siguran način života. Štaviše, pametne kuće mogu da uštede energiju i novac uz mogućnost ostvarivanja profita od prodaje „čiste“ energije iz obnovljivih izvora. Sa druge strane, mogućnost smanjenja potrošnje u domaćinstvima podstiče mnoge vlade da podrže koncept i tehnologiju pametnih kuća i zgrada.

Da bi se kontrolisalo unutrašnjim ambijentom kuće, pametna kuća je automatizovana tako što ima mogućnost kontrole nekih uređaja, kao što su uređaji za osvetljenje ili grejanje. Savremeno upravljanje ima kontrolu i nad ostalim uređajima, ono ima mogućnost da prati unutrašnje okruženje i aktivnosti stanara.

Pametne kuće se mogu definisati kao stambene zgrade koje koriste različite komunikacione šeme i algoritme optimizacije za predviđanje, analizu, optimizaciju i upravljanje potrošnjom energije tako da maksimalno povećaju beneficije za kuću bez velike promene stila i udobnosti života [6].

Kao i slučaju pametnih kuća, u pametnim zgradama se koriste inteligentni uređaji i komunikacioni sistemi kako bi se prikupile informacije o događajima u zgradi, što će omogućiti optimizaciju preformansi same zgrade.

Razlika između pametnih zgrada i pametnih kuća pre svega je u veličini i broju stanara, ali je sam koncept pametnog objekta isti.

7. PREDLOG REŠENJA I ZAKLJUČAK

Usled trenutne energetske krize sve više potrošača gubi poverenje u sigurnost elektroenergetskog sistema i traže rešenje u decentralizaciji. Primeri decentralizacije će biti obrađeni na stambenom kompleksu, industrijskom postrojenju i vojnoj bazi.

Stambeni kompleks može da se izgradi tako da formira mikromrežu. To bi zahtevalo vezu sa glavnom mrežom preko tačke zajedničkog spajanja (PCC) preko koje bi mikromreža dobijala električnu energiju u normalnom pogonu. Integracija obnovljivih izvora i skladišta električne energije su neophodni. Kako solarni paneli predstavljaju najpristupačniji izvor obnovljive energije u gradu, zgrade ovog kompleksa bi na svojim krovovima imale instalisane solarne elektrane, ali bi i implementacija mikro vetroturbina bila moguća. Da bi energija dobijena iz obnovljivih izvora bila korisna, potrebno je postaviti i skladišta za električnu energiju u vidu baterija ili gorivnih ćelija. Pošto je integracija električnih vozila u mrežu sve veća, ovaj stambeni kompleks mora da obezbedi veliki broj stanica za punjenje. Ukoliko bi broj stanica za dopunu bio isti sa brojem električnih vozila, sva vozila bi mogla da učestvuju u razmeni električne energije između baterije i mreže.

Veliki broj električnih vozila bi mogao da preoptereti mrežu, ali ukoliko bi se pravilno upravljalo energijom, baterije električnih vozila bi mogle da posluže kao dodatno skladište za električnu energiju. Još jedan način smanjenja potrošnje je implementacija pametnih zgrada, pomoću kojih bi se povećala energetska efikasnost i koje bi omogućile upravljanje opterećenjem odnosno demand response program. Na taj način bi se mogla održavati ravnoteža između proizvodnje i potrošnje. Pošto je cilj decentralizacija, uz pravilno upravljanje ovakva mikromreža bi mogla da funkcioniše u ostrvskom režimu duži vremenski period. Električna energija bi se proizvodila iz obnovljivih izvora i uz pomoć skladišta energije i baterija električnih vozila ona bi zadovoljavala potrebe potrošnje.

Industrijska postrojenja takođe mogu da se transformišu u mikromreže, ovakva postrojenja obični imaju svoj izvod iz prenosne mreže preko kojeg dobijaju električnu energiju. Kako su ovakva industrijska postrojenja uglavnom izolovana i prostiru se na velikim površinama, osim fotonaponskih panela mogli bi se implementirati i vetrogeneratori, a ako na teritoriji postoji izvor ili reka, može se konstruisati i mikro hidroelektrana. Baterije, zamajci i gorivne ćelije su neophodne za normalno funkcionisanje ovakve mikromreže tokom ostrvskog režima. Upravljanje odzivom opterećenja takođe može da se primeni, ali kako je to industrijski kompleks, većina mašina mora da bude u pogonu i njihovo gašenje nije opcija, tako da se može upravljati samo opterećenjima koja nisu ključna za nastavak rada postrojenja. Kako industrijsko postrojenje ima jednog vlasnika, upravljanje mikromrežom bi bilo centralizovano, jer korisnici mikromreže imaju zajedničke ciljeve tj. Zajedničko radno okruženje pa teže koordinaciji zarad ostvarivanja sopstvenih ciljeva.

Vojne baze zahtevaju visok stepen sigurnosti i bezbednosti, a kako je jedna od najvećih mana pametnih mreža mogućnost hakerskih napada i ispada celog sistema, vojska je veoma zainteresovana za decentralizaciju. Ispad celog sistema može da izazove velike ekonomske gubitke, ali u vojsci može da izazove i gubitke života. Uglavnom sve vojne baze poseduju dizel generatore koji mogu da obezbede proizvodnju električne energije, ali da bi se obezbedio ostrvski rad na duže vremenske periode to nije dovoljno, tako da je implementacija solarnih i vetro elektrana neophodna. Zgrade vojnih baza bi bile modernizovane pomoću pametnih brojila, senzora, solarnih panela i baterija za skladištenje električne energije. Na taj način bi objekti u vojnim bazama bili više energetski efikasni, trošili bi manje resursa i postojala bi mogućnost upravljanja odzivom opterećenja.

8. LITERATURA

[1] Koncept pametnih mreža (Smart Grids) u elektrodistributivnom sistemu, Ž.N. Popović, B.B. Radmilović, V.M. Gačić

[2] Smart Grids: Advanced Technologies and Solutions Second Edition, Stuart Borlase, 2018.

[3] Microgrid: Architecture, policy and future trends, Lubna Mariam, Malabika Basu, Michael F. Conlon

[4] Vehicle to Grid Technology, Priya Gupta, Devangee Bhurawalla, Pooja Shah, Prof. Bhushan Save

[5] Smart homes:potentials and challenges, Rasha El-Azab

Kratka biografija:

Milan Stojanović rođen je u Novom Sadu 1997. godine. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektroenergetski sistemi odbranio je 2021. godine.

Vladan Krsman rođen je u Sarajevu 1985. godine. Doktorsku disertaciju odbranio je 2017. godine na Fakultetu tehničkih nauka iz oblasti Elektroenergetski sistemi.

Зборник радова Факултета Техничких наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/23BE20Karadzic

ТРОСЛОЈНА АРХИТЕКТУРА КАО СОФТВЕРСКО РЕШЕЊЕ СИСТЕМА ЗА ЕВИДЕНЦИЈУ СТУДЕНАТА

THREE-TIER ARCHITECTURE AS A SOFTWARE SOLUTION FOR A STUDENT RECORDS SYSTEM

Игор Караџић, *Факултет Техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом чланку је описана имплементација и рад евиденције студената на одређеним предметима током школске године. Реч је о трослојном систему који укључује имплементацију сервера и клијента, као и креирање и управљање базом података. Пројекат представља приступ развијања web апликација уз технологије и програмска окружења као што су Node, Javascript i MongoDB. Подржано је додавање студената и предмета, додељивање студената да у задатој школској години слушају одређени предмет, дефинисање категорија за дати предмет, унос поена за датог студента на датом предмету у одговарајућим категоријама.*

Кључне речи: *Web, Node, Javascript, MongoDB*

Abstract – *This article describes implementation and functionality of student records on specific subjects during school year. It is about a three-layer system that includes the implementation of a server and a client, as well as the creation and management of a database. Project presents the approach to development of web application including technologies such as Node, Javascript and MongoDB. It supports creating students and subjects, assigning students to specific subjects in a given school year, defining categories for a given subject, assigning points to a given student on a given subject in specific categories.*

Keywords: *Web, Node, Javascript, MongoDB*

1. УВОД

Путем сајта за евиденцију студената корисник је у могућности да прегледа све студенте и предмете активне у систему, да провери за сваког студента које предмете слуша, да прегледа за сваки предмет које категорије има као и колико је поена студент освојио на одређеним категоријама на задатом предмету. У оквиру овог пројекта реализован је систем који се састоји од два основна блока. Први блок представља, серверску страну, колоквијално названу *Back-end*, која укључује серверску апликацију базирану на *Node.js* [1] платформи и њој специфичном *JavaScript* програмском језику, и базу података, која је у овом случају *MongoDB* [2]. Други блок обухвата клијентску страну, илити *Front-end* који садржи *HTML* и *CSS* за креирање и дизајнирање *web* страница као и *Javascript* језик за динамичко управљање садржајем *web* странице.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Предраг Теодоровић, доцент.

Такође је реализована *Android* апликација која је направљена коришћењем *React Native*.

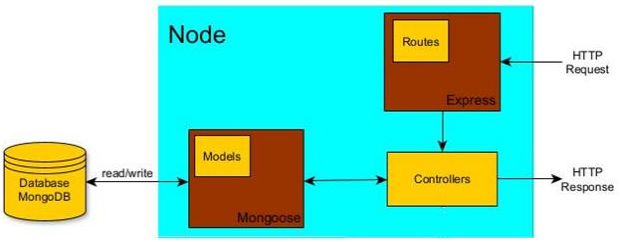
2. СИСТЕМ

У овом поглављу наведени су кључни аспекти и модули система и описани са теоријским осврtом.

2.1. Серверска страна

Сервер је сачињен од следећих ентитета:

- *Node.js* - за покретање сервера
- *MongoDB - за управљање базом података*
- *Express.js - за рутирање*
- *Контролера - за обраду захтева*



Слика 1. *Блок шема серверске стране*

Node.js

Node.js представља *JavaScript* платформу базирану на *Google V8 Engine*-у, која пружа једноставан и брз модел програмирања, а уз то подржава и асинхрони рад *web* апликације. Захтеве обрађује у само једном процесу, без потребе за креирањем додатне нити (енг. *Thread*) услед нових захтева. Захваљујући томе омогућен је бржи процес јер не постоји потреба за чекањем извршења захтева, већ апликација наставља даље са извршавањем наредне линије кода.

Уз *Node* долази и *Node Package Manager - npm*. У питању је алат који је задужен за добављање екстерних *Node.js* пакета, као и за складиштење и проналажење истих. Инсталација нових пакета обавља се помоћу *npm* команди командне линије (терминала), чиме се пакети аутоматски преузимају и додају у листу зависности (енг. *Dependency*) у датотеку *package* са *JSON* екстензијом. Пакети који су коришћени на *back-end*-у су:

- *Mongoose* - за рад са *MongoDB*
- *Dotenv* - за учитавање променљивих из *env* датотеке у процесу
- *Express* - за рутирање сервера и обраду *HTTP* захтева

MongoDB

Node поседује пакет (библиотеку) која омогућава приступање бази и извршавање различитих упита. Реч је о пакету који се назива *Mongoose* и пакету који поседује веома једноставан интерфејс за моделирање података објеката (енг. *Object Data Modeling - ODM*). Користи се као спрега између објеката у самом коду и њихову репрезентацију у бази. *MongoDB* представља нерелациону врсту базе података илити *NoSQL*, односно документациону базу, у оквиру које податке је могуће складиштити као *JSON* документе. Структура докумената може да варира, услед чега је смањена сложеност примене, а самим тим и бржи развој саме апликације.

Express.js

Express је *Node* пакет који представља уграђену библиотеку чије основно задужење јесте руковођење *HTTP* захтевима са клијентске стране. Такође, *Express* пружа комуникацију између сервера и базе података, где серверска страна тражи од модела извршавање операција од базе. Коришћене су *GET, POST, PUT* и *DELETE* методе.

Након што сервер пошаље захтев или добије исти, потребно је проследити одговор. Да би било лакше разазнати врсте захтева и упита, креирају се путање или руте (енг. *Routes*) које су креиране за одређену крајњу тачку (енг. *Endpoint*). Изворна (енг. *Root*) тачка у фази развоја јесте http://localhost:3000. Оваква путања могућа је само за машину у локалној мрежи, док би за остале уређаје у мрежи на рути уместо *localhost* кључне речи била прослеђена конкретна *IP* адреса уређаја на којем се сервер заправо извршава.

У продукцији би крајња тачка била назив *web* странице, која би била *host*-ована. Руте служе за прослеђивање захтева одговарајућим контролерима, који даље обављају сву потребну функционалност.

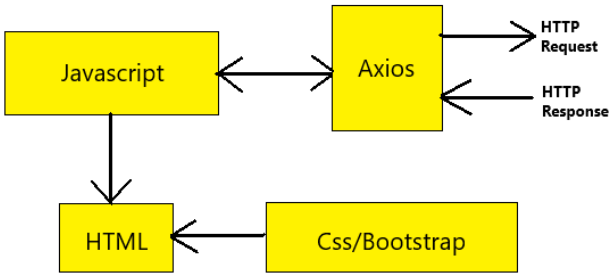
Kontroleri

Контролери функционишу тако што преко модела шаљу захтеве ка бази, креирају одређене упите у зависности од тога за шта су им потребни ти подаци и уколико испуњавају услове онда као одговор добијају жељене податке. Уколико би се узео пример додавања новог студента на *web* страници, први корак био би унос података (име, презиме и број индекса) на клијентској страни, након чега следи паковање унетих података у *HTTP* захтев ка рути на серверској страни http://localhost:3000/api/v1/students који сервер преузима и потом прослеђује одговарајућем контролеру.

2.2. Клијентска страна

За развој корисничког интерфејса коришћени су *HTML, CSS* и *Javascript*. Апликација је направљена тако да буде *single-page* што значи да ће се страница само једном учитати при иницијалном покретању а након тога се промене на страници динамички учитавају са сервера односно асинхроно без учитавања нове странице. Кључни ентитети на *front-end* делу су:

- *Javascript* – динамичко *понашање* садржаја *web* странице
- *HTML* – за креирање *web странице*
- *CSS* и *Bootstrap* – за изглед корисничког интерфејса
- *Axios* – за слање *HTTP захтева*



Слика 2. *Блок шема клијентске стране*

Javascript

Javascript је динамичан, слабо типизиран и интерпретиран програмски језик високог нивоа. Поред *HTML*-а и *CSS*-а, једна је од три водеће технологије за дефинисање садржаја на *web*-у; већина *web* сајтова користи *Javascript* а сви модерни *web* читачи га подржавају без потребе за инсталирањем додатака. Комбинован са *HTML*-ом и *CSS*-ом, *Javascript* чини *DHTML* (*Dynamic HTML*). Садржи *API* за рад са текстом, низовима, датумима и регуларним изразима, али не и улазно/излазне функционалности, као што су повезивање, складиштење података или графичке функционалности, за шта се ослања на окружење у коме се извршава.

Javascript се поред *web*-а користи у другим окружењима, као што су *PDF* документи, *web* читачи за специфичне *web* сајтове и десктоп виџети. Нове и знатно брже *Javascript* виртуелне машине и платформе засноване на њима, популаризовале су *Javascript* за израду *web* апликација на серверској страни. На клијентској страни, програмери најчешће имплементирају *Javascript* као интерпретирани језик, али све више новијих *web* читача обавља *just-in-time* компајлирање. *Javascript* се још користи и за развој видео игара, десктоп и мобилних апликација као и у мрежном програмирању на серверској страни са извршним окружењима као што је *Node.js*.

Javascript је најпопуларнији скриптни језик на Интернету којег подржавају сви познатији прегледачи (*Internet Explorer, Mozilla Firefox, Netscape, Opera, Safari*). Неке примене *JS*-а су:

- *Javascript* даје *HTML* дизајнерима алат за програмирање – *HTML аутори обично нису програмери, али Javascript је скрипти језик са веома једноставном синтаксом.*
- *Javascript може да динамички унесе код у HTML страну,*
- *Javascript може да реагује на догађаје – може да се подеси тако да се изврши кад се нешто деси, нпр. кад се страна учита или кад корисник кликне на HTML елемент,*
- *Javascript може да прочита или испише елементе – може да прочита и да промени садржај HTML елемената,*

- *Javascript може да се користи и за проверу исправности унетих података – може* да се користи за проверу исправности података унетих у форму, да провери исправност података пре него што се пошаљу серверу,
- *Javascript* може да се користи за детектовање *browser*-а корисника – у *зависности од browser-а, учитава се страна специјално дизајнирана за тај browser,*
- *Javascript може да се користи за креирање cookie*-ја – може да се користи за чување и враћање информација о рачунару посетиоца, итд.

HTML

HTML је описни језик специјално намењен опису *web* страница. Помоћу њега се једноставно могу одвојити елементи као што су наслови, параграфи, цитати и слично. Поред тога, у *HTML* стандард су уграђени елементи који детаљније описују сам документ као што су кратак опис документа, кључне речи, подаци о аутору и слично. Ови подаци су општепознати као мета подаци и јасно су одвојени од садржаја документа.

CSS

CSS је језик форматирања помоћу ког се дефинише изглед елемената *web* странице. Првобитно, *HTML* је служио да дефинише комплетни изглед, структуру и садржај *web* странице, али је од верзије 4.0 *HTML*-а уведен *CSS* који би дефинисао конкретан изглед, док је *HTML* остао у функцији дефинисања структуре и садржаја.

CSS синтакса се састоји од описа изгледа елемената у документу. Опис може да дефинише изглед више елемената, и више описа може да дефинише један елемент. На тај начин се описи слажу један преко другог да би дефинисали коначни изглед одређеног елемента (отуда назив *cascading* (енгл. *cascade* - цреп) да би се дочарало слагање једног стила преко другог у дефинисању коначног изгледа елемента).

Bootstrap

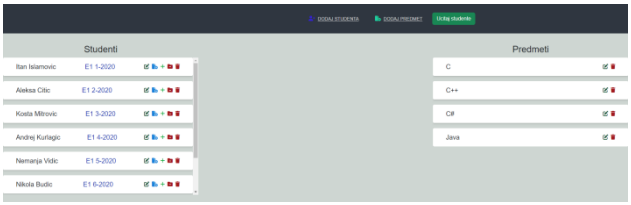
Bootstrap је *framework* који је задужен за естетику, тј. за изглед корисничког интерфејса. Представља колекцију *CSS* класа које су задужене за унапред дефинисан и стандардизован изглед *HTML* елемената. Поред лепог изгледа, пружа и додатне функционалности кроз уграђени *jQuery*, али једна од најбитнијих одлика јесте одзивност страница, односно скалиран и контролисан приказ елемената на различитим резолуцијама и величинама екрана.

3. ФУНКЦИОНАЛНОСТИ

У овом поглављу, урађен је осврт на неке од кључних функционалности *web* апликације.

Почетна страна

На почетној страници *web* апликације приказан је преглед свих студената и предмета. Наиме, приказани су основни подаци као што су име, презиме и број индекса студента за студенте и назив предмета за предмете. Кликом на име и презиме или број индекса студента отвара се прозор (модал) са детаљнијим информацијама о студенту: уколико нема предмета да слуша само су приказани име, презиме и број индекса студента а уколико има неки предмет да слуша онда је приказана школска година у којој слуша дати предмет као и освојени поени за одређене категорије за дати предмет. Кликом на назив предмета отвара се прозор (модал) са детаљнијим информацијама о предмету: уколико нема дефинисане категорије само је приказан назив предмета а уколико има дефинисане категорије онда су и оне приказане заједно са називом предмета. Такође, на овој страници обезбеђено је и додавање нових студената и предмета након чега је могуће њихово даље ажурирање, брисање, додељивање предмета студенту, додељивање категорија предмету као и додељивање поена студенту за категорије датог предмета.



Слика 3. *Изглед почетне странице*

Додавање новог студента

За додавање студента, мора се попунити поља за име, презиме и број индекса студента и тек тада се може додати. Уколико се то не уради од сервера се добија одговор о грешци да одговарајуће поље није попуњено или у случају броја индекса може добити и грешку да није добро унет формат за број индекса. Након додавања новог студента исти ће се појавити у колони студенти и над њим се онда могу радити следеће функционалности: ажурирање, брисање, додељивање предмета студенту, уклањање додељених предмета као и унос за дате предмете.

Додавање новог предмета

Уколико би корисник желео да дода нови предмет, мора попунити поље за назив предмета и уколико жели може попунити поље за категорије које ће дати предмет имати. Уколико поље за назив предмета остави празно, од сервера ће добити одговор о грешци да поље није попуњено. Након додавања новог предмета исти ће се појавити у колони предмети и над њим се онда могу радити функционалности ажурирања и брисања.

Ажурирање студента

Ажурирање информација о студенту постиже се притиском дугмета за ажурирање које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити потребне измене. Ажурирање студента подразумева измену података као што су име, презиме и број индекса. Приликом ажурирања било ког од датих поља важи исто правило као и код додавања новог студента, да сва поља морају бити попуњена. За ажурирање броја индекса важи још једно правило а то је да то поље мора бити написано у правилном формату, у супротном ће сервер вратити грешку о неправилном формату и неће ажурирати студента.

Ажурирање предмета

Уколико би корисник желео да ажурира неки предмет то може урадити тако што притисне дугме за ажурирање које стоји поред сваког предмета. Након што то

уради отвориће се прозор (модал) преко кога ће обавити потребне измене. Ажурирање предмета подразумева измену података као што су име предмета и категорије које ће дати предмет садржати. Приликом ажурирања поља за име предмета важи исто правило као и код додавања новог предмета, да то поље мора бити попуњено док поље за категорије може остати празно.

Брисање студента

Брисање студента ради се тако што се притисне дугме за брисање које стоји поред сваког студента. Након што то уради дати студент биће уклоњен из листе студената и листа ће бити ажурирана.

Брисање предмета

Уколико би корисник желео да обрише неки предмет то може урадити тако што притисне дугме за брисање које стоји поред сваког предмета. Након што то уради дати предмет биће уклоњен из листе предмета и листа ће бити ажурирана.

Додељивање предмета студенту у одређеној школској години

Додела предмета студенту извршава се тако што се притисне дугме за додељивање предмета студенту које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити дату функционалност. Додељивање предмета студенту подразумева следеће: студенту се мора доделити предмет као и школска година у којој би слушао дати предмет. Уколико се не попуни неко од поља, сервер ће вратити поруку о грешци да дато поље не сме бити празно. Додатно, уколико је попуњено поље за школску годину, сервер може вратити поруку о грешци да није добро унет формат за школску годину.

Уклањање додељеног предмета студенту

Уколико би корисник желео да уклони неки предмет који је додељен одређеном студенту то може урадити тако што притисне дугме за уклањање додељеног предмета које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити дату функционалност. Уклањање додељеног предмета студенту подразумева да се изабере неки предмет који би желели да уклонимо тако да га студент више не слуша.

Унос поена за датог студента на датом предмету у одговарајућим категоријама

Уколико би корисник желео да унесе поене за датог студента то може урадити тако што притисне дугме за унос поена које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити дату функционалност. Унос поена за датог студента подразумева следеће: прво се мора одабрати предмет за који би се желели унети поени, затим се могу унети поени за одређене категорије изабраног предмета.

Уколико се не одабере предмет, сервер ће вратити поруку о грешци да то поље не сме да буде празно. Поени нису обавезни да се унесу тј. у случају да се поље поени остави празно, то ће се протумачити као да је за сваку категорију датог предмета унето нула поена. У супротном се могу унети конкретни поени и потребно их је унети онолико колико има категорија изабраног предмета и ти поени треба да буду раздвојени зарезом.

Уношење више студената одједном путем JSON датотеке

Ова функционалност више служи као тест функционалност у случају да неко жели брже да тестира функционалност *web* сајта. Притиском на дугме учитај студенте, биће додато 8 студената уместо да се они ручно уносе.

4. ЗАКЉУЧАК

Може се закључити да је систем чији је циљ био примена клијент-сервер комуникације са практичном применом успешно реализован. Пројекат је укључио развијање *web* сајта са технологијама за развијање *single-page* апликације.

Предлози за побољшање:

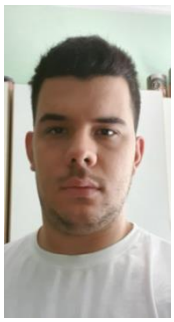
- Додавање опције за *претрагу студената и предмета у циљу њиховог лакшег налажења*
- *Додавање опције за сортирање студената и предмета*
- *Боља подршка за IOS кроз мобилну апликацију*
- *Прелазак са HTTP на HTTPS* трансфер протокол као стандардизовани гарант сигурности и безбедности интернет странице

5. ЛИТЕРАТУРА

[1] Shah, Dhruti Na. *Node .Js*. Bpb Publications, 2018.

[2] Chodorow, Kristina. *MongoDB*. O'Reilly, 2013.

Кратка биографија:



Игор Караџић рођен је Ужицу 1996. год. Дипломски рад на Факултету техничких наука из области Елктротехника и рачунарство – Примењено софтверско инжењерство одбранио је 2020. год. Област интересовања су му развој софтвера у програмским језицима *Java, Javascript* i *C#.*

контакт: igigotika@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE21Perisic

SISTEMI ZA GENERISANJE PROGRAMSKOG KODA

PROGRAM CODE GENERATION SYSTEMS

Ana Perišić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu prikazan je značaj i primena generisanja koda. Opisane su tehnike koje se koriste za generisanje koda. Navedeni su neki od alata za generisanje koda, kao i problemi koji su mogući u slučaju generisanja koda.*

Ključne reči: *generisanje koda, tehnike za generisanje koda, alati za generisanje koda, generatori koda*

Abstract – *This paper describes significance and usage of code generation. Code generation techniques are described as well as code generation tools. Also, some problems, that code generation causes, are listed and described in this paper.*

Keywords: *code generation, code generation techniques, code generation tools, code generators*

1. UVOD

U računarstvu, generisanje koda označava softverske tehnike ili sisteme koji generišu programski kod koji se zatim može koristiti nezavisno od sistema generatora u okruženju koje se izvršava. Osnovna četiri razloga za generisanje koda su: produktivnost, pojednostavljenje, prenosivost i doslednost [1]. Cilj ovog rada je opisati tehnike za generisanje koda, prikazati primenu generatora koda i navesti vrste alata za generisanje koda, kao i neke od problema pri generisanju koda.

U slučaju generisanja koda, generator se piše jednom i može se ponovo koristiti onoliko puta koliko je neophodno. Obezbeđivanje specifičnih ulaza u generator i njegovo pozivanje je znatno brže od ručnog pisanja koda, stoga generisanje koda omogućava uštedu vremena [1].

Sa generisanjem koda se generiše kod iz nekog apstraktnog opisa. Taj opis je obično lakše analizirati i proveriti u poređenju sa celim generisanim kodom [1].

Kada se dobije proces za generisanje koda za određeni jezik ili okvir (engl. *framework*), može se jednostavno promeniti generator i ciljati drugi jezik ili okvir. Takođe moguće je ciljati više platformi odjednom. Na primer, pomoću generatora parsera može se dobiti parser u C#, Javi i C++ [1]. Dakle, isti apstraktni opis se može koristiti za generisanje različitih vrsta artefakata.

Sa generisanjem koda uvek se dobija kod koji je očekivan. Generisani kod je dizajniran prema istim principima. Kod uvek radi onako kako se i očekuje, naravno ako se izuzme pojava grešaka u generatoru. Kvalitet koda je dosledan.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Dunja Vrbaški, docent.

2. TEHNIKE ZA GENERISANJE KODA

2.1. Šabloni i filtriranje

Ova tehnika opisuje najjednostavniji način za generisanje koda. Sav kod se već nalazi u šablonima, a filtriranje se koristi kako bi se odabrali delovi koda koji će biti uključeni na izlazu iz generatora [2]. Kod koji treba da se generiše nalazi se u šablonima. Promenljive u šablonima mogu biti vezane za vrednosti iz modela.

Generisanje pomoću šablona i filtriranja je prilično jednostavna tehnika, ali glavni nedostatak je kompleksnost stilova. Iz tog razloga, ovaj pristup je potpuno neprikladan za veće sisteme, posebno ako je specifikacija zasnovana na XMI [2].

2.2. Šabloni i metamodel

Da bi se rešio i izbegao problem direktnog generisanja koda iz XML modela, može se implementirati generator na više nivoa koji prvo analizira XML, zatim instancira metamodel, koji je prilagodljiv od strane korisnika, i na kraju ga koristi zajedno sa šablonima za generisanje. Ulaz se koristi za kreiranje metamodela, a potom se dati metamodel koristi zajedno sa šablonima kako bi se na osnovu njega generisao izlaz.

Prednost ovog pristupa je što se s jedne strane dobija veća nezavisnost od konkretne sintakse modela, na primer UML-a i njegovih različitih XMI verzija [2]. S druge strane, može se integrisati funkcionalnija logika za verifikaciju modela, ograničenja, u metamodel [2]. Za razliku od tehnike šablona i filtriranja, ovo se može implementirati u programskom jeziku, kao što je Java.

2.3. Procesiranje frejmova

Frejmovi su, u osnovi, specifikacije koda koje treba generisati. Kao i klase u objektno orijentisanim jezicima, frejmovi se mogu instancirati više puta. Frejm je sličan klasi u objektno orijentisanom programiranju, a instanca objektu date klase.

Tokom instanciranja, promenljive, koje se nazivaju slotovi, su vezane za konkretne vrednosti. Različito ponašanje istog frejma se postiže prosleđivanjem različitih parametara prilikom instanciranja. Svaka instanca može da poseduje sopstvene vrednosti za slotove, baš kao i objekti klasa.

Vrednosti koje su dodeljene slotovima mogu biti različite, od stringova do instanci frejmova. U toku izvršavanja, ovo rezultira strukturom stabla frejmova koja na kraju predstavlja strukturu programa koji treba da se generiše [2].

2.4. Generisanje zasnovano na API specifikaciji

Ubraja se u najpoznatiju tehniku generisanja koda. Kod generisanja zasnovanog na API (engl. *Application Programming Interface*) specifikaciji generisanje se vrši tako što se kreira ciljni model upotrebom namenskog API-ja [4]. Konceptualno se ovo generisanje zasniva na apstraktnoj sintaksi, odnosno metamodelu ciljnog jezika.

Korisniku je dostupan API koji je uprošćena verzija ciljnog jezika i sa njim najčešće deli apstraktnu sintaksu [4]. Klijent koristi API koji je baziran na apstraktnoj sintaksi ciljnog jezika i njime kreira elemente izlaznog programa. Elementi se kreiraju na nivou apstraktne sintakse. Stoga nije moguće kreirati sintaksno neispravan kod. Na primer, može da postoji API koji ima funkciju za kreiranje neke klase u izlaznom programu [4]. Toj kreiranoj klasi je moguće dodati metode, atribute i slično. Na kraju se apstraktno sintaksno stablo izlaznog fajla automatizovano pretvara u tekst, odnosno programski kod [4].

Nedostatak je što se često mora pisati ponavljajući kod koji se u nekim drugim tehnikama nalazi u sastavu šablona [2]. Takođe, mana je složenost i imperativni tok zadavanja izgleda izlaznog programa što može prouzrokovati nečitkost kod složenijih transformacija [4]. Ipak prednost ovog pristupa je što se radi na apstraktnom nivou i nije moguće kreirati sintaksno neispravne izlaze.

Međutim, problem sa ovom vrstom generatora je u tome što postoje potencijalno velike količine konstantnog koda, koda koji ne zavisi od modela, koji mora biti programiran umesto da se jednostavno kopira u šablone [2].

2.5. Generisanje koda in-line

Generisanje koda *in-line* se odnosi na slučaj u kojem izvorni kod sadrži konstrukcije koje generišu više izvornog koda ili mašinskog koda tokom kompajliranja ili neke vrste pretprocesiranja [2]. Predstavlja kombinaciju izvornog i generisanog koda. U izvornom kodu se nalaze sekcije koje kompajler ili interpreter koriste za generisanje dodatnog izvršnog koda. Primer bi bili šabloni u C++ programskom jeziku [2].

Ciljni jezik poseduje iskaze koji se u vreme kompajliranja zamenjuju drugim konstrukcijama istog jezika. Ova zamena se obavlja najčešće posebnim alatom koji se naziva pretprocesor. Pretprocesiranje može da se obavlja u više koraka, odnosno moguće je imati više pretprocesora koji će vršiti ekspanziju koda pre faze kompajliranja.

2.6. Generisanje koda atributima

Tehnika koja se često primenjuje u programskom jeziku Java. Postala je popularna zahvaljujući *JavaDoc* u Java programskom jeziku, gde su se specifični komentari koristili da bi omogućili automatsko generisanje HTML dokumentacije [2]. S obzirom na proširivost *JavaDoc* arhitekture, moguće je uključivanje prilagođenih oznaka (engl. *Custom tags*), kao i generatora koda.

Najpoznatiji primer predstavlja *XDoclet*, poznatiji kao XDOC [2]. *XDoclet* omogućava generisanje EJB interfejsa (engl. EJB *Remote/Local Interfaces)* i deskriptora (engl. *Deployment descriptors)*. Programer ručno piše klasu implementacije i dodaje neophodne *XDoclet* komentare u klasu, koje zatim čita *XDoclet* generator koda. Štaviše, generator ima pristup sintaksnom stablu izvornog koda, kome se dodaju komentari. Na ovaj način generator može da izvede informacije iz komentara kao i iz samog koda [2].

2.7. Generisanje zasnovano na aspektnom pristupu

Spada u tehniku koja opisuje spajanje odvojenih, ali sintaksno potpunih i nezavisnih delova koda. Stoga je neophodno definisati način na koji će se različiti delovi koda spojiti, pa se uvodi pojam tačke spajanja (engl. *join points, hooks*). *AspectJ* [ASPJ] je dobro poznati primer ove vrste generatora: regularni OO programski kod i kod aspekta su isprepleteni, bilo na nivou izvornog koda ili na nivou bajt koda [2]. Aspekti opisuju sveobuhvatne probleme, odnosno, funkcionalnost koja se ne može adekvatno opisati i lokalizovati korišćenjem dostupnih konstrukcija objektno orijentisanog programiranja [2].

3. PRIMENA GENERATORA KODA

U zavisnosti od konteksta, generatori koda su našli primenu u različitim oblastima, s obzirom da predstavljaju značajnu komponentu razvojnog procesa. Primer primene generatora koda je njegova upotreba u modernim razvojnim okruženjima, gde omogućavaju kreiranje kostura klase za implementaciju interfejsa, pritisnuvši svega jedno dugme [1]. Na taj način je moguća ušteda vremena onome ko razvija softver.

Postoji više različitih načina kako dizajnirati pajplajn generisanja koda (engl. *pipeline*) [1]. Prvobitno je neophodno definisati dva elementa, kao što su ulazi i izlazi. Ulazi predstavljaju mesto odakle dolaze informacije koje će se koristiti prilikom generisanja koda, dok izlazi predstavljaju način kako će se dobiti izgenerisan kod. Takođe, moguće je primeniti niz transformacionih koraka između ulaza i izlaza, što dovodi do pojednostavljenja izlaznog sloja i nezavisnosti ulaza od izlaza. Međutim, primena ovih transformacija predstavlja opcioni izbor.

U nastavku će biti opisani mogući ulazi [1]. Neki od njih su:

- *DSL jezici (engl. Domain Specific Language)* - moguće je koristiti alate kao što je ANTLR kako bi se opisala gramatika jezika, iz čega je posle moguće generisati parser.
- *Kod u različitim formatima* - šeme baza podataka iz kojih se generiše DAO, obrazac objekta pristupa podacima koji odvaja klijentski interfejs od mehanizma pristupa podacima.
- *Obrnuto inženjerstvo (engl. Reverse engineering)* - informacije se mogu dobiti obradom složenih artefakata koda.
- *Pomoćnik (engl.Wizard)* - oni dozvoljavaju zahtevanje informacije od korisnika.
- Izvori podataka kao što su: DB, CSV datoteka ili tabela.

Mogući izlazi su :

- *Obrađivači šablona* (engl. *Template engines*) - većina veb programera poznaje mehanizme šablona koji se koriste za *popunjavanje* HTML korisničkog interfejsa podacima.
- API-ji za pravljenje koda: na primer, Java parser se može koristiti za programsko kreiranje Java datoteka.

U nastavku će biti proučeni neki od pajplajnova, kao što su:

- *Parser generisanje* - odnosi se na generisanje parsera na osnovu zadate formalne gramatike. Ulaz može biti DSL, a izlaz se generiše upotrebom šablonskih mehanizama.
- *Model vođen dizajnom* (engl. *Model driven design*) - dodaci za razvojna okruženja (engl. *Plugins for IDEs*) ili sama razvojna okruženja, koja omogućavaju da se opiše model aplikacije. Često se koristi grafički interfejs i potom se iz *tog* modela generiše ili cela aplikacija ili kosturi klasa.
- *Metaprogramski jezici* - uključuju jezike koji omogućavaju skoro potpunu manipulaciju kodom programa, izvorni kod je samo *još* jedna struktura podataka kojom se može manipulisati.
- *Kod koji se odnosi na bazu podataka* - slično kao i kod modela vođenog dizajnom i šablonima. Obično programer definiše šemu baze podataka iz koje se mogu generisati čitave CRUD aplikacije ili samo kod za rukovanje bazom podataka. Postoje i alati koji obavljaju obrnuti proces: iz postojeće baze podataka kreiraju *šemu* baze podataka ili kod za rukovanje.
- *Ad-hok aplikacije* - ova kategorija uključuje sve: od alata dizajniranih za rukovanje jednom stvari do sistema koji se koriste u poslovnom okruženju, koji mogu generisati čitave aplikacije iz formalnog prilagođenog opisa. Ove aplikacije su uglavnom deo specifičnog toka posla. Na primer, korisnik koristi grafički interfejs da opiše aplikaciju i jedan ad-hok sistem generiše šemu baze podataka za podršku ovoj aplikaciji, drugi generiše CRUD interfejs i slično.
- *IDE generisani kod* - mnogi jezici zahtevaju mnogo šablonskog koda za pisanje i IDE obično može da generiše neke od njih: klase sa opisom za metode koje treba implementirati, *hashCode* i *toString* metode, *get* i *set* metode za sva postojeća svojstva i slično.

Generatori se takođe mogu koristiti za proizvodnju prototipova umesto produkcionog koda (engl. *production code*) [3]. Izrada prototipa se obično koristi za dobijanje ranih povratnih informacija, a generatori mogu proizvesti kod za potpuno drugačiju ciljnu platformu i na drugom programskom jeziku od finalnog koda. Ovde generatori ne moraju nužno da optimizuju kod, već su tu da bi omogućili funkcionalnost, upotrebljivost, izgled ili druge karakteristike relevantne za izradu prototipa [3]. Jezik modelovanja, međutim, može biti isti za razvoj i prototipa i proizvodnog koda. Model se takođe može koristiti za simulaciju. Generatori koda onda obezbeđuju potreban izlaz u simulatoru koda [3].

Generatori se mogu koristiti i za izradu dokumentacije [3]. Takođe je vredno napomenuti da se generisana dokumentacija ne odnosi samo na implementaciju već pokriva i rešenje opisano u domenu. Na kraju krajeva, u DSM (engl. *Domain Specific Model*) metodologiji softverskog inženjerstva, modeli uglavnom specificiraju domen problema, a ne rešenje.

U DSM metodologiji, generatori se takođe koriste za proveru konzistentnosti i kompletnosti dizajna. Ovo je neophodno jer obično nema smisla, ili čak nije moguće, staviti sva pravila u metamodel i proveriti ih tokom svake radnje modelovanja. Ovo je posebno tačno prilikom provere delimičnih modela, kada postoji više modela ili kada se integrišu modeli napravljeni od strane različitih programera [3].

Generatori za analizu modela se takođe mogu koristiti za vođenje rada na modelovanju i informisanju o potrebnim akcijama. Tipičan takav primer je da se pogleda da li je model ili su modeli nedovršeni i da se prijave moguće radnje koje su neophodne da bi model bio potpun. Takva provera modela se može pokrenuti slično kao kod generatora: kada je neophodno ili nakon sprovođenja određenih radnji modelovanja.

4. ALATI ZA GENERISANJE KODA

4.1. Primeri alata za generisanje koda

Obrađivači šablona (engl. *Template engine*) predstavljaju grupu alata koja se najčešće koristi. *Template engine* predstavlja mini kompajler koji je u mogućnosti da prevede jednostavne šablone za odgovarajući programski jezik [1]. Postoji fajl za šablon koji sadrži posebne oznake koje se mogu tumačiti upravo pomoću šablona.

Najjednostavnije što može da uradi jeste da zameni ovu specijalnu notaciju odgovarajućim podacima koje dobije tokom izvršavanja. Većina ovih alata omogućava upotrebu naredbi za kontrolu toka, kao što su: *for* petlje, *if-else* iskazi što korisniku dozvoljava opis jednostavnijih struktura.

Neki od predstavnika obrađivača šablona su:

- U Javi: FreeMarker, Velocity, JSP, StringTemplate, i slično.
- U Pajtonu: Jinja, Django template system, Cheetah, Mako, Genshi, i slično.

Parser generatori predstavljaju grupu alata za automatsko i brzo kreiranje parsera za jezik. Na osnovu formalne gramatike koju dobiju na ulazu generišu programski kod parsera koji će prepoznavati rečenice na datom jeziku i zatim pretvarati ulazne stringove u stabla parsiranja. Sem toga, mogu generisati i skenere, odnosno leksere. Često implementiraju mehanizam za obilazak stabla parsiranja i njegovu transformaciju. Primeri poznatijih parser generatora su: ANTLR, JavaCC, Bison i slično.

4.2. Problemi pri korišćenju alata za generisanje koda

Kada se koristi alat za generisanje koda tada kod postaje zavisan od njega. Zaključuje se da je neophodno održavati alate za generisanje koda, tako da je jedan od problema generisanog koda upravo održavanje generatora koda [1]. Neophodno je konstantno ažurirati postojeći kod alata za generisanje koda.

Još jedan od problema generisanog koda jeste njegova kompleksnost. Automatski generisani kod ima tendenciju da bude složeniji od ručno pisanog koda. Generisani kod je takođe sigurno manje optimizovan od onog koji je moguće ručno napisati. Ponekad je razlika mala i nije značajna, ali ako su aplikaciji značajne performanse, generisanje koda možda neće biti optimalno.

Postoje različita mišljenja o tome koje vrste koda se mogu generisati i koji je nivo kvaliteta tog koda [3]. Na primer, automatizacija za proizvodnju statičkih deklarativnih definicija u odnosu na uobičajeni dizajn, kao što su interfejsi ili šeme baza podataka, postoji već godinama, tako da je na raspolaganju više gotovih generatora [3]. Međutim, situacija se razlikuje kada je u pitanju generisanje ponašanja funkcionalnog koda. Često se zahtevaju opsežni i detaljni

UML modeli za specifikaciju funkcionalne strane, ali još uvek nisu adekvatni u detaljima za generisanje koda.

Mnogi programeri su imali loše iskustvo sa dostupnim generatorima (engl. *third-party generators*) jer je proizvođač generatora popravio način proizvodnje koda [3]. Uprkos postojanju više načina za pisanje koda za određeno ponašanje, proizvođač je izabrao samo jedno od njih. Taj izabrani način nije uvek zadovoljavajući i optimalan za neke postojeće specifične zahteve, uzimajući u obzir generisani ciljni jezik, korišćeni model, memoriju i slično. Generatori koda ne znaju dovoljno o specifičnim zahtevima organizacija da bi generisali idealan kod, tako da je isto tako to jedan od razloga zbog čega se smatra nezadovoljavajućim i nedovoljno korektnim. Pošto izmena generisanog koda obično nije optimalna opcija, organizacije odbacuju generisani kod. Vrednost generisanog koda je tada ograničena na izradu prototipa i fazu prikupljanja zahteva [3].

5. ZAKLJUČAK

U radu je prikazana primena i značaj generatora koda, kao i tehnike i alati koji se koriste za generisanje koda. Zaključuje se da generisanje koda obezbeđuje: produktivnost, pojednostavljenje, prenosivost i doslednost, što omogućuje primenu generatora koda u različitim oblastima.

Na primer, generatori koda zauzimaju značajno mesto u izradi dokumentacije, proizvodnji prototipova, u DSM metodologiji, gde proveravaju konzistentnost i kompletnost dizajna.

Takođe, postoje alati za generisanje koda, kao što su parser generatori koji služe da automatski i brzo kreiraju parser za određeni programski jezik.

Postoji još jedna grupa alata, kao što su obrađivači šablona koji omogućavaju prevođenje napisanih šablona od strane programera na željeni programski jezik.

Međutim, ukazano je i na potencijalne probleme sa kojim se moguće suočiti tokom korišćenja alata za generisanje koda. Zaključuje se da su to: kontinuirano održavanje i ažuriranje programskog koda alata za generisanje koda, kompleksnost generisanog koda, zavisnost koda i alata, kao i kvalitet generisanog koda, neki od problema koje generisanje koda može da prouzrokuje.

6. LITERATURA

[1] A Guide to Code Generation, https://tomassetti.me/code-generation/ (pristupljeno u oktobru 2022.)

[2] T. Stahl, M. Völter, J. Bettin, A. Haase, S. Helsen, *Model-Driven Software Development, Technology, Engineering, Management*, Chichester: John Wiley & Sons Ltd, 2006, pp 186-198.

[3] S. Kelly, J. P. Tolvanen, *Domain-Specific Modeling: Enabling Full Code Generation*, New Jersey: John Wiley & Sons Ltd , 2008, pp 79-86.

[4] I. Dejanović, *Jezici specifični za domen*, Novi Sad: Fakultet tehničkih nauka, 2021.

Kratka biografija:

Ana Perišić rođena je u Trebinju 1998. god. Fakultet tehničkih nauka u Novom Sadu, studijski program Računarstvo i automatika, upisala je 2017.god. Diplomirala je 2021.god., a potom upisala master akademske studije iz iste oblasti.

kontakt: anaperisic129@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE23Ivic

ЈЕДНА ИМПЛЕМЕНТАЦИЈА СИСТЕМА ЗА КОНТРОЛУ ПРИСТУПА МАКСИМАЛНЕ УПОТРЕБЉИВОСТИ

ONE IMPLEMENTATION OF ACCESS CONTROL SYSTEM WITH MAXIMUM USABILITY

Мирко Ивић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У овом раду представљен је развој једног сиситема за контролу приступа објектима у чијој су имплементацију коришћене различите технолохије. На почетку су дефинисани процеси и корисници система. Након тога описан је доменски модел система, а затим и имплеметација сваке компоненте система.*

Кључне речи: *Контрола приступа, Ардуино, RFID, NFC, Веб аплиакција,*

Abstract – *This papper presents the development of access control system for facilities, in which implementation various technologies were used. At the beginning, processes and users of the system are defined. After that, the domain model of the system is described, followed by the implementation of each component of the system.*

Keywords: *Access control, Arduino, RFID, NFC, , Web application, Web application*

1. УВОД

Контролисан приступ стамбеним и пословним објектима представља значајан фактор за безбедност и заштиту особа које бораве унутра, као и поверљивих докумената чије би компромитовање могло да угрози пословање. Систем, који је тема овог рада, представља решење за контролу приступа као и за контролу кретања унутар јендог предузећа. Осим запослених, у предузеће долазе разни посетиоци, којима треба омогућити ауторизован приступ објекту. Осим контроле приступа, систем има и могућност бележења колико времена запослени проведе на послу у току радног дана. На овај начин обезбеђује се максимална употребљивост и аутоматизација у раду, а смањује могућност грешке и злоупотребе. На брз и поуздан начин бележи се радно време запосленог, а степен безбедности унутар објекта се подиже на виши ниво.

Најпоузданији начин за идентификацију особа је свакако идентификација путем скенирања отиска прста или зенице ока. Међутим, ови подаци спадају у биометријске податке особе што захтева посебне дозволе од надлежних институција и посебан третман приликом чувања ових података 0. У овом случају, паметни телефони, које велики део популације користи, представља идеалан уређај. Запослени не може бити приморан да поседује паметни уређај. Због тога је за такве кориснике потребно имплементирати и алтернативни начин идентификације.

2. ОПИС СИСТЕМА

У систему се препознају три типа корисника, тј три улоге:

- администратор,
- запослени,
- посетилац.

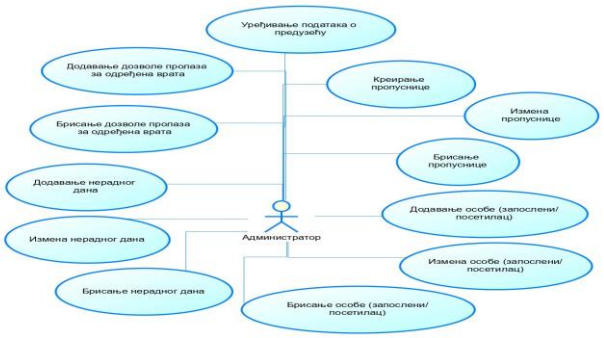
Улоге администратор и запослени имају директну интеракцију са системом, што значи да преко апликације имају могућност извршавања акција над ентитетима и мењања стања система. Улога посетилац непосредно интерагује са системом, тј. посетилац не користи софтвер директно и није свестан своје интеракције са системом.

Већина функционалности су у надлежности администратора, који има главну улогу и уређује систем према потребама предузећа. Задатак администратора је да одржава систем у конзистентном стању, тј. да систем у сваком моменту осликава реално стање ствари у предузећу.

На слици 1. и на слици 2., путем дијаграма случајева коришћења [2], приказане су функционалности које су доступне администратору.

Слика 1. *Функционалности доступне администратору (први део)*

НАПОЕМНА:
Овај рад проистекао је из мастер рада чији ментор је био др Драган Иветић, ред. проф.



Слика 2. *Функционалности доступне администратору (други део)*

Фокус система је да омогући запосленима ауторизован приступ предузећу и унутрашњим просторијама истог. Из тог разлога запослени има знатно мање доступних функционалности у односу на корисника типа администратор. Осим чекирања на вратима, запослени може да формира захтев за годишњи одмор и да преузима одређене документе. С обзиром на то да администратор ради у предузећу, њему су такође доступне све функционалности које има и запослени.

Као што је већ речено, корисник типа посетилац нема директне интеракције са системом. Једниа функционалост која је њему додељена је чекирање на вратима. Њему се издаје привремена пропусница како би имао приступ тачно одређеним просторијама унутар предузећа и на тај начин се спречио неовлашћени приступ просторијама у којима се налазе поверљиви документи или одвијају процеси који су део пословне тајне.

Систем има могућност генерисања следећих докумената:

- потврда о радном односу, тј. потврда о запослењу,
- листинг са именима свих запослених и њиховим подацима,
- листинг са именима свих запослених и укупним бројем радних сати са жељени временски период,
- листинг са датумом и временом откуцавања запосленог у току месеца и бројем сати проведених на радном месту за сваки дан.

Потврда о радном односу се генерише на основу захтева запосленог и може бити извезена (експортована) у *PDF (Portable Document Format)* формату. Потврда треба да садржи личне податке запосленог, податке о предузећу и намену за коју се потврда издаје. Листинг са именима свих запослених и њиховим подацима се генерише на захтев администратора и може бити извезен у *PDF* или *CSV (Comma-Separated Values)* формату. Пре самог генерисања администратор може да одреди по ком критеријуму ће листинг бити сортиран. Овај листинг поред имена и презимена запослених садржи и додатне податке о њима попут адресе, јмбг-а, занимања, позиције итд. Листинг са именима свих запослених и укупним бројем радних сати за жељени временски период генерише се на захтев администратора и може бити извезен у *PDF* или *CSV* формату. Администратор пре самог генерисања дефинише временски период који га интересује и критеријум по ком ће листинг бити сортиран. На документу се налазе датуми за изабрани период, подаци о запосленима и збир сати проведених на послу за сваког запосленог. Листинг са датумом и временом откуцавања запосленог и бројем радних сати за сваки дан у току месеца се генерише на захтев запосленог или администратора и може бити извезен у *PDF* или *CSV* формату. На листингу се налазе подаци о запосленом и назив месеца за који се лиситнг издаје. Осим тога, приказани су и појединачни подаци за сваки дан у месецу као што су време доласка и одласка са посла и разлика та два времена тј. колико је времена запослени провео на послу тог дана.

Да би систем осликавао реално стање у предузећу задужен је администратор. Путем апликативног софтвера администратор уноси податке о предузећу и о ентитетима унутар предузећа као што су: врата, нерадни дани, позиције, запослени, посетиоци итд.

Приликом запошљавања новог заполсеног администратор уноси његове податке у систем. Након тога запосленом се додељује једна или више пропусница које може да користи за приступ предузећу и кретање кроз њега. Запосленом се издаје перманентна пропусница, тј. пропусница без датума истека. Уколико запосленом престаје радни однос, тада се дефинише датум престанка валидности пропуснице. Доласком до одређених врата запослени користи своју пропусницу ради идентификације. Потом систем врши проверу да ли тај запослени има дозволу за пролаз. Уколико има, биће му омогућен пролаз кроз врата, у супротном биће му сигнализовано да нема приступ тим вратима. Такође, пропусница се издаје и за посетиоце. За разлику од пропуснице која се издаје запосленом, посетиоцу не може бити издата перманентна пропусница. Када су нека од врата дефинисана као улазно-излазна, тада се приликом дозволе пролаза, запосленом бележи датум и време проласка. На основу тих записа, остварује се могућност евиденције колико је времена запослени провео на свом радном месту.

Евиденцију коришћења годишњих одмора и боловања води администратор. Његов задатак је да бележи свако коришћење како би се на ефикасан начин водила евиденција колико дана одмора је запослени искористио у току године, тј. колико дана је провео на боловању. Запослени би требало да свој годишњи одмор најави унапред тако што поднесе захтев. Администратор врши одобравање захтева уколико одсуство запосленог за тај период не ремети унапред испланирану активност предузећа.

3. МОДЕЛ ПОДАТАКА

Централни ентитети овог система су *Preduzece i Osoba.* Све битне информације које описују један пословни субјекат попут назива, адресе, телефона, матичног броја се налазе у истоименом ентитету. За поља која репрезентују ПИБ и матични број предузећа потребно је увести ограничење јединствености тј. забрану дефинисања два ентитета предузећа са истим вредностима за ова два поља.

Нерадни дани у предузећу дефинисани су путем ентитета *NeradniDan.* Ентитет садржи поља датума нерадног дана и описа који објашњава зашто се тог дана није радило. Поред ова два поља постоји и поље *BrojSati* које, у предузећима где се зарада рачуна на основу броја радних сати, говори колико се радних сати приписује раднику тог дана. Врата на којима се врши контрола приступа репрезентована су путем ентитета *Vrata.* Једно од поља овог ентитета је и поље *Ulazna* типа *boolean.* На основу овог поља дефинише се да ли су одређена врата улазно-излазна тј. да ли се особи приликом проласка бележи време и датум. Ентитетом *Pozicija* дефинисане су позиције унутар једног предузећа.

Ентитет *Osoba* репрезентује људе који интерагују са системом. У овом ентитету налаза се поља која дефинишу једног човека унутар система као на пример име, презиме, адреса, број телефона итд. Енитети *Zapolseni* и *Posetilac* наслеђују ентитет *Osoba*. Како би се запослени могао пријављивати на систем истоименом ентитету додата су поља корисничко име и лозинка. Тренутно, ентитету *Posetilac* нису додата никаква додатна поља. Разлог имплементирања у три табеле је у томе што би у случају једне табеле, у којој су и запослени и посетиоци, дошло до усложњавања проблема. Тада би приликом уноса новог посетиоца требало изоставити унос података за поља која су везана само за запосленог, а нека од тих поља су неопходна у случају уноса запосленог. Ту долази до компликације приликом валидације уноса која је избегнута овако дизајнираним моделом. Свим особама у систему додељују се пропуснице како би се могли чекирати на вратима. Ентитет *Propusnica* садржи податке о пропусници попут датума издавања, датума истека и којег је типа. Датум истека дефинише се уколико је пропусница додељена посетиоцу или уколико се зна датум престанка радног односа запосленог. За сваку позицију унутар предузећа, ентитетом *PozicijaVrataPreduzece* унапред су предодређена врата за која та позиција има пролаз. То значи да је за додељивање дозволе пролаза особи, довољно да се приликом уноса особе у систем дефинише њена позиција. Дозволе пролаза, које се односе на неку особу и одређена врата, репрезентоване су путем ентитета *DozvolaProlaza.* На овај начин остварена је могућност прилогађавања приликом доделе пропусница, тј. особа може имати приступ још неким вратима, иако то није предвиђено у односу на позицији. Такође, приступ неким вратима може бити и одузет.

Евидентирање годишњег одмора имплементирано је коришћењем ентитета *GodisnjiOdmor* и *KoriscenjeOdmora.* Календарску годину током које запослени може да користи годишњи одмор описује ентитет *GodisnjiOdmor.* Овај ентитет садржи поље које даје информацију о календарској години, чија вредност мора бити јединствена за једног запосленог, и поље које показује колико је дана годишњег одмора запослени искористио у тој години.

Коришћење одмора у години евидентра се путем ентитета *KoriscenjeOdmora.* Најпре треба поменути да ентитет *KoriscenjeOdmora* садржи поље *StatusGodisnjiOdmor* које је типа *enum*, а могуће вредности су *Predlog* и *Prihvacen.* Ово поље сигнализира да ли је то изнети предлог запосленог у вези са годишњим одмором или годишњи који треба урачунати у укупан збир дана.

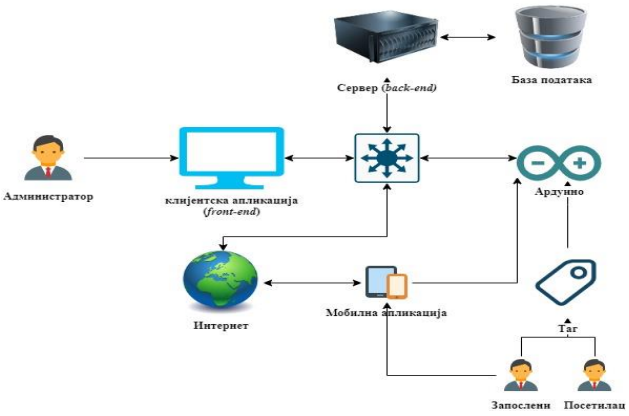
Осим овог поља, енитет *KoriscenjeOdmora* садржи податке о датуму почетка и датуму завршетка одмора и броју искоришћених дана. Приликом рачунања броја искоришћених дана не рачунају се дани који су у предузећу дефинисани као нерадни.

Имплементација евидентирања боловања решена је на сличан начин као евиденција годишњих одмора. Евиденција боловања садржи енитете *Bolovanje* и *KoriscenjeBolovanja.* Енитет *Bolovanje* описује календарску годину у којој запослени користи боловање.

Поље *Godina* представља годину коју описује ентитет и вредност овог поља мора бити јединствена за запосленог. Вредност поља *BrojDana* представља укупан број дана које је запослени провео на боловању током године. Појединачно коришћење боловања у години евидентира се помоћу ентитета *KoriscenjeBolovanja.* Овај ентитет садржи податке о датуму отварања, датуму затварања и броју дана проведених на боловању том приликом.

4. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

Систем је изграђен од неколико компоненти које међусобно интерагују. У наставку је описана свака појединачна компонента архитектуре система уз додатни опис технологија које су коришћене приликом имплементације. На слици 3. приказана је архитектура система.



Слика 3. *Архитектура система*

4.1 Серверска апликација

Серверска апликација заузима централно место у систему. Све остале компоненте комуницирају са њом или преко ње. За развијање серверске апликације коришћено је *IntelliJ IDEA* интегрисано радно окружење. Програмски код написан је у програмском језику *Java* верзије 17 и у радном оквиру *Spring Boot.*

Приликом дизајнирањаа дизајнирање комуникације између сервера са клијентом и ардуином коришћен је *REST* архитектонски стил [3], а сама комуникација се одвија путем *HTTP* протокола.

4.2 База података

Подаци који циркулишу унутар система чувају се у јединственој бази података. За потребе овог система коришћена је *SQL Server 2012* релациона база података, а комуникација сервера са базом података остварена је уз помоћ *Hibernate JPA (Java Persistance API)* радног оквира. База података генерисана је *Code-First* методом. То значи да су прво направљене класе модела, а потом на основу анотација које се додају пољима класе *Hibernate JPA* креира табеле унутар базе. Мапирање доменског модела на базу података врши се на следећи начин. На основу ентитета доменског модела формиране су објектне класе, а потом коришћењем функционалности *Hibernate JPA,* објектне класе су намапиране на табеле унутар базе података. Атрибути ентитета су мапирани на поља објектне класе, а они на колоне у табели. Сваки ентитет поседује атрибут *id,* који представља примарни кључ ентитета.

4.3 Клијентска апликација

Преко клијентске апликације администратор врши ажурирање података унутар система. Администратор има своје креденцијале помоћу којих се пријављује на систем. Приступ систему није дозвољен непријављеним корисницима. Клијентска апликација је развијана у *Visual Studio Code* интегрисаном радном окружењу, а имплементирана је коришћењем *Angular JavaScript* библиотеке верзије 15.0.2.

Приликом уноса података, администратор користи различите форме за унос. Приликом уноса потребно је обезбедити валидацију унетих података. За валидацију унетих података кориштени су *Strategy* шаблон и *Angular*-ове библиотеке за валидацију форме уноса. Код *Strategy* шаблона свака класа модела имплеметира интерфејс са методом *validate.* Унутар сваке класе модела, метода *validate* је редефинисана и имплементирана тако да се изврши валидација према захтевима за специфичну класу.

4.4 Ардуино платформа

Ардуино је платформа отвореног кода (енг. *open-source*) која омогућава реализацију разноврсних пројеката у домену електронике [4]. Једна од великих предности ове платформе јесте њена отвореност. То значи да су сви делови платформе отворени и доступни за проучавање и измене. Ардуино штит (енг. *Shield)* је хардверска компонента која се надограђује на врх микроконтролера и садржи склопове који су специјализовани за одређену намену. Ово значи да корисник, уколико жели да омогући контролеру приступ интернету или управљање електро мотором, не мора самостално да развија модул за ту функционалност. Уместо тога, довољно је да прикључи одговарајући штит контролеру и користи већ доступне библиотеке како би управљао том додатном функционалношћу.

Ардунио контролер сам по себи нема могућност приступа интернету, нити могућност детектовања *RFID* или *NFC* сигнала. За потребе система ардуино уређају придодати су интернет штит (енг. *Ethernet Shield) W5100* и *RFID* читач *RC522*. Захваљујући овим додацима, ардуино има приступ интернету и могућност читања *RFID* тагова. Пошто овај читач ради на фреквенцији од 13,56 MHz, то му омогућава детектовање и *NFC* сигнала.

4.5 Андроид апликација

Мобилна апликација комуницира са сервером преко интернета. Запослени користе апликацију ради преузимања докумената и увида у своје радне сате, а запослени чији паметни телефони имају *NFC* могу се чекирати на вратима и помоћу паметног телефона. У супротном, запослени користе *RFID* таг за чекирање. Приликом развијања мобилне апликације коршћено је *Android Studio* интегрисано развојно окруже, а апликација је рађена у андроид радном оквиру и писана у *Java* програмском језику.

5. ЗАКЉУЧАК

Тема овог рада јесте креирање система за контролу приступа објекту једног предузећа употребом различитих технологија ради добијања максималне ефикасности у раду. Представљен је систем који путем додељених пропусница препознаје особе на вратима, а затим на основу дефинисаних дозвола омогућава или забрањује даљи пролаз. Такође, једна од бенефиција овог система јесте аутоматско израчунавање броја радних часова које запослени порведе у предузећу.

Даљи правци развоја и унапређивања система може бити имплементација *HTTPS* (*Hypertext Transfer Protocol Secure)* протокола како би се осигурала сигурна комуникација како између клијента и сервера, тако и између андорид апликације и сервера. Ардуино платформа се показала врло захвалном за интеграцију унутар система. Доступност компоненти и кодова за учење знатно су олакшали имплемнтацију. Међутим, *RFID* читач *RC522* се није нарочито показао приликом употребе са *NFC*-ом. Иако читач ради на фреквенцији од 13,56 MHz, дешавало се да не успева да прочита сигнал који му је упућен од стране паметног телефона.

6. ЛИТЕРАТУРА

[1] Службени гласник РС, „Правно информациони систем“, https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html.

[2] UML, „UML Use Case Diagrams“, https://www.uml-diagrams.org/use-case-diagrams.html.

[3] R. Thomas Fielding, Architectural Styles and the Design of Network -Based Software Architectures, ProQuest Dissertations Publishing, 2000, pp.76-105.

[4] Arduino, „What is Arduino?“, https://docs.arduino.cc/learn/starting-guide/whats-arduino.

Кратка биографија:

Мирко Ивић рођен је у Новом Саду 1996. год. Основне академске студије завршио је 2020. год. на Факултету техничких наука у Новом Саду. Мастер рад на Факултету техничких наука из области Рачунарство и аутоматика одбранио је 2023.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4
DOI: https://doi.org/10.24867/23BE25Simic

PREDVIĐANjE POTROŠNjE ELEKTRIČNE ENERGIJE KORIŠĆENjEM LGBM ALGORITMA U ML.NET-U I PYTHON-U

ELECTRICITY CONSUMPTION PREDICTION USING LGBM ALGORITHM IN ML.NET AND PYTHON

Nemanja Simić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Predviđanjem potrošnje električne energije, elektrodistribucije mogu efikasnije da planiraju režim upotrebe generatorskih postrojenja, redovna održavanja elemenata mreže kao i potencijalnu trgovinu na marketu električne energije. Algoritmi mašinskog učenja mogu poslužiti kao alat za precizno predviđanje potrošnje u elektroenergetskim sistemima. Kroz ovaj rad implementirana su dva softverska rešenja za prognozu potrošnje električne energije na osnovu podataka vremenskih prilika koristeći LGBM algoritam u ML.NET-u i Pythonu, dok su rezultati predikcija opisani i analizirani.*

Ključne reči: *LGBM, ML.NET, Python, Mašinsko učenje, Algoritam*

Abstract – *By predicting electricity consumption, electricity distribution companies can more efficiently plan the mode of use of generating plants, regular maintenance of network elements, as well as potential trade on the electricity market. Machine learning algorithms can serve as a tool for accurate consumption forecasting in power systems. Through this work, two software solutions for forecasting electricity consumption based on weather data were implemented using the LGBM algorithm in ML.NET and Python, while the prediction results were described and analyzed.*

Keywords: *LGBM, ML.NET, Python, Machine Learning, Algorithm*

1. UVOD

Stabilan elektroenergetski sistem odlikuje, između ostalog, pouzdana proizvodnja električne energije i redovno održavanje postrojenja. Precizna prognoza potrošnje unapređuje proces planiranja upotrebe i održavanja proizvodnih postrojenja, kao i kupovine električne energije. Stoga, počeo je razvoj raznih modela mašinskog učenja koji predviđaju potrošnju električne energije u stambenim i poslovnim zgradama koristeći karakteristike kao što su podaci o vremenskoj prognozi ili istorijski podaci o računima za električnu energiju [1]. U radu [2], Kolter i Ferreira koriste mesečne račune električne energije i gasa, kao i karakteristike zgrade kako bi modelovali potrošnju. Raspostranjena upotreba mašinskog učenja dovela je do razvoja različitih tehnologija u ovom polju. U programskom jeziku *Python* napisane su popularne biblioteke za obradu i analizu podataka *Numpy* i *Pandas*, zajedno sa velikim brojem algoritama mašinskog učenja različitih namena, od kojih je većina deo *scikit-learn* paketa. Tehnološki gigant *Microsoft*, prateći trendove tržišta, C# programerima nudi mogućnost upotrebe ovakvih algoritama pomoću *ML.NET* paketa koji je razvijen u kao deo *.NET* radnog okvira-a.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Sebastijan Stoja, docent.

2. TEORIJSKE OSNOVE

2.1. Prognoza potrošnje u elektroenergetskim sistemima

Predviđanje potrošnje električne energije ima za cilj da obezbedi neprekidno snabdevanje i smanji operativne troškove preduzeća [3]. Predstavlja proračun koji pruža kompanijama jasniju sliku sistema u budućnosti i daje indikatore o potrebnim intervencijama. Poznavanje stanja mreže u određenom trenutku u budućnosti pruža uvid u potencijalne probleme sistema i olakšava donošenje biznis odluka [4]. Ovakve informacije su od velikog značaja prilikom kupovine i prodaje električne energije [5], kao i tokom planiranja radova i proširenja infrastrukture mreže [6]. Na osnovu vremenske skale, prognoza potrošnje električne energije se klasifikuje na:

- Veoma kratkoročno predviđanje potrošnje (*VSTLF*) koristi se za vremenski period od 10 do 30 minuta [7].
- Kratkoročno predviđanje potrošnje (*STLF*) koristi se za vremenski period od nekoliko sati, dana ili jedne nedelje [8].
- Srednjoročno predviđanje potrošnje (*MTLF*) koristi se za vremenski period od jedne nedelje do jedne godine [9].
- Dugoročno predviđanje potrošnje (*LTLF*) koristi se za vremenski period od jedne do 20 godina [10].

2.2. Mašinsko učenje

Arthur Samuel, još 1959. godine, mašinsko učenje definiše kao oblast studija koja računarima daje mogućnost učenja koje nije eksplicitno programirano. Sposobnost generisanja znanja na osnovu od ranije poznatih podataka i korišćenja tog znanja u pronalaženju odgovora na pitanja o novim, do tada nepoznatim, podacima.

Tom Mitchell 1998. godine iznosi moderniju definiciju: „Kaže se da kompjuterski program uči iz iskustva E u odnosu na neku klasu zadatka T i meru učinka P, ako se njegov učinak na zadacima T, mereći sa P, poboljšava iskustvom E.“.

Drugim rečima, mašinsko učenje predstavlja podoblast veštačke inteligencije u kojoj algoritmi imaju sposobnost iterativnog otkrivanja pravila među podacima iz kojih donose konačne zaključke spremne za korišćenje u obliku modela. Proces iterativnog obrađivanja ulaznih podataka od strane algoritma naziva se treniranje modela. Ovo nije egzaktna nauka, što znači ne postoji jedno tačno rešenje niti jedan savršeno istreniran model. Više različitih algoritama se mogu koristiti u rešavanju istog problema, ne postoji jedan algoritam koji odgovara samo jednom problemu. Ono što će odlučiti koji će se algoritam koristiti u konačnom rešenju jeste sama preciznost konačnog zaključka tog algoritma, odnosno istreniranog modela. Na preciznost predikcije može uticati sam oblik podataka, količina i kompleksnost, ali i priroda algoritma kao i podešavanje njegovih parametara.

Prema načinu obučavanja modela, prepoznajemo tri tipa mašinskog učenja:

1. Nadgledano učenje (eng. *supervised learning*) podrazumeva skup ulaznih podataka (x1, x2, …, xn) i skup njima odgovarajućih izlaznih podataka (y1, y2, …., yn). Drugim rečima, model obučavamo trening podacima čija tražena prognoza je poznata unapred algoritmu i na osnovu koje se traži logička veza sa ulaznim podacima. Algoritmi sa ovakvim načinom učenja se dele na regresione i klasifikacione.
2. Nenadgledano učenje (eng. *unsupervised learning*) pruža mogućnost izračunavanja izlaznih podataka kada su poznati samo ulazni. Kod ovakvog pristupa rešavanja problema algoritam ima zadatak otkrivanja zakonitosti između ulaznih podataka. Zaključci koje donosi algoritam su do tog trenutka nepoznati i ne mogu se znati efekti ulaznih podataka pre obučavanja. Algoritmi se dele na algoritme za klasterovanje i za pronalazak novih struktura.
3. Učenje podsticajem (eng. *reinforcment learning*) je oblast mašinskog učenja u kojoj se algoritam „nagrađuje“ kada se u iteraciji izračuna dovoljno precizna prognoza, dok se u suprotnom „kažnjava“. Metafora nagrade i kazne ogleda se u jednostavnom dodavanju i oduzimanju od sume za koju algoritam teži da bude što veća. Na taj način algoritam podešava svoje parametre kroz cikluse sa ciljem što veće preciznosti.

2.2.1 Stablo odluke

Stablo (eng. *Tree*) je oblik strukture podataka u računarstvu. Podaci su uvezani u hijerarhiju imitirajući obrnuto drvo, koren je gore dok se grane i listovi razvijaju ka dole. Osnovni elementi ove strukture jesu čvorovi i grane. Prvi čvor u stablu se naziva koren, dok čvorovi ispod kojih ne postoji grananje se nazivaju listovi.

Stablo odluke (eng. *Decision tree*) [11] je struktura podataka u čiji čvorovi predstavljaju odluke, svaka grana je odluka ili pravilo, dok je svaki list konačan ishod. Smatra se kao jedan od najpopularnijih pristupa prikaza klasifikacije. Pronalazi upotrebu u raznim istraživačkim disciplinama kao što su statistika, mašinsko učenje, prepoznavanje obrazaca kao i rudarenju podataka (eng. *data mining)* [12]. Čitav proces prolaska kroz stablo odluke podseća na način ljudskog odlučivanja i lako ga je razumeti.

Primena ove strukture je popularna među algoritmima nagledanog učenja. Svoju upotrebu pronalazi i u regresiji i u klasifikaciji. Ovakvi algoritmi imaju mnogo “šta ako” ograničenja i ovo stablo im pomaže tokom izvršavanja.

2.2.2 Light Gradient Boosting Machine

*Light Gradient Boosting Machin*e (*LGBM*) [13] je algoritam mašinskog učenja zasnovan na stablu odluke. „Laka“ varijanta *gradient boosting machine* algoritma koja je preciznija i boljih performansi od ostalih iz ove porodice algoritama. Svoj epitet je dobio zahvaljujući smanjenoj potrošnji memorije tokom izvršanja i brzim procesom obučavanja modela. Glavna razlika između ovog i ostalih *Gradient Boosting Machine* (GBM) algoritama jeste mogućnost vertikalnog širenja stabla, što znači gradi nove listove. Izlaz algoritma je maksimalno efektivan list sa najmanjom greškom. Pored brzine, preciznosti, ekonomičnog trošenja memorije, ovaj algoritam karakteriše i mogućnost paralelizacije, distribucije i učenja koristeći resurse grafičkog procesora. Takođe, pogodan je za rešavanje problema sa velikom količinom podataka. Sama implementacija algoritma je jednostavna za korišćenje tokom softverskog razvoja.

3. IMPLEMENTACIJA REŠENJA

3.1. Priprema podataka

U ovom radu se rešava problem predviđanja potrošnje električne energije na osnovu podataka vremenskih prilika. Stoga, prvi koraci su pronalaženje javno dostupnih pouzdanih izvora, analiza, obrada i spajanje dva seta podataka.

3.1.1 Potrošnja električne energije

Nakon analize javno dostupnih izvora odabran je set podataka koji predstavlja potrošnju električne energije u Njujorku. Ovaj set je dostupan na veb sajtu *NYC Open Data* [14] generisan od strane uprave za stambena pitanja grada Njujorka (eng. *New York City Housing Authority* (*NYCHA*)). Sadrži različite informacije vezane za potrošnju električne energije na području grada u periodu od 2010. do 2021. godine. U tabeli se nalazi 362,630 zapisa koji su opisani sa 27 atributa.

3.1.2 Vremenske prilike

Kao izvor za skup podataka o vremenskim prilikama korišćen je veb sajt nacionalnog centra za informacije o životnoj sredini *National Oceanic and Atmospheric Administration* (*NOAA*) [15], američka naučna regulatorna agencija u okviru Ministarstva trgovina Sjedinjenih Američkih Država. Podaci su dostupni u obliku prosečnih mesečnih vrednosti za odabrani grad.

3.2. Model mašinskog učenja u C#-u

Visual Studio 2019 nudi proces kreiranja modela mašinskog učenja sa korisničkim interfejsom korišćenjem *ML.NET Model Builder*-a [16]. Korišćenjem ove funkcionalnosti moguće je učitati obrađene podatke, istrenirati model sa algoritmom koji daje najpreciznije rezultate, testirati i isporučiti rešenje bez kucanja koda, u šest koraka:

1. Scenario – Odabir tipa problema, čime se automatski određuje grupa adekvatnih algoritama.

2. Okruženje – Odabir mašine na kojoj se izvršava obučavanje, lokalna ili mašina na *Azure*-u.
3. Podaci – Učitavanje podataka iz fajla ili sa povezanog *SQL* servera. Odabir kolona za ignorisanje tokom treniranja kao i kolone čija se vrednost prognozira.
4. Treniranje modela – Grupa algoritama, adekvatnih za odabrani scenario, se iterativno treniraju pri čemu se nakon svake iteracije beleži preciznost modela. Tokom ovog repetitivnog procesa parametri algoritama se automatski podešavaju sa ciljem da svaki sledeći ciklus bude precizniji od prethodnog.
5. Evaluacija modela – Manuelno testiranje preciznosti najefikasnijeg modela iz prethodnog koraka.
6. Generisanje koda – Gotovo rešenje se može izgenerisati u obliku konzolne aplikacije ili kao *Web API*.

3.3. Model mašinskog učenja u Python-u

Python model kreiran je uz pomoć razvojnog okruženja *Jupyter Notebook* i metode *lightgbm.LGBMRegressor* [17] u okviru *scikit-learn* biblioteke. Parametri algoritama zasnovanih na strukturi drveta (eng. *tree based*), kao što je i *LGBM*, se dele u četiri glavne kategorije:

- Parametri koji utiču na strukturu stabla (*num_leaves* i *max_depth*)
- Parametri koji utiču na brzinu učenja (*num_threads* i *use_gpu*)
- Parametri koji utiču na preciznost (*n_estimators* i *learning_rate*)
- Parametri koji štite od overfit-a (*lambda_1* i *lambda_2*)

4. REZULTATI TESTIRANJA

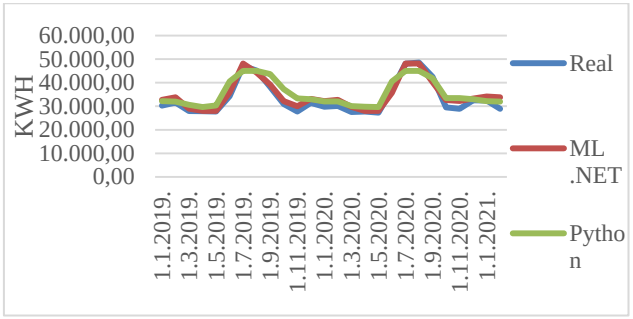
Prilikom kreiranja modela mašinskog učenja i samog testiranja rešenja, u ovom radu korišćeno je isto radno okruženje za *Python* i *ML.NET* kako bi poređenje dve implementacije bilo verodostojno, čije se karakteristike mogu videti u tabeli:

Tabela 1. *Karakteristike testnog okruženja korišćenog u ovom istraživanju.*

OS	Windows 10 Pro 21H2, 64-bit
Procesor	AMD Ryzen 5 2600 Six-Core 3.40 GHz
RAM	16.0 GB

Nakon završetka procesa obučavanja, oba modela mašinskog učenja su izvršavana nad istim setom ulaznih podataka, a njihovi rezultati upoređivani i analizirani. Podaci koji se nisu našli u trening setu, period od januara 2019. do februara 2021. godine, a za koje su poznate mesečne prosečne vrednosti potrošnje zgrada, su korišćeni u testiranju modela. Mesečne prognoze oba modela zajedno sa realno izmerenom potrošnjom se mogu videti na slici 1.

Maksimalna i minimalna odstupanja u prognozi izračunata u KWH a potom predstavljena u % za taj mesec za oba modela se mogu videti tabeli 2.

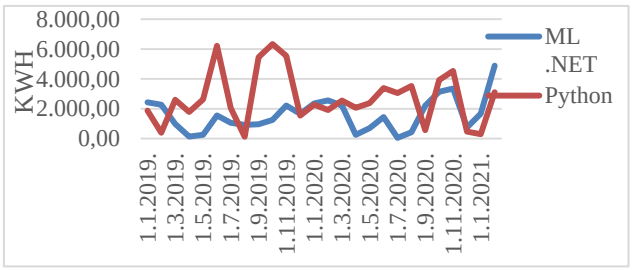


Slika 1. *Vizuelni prikaz prognoziranih i realnih vrednosti potrošnje.*

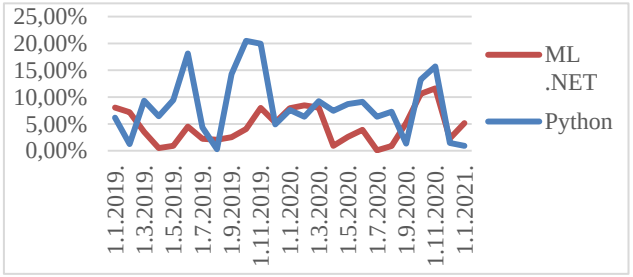
Tabela 2. *Maksimalna i minimalna odstupanja u prognozi oba modela.*

	MAX KWH	MAX %	MIN KWH	MIN %
ML.NET	4885.79	16.92	43.04	0.09
Python	6334.9	20.47	128	0.28

Preciznosti modela za svaki mesec se razlikuju. Drugim rečima, ne postoji fiksno odstupanje od tražene vrednosti za svaku prognoziranu već to u mnogome zavisi od ulaznih podataka i analogije koju je napravio algoritam na osnovu njih. Na slici 2 se mogu videti odstupanja u KWH oba modela za svaki prognozirani mesec, dok se na slici 3 može videti to isto odstupanje u procentima.



Slika 2. *Odstupanja prognozirane potrošnje oba modela u KWH.*



Slika 3. *Odstupanja prognozirane potrošnje oba modela u %.*

Matematički izračunata preciznost prognoza može se videti u tabeli 3.

Tabela 3. *Metrike preciznosti prognoza dva modela.*

	MAPE	**ME**	**MAE**	**MPE**	**CORR**
ML.NET	0.05	1218.34	1602.77	0.042	0.97
Python	0.08	1974.66	2713.95	0.069	0.93

Tumačenjem ovih vrednosti, izračunatim statističkim formulama, utvrđuje se da li prognoze mogu imati upotrebnu vrednost u rešavanju konkretnog problema.

5. ZAKLJUČAK

U ovom radu korišćene su dve implementacije *LGBM* algoritma sa ciljem predviđanja prosečne potrošnje zgrade na Menhetnu na osnovu podataka vremenskih prilika. Model kreiran uz pomoć *ML.NET*-a pokazao je za 3% preciznije rezultate od modela implementiranog u *Python*-u i na taj način dokazao da algoritmi pisani u *C#* jesu konkurencija na tržištu.

Važan deo procesa stvaranja modela mašinskog učenja jeste priprema podataka. Veliki broj praktičnih i efikasnih biblioteka napisanih u *Python*-u čine ovaj jezik liderom u obradi podataka (eng. *Data proccessing*).

Proces kreiranja modela u *C#* je znatno olakšan korišćenjem *Model Builder*-a, ali njegovo glavno ograničenje jeste to što na raspolaganju ima samo one algoritme koje je *Microsoft* podržao. Drugim rečima *LGBM* algoritam se za potrebe ovog rada jeste pokazao preciznijim kada se koristi njegova *C#* impelementacija, međutim *Python* ima veći broj podržanih algoritama i postoji mogućnost da korišćenjem nekih od njih koji nisu podržani u *.NET*-u imaju još preciznije rezultate. Takođe, radno okruženje *Jupyter Notebook* koje se koristi tokom obrade podataka i kreiranjem modela u *Python*-u je jednostavnije i lakše za korišćenje od *Visual Studio*-a, što ovom jeziku daje prednost.

Dalje usavršavanje rešenja bi moglo ići u pravcu novih tehnika za podešavanje parametara. Sve češće se može videti upotreba algoritama veštačke inteligencije baziranih na učenju nagradom i kaznom (eng. *reinforcment learning*) u podešavanju parametara drugih algoritama. Na taj način mašinskim učenjem stvaramo novi što precizniji model mašinskog učenja. Algoritam kroz iteracije podešava različite vrednosti ulaznih promenjivih, i u zavisnosti od preciznosti novodobijenog modela biva nagrađen ili kažnjen. Takođe, *Python* zajednica nudi širok spektar algoritama regresije, gde bi neki od njih mogao dati preciznije rezultate od *LGBM* implementiranog u *.NET*-u. Konačno, *Python* i *.NET* možda jesu popularne i velike zajednice ali nisu jedine. Istraživanja treba proširivati novim tehnologijama, davati šansu novim algoritmima.

6. LITERATURA

[1] Zhao, Hai-xiang & Magoulès, Frédéric, „*A review on the prediction of building energy consumption*“, Renewable and Sustainable Energy Reviews. 16. 3586-3592. 10.1016/j.rser.2012.02.049.

[2] J. Kolter,J. Ferreira, „*A Large-Scale Study on Predicting and Contextualizing Building Energy Usage*“, AAAI, vol. 25, no. 1, pp. 1349-1356, Aug. 2011.

[3] Slobodan Ilić, „*Kratkoročno predviđanje potrošnje električne energije u velikim elektroenergetskim sistemima*“, Fakultet tehničkih nauka, Univerzitet u Novom Sadu.

[4] Sebastijan Stoja, „*Arhitektura softverskog sistema za elektroenergetske proračune zasnovana na mikroservisima*“, Fakultet tehničkih nauka, Univerzitet u Novom Sadu.

[5] Gama João, Rodrigues Pedro, „*Stream-Based Electricity Load Forecast*“, 446-453. 10.1007/978-3-540-74976-9_45, 2007.

[6] Khuntia, Swasti R. & Rueda, José & Meijden, Mart, „*Forecasting the Load of Electrical Power Systems in Mid- and Long-term Horizons - A Review*“, IET Generation Transmission & Distribution. 10, 2016. 10.1049/iet-gtd.2016.0340.

[7] J. W. Taylor, „*An evaluation of methods for very short-term load forecasting using minute-by-minute British data*“, International Journal of Forecasting, vol. 24, no. 4, pp. 645-658, 2008.

[8] G. Gross, F.D. Galiana, „*Short-term load forecasting*“, Proceedings of the IEEE, vol. 75, no.12, pp. 1558-1572, 1987.

[9] N. Amjady, F. Keynia, „*Mid-term load forecasting of power systems by a new prediction method*“, Energy Conversion and Management, vol. 49, no. 10, pp. 2678-2687, 2008.

[10] H. Daneshi, M. Shahidehpour, A. L. Choobbar, „*Long-term load forecasting in electricity market*“, IEEE International Conference on Electro/Information Technology, USA, 2008.

[11] Harsh H. Patel, Purvi Prajapati, „*Study and Analysis of Decision Tree Base Classification Algorithms*“, Dept. Of Information Technology, CSPIT, Charotar University of Science and Technology, Changa, Gujarat, India.

[12] Lior Rokach, Oded Maimon, „*The Data Mining and Knowledhe Discovery Handbook Chapter:Decision Trees*“, DOI:10.1007/-387-25465-X_9.

[13] https://lightgbm.readthedocs.io/en/v3.3.2/ (pristupljeno u martu 2023.)

[14] https://data.cityofnewyork.us/Housing-Development/Electric-Consumption-And-Cost-2010-April-2020-/jr24-e7cr (pristupljeno u martu 2023.)

[15] https://www.ncdc.noaa.gov/cdo-web/datasets/GSOM/locations/CITY:US360019/detail (pristupljeno u martu 2023.)

[16] https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet/model-builder (pristupljeno u martu 2023.)

[17] https://lightgbm.readthedocs.io/en/latest/index.html (pristupljeno u martu 2023.)

Kratka biografija:

Nemanja Simić rođen je u Šapcu 1996. godine. Osnovne akademske studije na Fakultetu tehničkih nauka Univerziteta u Novom Sadu upisao je 2015. godine. Diplomirao je 2019. godine na smeru Primenjeno sofversko inženjerstvo i iste godine upisao master akademske studije na istom smeru.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4
DOI: https://doi.org/10.24867/23BE26Grbic

SOFTVERSKO REŠENjE ZA DETEKCIJU ANOMALIJA POTROŠNJE ELEKTRIČNE ENERGIJE U ML .NET-U I PYTHON-U

SOFTWARE SOLUTION FOR THE DETECTION OF ELECTRICAL ENERGY CONSUMPTION ANOMALIES IN ML .NET AND PYTHON

Nina Grbić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Podatke iz Smart Grid sistema moguće je analizirati za detekciju abnormalnih pojava u različitim oblastima poput sajber bezbednosti, detekcije krađe, detekcije kvara i slično. Algoritmi mašinskog učenja mogu poslužiti kao alat za analizu i obradu sirovih podataka koji dolaze iz sistema u realnom vremenu. Kroz ovaj su rad opisani, analizirani i prikazani rezultati implementiranog softverskog rešenja koje vrši detekciju anomalija na osnovu podataka o potrošnji električne energije nad algoritmima u ML .NET-u i Python-u.*

Ključne reči: *Detekcija anomalija, Mašinsko učenje, ML .NET, Python*

Abstract – *Data from the Smart Grid system can be analyzed to detect abnormal phenomena in various areas such as cyber security, theft detection, failure detection, etc. Machine learning algorithms can be used as a tool for analyzing and processing raw data coming from real-time systems. This paper describes, analyzes and presents the results of the implemented software solution that detects anomalies based on data containing electricity consumption over algorithms in ML .NET and Python.*

Keywords: *Anomaly detection, Machine learning, ML .NET, Python*

1. UVOD

Gubici koji se javljaju u elektroenergetskim sistemima (EES) u velikoj meri utiču na stabilnost mreže, kvalitet napajanja kao i procenat troškova u konačnoj ceni električne energije. Ukupni distributivni gubici Elektroprivrede Srbije (EPS) su među najvećim u Evropi i kao direktna posledica postoji značajna potreba da se tehnički gubici na mreži i krađe električne energije svedu na prihvatljivu meru **Error! Reference source not found.**.
Ovo predstavlja jasan pokazatelj da tradicionalan pristup upravljanja električnom mrežom mora da se zameni sa inovativnim tehnologijama koje će optimizovati procese prilikom nadgledanja električne mreže. U softver za monitoring elektroenergetskog sistema moguće je integrisati takvo rešenje da algoritmima mašinskog učenja i analizom podataka se detektuju oblasti ili potrošači sa neuobičajenim karakteristikama ili potrošnjom.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Sebastijan Stoja, docent.

Najpopularnija tehnologija za implementaciju algoritama mašinskog učenja jeste *Python* odnosno njegova moćna biblioteka *SciKit-learn*. Međutim, *Microsoft* 2019. godine objavljuje svoju tehnologiju pod nazivom *ML .NET* sa ciljem da bude konkurentan na tržištu tehnologija za integrisanje modela mašinskog učenja u softverska rešenja. Glavna motivacija ovog rada jeste analiza preciznosti algoritama za detekciju anomalija u krivama potrošnje uz pomoć algoritama mašinskog učenja u *ML .NET*-u i *Python*-u.

2. TEORIJSKE OSNOVE

2.1. Elektroenergetski sistemi

2.1.1. Smart Grid

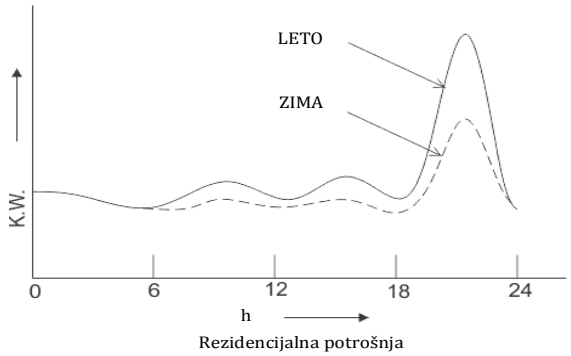
Smart Grid se odnosi na električne mreže sledeće generacije zasnovane na digitalnoj tehnologiji za snadbevanje potrošača električnom energijom putem dvosmerne digitalne komunikacije. Ovaj sistem isporučuje električnu energiju između potrošača i dobavljača i kontroliše digitalne uređaje kako bi obezbedio uštedu energije uz smanjenje potrošnje energije i troškova, a maksimiziranje transparentnosti i pouzdanosti lanca snadbevanja energijom [2].
Pored tradicionalnih proizvodnih objekata i prenosne mreže, Smart Grid se sastoji od tri nove komponente: pametni kontrolni i merni uređaji, digitalni komunikacioni sistemi i računarski softverski program. Pametni uređaji podrazumevaju kompjuterski kontrolisane generatore i druge izvore energije, kao i brojila i inteligentne elektronske uređaje koji prikupljaju informacije o potražnji električne energije, njenoj dostupnosti iz različitih izvora, kapacitetu isporuke svakog dela mreže i protoku energije [3].

2.1.2. Pametna brojila

Pametna brojila (eng. *Smart Meter*) su ključan faktor u *Smart Grid* sistemu i čini opremu koja prati potrošnju električne energije u predefinisanim intervalima. Digitalna brojila zajedno sa sistemom za automatsko očitavanje merenja (AMR), obradom i arhiviranjem merenih podataka i automatskim upravljanjem merenjima (AMM) čini integrisani sistem za naprednu infrastrukturu za merenje (AMI) koja danas u velikom broju zamenjuje tradicionalna brojila [4].
U naprednim sistemima za upravljanje EES, praćenje potrošnje u realnom vremenu omogućava i ažuriranje krive potrošnje korisnika kako bi se dobila preciznija kriva sa rezultatima sa polja. Vrednosti sa potrošačke krive se koriste prilikom izvršavanja glavnih funkcija proračuna stanja mreže.

2.1.3. Krive potrošnje

Kriva potrošnje predstavlja dijagram potražnje za snagom (y osa) u odnosu na vreme (x osa) u hronološkom redu. Ova kriva reprezentuje varijaciju opterećenja nekog elementa elektroenergetskog sistema (npr. potrošača) u odnosu na vreme (Slika 1).



Slika 1. *Primer krive potrošnje*

2.1.4. Gubici električne energije

Gubici električne energije u EES su jasan pokazatelj efikasnosti rada infrastrukture mreže. Gubici na nivou prenosa i distribucije se mogu podeleti na tehničke i netehničke gubitke. Tehnički gubici su neizbežni, jer su rezultat fizičkog procesa prenosa odnosno transformacije električne energije, kvarova ili problema sa opremom i infrastrukturom tokom procesa prenosa i distribucije [5].
Za razliku od tehničkih, netehnički gubici nastaju kao posledica situacija koje je moguće kontrolisati poput nenaplaćenu ili pogrešno naplaćenju potrošnju električne energije, kao i nezakonitu upotrebu električne energije.

2.2. Mašinsko učenje

Mašinsko učenje predstavlja proces analize podataka koji korišćenjem posebno podešenih algoritama i iterativne obrade podataka na ulazu, omogućavajući sistemu koji implementira da otkrije skrivena znanja u modelu podataka, bez eksplicitnog programiranja [6]. Glavna ideja mašinskog učenja predstavlja razvoj matematičkog modela koji će analizom i obradom postojećih podataka generisati zaključke i predviđanja o novim podacima. Ako to izrazimo matematički, pokušavamo da aproksimiramo funkciju mapiranja – f od ulaznih promenjivih *y* u izlazne promenjive *x*(1):

$$f(x) = y \qquad (1)$$

Kreiranje modela mašinskog učenja se svodi na sledeće korake:

- Analiza i priprema podataka – prikupljanje atributa, analiza značenja atributa i obeležja, redukcija suvišnih atributa ili podataka koji unose šum, normalizacija vrednosti atributa kako bi rezultovali definisanjem krajnjeg skupa podataka koji će se koristiti za treniranje, validaciju i testiranje modela mašinskog učenja.
- Primena odgovarajućeg algoritma i podešenje parametara – konfigurisanje modela biranjem odgovarajućeg tipa algoritma i definisanjem neophodnih parametara ili hiperparametara.
- Evaluacija – eksperimentalna evaluacija se sprovodi radi određivanja performansi rada nad istim skupom podataka, pruža uvid u tačnost i preciznost matematičkog modela i vrše se iterativne korekcije u konfiguraciji parametara kako bi dobili bolje rezultate [7].

2.2.1. Algoritmi mašinskog učenja

Algoritmi mašinskog učenja se najčešće kategorizuju po načinu na koji algoritam vrši predikcije:

- Nadgledano obučavanje – sistem se obučava uz pomoć obeležja podataka, algoritam generiše funkciju koja mapira ulazne podatke na željene izlaze.
- Nenadgledano obučavanje – sistem se obučava uz pomoć ulaznih podataka i bez dobijanja obeležja utvrđuje glavne karakteristike podataka i formira prirodne klastere.
- Polunadgledano obučavanje – obeležja podataka su dostupna samo za normalne podatke, dok za obučavanje koristi se modifikovani model klasifikacije čime detektuju instance podataka koje se ne uklapaju u predviđeni šablon.
- Pojačano obučavanje – sistem obučava softverske agente koje akcije da preduzmu u svom okruženju. Učenje se vrši na osnovu datih ulaznih podataka i signala podrške [8].

2.2.2. Detekcija anomalija

Anomalijom se smatra tačkom podataka koja se značajno razlikuje od drugih tačaka u datom skupu. Detekcija anomalija predstavlja proces pronalaženja odstupanja u podacima. Anomalije se u globalnom smislu mogu podeliti u 3 glavne kategorije opisane uz primere u tabeli:

Tabela 1. *Osnovna kategorizacija anomalija*

Kategorija	Opis
Tačkaste anomalije	Određena instanca podatka koja u velikoj meri odstupa u odnosu na očekivano ponašanje podataka sistema
Kontekstualne anomalije	Instance odstupanja podataka koje u velikoj meri zavise od konteksta podataka.
Zbirne anomalije	Analiza pojedinačne instance ne predstavlja anomaliju, ali njihovo grupisanje se smatra neuobičajenim ponašanjem.

3. METODOLOGIJA

3.1 Formulacija problema

Kreirano je softversko rešenje u *ML .NET*-u sa njegovim implementiranim algoritmima za detekciju anomalija, kao i rešenje u *Python*-u sa *Isolation Forest* odabranim algoritmom za detekciju anomalija na osnovu radova [9][10]. Od podataka kao ulaz u model mašinskog učenja koristi se fajl koji sadrži 24-časovna očitavanja električne energije niskonaponskih potrošača proizvoljno generisanih. Analiziraju se rezultati dobijeni kao izlaz iz svih funkcija, kao i eksperimentalna evaluacija radi određivanja performansi algoritama nad istim ulazom.

3.2. Odabrani algoritmi

3.2.1. ML .NET

DetectSpikeBySSA – SSA je skraćenica za analizu singularnog spektra i predstavlja složenu metodu analize koja kombinuje delove standardne analize podataka

vremenske serije, multivarijantnu statistiku, multivarijantnu geometriju, dinamičke sisteme i obradu signala [11].

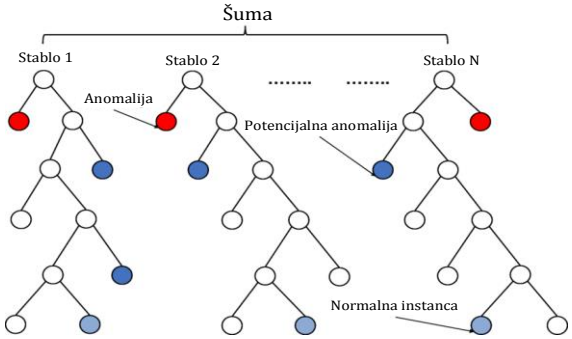
DetectIidSpike – U teoriji verovatnoće i statistici, kolekcija slučajnih promenljivih je nezavisna i identično raspoređena ako svaka slučajna promenljiva ima istu distribuciju verovatnoće kao i druge i sve su međusobno nezavisne [12].

DetectEntireAnomalyBySrCnn – Algoritam konvolucione neuronske mreže super rezolucije Sr-Cnn (eng. *Super Resolution Convolutional Neural Network*) koristi kompjuterski pogled na problem detekcije anomalija. Sr-Cnn pozajmljuje model super rezolucije iz domena otkrivanja vizuelne istaktnutosti i primenjuje ga na otkrivanje anomalija u tipovima podataka poput vremenskih serija [13].

RandomizedPCA – Randomizovani PCA (eng. *Principal Component Analysis*) algoritam funkcioniše tako što traži korelacije između varijabli i utvrđuje kombinaciju vrednosti koja najbolje predstavlja razlike u rezultatima. Zatim se ove kombinovane vrednosti obeležja koriste za kreiranje kompaktnijeg skupa obeležja, koji se naziva glavnim komponentama. Randomizovani PCA označava približni model analize glavnih komponenti koji koristi algoritam randomizovane dekompozicije singularnih vrednosti [14].

3.2.2. Python

Isolation Forest izoluje zapažanja tako što nasumično bira obeležje, a zatim nasumično određuje vrednost između maksimalne i minimalne vrednosti izabranog obeležja.



Slika 2. *Vizuelni prikaz detekcije anomalija koristeći Isolation Forest algoritam*

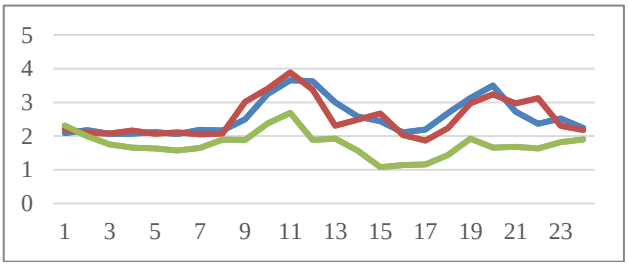
Pošto se rekurzivno particionisanje može predstaviti sktrukturom stabla, broj particionisanja potrebnih za izolovanje uzorka je jednak dužini putanje od korenskog čvora do završnog čvora. Nasumično particionisanje proizvodi primetno kraće putanje za anomalije. Stoga, ukoliko skup nasumičnih stabala zajedno proizvodi kraće dužine puta za određene uzorke, veća je verovatnoća da će te tačke biti okarakterisane kao anomalije [15].

3.3. Podaci

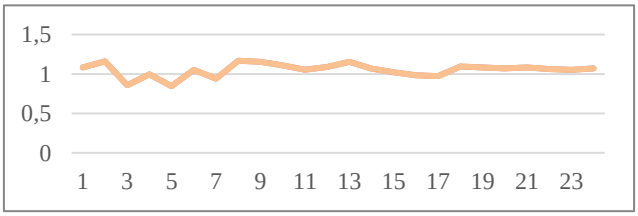
U sistemima koji rade u realnom vremenu, jedan od glavnih poslovnih zahteva predstavlja praćenje ključnih signala kako bismo bili sigurni da prate očekivane obrasce ponašanja. Podaci ove prirode su poznati kao vremenske serije (eng. *time series*) i predstavljaju hronološki uređen niz podataka pojave koja se izučava. Pojavu od interesa najčešće pratimo po danima, mesecima, kvartalima, godinama itd.

Za potrebe analize problema, podaci su izgenerisani uz pomoć ručno kreirane skripte na osnovu podataka standardne krive potrošnje. Kriva potrošnje je predstavljena sa 24 tačke gde svaka tačka reprezentuje očitanu vrednost potrošnje u datom satu u kW. Da bi se adekvatno evaluirali razmatrani modeli mašinskog učenja, ručno su kreirani različiti problemi koji su tipični u analizi podataka dnevne potrošnje poput:

- Odsecanje pikova potrošnje – predstavlja jedan od načina upravljanja troškovima komunalnih usluga tako što se eliminišu kratkoročni skokovi potražnje odgovorne za visoke troškove potrošnje električne energije u datom momentu.
- Profilisanje potrošnje – algoritmima mašinskog učenja moguće je utvrditi i nove tipove potrošača u sistemu. Naknadom analizom rezultata novih potrošačkih krivi, distribucija utvrđuje validnost novodetektovanih potrošača ili potencijalnu malverzaciju podacima očitavanja električne energije.
- U podacima su ručno uneta 3 primera generisanih novih tipova potrošača (Slika 3) a 1 slučaj ručno simuliranog odsecanja pikova potrošnje (Slika 4).



Slika 3. *Ručno generisani primeri novih tipova potrošača*

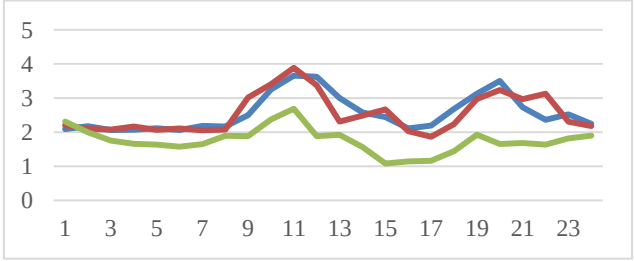


Slika 4. *Ručno generisan primer odsecanja potrošnje*

Uz pomoć ovih primera evaluira se uspešnost algoritama za detekciju anomalija u krivama potrošnje.

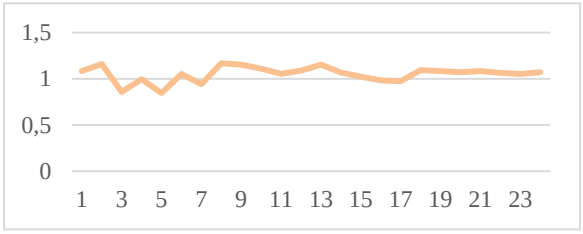
4. REZULTATI REŠENjA

U ML .NET-u najuspešnije se pokazala *DetectEntireAnomalyBySrCnn* funkcija. Ona je detektovala sve nove tipove potrošača ali nije detektovala odsecanje potrošnje kao vid anomalije u ulaznim podacima (Slika 5).



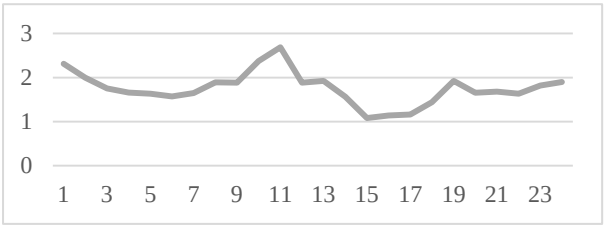
Slika 5. *Rezultati funkcije DetectEntireAnomalyBySrCNN u ML .NET-u*

Funkcija *DetectSpikeBySSA* daje rezultate nasuprot prethodno testirane funkcije i kao rezultat prikazuje samo primer odsecanja potrošnje (Slika 6).



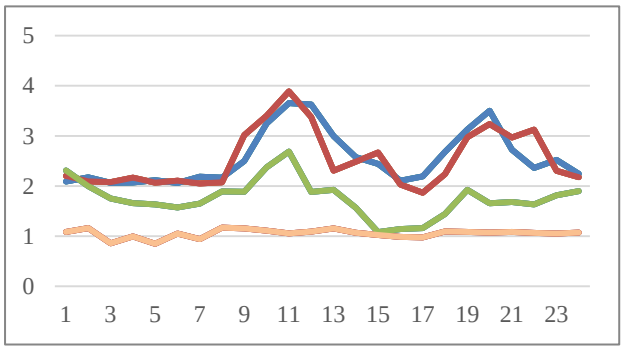
Slika 6. *Rezultati funkcije DetectSpikeBySSA u ML .NET-u*

DetectIidSpike nije dao pozitivne rezultate ni za jedan primer anomalije u ulaznom skupu podataka i nije dalje razmatran. *DetectSpikeBySSA*, *DetectIidSpike* i *DetectEntireAnomalyBySrCnn* funkcije kao ulaz primaju jednodimenzioni niz, što je zahtevalo transformaciju 24 dimenzije podataka vremenske serije na jednu dimenziju po potrošaču. Prilikom smanjenja dimenzionalnosti, gubimo potencijalno bitne karakteristike podataka što može uticati na performanse i preciznost algoritama. Izuzetak prethodno istaknutog problema *ML .NET* je izbegao u implementaciji *RandomizedPCA* algoritma. Međutim, ova metoda pokazuje najlošije rezultate jer je detektovala samo 1 novi tip potrošača od ukupno 3, a nije detektovala odsecanje pikova potrošnje (Slika 7).



Slika 7. *Rezultati funkcije RandomizedPCA u ML .NET-u*

Python je opravdao svoju vodeću poziciju na tržištu tehnologija za integrisanje modela mašinskog učenja u softverska rešenja, i pokazao najbolje rezultate sa detektovanim svim novim tipovima potrošnje, kao i primer odsecanja pikova potrošnje (Slika 8).



Slika 8. *Rezultati funkcije IsolationForest u Python-u*

5. ZAKLJUČAK

U ovom radu, analizirani su algoritmi za detekciju anomalija u *Python*-u i *ML .NET*-u. Rezultati su pokazali da *Python*-ova implementacija *Isolation Forest* algoritma pruža najbolje rezultate, dok kod *ML .NET*-a možemo uočiti detekcije pojedinih slučajeva ali ni jedan zaseban algoritam ne detektuje sve anomalije sa generisanim skupom podataka. Dalji pravci unapređenja softverskog rešenja za detekciju anomalija u *ML .NET*-u mogu biti fokusirani na kreiranju hibridnog modela koji će uz pomoć više implementacija algoritama imati model mašinskog učenja koji može da pokrije sve slučajeve neuobičajenog ponašanja. Jedno od zanimljivih rešenja bi bilo i istraživanje benefita hibridne softverske implementacije *Python* i *ML .NET* algoritama za analizu i pripremu podataka, tako i rešavanje problema uz pomoć algoritama mašinskog učenja.

6. LITERATURA

[1] https://bif.rs/2021/12/ukupni-distributivni-gubici-eps-a-su-medju-najvecim-u-evropi/ (pristupljeno u martu 2023.)
[2] Raja Masood Larik, Mohd Wazir Mustafa, Sajid Hussain Qazi, „*Smart Grid Technologies in Power System*“s, University of Technology Malaysia, Malaysia
[3] https://www.techopedia.com/definition/692/smart-grid (pristupljeno u martu 2023.)
[4] Srdjan Milošević, Emil Naumovski, „*Sistem za daljinsko očitavanje i upravljanje potrošnjom u PD „Elektrodistribucija Beograd“*“, Zbornik Međunarodnog kongresa o KGH, [S.l.], v. 44, n. 1, p. 1-6, oct. 2017
[5] https://anyline.com/news/detect-non-technical-losses-energy-utility (pristupljeno u martu 2023.)
[6] Bishop, C. M., „*Pattern Recognition and Machine Learning, Springer“*
[7] Nenad Rakić, *“Primena metoda mašinskog učenja za rangiranje individualnih sposobnosti”*, Prirodni Matematički Fakultet, Univerzitet u Novom Sadu, 2020,
[8] as. ms Vladimir Jocović, as. ms Adrian Milaković, “*Inteligentni sistemi*”, Elektrotehnički fakultet, Univerzitet u Beogradu
[9] https://blog.paperspace.com/anomaly-detection-isolation-forest/ (pristupljeno u martu 2023.)
[10] Kumar Reddy Shabad, Abdulmueen Alrshide, „*Anomaly Detection in Smart Grids using Machine Learning*, Miami, Florida, USA, 2021.
[11] https://learn.microsoft.com/en-us/dotnet/api/microsoft.ml.timeseriescatalog.detectspikebyssa?view=ml-dotnet (pristupljeno u martu 2023.)
[12]https://en.wikipedia.org/wiki/Independent_and_identically_distributed_random_variables (pristupljeno u martu 2023.)
[13] https://towardsdatascience.com/time-series-anomaly-detection-b10fdb542974 (pristupljeno u martu 2023.)
[14] Vladimir Rokhlin, Arthur Szlam, Mark Tygert, „*A randomized algorithm for principal component analysis“*, Jul 2009.
[15] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html (pristupljeno u martu 2023.)

Kratka biografija:

Nina Grbić rođena je u Novom Sadu 1996. godine. Osnovne akademske studije na Fakultetu tehničkih nauka Univerziteta u Novom Sadu upisala je 2015. godine. Diplomirala je 2019. godine na smeru Primenjeno sofversko inženjerstvo i iste godine upisala master akademske studije na istom smeru.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE27Pualic

ULOGA I ZNAČAJ SOFTVERA ZA UPRAVLJANJE PROJEKTIMA U KONTEKSTU POSLOVANJA IT FIRME

THE ROLE AND IMPORTANCE OF PROJECT MANAGEMENT SOFTWARE IN THE CONTEXT OF IT COMPANY OPERATIONS

Nemanja Pualić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – *U ovom radu ispitana je uloga upravljanja projektima u funkcionisanju IT preduzeća i načini na koje se može efikasno sprovesti. Predstavljeni su koraci ka uspešnom vođenju IT Firme, kao i šta se ne sme raditi tokom poslovanja IT firme. Takođe, detaljno je opisana važnost softvera za upravljanje projektima u procesu vođenja IT preduzeća. Dalje, objašnjena su opšta projektna obeležja, kao i koncept upravljanja projektima, njegove faze i procesi.. Pored toga, sprovedena je uporedna analiza najčešće korišćenih softvera za upravljanje projektima, a posebno je analizirana Jira platforma. U radu je takođe prikazan pregled Jira platforme i njenog radnog panela, kreiranje tabele, nove kolone i verzije, kao i praćenje vremena stavki i izveštavanje o odrađenom poslu. Takođe, predstavljena je i napredna pretraga u Jiri pomoću JQL jezika koji je takođe detaljno obrađen i opisan.*

Ključne reči: *IT Firma, Jira platforma, JQL jezik, Upravljanje projektima, Napredna pretraga*

Abstract – *In this paper, the role of project management in the functioning of IT companies and the ways in which it can be effectively implemented are examined. The steps towards successful management of an IT Company are presented, as well as what must not be done during the operation of an IT Company. Also, the importance of project management software in the process of running an IT company is described in detail. Furthermore, the general project features are explained, as well as the concept of project management, its phases and processes. In addition, a comparative analysis of the most commonly used project management software was conducted, and the Jira platform was analyzed in particular. The paper presents an overview of the Jira platform and its work panel, creating a table, new columns and versions, as well as tracking the time of items and reporting on work done. Also, advanced search in Jira using the JQL language is presented, which is also covered and described in detail.*

Keywords: *IT Company, Jira Platform, JQL Language, Project Management, Advanced Search*

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, red. prof.

1. UVOD

Softver za upravljanje projektima predstavlja jedan od ključnih alata za uspešno funkcionisanje IT kompanija. Ovaj softver omogućava organizaciju i planiranje svih aspekata projekta, kao i praćenje napretka i evaluaciju rezultata. Zahvaljujući ovom alatu, IT kompanije su u stanju da efikasnije rade na više projekata istovremeno, što rezultira povećanju njihove produktivnosti i profitabilnosti. Softver za upravljanje projektima obuhvata širok spektar funkcija koje su neophodne za uspešno funkcionisanje projekata u IT industriji. To uključuje planiranje projektnog budžeta i rokova, raspodelu resursa, praćenje napretka, identifikaciju i rešavanje problema, kao i evaluaciju rezultata.

2. OPŠTA PROJEKTNA OBELEŽJA

Projekat je ciljno usmerena, jednokratna, relativno nova i kompleksna namera, produkt ili celovitost međusobno povezanih aktivnosti čije je trajanje vremenski ograničeno, a ispunjenje odnosno realizacija povezana s korištenjem ogromnih resursa i visokim rizikom, pa zbog toga zahteva saradnju različitih stručnjaka (timski rad), ocenjivanje valjanosti i posebno organizovanje.

2.1. Projektni zadaci

Projektni zadaci su sve aktivnosti potrebne za realizaciju unapred definisanih projektnih rezultata. Oni su specifični za svaki projekat, međutim mogu se razlikovati zadaci planiranja od zadataka realizacije projekta. Ovi poslednji dalje se dele na zadatke pribavljanja i korištenja resursa (npr. kadrova za jednu projektnu fazu) i na zadatke koji su neposredno usmereni na realizaciju projektnog cilja (npr. rešenje projektnog problema) [1].

2.2. Koordinacija projektom

Pojam koordinacija projektom odnosi se na sve one nosioce projektnih zadataka koji su nadležni za osiguranje i korištenje projektnih resursa i za koordinaciju projektnih aktivnosti. I kod individualne i kod čiste projektne organizacije o kojima će biti reči kasnije, postoji vođa projekta koji rukovodi i koordinira projektom odnosno zadaje zadatke projektnom timu i koordinira njihov rad kako bi projekat bio ostvaren u predviđenom vremenu i uz minimalne troškove [1].

2.3. Izvođenje projekta

Izvođenjem projekta upravljaju projektni saradnici koji izvršavaju projektne zadatke uz korištenje raspoloživih resursa, a pritom oni nemaju pravo izdavanja uputstva. (projektni timovi) [1].

3. FAZE UPRAVLJANJA PROJEKTIMA

Životni ciklus projekta može se definisati kao okvir specifičnih faza visokog nivoa koji pomažu da se ideja ostvari na organizovan način. Upravljanje projektom nije lak podvig, bez obzira na razmere i obim. Kada podelite projekat na faze kojima se može upravljati, od kojih svaka ima svoje ciljeve i rezultate, lakše je kontrolisati projekat i kvalitet rezultata [2].

3.1. Pokretanje (inicijacija)

Inicijacija je prva faza upravljanja softverskim projektom. Ukratko, inicijacija je proces pokretanja projekta, gde se utvrđuje da li je projekat izvodljiv i potreban, utvrđuju se ključni kriterijumi uspeha, definiše se projekat, analizira i formira tim za projekat. Nakon što projekat dobije odobrenje, prelazi se u sledeću fazu – planiranje [2].

3.2. Planiranje

Sledeći korak je da počnete da shvatate kako ćete tačno završiti projekat. Ko je uključen u njega? Koji zadaci su uključeni? Koji zadaci zavise od drugih zadataka? Koliko dugo će trajati rad na projektu? Odlična vizuelna alatka za ovaj aspekt procesa jeste Gantov grafikon. Gantovi grafikoni pomažu u definisanju zadataka projekta i kada će se oni obavljati, sve do definisanja ciljeva za svaki dan [2].

3.3. Izvršenje

Faza izvršenja projekta je u kojoj vaš tim obavlja stvarni posao. Kao menadžer projekta, vaš posao je da uspostavite efikasne tokove posla i pažljivo pratite napredak vašeg tima. Jedan od alata koji će vam pomoći u ovoj fazi je Jira alat koji će biti obrađen u nastavku [2].

3.4. Praćanje i kontrola projekta

U procesu upravljanja projektom, treća i četvrta faza nisu po prirodi sekvencijalne. Faza praćenja i kontrole projekta odvija se istovremeno sa izvođenjem projekta, čime se osigurava da su ciljevi i rezultati projekta ispunjeni. U ovoj fazi se uspostavljaju kritični faktori uspeha (CSF) i ključni pokazatelji učinka (KPI). Tokom faze praćenja upravljanja projektom, menadžer je takođe odgovoran za kvantitativno praćenje napora i troškova tokom procesa. Ovo praćenje ne samo da osigurava da projekat ostane u okviru budžeta, već je i važno za buduće projekte [2].

3.5. Završna faza, faza zatvaranja

Ovo je završna faza procesa upravljanja projektom. Faza zatvaranja projekta označava kraj projekta nakon konačne isporuke. Konačni zadatak ove faze je da se pregleda ceo projekat i završi detaljan izveštaj koji pokriva svaki aspekt. Svi potrebni podaci se čuvaju na bezbednom mestu kome mogu da pristupe projektni menadžeri te organizacije [2].

4. UPOREDNA ANALIZA SOFTVERA ZA UPRAVLJANJE PROJEKTIMA

U ovom poglavlju je uporedno prikazano i analizirano jedanaest softvera koji se primenjuju za upravljanje projektima. Svaki od ovih softvera je analiziran prema elementima koje sadrže. Osnovni analizirani elementi su: mogućnost kolaboracije, praćenje, raspoređivanje, projektni portfolio menadžment, upravljanje resursima, upravljanje dokumentima, sistem toka rada, izveštavanje i analiza, baziranost na mreži i licenciranost [3].

4.1. Softveri za upravljanje projektima

Danas generalno postoje dva tipa računarskih programa za podršku upravljanju projektima, a to su računarski programi koji se implementiraju kao desktop aplikacije i programi koji se implementiraju kao internet aplikacije, kojima se pristupa preko interneta ili extraneta i koji koriste veb pretraživač. U ovom radu je prikazano i upoređeno jedanaest računarskih programa prama komponentama koje sadrže. Pored prikazanih programa postoji još oko 100 softvera za upravljanje projektima, od kojih čak 23 ima većinu analiziranih elemenata.

4.2. Uporedna analiza softvera za upravljanje projektima

Analiza softvera je vršena preko komponenti koje sadrže. Komponente koje se posmatraju su: mogućnost kolaboracije, praćenje, raspoređivanje, projektni portfolio menadžment, upravljanje resursima, upravljanje dokumentima, sistem toka rada (workflow system), izveštavanje i analize, baziran na mreži (web-based) i da li je licencir. Projekti ne mogu funkcionisati tako dobro u izolaciji od ljudi koji rade u istoj organizaciji, te ova opcija omogućava unapređenje rada preko razmene fajlova sa projektnim informacijama, ali i menadžerima da obaveštavaju o statusu projekta ili da kreiraju izveštaje koji se mogu videti u celoj organizaciji.

Softveri su analizirani i po pitanju posedovanja sistema za praćenje. Ovakvi sistemi kao delovi softvera za upravljanje projektima omogućavaju praćenje projekata po određenim elementima, poređenje i upozoravanje na moguća ili nastala odstupanja u odnosu na planirane veličine, kao i upravljanje i rešavanje nastalih odstupanja. Treća analizirana komponenta koja je komponenta raspoređivanja. Ova komponenta se odnosi na proces raspoređivanja resursa na planirane zadatke. Sledeća komponenta je projektni portfolio menadžment. Pod ovim terminom misli se na analiziranje i upravljanje grupom trenutnih ili predloženih projekata na osnovu brojnih karakteristika. Osnovna svrha ove komponente je pomoć menadžerima da naprave optimalan miks projekata koji će omogućiti da organizacija na najbolji mogući način postigne svoje ciljeve.

Softveri koji sadrže komponentu upravljanja resursima daju mogućnost za efikasno i efektivno upošljavanje istih. Ovim se želi omogućiti na najbolji način raspoređivanje postojećih resursa i postizanje ciljeva u okviru planiranog. Takođe komponenta koja se ispituje, a značajna je za uspešnu saradnju svih na projektu je sistem za upravljanje dokumentima. Ovim sistemom se omogućava praćenje i arhiviranje elektronskih dokumenata, kao i različitih verzija istog dokumenta kojima su pristupili razljičiti članovi tima. Takođe softveri su analizirani i sa aspekta obuhvata komponente sistema toka rada (workflow system). Workflow predstavlja postavljanje tokova posla, pravila koja povezuju niz zadataka. Ovaj sistem omogućava upravljanje, kontrolu, kao i uvid u efikasnost izvršenja. Smanjenje potrebnog vremena, povećanje produktivnosti, poboljšanje kvaliteta i smanjenje troškova koje omogućava workflow sistem su dokazi značaja ove komponente. Sledeća bitna komponenta koju softveri treba da poseduju je komponenta izveštavanja i analize. Zatim sledi komponenta baziranosti na mreži pod čim se ispituje da li softver ima mogućnost pristupa putem intraneta ili intrneta. Poslednja komponenta po kojoj se

softver ispituje jeste da li je softver licenciran ili besplatan za upotrebu.

U tabeli 1 prikazano je jedanaest softvera sa aspekta obuhvata svake od navedenih devet komponenti [3].

Tabela 1. *Uporedna analiza softvera za upravljanje projektima*

	Kolaboracija	Sistem za praćenje	Raspoređivanje	Projektni portfolio menadžment	Upravljanje resursima	Upravljanje dokumentima	Workflow sistem	Izveštavanje i analize	Web-based	Licencirano
Cooper project	✓	✓	✓	✓	✓	✓		✓	✓	✓
HP Software Division	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
JIRA	✓	✓	✓				✓	✓	✓	✓
Merlin	✓	✓	✓	✓	✓	✓		✓	✓	✓
Microsoft Office Project Server	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Project			✓		✓			✓		✓
Microsoft SharePoint Server	✓	✓	✓			✓	✓		✓	✓
Onepoint Project	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Open ERP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Primavera Project Planner	✓	✓	✓	✓	✓	✓			✓	✓
Project Open	✓	✓	✓	✓	✓	✓	✓	✓	✓	
SAP PPM	✓	✓	✓	✓	✓	✓			✓	✓
WorkPLAN Enterprise	✓	✓	✓	✓	✓	✓	✓	✓		✓

4.4. Zaključak uporedna analiza softvera za upravljanje projektima

Iz rezultata analize može se zaključiti da većina softvera poseduje na samo osnovne već i napredne opcije za upravljanje projektima.

Budući da većina ovih softvera u određenoj meri nastaje kao rezultat saradnje softverske kompanije i klijenta može se doći do zaključka da kompanije već uveliko imaju razvijene potrebe za naprednim korišćenjem softvera za upravljanje projektima.

Trend upravljačkih potreba i nivo razvijenosti softvera ukazuju na to da će se ova industrija dalje razvijati ubrzanim tempom u funkciji unapredjenja efikasnosti, efektivnosti i ekonomičnosti poslovanja kompanija, privrednih subjekata i organa državne uprave [3].

5. JIRA PLATFORMA ZA UPRAVLJANJE PROJEKTIMA

Jira je popularna platforma za upravljanje projektima i zahtevima korisnika koja se koristi u različitim industrijama, uključujući softverski razvoj, IT, proizvodnju i marketinške agencije.

Jira pruža alate za planiranje, praćenje i analizu projekata. Korisnici mogu da kreiraju i prihvate zahteve korisnika (tikete), da ih raspoređuju i praćenju njihovog statusa i napretka. Takođe, Jira omogućava timovima da komuniciraju i saradjuju na projektima putem integrisanih alata za kolaboraciju.

6. PREGLED FAZA RAZVOJA JEDNOG KONKRETNOG FEATURE-A

Od ideje do realizacije jednog konkretnog feature-a na projektu potrebno je proći kroz različite faze. Faze kroz koje prolazimo su: kreiranje tiketa od strane project manager-a, upoznavanje developera sa tiketom i procenjivanje (estimiranje) vremena potrebnog za izradu nove funkcionalosti (feature-a). Sledeća faza je implementacija rešenja od strane developera, nakon te faze ide testiranje rešenja od strane testera. Ukoliko je tester pronašao neku grešku unutar rešenja developer ispravlja grešku i ponovo šalje rešenje na testiranje. Project manager i Product owner proveravaju da li je ticket realizovan u skladu sa očekivanjima i daju dozvolu za release-ovanje. U ovom poglavlju obrađujemo detaljno sve gore navedene faze. Koristićemo Kanban Jira board. Kanban je jedan od najjednostavnijih okvira koji se koristi, jer omogućava menadžerima projekata da efikasno upravljaju i prate svoje projekte.

6.1. Kreiranje tiketa od strane project manager-a

Project manager kreira ticket u kom precizno objašnjava šta je potrebno softverski implementirati, kako treba da izgleda flow i koji su mogući use case-ovi, takođe postavlja unutar tiketa dizajn (ako postoji) ili primer kako feature treba da izgleda. Kreirani tiket Project manager postavlja u Backlog kolonu. U zavisnosti od organizacije rada koju tim koristi ponekad tiketi iz backloga prolaze kroz dodatne faze definisanja na određenim sastancima koji se nazivaju refinement sastanci, radi jednostavnosti primera ovu fazu nećemo obrađivati.

6.2. Estimacija tiketa

Unutar ove faze Developer estimira tj. procenjuje vreme potrebno za izradu zadatka i takođe ako je potrebno traži dodatna pojašnjenja ukoliko ima nekih nedoumica, uglavnom ova pojašnjenja traži od Project Manager-a ili Product Owner-a, ali ponekad u zavisnosti od strukture i organizacije tima moguće je dodatna pojašnjenja tražiti i od klijenta. Estimacija zadatka se uglavnom vrši u satima ili story poenima. Obe metodologije su podjednako zastupljene.

6.3. Implementacija rešenja od strane developera

U ovoj fazi Developer implementira rešenje u tejnologiji u kojoj se razvija projekat, u nastavku je prikazana uproštena implementacija feature-a.

6.4. Testiranje implementirane funkcionalnosti

Testiranje softvera obuhvata različite vrste testiranja kako bi se osiguralo da softverski proizvod neće imati funkcionalne i nefunkcionalne nedostatke, a sve u cilju smanjenja ukupnih troškova razvoja softvera, poboljšanja njegovog kvaliteta i udobnosti korišćenja. Ako program ne obavlja funkciju za koju je razvijen, postaje neprofitabilan i postoji šansa da će ga potencijalni korisnik zameniti konkurentnim softverom.

6.5. Acceptance testing (testiranje od strane Product Manager-a i/ili Product Ownera)

Project manager i Product owner proveravaju da li je ticket realizovan u skladu sa očekivanjima. Ako tiket zadovoljava sve kriterijume prebacuju ga iz statusa „Testing" u status „Done". Nakon toga tiket tj. Funkcionalnost ide na release odnosno puštanje u produkciju.

7. PREGLED JIRA JQL JEZIKA ZA NAPREDNU PRETRAGU

JQL je skraćenica od Jira Query Language i najmoćniji je i najfleksibilniji način za traženje vaših tiketa na Jira platformi. JQL je za sve: programere, testere, agilne menadžere projekata i poslovne korisnike. U nastavku sledi pregled Jira JQL jezika za naprednu pretragu.

7.1. Osnovne i napredne pretrage u Jiri

Postoje dve vrste pretraga u Jiri: osnovna i napredna. Osnovne pretrage, predstavljaju skup obrazaca koje možete popuniti, kao što su naziv projekta, tip problema, status i primalac. Osnovna pretraga može biti korisna za dobijanje pregleda na visokom nivou o vašim problemima i statusu [6]. Napredno pretraživanje je mesto gde ćete ući u JQL, koristeći ga za formiranje upita. Upiti su niz jednostavnih elemenata nanizanih zajedno da formiraju složenije pitanje. Upit ima tri osnovna dela: polja, operatore i vrednosti.

7.2. Važne ključne reči i operatori

Atlassian je napravio JQL referencu u kojoj možete pronaći sve ključne reči, operatore i ostale potrebne informacije. Evo nekih od najčešćih ključnih reči i operatora koje ćete koristiti:

- AND - Primer upotrebe: project = Collaboration AND status = "In Progress" - Ovo će vratiti samo tikete koji odgovaraju obema klauzolama (deo su projekta saradnje i njihov status je podešen na „U toku“).
- OR - Primer upotrebe: project = Collaboration OR status = "In Progress" - Vraća sve tikete iz projekta saradnje ili koji imaju status postavljen na „U toku“.
- IS - Primer upotrebe: opis IS EMPTY - Ovo će vratiti sve tikete koji nemaju opis.
- != - Primer upotrebe: status != "To Do" - Vraća sve probleme osim onih čiji je status postavljen na „To Do“.
- >= - Primer upotrebe: "Story Points" >= 5. Pronađite sve tikete koji imaju poene priče koji su veći ili jednaki datoj vrednosti. Postoje još i opcije >, <, <=.
- IN - Primer upotrebe: status IN ("To Do", "In Progress", "Closed"). Ovo će pronaći sve probleme koji imaju status „To Do“, „In Progress“ ili „Closed“.
- Reverse - Primer upotrebe: NOT IN

7.3. Konstrukcija JQL upita

Jednostavan upit u JQL-u (takođe poznat kao 'klauzula') sastoji se od polja, praćenog operatorom, praćenom jednom ili više vrednosti ili funkcija. Na primer: project = "TEST" - Ovaj upit će pronaći sve probleme u projektu „TEST“. Koristi polje „project“, operator EQUALS i vrednost „TEST“. Složeniji upit može izgledati ovako: project = "TEST" AND assignee = currentuser() - Ovaj upit će pronaći sve probleme u projektu „TEST“ gde je primalac trenutno prijavljen korisnik. Koristi polje „project“, operator EQUALS, vrednost „TEST“, ključnu reč „AND“ i funkciju „currentuser()“ [6].

7.4. Postavljanje prioriteta operatora

Možete koristiti zagrade u složenim JQL izjavama da biste primenili prioritet operatora. Na primer, ako želite da pronađete sve rešene probleme u projektu 'SysAdmin', kao i sve probleme (bilo koji status, bilo koji projekat) koji su trenutno dodeljeni administratoru sistema (bobsmith), možete koristiti zagrade da biste primenili prioritet logički operatori u vašem upitu, tj. (status= resolved AND project=SysAdmin) OR assignee= bobsmith [6].

7.5. Rezervisane reči i znakovi

JQL ima listu rezervisanih znakova: space (" ") + . , ; ? | * / % ^ $ # @ [ ]. Ako želite da koristite ove znakove u upitima, potrebno je okružiti ih navodnicima (možete koristiti ili pojedinačne navodnike (') ili dvostruke navodnike ("")), ako pretražujete tekstualno polje i znak se nalazi na listi rezervisanih znakova za tekstualne pretrage. Ili Ako želite da koristite ove znakove u upitima, potrebno je ispred njih navesti dve obrnute kose crte. Na primer: version = "[example]". Neke od rezervisane reči u JQLu su: "a", "an", "abort", "access", "add", "after", "alias", "all","alter", "and", "any", "are", "as"...

8. ZAKLJUČAK

Zaključak ovog rada potvrđuje važnost efikasnog upravljanja projektima u funkcionisanju IT firme i potrebu za korišćenjem softvera za upravljanje projektima. U tom smislu, Jira platforma se pokazala kao korisno i jednostavno sredstvo za praćenje projekata, te bi se njeno korišćenje moglo preporučiti IT firmama.

9. LITERATURA

[1] Seminarski rad Predmet: Upravljanje projektima Profesor: Dr Miloš Petronijević

[2] https://kissflow.com/ 5 Phases of Project Management – A Complete Breakdown

[3] Uporedna analiza softvera za upravljanje projektima: http://spin.fon.bg.ac.rs/doc/ret/SPIN%202011/Sekcije/06upr%20projektima-pdf/605_UPOREDNA%20ANALIZA%20SOFTVERA%20ZA%20%20UPRAVLJANJE%20PROJEKTIMA.pdf

[4] Sarah T., A Comprehensive Guide to Project Management Solutions, 2010

[5] Patrick Li, JIRA 4 Essentials , 2011

[6] https://www.atlassian.com/blog

Kratka biografija:

Nemanja Pualić rođen je u Novom Sadu 1998. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke odbranio je 2023.god.
kontakt: pualic.nemanja@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/23BE28Jovanovic

ПОБОЉШАЊЕ РЕЗОЛУЦИЈЕ ТЕРМАЛНЕ КАМЕРЕ КОРИШЋЕЊЕМ СТАНДАРДНОГ *CMOS* КАМЕРА МОДУЛА

IMPROVING RESOLUTION OF THE THERMAL CAMERA BY USING A STANDARD *CMOS* CAMERA MODULE

Живорад Јовановић, *Факултет техничких наука, Нови Сад*

Област – ПРИМЕЊЕНА ЕЛЕКТРОНИКА

Кратак садржај – *У овом чланку је описана имплементација побољшања резолуције термалне камере. Потребно је комбиновати приказе са OV7670 камера модула и са MLX90640 термалне камере да би се добио термални приказ са доста више детаља на слици.*

Кључне речи: *OV7670 камера модул, MLX90640 термална камера, Mikromedia 7 развојно окружење*

Abstract – *This paper describes implementation of thermal camera resolution enhacement. It is necessary to combine the views from the OV7670 camera module and from the MLX90640 thermal camera to get a thermal view with much more details on the image.*

Keywords: *OV7670 camera module, MLX90640 thermal camera, Mikromedia 7 development environment*

1. УВОД

Како је технологија временом напредовала, данас је тешко замислити човека савременог доба који нема бар неки „паметни“ уређај у свом дому. Данас се у домаћинствима скоро па и не може наћи мобилни уређај који у себи нема интегрисану камеру. Осим у мобилним телефонима, камере се сада могу наћи у свим другим савременим системима попут преносних рачунара (лаптопова), таблет рачунара и других. Запањујућа чињеница је да је данашња технологија, након развоја камера драстично великих резолуција отишла још један корак даље. У складу са тим долази се до појма **термовизије** (од грчке речи *termo* – топло и латинског глагола *video,videre* – видети, гледати). У буквалном смислу овај појам се може превести као „гледање топлоте“. На основу овога, термовизија дефинише приступ којим се врше снимања топлоте објеката. Термовизијско снимање представља бесконтактни метод којим се може регистровати емитовање топлоте односно инфрацрвено зрачење које сва тела емитују у већој или у мањој мери. Овакве врсте камера се зову термалне камере [1]. У данашње време не постоје термалне камере великих резолуција, чиме би се могло приказати доста више детаља на слици.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Јован Бајић.

Циљ рада јесте да се приказ са термалне камере резолуције 32x24 пиксела представи помоћу *OV7670* камера модула који снима у резолуцији 320x240 пиксела. Притиском на екстерни тастер појављује се комбинација приказа слике са термалне камере и са *OV7670* камера модула.

2. *OMNIVISION 7670* КАМЕРА МОДУЛ

За реализацију система је искоришћен камера модул ознаке *OV7670* што представља нисконапонски *CMOS* уређај који пружа пуну функционалност *VGA* камере и процесора слике у малом кућишту [2].

OV7670 камера модул шаље податке у паралелном синхроном формату. Пре свега потребно је прво доставити сигнал такта на *XCLK* пин камера модула. Након што камера добије сигнал такта на *XCLK* пину, почеће да активира своје пинове *VSYNC, HSYNC/HREF* и *D0-D7* пинове. Подаци о пикселима се узоркују на растућој ивици *PCLK* сигнала и у времену када је *HREF* сигнал на високом логичком нивоу. Растућа ивица *HREF* сигнала сигнализира почетак линије за читање док опадајућа ивица сигнализира крај линије. Силазна ивица *VSYNC* сигнала сигнализира почетак оквира а растућа ивица сигнализира крај оквира [3].

2.1 *YUV* простор боја

OV7670 камера модул подржава *RGB* као и *YUV* формат фотографије. За снимање слике са овог камера модула одабран је *YUV* формат боја јер даје више информација од приказа саме конкретне боје на излазу модула. У *YUV* простору боја, *Y* параметар представља монохроматски сигнал који се односи само на осветљеност пиксела. Параметри U и V представљају конкретну боју и називају се још црвена пројекција и плава пројекција боје [4].

3. *MLX90640* ТЕРМАЛНА КАМЕРА

Термовизијска камера, позната и као инфрацрвена камера или термална камера, је уређај који формира слику користећи инфрацрвено зрачење. Инфрацрвено зрачење се емитује од стране свих објеката са температуром изнад апсолутне нуле, стога термографија омогућава да се види нечија животна средина, са или без видљивог осветљења. Количина зрачења, емитована од стране објеката повећава се са температуром, стога се помоћу термографије могу видети разлике у температури. У складу са претходно реченим, може се закључити да термална камера детектује и мери инфрацрвену енергију објеката [5].

IR 3 GRID клик плочица је опремљена са *MLX90640ESF-BAA* термалном камером која представља *IR* сензор резолуције 32x24 пиксела развијен од стране фирме *Melexis*.

Ова термална камера садржи *PTAT* (енг. *proportional to absolute temperature*) компензациони сензор.

У случају *PTAT* принципа мерења, у електричном колу сензора се генерише напон који директно зависи од температуре.

Видно поље ове камере износи 110°x75° са елементима распоређеним у 32x24 мрежу. Сваки пиксел (сензор) мери температуру свог видног поља на основу чега се креира термална слика односно рачуна се температура сваке тачке у видном пољу [6].

3.1 *HSV* модел боја

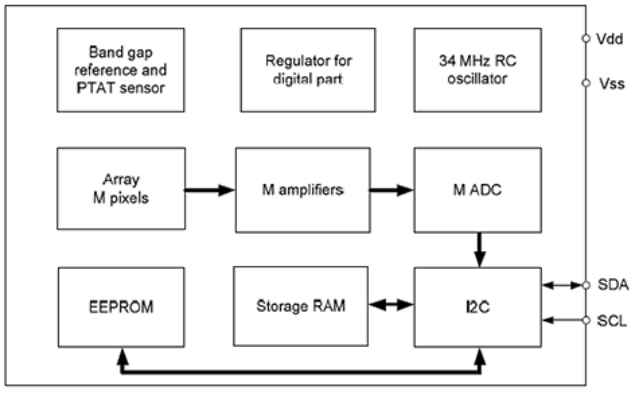
Код овог простора боја, *H* параметар представља конкретну боју односно тон боје, *S* параметар представља засићеност боје и *V* параметар је вредност боје.

Свака боја коју човек запази, може описати на основу њеног тона, засићености и светлине. *H* параметар односно тон боје се може представити на кружници у опсегу 0°-360° и представља локацију доминантне таласне дужине (таласна дужина око које је концентрисана највећа енергија).

Како је речено, засићеност боје се дефинише *S* параметром и представља садржај беле светлости у боји. *S* параметар може имати вредности у опсегу 0% - 100%. *V* параметар представља сјајност боје и одређује меру колико је боја светла или тамна. Као и *S* параметар, *V* параметар може бити у опсегу 0% - 100% [7].

3.2 Блок шема *MLX90640* термалне камере

На слици 1 је приказана блок шема *MLX90640* термалне камере. Као што показује слика 1, излази свих сензора (*IR* елемената) се смештају у интерну *RAM* меморију и подаци са сензора се могу преузети из *RAM* меморије путем I^2C интерфејса.
Ови излазни сигнали се претходно појачавају и пропуштају кроз *AD* конверторе. Такође, путем I^2C интерфејса се могу прочитати подаци из *EEPROM* меморије. Што се тиче брзине I^2C комуникације, овај сензор подржава фреквенције сигнала такта до 1MHz [8].



Слика 1. *Блок шема MLX90640 термалне камере [8]*

3.3 Организација меморије *MLX90640* термалне камере

MLX90640 термална камера садржи више различитих меморијских модула у зависности од намене. Потребно је издвојити два најзначајнија блока меморије у овом случају а то су *RAM* меморија и *EEPROM* меморија. Што се тиче *RAM* меморије, додељене су јој меморијске адресе у опсегу 0x0400 – 0x07FF и у ову меморију се смештају вредности тренутних стања пиксела. *EEPROM* меморији су додељене меморијске локације у опсегу адреса 0x2400 – 0x273F што је еквивалентно са 832 меморијске локације. У ову меморију се смештају параметри компензације и калибрације сензора. При укључењу уређаја, ови параметри се смештају у *RAM* меморију чиме се обезбеђује правилан рад термалне камере [8].

3.4 Обрасци читања стања пиксела *MLX90640* термалне камере

Постоје два начина распореда пиксела односно два начина читања тренутног стања пиксела са *MLX90640* термалне камере:

- *Chess pattern mode* (режим шаховског узорка).
- *TV interleave mode.*

У случају режима шаховског узорка, као што само име каже пиксели су организовани тако да се прва подстраница односи на пикселе са поља једне боје (на пример сва бела поља на шаховској табли), док друга подстраница обухвата пикселе са поља друге боје (сва црна поља са шаховске табле). Ако је у питању *TV interleave* режим, стања пиксела се читају тако да првој подстраници одговарају пиксели сваке друге линије, док при читању следеће подстранице остају да се прочитају стања пиксела из осталих линија [8].

4. *MIKROMEDIA 7* РАЗВОЈНИ СИСТЕМ

За реализацију овог рада искоришћен је *Mikromedia 7* развојни систем који је развијен од стране фирме *Mikroelektronika*. Овај развојни систем на себи садржи микроконтролер ознаке *STM32F746ZG.* Изглед развојног система приказан је на слици 2. Главни периферијски модул на овом систему представља *TFT* дисплеј резолуције 800x480 пиксела чија дужина дијагонале износи 7 инча и осетљив је на додир. Комуникација између микроконтролера и самог дисплеја омогућена је коришћењем *SSD1936* графичког контролера који се такође налази интегрисан на развојном систему [9].

Слика 2. *Mikromedia 7 развојни систем [9]*

4.1 *STM32F746ZG* микроконтролер

STM32F746ZG микроконтролери су базирани на 32-битном *ARM Cortex – M7* језгру високих перформанси чија је максимална фреквенција рада до 216MHz. Такође имају стандардне и напредне комуникационе интерфејсе: до четири I^2C модула, шест *SPI* модула, четири *UART* модула, два *CAN* модула, два *SAI* модула, *DCMI* интерфејс (интерфејс за повезивање камера модула), *LCD-TFT* дисплеј контролер, *SDMMC* интерфејс (омогућава комуникацију са *SD* меморијским картицама и са *SD* улазно/излазним уређајима) [10].

5. РЕАЛИЗАЦИЈА ЗАДАТКА

5.1 *STM32CubeMX* конфигуратор кода

Задатак је реализован у *STM32CubeIDE* развојном окружењу користећи *STM32CubeMX* конфигуратор кода. Главна предност овог конфигуратора је то што се сва иницијална подешавања врше графичким путем. *STM32CubeMX* конфигуратор кода генерише иницијални код у ком су ажуриране вредности одређених регистара у складу са траженим захтевима.

5.2 Идеја реализације задатка

Приказ са *OV7670* камера модула се може посматрати као матрица пиксела која садржи 240 врста и 320 колона, док је код термалне камере у питању 32x24 матрица пиксела. Ове две матрице треба довести у међусобну везу на основу које ће се извршити мапирање пиксела са термалне камере на пикселе *OV7670* камера модула.

Како је већ наведен однос резолуција обе камере, није тешко доћи до закључка да један пиксел са термалне камере представља 10x10 матрицу пиксела са *OV7670* камера модула. Дакле, 320x240 матрица пиксела се треба поделити на низ 10x10 матрица при чему свака подматрица представља један пиксел са термалне камере.

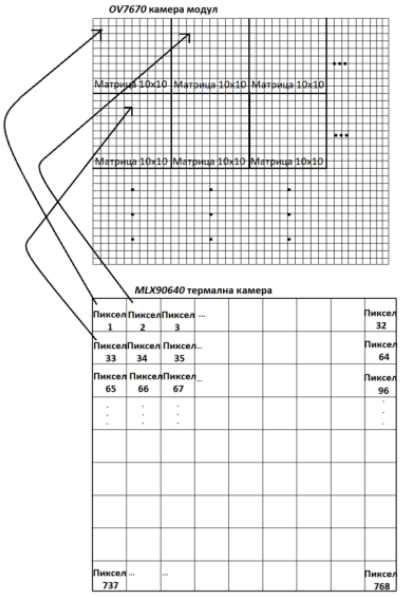
Следеће што треба урадити је приступити *Hue* параметру за сваки пиксел термалне камере, при чему ће се вредност овог параметра додељивати 10x10 матрицама пиксела са *OV7670* камере.

Принцип мапирања пиксела заснован на претходно објашњеном поступку приказан је на слици 3. Са термалне камере треба добити податке о *Hue* параметру док се са *OV7670* камера модула користи само *Y* параметар који даје информацију о осветљености сваког појединачног пиксела.

Комбинацијом *Hue* и *Y* параметара добија се комбинација приказа са обе камере, што и јесте циљ задатка.

5.3 Софтверска реализација задатка

Програм је написан тако да постоје два приказа на *TFT* дисплеју, на једном приказу се налази оно што снима само термална камера, док је на другом приказу циљ задатка а то је комбинација приказа слике са *OV7670* камера модула и са *MLX90640* термалне камере.



Слика 3. *Мапирање пиксела са MLX90640 термалне камере на OV7670 камера модул*

Једна од најбитнијих функција за цртање приказа са камера представља *SSD1936_WriteData* функција којој се као параметар прослеђује боја у оквиру 16-битног податка. Пре приказивања саме боје на дисплеју потребно је извршити конверзије простора боја у *RGB* формат. У ту сврху су искоришћене функције *YUVtoRGB* односно *HSVtoRGB*. Основне формуле од којих се полази при конверзији *YUV* простора боја у *RGB* простор представљају формуле (1) - (3). На основу ових формула израчунају се компоненте *RGB* простора боја [4].

$$R = Y + 1.140V \tag{1}$$

$$G = Y - 0.395U - 0.581V \tag{2}$$

$$B = Y + 2.032U \tag{3}$$

У случају конверзије *HSV* простора боја у *RGB* простор боја, користе се формуле (4) – (8).

$$C = V \times S \tag{4}$$

$$X = C \times (1 - |(H/60^\circ) \bmod 2 - 1|) \tag{5}$$

$$m = V - C \tag{6}$$

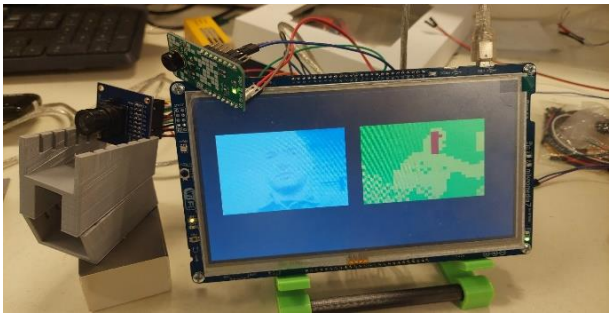
$$(R', G', B') = \begin{cases} (C, X, 0), 0^\circ \le H < 60^\circ \\ (X, C, 0), 60^\circ \le H < 120^\circ \\ (0, C, X), 120^\circ \le H < 180^\circ \\ (0, X, C), 180^\circ \le H < 240^\circ \\ (X, 0, C), 240^\circ \le H < 300^\circ \\ (C, 0, X), 300^\circ \le H < 360^\circ \end{cases} \tag{7}$$

$$(R, G, B) = \begin{cases} (R' + m) \times 255 \\ (C' + m) \times 255 \\ (B' + m) \times 255 \end{cases} \tag{8}$$

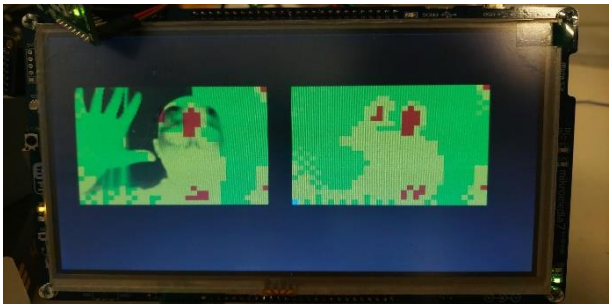
На крају конверзије, вредности *R*, *G* и *B* компоненти ће бити у опсегу 0-255 [11].

6. РЕЗУЛТАТИ ТЕСТИРАЊА

На сликама 4. и 5. су приказани резултати тестирања у случају основног приказа са *OV7670* камера модула односно у случају коначног система. Са слике 5. се може видети да није било могуће преклапање слика са обе камере из разлога што је коришћена термална камера са видним пољем од 110°x75°. Коришћењем термалне камере са мањим видним пољем, овакав систем би се могао унапредити.



Слика 4. *Изглед система у случају основног приказа са OV7670 камера модула*



Слика 5. *Изглед коначног система*

7. ЗАКЉУЧАК

У складу са свим претходно изнетим чињеницама, коначан систем који је тема овог рада је реализован у складу са захтевима и испуњава жељену функционалност. Разлика између температуре околине и температуре људског тела је мала, па стога температура околине може стварати шумове чиме се добијају делимичне промене у боји на слици са термалне камере. Зато би сва тестирања најбоље било вршити у условима где су веће разлике у температури(на пример где је додатно охлађена просторија), на тај начин добијају се „чистије“ слике са термалне камере.

Реализација оваквог система би се могла унапредити коришћењем термалне камере са мањим видним пољем (55°x35°). Са оваквим параметром би био једноставнији процес преклапања слика са једне и друге камере јер камере не би морале бити на већој међусобној удаљености да би снимале исту сцену.

Такође, уместо коришћења екстерних тастера, систем би се могао додатно унапредити реализацијом графичког корисничког интерфејса.

Мотивација за реализацију оваквог задатка је та што у данашње време не постоје термалне камере великих резолуција, па је оваквим комбиновањем приказа ближе демонстрирано како би то изгледало.

8. ЛИТЕРАТУРА

[1] Појам термовизије и основни принципи рада термалних камера, https://www.wikiwand.com/sr/Termovizija [Приступљено у фебруару 2023.]

[2] Основни принцип рада *OV7670* камера модула, https://www.elprocus.com/cmos-ov7670-camera-module/ [Приступљено у фебруару 2023.]

[3] Принцип добијања слике помоћу *OV7670* камера модула, http://embeddedprogrammer.blogspot.com/-2012/07/hacking-ov7670-camera-module-sccb-cheat.html [Приступљено у фебруару 2023.]

[4] *YUV* простор боја (основни појмови), https://dexonsystems.com/blog/rgb-yuv-color-spaces [Приступљено у фебруару 2023.]

[5] Основни појмови о термалним камерама, https://www.antenall.rs/sr/vesti/sta-je-termalna-kamera-za-detekciju-telesne-temperature; [Приступљено у фебруару 2023.]

[6] *IR 3 GRID click* плочица (основне информације), https://www.mikroe.com/ir-grid-3-click, [Приступљено у фебруару 2023.]

[7] Ј.Бајић, “Лабораторијске вежбе из Оптоелектронике“, https://www.optolab.ftn.uns.ac.rs/images/NASTAVA/OE/files/lab/pdf/2021/8-Senzor-boje.pdf [Приступљено у фебруару 2023.]

[8] *Datasheet MLX90640* сензора, https://download.mikroe.com/documents/datasheets/MLX90640.pdf [Приступљено у фебруару 2023.]

[9] *Mikromedia 7* развојни систем, https://www.mikroe.com/blog/mikromedia-7-stm32f7-user-manual [Приступљено у фебруару 2023.]

[10] *STM32F746ZG* микроконтролер *datasheet*, https://www.st.com/resource/en/datasheet/stm32f746zg.pdf [Приступљено у фебруару 2023.]

[11] Формуле за конверзију *HSV* простора боја у *RGB* простор боја, https://www.rapidtables.com/convert/color/hsv-to-rgb.html [Приступљено у марту 2023.]

Кратка биографија:

Живорад Јовановић рођен је 15.02.1997. у Ужицу. Завршио је Техничку школу у Ужицу 2016. год. Уписао је Факултет техничких наука у Новом Саду 2017. год. на студијском програму Енергетика, електроника и телекомуникације. У октобру 2021. год. завршио је основне академске студије, након чега уписује мастер студије на усмерењу Примењена електроника.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/23BE29Katanic

CNC MAŠINA ZA IZRADU ŠTAMPANIH PLOČICA

CNC MACHINE FOR MAKING PRINTED CIRCUIT BOARDS

Miroslav Katanić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je prikazan način implementacije CNC mašine pokretane putem koračnih motora. Za upravljanje su korišćeni Arduino Uno, prilagodna pločica i drajveri koračnih motora. Upravljanje je vršeno preko programskih paketa Altium Designer, FlatCam i Candle.*

Ključne riječi: *CNC mašina, drajveri za koračne motore, koračni motori, motori sa četkicama.*

Abstract – *This paper describes the method of implementation of a CNC machine driven by stepper motors. Arduino Uno, CNC Shield and stepper motor drivers were used for control. Software packages for control is Altium Designer, FlatCam and Candle.*

Keywords: *CNC machine, Stepper motor drivers, Stepper motor, motor with brushes.*

1. UVOD

Štampana ploča (*engl*. Printed Circuit Board) predstavlja pasivnu električnu komponentu čiji je osnovni zadatak ostvarivanje veza između elektronskih komponenti (otpornici, kondenzatori, tranzistori itd.). Pored električne veze na štampanoj ploči koja se ostvaruje uz pomoć provodnih putanja (provodnih vodova), ostvaruju se i mehaničke veze između ploče i lemljenih komponenti uz pomoć provodnih podloga u oblicima koji su dizajnirani za prihvatanje priključaka komponente. Dakle komponente se leme na ploču kako bi se električno spojile i mehanički pričvrstile za nju.

Do same proizvodnje ploče potrebno je prvo izvršiti izradu potrebnih biblioteka sa komponentama od interesa, zatim projektovanje električne šeme, a na kraju i izrada rasporeda komponenti (*engl*. Layout). Sve prethodno navedene korake moguće je ostvariti u okviru određenog softverskog alata koji obavlja ogroman posao, neki od poznatijih alata su Altium Designer, Eagle, KiCad itd.

Standardna štampana ploča sastoji se od ploče izolacionog materijala i povrh njega provodnog bakarnog sloja. Ideja jeste da nakon izrade pločice, izrađene provodne staze predstavljaju fiksirane žice i međusobno su izolovane vazduhom i materijalom izolacione podloge. Takođe, površinu štampane ploče dobro je zaštiti premazom koji štiti bakar od korozije i oksidacije i pored toga smanjuje šansu za pojavu kratkog spoja između provodnih staza ili zalutalih golih žica. Ovaj sloj se naziva lemno otporni sloj ili lemna maska.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Jovan Bajić, vanr. prof.

U pogledu broja slojeva ploče, u zavisnosti od složenosti ili potrebe, ploče mogu biti jednoslojne ili višeslojne. Kod višeslojnih ploča imamo da su slojevi međusobno povezani uz pomoć električnih tunela (engl. via). Nakon dvoslojnih došlo se do četvoroslojnih gdje se najčešće dva sloja koriste za napajanje i masu, dok se druga dva koriste za ožičavanje signala komponenti.

Razlikuju se dva tipa komponenti na osnovu načina njihovog montiranja na ploču i to komponente „kroz otvor“ (engl. Through Hole) i komponente površinskog montiranja (engl. sourface mounted). Jedna ploča može koristiti i jedan i drugi tip komponenti. Komponente „kroz otvor“ se sve manje koriste, međutim kod nekih komponenti su skoro pa nezamjenljive, recimo konektori, elektrolitski kondenzatori i slično. Dok površinsku montažu koristimo mnogo više zbog toga što zauzimaju mnogo manje prostora, pouzdanije su, pri površinskoj montaži koristi se napredne brze tehnologije lemljenja itd.

Glavni zadatak ovog rada jeste izrada mašine za pravljenje štampanih pločica. U nastavku rada će biti prikazana mašinska konstrukcija (odnosno od kojih je komponenti je sastavljena mašina). Upravljanje *CNC* mašinom je vršeno drajverima koračnih motora koje je pokretao *Arduino Uno*. Zatim će biti objašnjeni programski paketi koji upravljaju mašinom, a na samom kraju će biti predstavljeni rezultati testiranja različitih vrsta glodala.

2. NAČINI IZRADE ŠTAMPANIH PLOČICA

Izrada štampanih ploča se može realizovati ručno i mašinski. Prije same izrade pločica i jedne i druge realizacije potrebno ih je projektovati u određenom programskom paketu kao što je već u prethodnoj glavi navedeno (Altium Designer, Eagle, KiCad itd.). U tim programima potrebno je prvo izvršiti izradu biblioteka komponenti koje se koriste, zatim projektovanje šematika, raspored komponenti na samoj ploči i na kraju povezivanje komponenti određenom širinom vodova.

2.1. Ručna izrada štampanih pločica

Kao što se prema samom imenu može zaključiti, ručna vrsta izrade je prva počela da se primjenjuje i predstavlja preteču gotovo svih današnjih modela izrade pločica. Ugrubo rečeno može se reći da se svaki ručni postupak odlikuje istim postupcima pripreme koji uključuju sledeće procese. Na podesan način potrebno je prvo izvršiti isijecanje pločice prema potrebama koje su diktirane zahtjevima tog projekta. Procesa sječenja pločice može se vršiti raznim alatima i tehnikama gdje se svaka odlikuje svojim karakteristikama, od kojih su glavne cijena i preciznost. Nakon što su željene dimenzije postigunte, potrebno je

pristupiti procesu čišćenja bakarne površine odnosno odstraniti sve nečistoće koje se nalaze na istoj. Nakon što je izvršena priprema pločice može se pristupiti procesu izrade ručnim putem koji mogu biti: korišćenjem flomastera ili laka za nokte, korišćenje rapidografa, transfer papira i izrada fotopostupkom [1]. Izrada fotopostupkom predstavlja najprecizniju metodu od prethodno navedenih.

2.2. Mašinska izrada štampanih pločica

Pod pojmom mašinske izrade podrazumjevamo da ključni dio pri izradi štampane ploče ima najčeće neka „*CNC*“ mašina. Te mašine možemo klasifikovati prema alatima i materijalima koje koriste, gdje oni mogu biti različiti počevši od glodala, burgije pa sve do naprednijih mašina koje rade sa laserima velike snage.

Pošto je tema ovog rada izrada CNC mašine za proizvodnju štampanih pločica u nastavku rada će prvo biti napravljen osvrt na fizički dio uređaja, elektroniku, upravljanje kao i softversko prilagođenje alata.

3. MAŠINA ZA IZRADU ŠTAMPANIH PLOČICA

Tema ovog rada jeste postupak projektovanja i sklapanja sistema čija je uloga proizvodnja štampanih pločica. Potrebno je bilo sastaviti mašinsku konstrukciju koja je upravljana putem tri motora (*engl*. Stepper Motor), dok četvrti motor upravlja alatom koji može biti različit u zavisnosti od potrebe, to može biti glodalo, bušilica i sl.

Kao što se može naslutiti, prema vrsti alata koji se koristi mogućnosti ovog sistema mogu biti procesi bušenja (kao alat koristi burgiju) ili procesi glodanja (u slučaju da je alat glodalo) ili eventualno proces sječenja. Sva tri navedena procesa predstavljaju značajan postupak u okviru procesa proizvodnje štampane pločice.

Čitav sistem se ugrubo rečeno sastoji iz mašinske konstrukcije sastavljene iz kvadratnih aluminijumskih profila, tri navojna vretena za tri različite ose i nosač motora proizveden 3D štampačem. Montirana su tri pogonska motora za tri radne ose i dodatno obradni motor kao što je prikazano na slici 1. Na kraju postoji upravljački sistem koji pogoni odnosno upravlja motorima.

Slika 1. *Izgled* CNC *mašine*

Dakle, cilj je bio isprojektovati sistem koji zadovoljava sve potrebe vezane za osnovne mašinske postupke pri procesu proizvodnje štampane ploče koji podrazumjevaju sječenje, glodanje i bušenje.

4. UPRAVLJANJE *CNC* MAŠINOM

Za upravljanje CNC mašinom koristi se upravljačka jedinica čiji je zadatak da komande od strane korisnika proslijedi uređaju. Da bi se te komande proslijedile potrebno je koristiti određene programe kao što su Altium Designer, FlatCAM, Candle i drugi. O programima će biti detaljnije rečeno u nastavku. Osnovni uređaji preko kojih se vrši upravljanje su MCU (engl. Microcontroller) Arduino Uno i prilagodna pločica za koračne motore. Arduino Uno predstavlja razvojno okruženje namijenjeno učenju kao i projektima manje složenih zadataka. Pokazalo se da je za potrebe ovog zadatka bio sasvim dovoljan da se ispune svi zahtjevi.

Veoma bitnu ulogu u okviru upravljanja i napajanje koračnih motora ima prilagodna pločica koja je priključena na Arduino Uno preko priključaka za komunikaciju koji su izvedeni u ženske ljestvice, što je prikazano na slici 2.

Slika 2. *Prilagodna pločica za koračne motore* [2]

Kao što se primjećuje sa slike, ova ploča je dizajnirana za upotrebu sa Arduino Uno razvojnim okruženjem, pa samim tim dimenzije tačno odgovaraju i moguće je uklopiti priključke direktno [2].
Na prilagodnu pločicu moguće je priključiti do četiri drajvera za koračne motore, dok su za potrebu ovog rada bila potrebna tri.

5. PROGRAMI ZA POKRETANJE CNC MAŠINE

Kako bi se uspješno sproveo proces upravljanja CNC mašinom neophodno je koristiti se određenim softverskim paketima. Paketi, odnosno alati koji se koriste u ovom radu su Altium Designer, FlatCam i Candle.

Uloga programskog paketa Altium Designer je kreiranje šematika i rasporeda komponenti. Kada se sve što je potrebno generiše onda se izvozi greber format.

Programski paket FlatCam ima ulogu da učita greber format i generiše g-code. G-code predstavlja niz instrukcija prema kojima se pokreću motori CNC mašine [3]. Programski paket Candle ima ulogu slanja instrukcija motorima CNC mašine putem g-code-a koji je generisam pomoću prethodno objašnjenog programskog paketa FlatCAM.

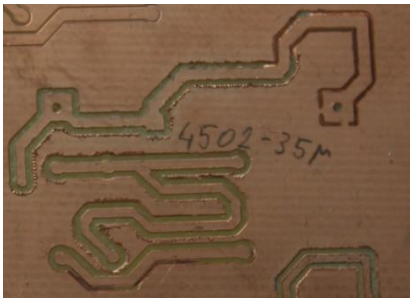
Candle program je besplatan za korišćenje, pa je iz tog razloga veoma rasprostranjen. Grafički prikazuje g-code i osnovne funkcije za pokretanje istog [4]. Candle koristi GRBL kod čiji se program pokreće preko Arduino platforme.

6. REZULTATI TESTIRANJA

Poglavlje koje slijedi posvećeno je procesu praktičnog testiraranja funkcionalnosti kao i analizi dobijenih rezultata. Testiranja su vršena na različitim postavkama, odnosno za različite ulazne parametre poput napona napajanja obradnog motora, promjena dubine prodiranja alata, kao i vrstu alata.

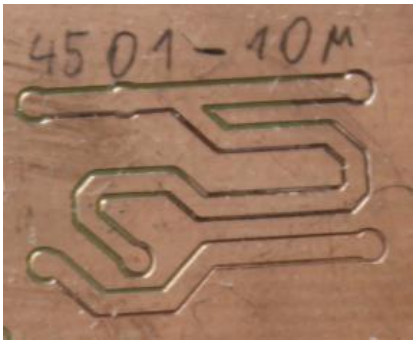
Na slici 3 je prikazan rad sa glodalom V tipa ugaonog raspona od 45° i širine 0.2mm. Dubina skidanja bakarnog sloja je 35µm. Brzina na kojoj je rađeno je 30000 obrtaja u minuti što predstavlja maksimalnu brzinu ovog motora. Kao što može da se vidi sa slike ovaj način nije uspješno izveden.

Vodovi koji treba da su izolovani jedan od drugog su u kratkom spoju. Ivice vodova su oštećene. Sloj skidanja bakra u odnosu na Z osu nije konstantan. Zbog ofseta obradnog alata po Z osi, u procesu glodanja je dolazilo do zaglavljivanja glodala.



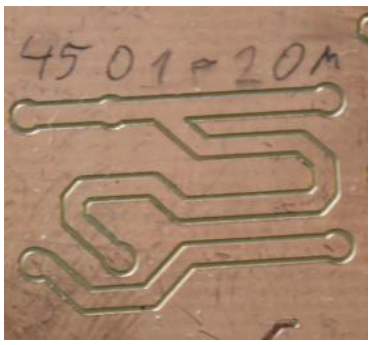
Slika 3. *Korišćenje prvog motora sa golodalom V tipa ugaonog raspona od 45° i širine 0.2mm, dubina gloldanja je 35µm*

Na slici 4 je prikazan rad sa glodalom V tipa ugaonog raspona od 45° i širine 0.1mm. Skidanje bakarnog sloja je vršeno na 10µm.
Brzina obradnog alata kojim je rađeno je 3000 obrtaja u minuti (engl. rpm) koji se napaja naponom od 12 V. Sa slike se može zaključiti da dubina na kojoj je vršeno skidanje bakarne površine nije dovoljna jer vodovi nisu dovoljno izolovani. Ivice vodova nisu oštećene.



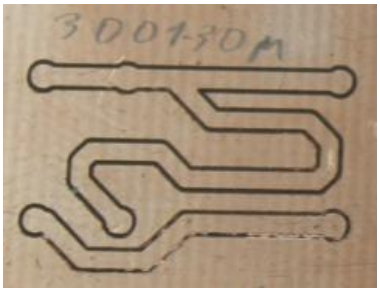
Slika 4. *Korišćenje drugog motora sa glodalom V tipa ugaonog raspona od 45° i širine 0.1mm, dubina gloldanja je 10µm*

Na slici 5 je korišćeno isto glodalo sa istom brzinom obrtanja obradnog alata kao na slici 4, samo je povećana dubina glodanja na 20µm. Sa slike može da se zaključi da su u ovom slučaju uspješno izolovani vodovi.
Dubina prodiranja je dovoljna, nije preduboka. Ivice nisu oštećene.



Slika 5. *Korišćenje drugog motora sa glodalom V tipa ugaonog raspona od 45° i širine 0.1mm, dubina gloldanja je 20µm*

Na slici 6 je prikazan rad sa glodalom V tipa ugaonog raspona od 30° i širine 0.1mm. Dubina skidanja bakarnog sloja je 30µm. Brzina obradnog alata kojim je rađeno je 3000 obrtaja u minuti (engl. rpm) naponom od 12V. Vodovi nisu dovoljno izolovani što može da se vidi sa slike. Ivice nisu oštećene.

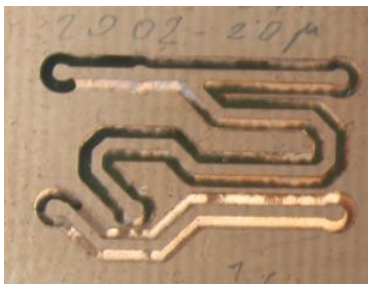


Slika 6. *Korišćenje drugog motora sa glodalom V tipa ugaonog raspona od 30° i širine 0.1mm, dubina gloldanja je 30µm, dok je brzina 3000 obrtaja u minuti*

Na slici 7 je prikazan rad sad istim glodalom kao na slici 6. Dubina glodanja je takođe ostala ista. Promjenjena je brzina obrtanja obradnog alata na oko 7500 obrtaja u minuti (*engl*. rpm) pri naponu napajanja od 29V. Sa ove slike može da se zaključi da je pri ovim parametrima zadovoljavajuće izvršeno izolovanje vodova. Ivice nisu oštećene.

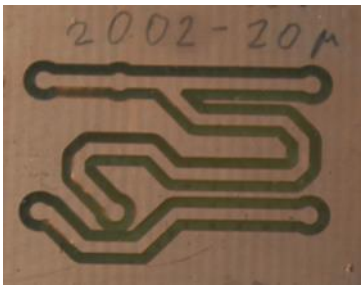
Slika 7. *Korišćenje drugog motora sa glodalom V tipa ugaonog raspona od 30°i širine 0.1mm, dubina gloldanja je 30µm, dok je brzina 7500 obrtaja u minuti*

Na slici 8 je prikazan rad sa glodalom V tipa ugaonog raspona 20° i širine 0.2mm. Dubina skidanja bakarnog sloja je 20µm. Brzina obradnog alata kojim je rađeno je 3000 obrtaja u minuti (engl. rpm) naponom od 12V. Vod nije dovoljno izolovan. Nije dovoljna dubina prodiranja bakarnog sloja. Ivice nisu oštećene.



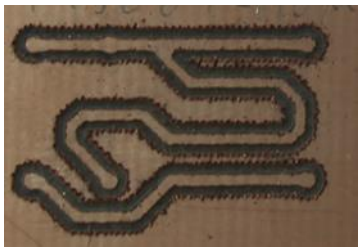
Slika 8. *Korišćenje drugog motora sa glodalom V tipa ugaonog raspona od 20° i širine 0.2mm, dubina gloldanja je 20µm, dok je brzina 3000 obrtaja u minuti*

Na slici 9 je prikazan rad sa istim glodalom kao u prethodnoj slici. Dubina skidanja bakarnog sloja je takođe ostala ista, ali je u ovom slučaju povećana brzina obrtanja obradnog alata sa 3000 na 7500 obrtaja u minuti (engl. rpm) gdje je napon napajanja povećan na 30V. Kao što se može vidjeti sa slike vodovi su korektno izolovani tako da nemaju električne veze sa spoljašnjom površinom bakra i ostalim vodovima. Ivice nisu oštećene.



Slika 9. *Korišćenje drugog motora sa glodalom V tipa ugaonog raspona od 20° i širine 0.2mm, dubina gloldanja je 20µm, dok je brzina 7500 obrtaja u minuti*

Na slici 10 je prikazan rad spiralnog glodala ugaonog raspona od 30° i širine 0.1mm. Dubina prodiranja bakarnog sloja podešena je na 40µm. Brzina pokretanja obradnog alata je podešena na 7500 obrtaja u minuti pri naponu napajanja od 30V. Vodovi su međusobno izolovani što može da se vidi sa slike. Sa slike se još može zaključiti da su ivice veoma oštećene.



Slika 10. *Koriščenje drugog motora sa spiralnim glodalom ugaonog raspona od 30° i širine 0.1mm, dubina glodanja je 40µm, dok je brzina 7500 obrtaja u minuti*

7. ZAKLJUČAK

Prvobitni cilj koji je postavljen u ovom radu podrazumjevao je izradu CNC mašine koja će biti u stanju da sprovede sve neophodne mašinske procese pri izradi štampane pločice.
Pored toga potrebno je bilo riješiti problem kontrole odnosno upravljanja, a naravno finalno i sprovesti proces testiranja čime bi se kompletirao ovaj rad.
Izrađena CNC mašina u stanju je da obavlja postupke glodanja, bušenja kao i sječenja, što su zapravo ključni procesi prilikom postupka izrade štampane pločice.
Kada se pogleda čitav sistem, treba imati na umu da je za relativno mali budžet proizvedena mašina koja je u stanju da prođe kroz sve osnovne mašinske procese u postupku proizvodnje štampane ploče.

8. LITERATURA

[1] Uputstvo za izradu štampanih pločica: http://arhiva.elitemadzone.org/t23768-Uputstvo-za-izradu-stampanih-plocica (pristupljeno u septembru 2022.)
[2] Prilagodna pločica: https://osoyoo.com/2017/04/07/arduino-uno-cnc-shield-v3-0-a4988/ (pristupljeno u novembru 2022.)
[3] Programski pakte *FlatCam*: https://www.mischianti.org/2019/02/22/design-and-mill-pcb-easy-and-cheap-part-3/ (pristupljeno u januaru 2023.)
[4] Programski paket *Candle*: https://cncphilosophy.com/candle-grbl-software-tutorial/ (pristupljeno u januaru 2023.)

Kratka biografija:

Miroslav Katanić rođen je u Derventi 1998. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjena elektronika odbranio je 2021.god.
kontakt: miroslavkatanic@uns.ac.rs

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE30Predojevic

UPOTREBLJIVOST PROGRESIVNE VEB APLIKACIJE

THE USABILITY OF THE PROGRESSIVE WEB APPLICATIONS

Dejan Predojević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su predstavljene progresivne web aplikacije kao i njihova istorija razvoja od prvog pominjanja ovog koncepta. Date su osnovne karakteristike, prednosti i mane progresivnih web aplikacija. Takođe su detaljno opisane tehnologije i korišćeni alati potrebni za razumevanje progresivnih web aplikacija. Prikazana je sve veća potreba za mobilnim rešenjima postojećih web sajtova, kao i problem razvijanja više od jedne aplikacije zbog podrške na različitim uređajima koji uzrokuje velika finansijska i vremenska ulaganja. Predstavljena je tržišna potreba koja je dovela do razmišljanja o jedno-aplikacijskom rešenju koje progresivne web aplikacije obezbeđuju. Pored navedenog opisane su i razlike između web, nativnih i progresivnih web aplikacija čija popularnost neprestano raste. U cilju demonstracije napravljena je aplikacija namenjena za rad u biblioteci. Aplikacija je razvijena uz pomoć Angulara i Ionika, a poseban akcenat je stavljen na prikazivanje mogućnosti da se od web aplikacije napravi progresivna web aplikacija, koja će moći da se koristi i na mobilnim uređajima.*

Ključne reči: *PWA, mobilna, aplikacija, web*

Abstract – *The paper presents progressive web applications as well as their history of development since the first mention of this concept. Basic features, advantages and disadvantages of progressive web applications are given. The technologies and tools used to understand progressive web applications are also described. The growing need for mobile solutions for existing websites is shown, as well as the problem of developing more than one application due to support on different devices, which causes large financial and time investments. A market need was presented that led to the idea of a one-app solution that progressive web applications provide. Differences between web, native and progressive web applications, whose popularity is constantly growing, are also described. For the purpose of demonstration, an application intended for work in the library was created. The application was developed with the help of Angular and Ionic, and a special emphasis was placed on showing the possibility of making a web application a progressive web application, which can also be used on mobile devices.*

Keywords: *PWA, mobile, appwlication, web*

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

1. UVOD

U proteklih nekoliko decenija tehnološki razvoj doveo je do velikog napretka u performansama računara kao i do revolucionarnih rešenja u razvoju mobilnih uređaja. Web aplikacije današnjice napravile su veliki iskorak ka boljem korisničkom iskustvu i dovele do toga da se sve više teži ka što boljem prilagođavanju aplikacija krajnjim korisnicima. Sve navedeno podstaklo je ljude da razmišljaju o jedno-aplikacijskom rešenju koje će moći da se koristi na različitim operativnim sistemima. Jedan kod koji će moći da se instalira na više različitih uređaja, jedan dizajn prilagođen da pruža najbolje moguće korisničko iskustvo, jedan programer koji će da razvija više aplikacija kucajući samo jedan kod.

U radu su predstavljene progresivne web aplikacije, kao i implementacija aplikacije za podršku rada biblioteke koja će biti korištena kao web i mobilna aplikacija. Predstavljena je istoriska, tehnološka i tržišna potreba za upotrebom progresivnih web aplikacija. Takodje je dat opis progresivnih web aplikacija, njihove karakteristike kao i prednosti i mane ovog pristupa.

2. PROGRESIVNE WEB APLIKACIJE

Progresivne web aplikacije su aplikacije koje predstavljaju kombinaciju najboljih osobina web i nativnih aplikacija. Pišu se u web tehnologijama koristeći standardne šablone koji im to omogućavaju. Progresivne web aplikacije prave vezu između web sajtova i mobilnih aplikacija, koja do njihove pojave nije bila moguća.

2.1. Istorija progresivnih web aplikacija

Iako su se progresivne web aplikacije zvanično pojavile 2015. godine, prvi put se spominju 2007. godine od strane Apple CEO Steve Jobs-a, koji je tada predstavio revolucionarno rešenje, poznato pod nazivom Iphone. Skoro celu deceniju ova ideja je ostala zanemarena i to vreme pripada velikoj ekspanziji nativnih aplikacija koje su dominirale u mobilnom svetu. Pojavom promenljivog web dizajna, web stranice su postale mnogo fleksibilnije i njihov prikaz na mobilnim uređajima postao je drastično bolji.

Progresivne web aplikacije su čekale svoj momenat i u 2015. godini programer kompanije Chrome, Alex Russel i dizajner Frances Berriman su prvi put javno predstavili progresivnu web aplikaciju koja je omogućavala mnogo bolje iskustvo rada na uređajima a pritom se zasnivala na samo jednom baznom kodu. Predstavljeno rešenje se bazira na Apple-ovoj ideji, koja je unapređena sa novim idejama koje su pojavile do 2015. godine.

Dva velika rivala, Google i Microsoft, čija trka u svetu tehnologije traje decenijama našli su zajednički jezik i postali veliki pobornici ideje o progresivnim web aplikacijama. Prvi koji su usvojili ovaj standard u svetu web-a bili su Chrome OS, Android i Windows. Iako je Microsoft morao da se odrekne svoje univerzalne Windows platforme, kako bi napravio prostora da se posveti progresivnim web aplikacijama u kojima je video veliki potencijal [1].

2.2. Tržište

Pojava progresivnih web aplikacija imala je veliki uticaj na promenu na tržistu aplikacija i dovela do toga da svi oni koji ne razmišljaju u ovom pravcu budu u velikom zaostatku. Zbog čega im je potrebno znatno više vremena za razvoj, a samim tim i više novca. Velike kompanije su veoma brzo uvidele prednost ovog pristupa i brzo se priključile novom trendu za šta su nagrađene velikim uspehom. Sa druge strane, manje kompanije ostale su skeptične i inicijalno nisu bile spremne za ovaj veliki korak. Međutim, ubrzo su i one uvidele prednost ovog pristupa. Velikom brzinom, za samo nekoliko godina, progresivne web aplikacije postaju prvi izbor u pravljenju web i mobilnih rešenja, predstavljajući nezaobilaznu opciju za razmatranje čak i za manje kompanije koje žele da budu u koraku za velikim kompanijama [2].

2.3. Dizajniranje za mobilne uređaje kao prioritet

Era mobilnih telefona je uveliko došla, protok internet saobraćaja na mobilnim telefonima je iz godine u godinu sve veći. Iz tog razloga, dizajn mobilnih aplikacija postaje jedna od najbitnijih stavki. Prvenstveno se obraća pažnja na dizajn aplikacija za najmanje ekrane, a zatim uz određene dorade dizajn se prilagođava velikim ekranima. Neki statistički podaci ukazuju na to da preko dve milijarde ljudi danas koristi internet isključivo preko mobilnog telefona, a očekuje se da do 2025. godine taj broj skoči za 72.5% [3].

Pored navedenih prednosti ovog pristupa, takođe je bitno napomenuti Google-ov algoritam za otkrivanje aplikacija, gde je u interesu samih kompanija da naprave što bolje mobilno rešenje jer algoritam favorizuje web aplikacije prilagođene mobilnim uređajima. Takođe, bitna činjenica je da u današnje vreme, reklamiranje i oglašavanje sve više prelazi sa TV uređaja na društvene mreže, gde se korisniku plasiraju informacije na najbrži i najpregledniji način. Dizajn savremenih mobilnih aplikacija mora da obezbedi dobru brzinu i preglednost sadržaja.

2.4. Prednosti i karakteristike

Progresivne web aplikacije bez poteškoća mogu da se prikažu na praktično svim uređajima koji se danas koriste, nezavisno od veličine ekrana. Često se pri pominjanju progresivnih web aplikacije čuje i naziv aplikacije koje rade na svim uređajima, što je veoma značajno za korisnike koji često menjaju uređaje u toku rada. Omogućavaju veoma dobro korisničko iskustvo, približno nativnim aplikacijama, jer su progresivne mobilne aplikacije prvenstveno dizajnirane za mobilne uređaje. Progresivna web aplikacija radi u uslovima gde je internet u delimičnom ili potpunom prekidu, ovo je jedna od ključni osobina ovih aplikacija.

Notifikacije omogućavaju progresivnim web aplikacijama da ostanu u stalnom kontaktu sa krajnjim korisnicima kroz niz notifikacija i strategiskih poruka čiji je esencijalni cilj da korisnika drži zainteresovanim i motivisanim za ponovno korišćenje aplikacije. Optimizacija za pretraživače omogućava da progresivna web aplikacija bude lako identifikovana i vidljiva od strane pretraživača poput Google-a, Bing-a i drugih. Deljenje aplikacije je jednostavno i lako uz pomoć URL-a aplikacije, a samim tim ona je lako čitljiva od strane ptreživača. Progresivna web aplikacija omogućava da jedna aplikacija pokrije praktično sve oblasti rada i da predstavja nativnu i web aplikaciju u jednoj što u značajnoj meri štedi finansijska i vremenska sredstva [4].

2.5. Brzina progresivnih web aplikacija

Statistički podaci ukazuju da ukoliko učitavanje stranice traje duže od 3 sekunde, postoji velika verovatnoća da će korisnik zatvoriti aplikaciju. Stopa konverzije predstavlja statistički podatak koji utvrđuje broj korisnika koji su izrvšili određene akcije na aplikaciji. Viša stopa konverzije obično znači da je korisnički interfejs bolje prilagođen krajnjim korisnicima. Stopa konverzije je direktno povezana sa brzinom učitavanja stranice.

Stopa konverzije se računa po formuli *broj sesija završenih akcijom / ukupan broj sesija*. Neki statistički podaci ukazuju da je mnogo manja stopa konverzije na mobilnim uređajima nego na desktop računarima. Što znači da se jako veliki broj korisnika ne odluči da odradi kompletan proces poput kupovine, ispunjavanja formi na telefonu, već pribegava tradicionalnim desktop aplikacijama. Posledice i razlozi zašto se ovo dešava mogu biti brojni, a neki od njih su loš korisnički interfejs, nejasno prikazani podaci, komplikovan proces i drugi. Progresivne web aplikacije u značajnoj meri utiču na ove nedostatke što utiče na povećanje stope konverzije [5].

2.6. Odnos nativnih, web i progresivnih web aplikacija

Izbor tipa aplikacije koja će se praviti prvenstveno zavisi od vizije ali i od biznisa za koji je aplikacija namenjena. Danas su poznate web, nativne i progresivne web aplikacije, svaka od njih ima neke svoje prednosti ali takođe ima i neke od mana. Ključan element je da su web i nativne aplikacije potpuno različite i imaju esencijalno drugačije osobine, dok progresivne web aplikacije čine kombinaciju osnovnih osobina web i nativnih aplikacija. Web aplikacije predstavljaju loš izbor ukoliko aplikacija treba da pristupa nekim od specifičnosti za mobilne uređaje, poput kamere, notifikacija i drugih. U slučaju da je potrebno web rešenje koje ne zahteva mobilnu aplikaciju, gde će pritom ono biti veoma lako dostupno i vidljivo, web aplikacije su pravi izbor.

U slučaju da aplikacija treba da bude brza i da se iskoriste sve prednosti mobilnih uređaja, pravi izbor je nativna aplikacija. Međutim, ukoliko je ipak potrebna jednostvna aplikacija, za čiju izradu se neće potrošiti ogromna količina novca, a da pritom bude omogućeno da radi na svim uređajima, pravi izbor je progresivna web aplikacija. Progresivna web aplikacija predstavlja zlatnu sredinu, te se ovom tipu aplikacija, s obzirom da obuhvataju sve esencijalne osobine, pruža velika prednost prilikom izbora.

2.7. Ograničenja

Određene hardverske i softverske platforme i kompanije nemaju ili tek razvijaju podšku za progresivne web aplikacije. Mnoge bitne karakteristike mobilnih uređaja su delimično ili potpuno ograničene za korišćenje od strane progresivnih web aplikacija. Pošto progresivne web aplikacije koriste poslednje tehnologije i principe, njihova podrška za starije uređaje i pretraživače je limitirana ili potpuno onemogućenja jer oni ne pružaju podršku za rad progresivnih web aplikacija.

2.8. Tehnički aspekt

Progresivna web aplikacija može da kontroliše internet zahteve kao i keširanje podataka ovo joj omogućuju service worker-i koji su ujedno i najbitniji element progresivnih web aplikacija. Pravilno čuvanje podataka omogućava rad progresivnih web aplikacija u režimu bez internet konekcije, jer u svakom trenutku na raspolaganju imaju podatke, kao i mnogo veću brzinu aplikacija jer omogućavaju asinhroni rad aplikacije, gde oni vode računa o komunikaciji van aplikacije. Ključna tehnologija progresivnih web aplikacija su svakako service workeri, da bi oni mogli da obavljaju svoju ulogu potrebno je obezbediti sigurno radno okruženje. Progresivna web aplikacija koristi TLS protokol kao standard za razmenu podataka između dve aplikacije a on omogućava da internet stranica bude sigurna.

Takođe bitan element svake progresivne web aplikacije je manifest fajl koji predstavlja konfiguracioni fajl sa svim potrebnim informacijama koje omogućavaju da se aplikacija može prilagoditi svim platformama, odnosno omogućuje da se web aplikacija prezentuje kao nativna aplikacija ili najpribližnije nativnoj aplikaciji. Bitno je napomenuti da progresivne web aplikacije uz pomoć service worker-a osnovni deo aplikacije drže keširan i on omogućava prikazivanje aplikacije u stanju kada je internet konekcija u prekidu. Prvo se učitava kostur aplikacije, odnosno ključni elementi a zatim se dinamički učitavaju ostali podaci [6].

3. APLIKACIJA ZA RAD U BIBLIOTECI

Library je aplikacija namenjena za podršku rada biblioteke. Razvijena je kao progresivna web aplikacija, tako da može da se prikaže na web-u ali i na mobilnim uređajima. Aplikaciju mogu da koriste dva tipa korisnika: administratori (zaposleni u biblioteci) i obični korisnici (korisnici usluga biblioteke). Cilj je da se uz pomoć Ionica omogući prikazivanje web rešenja i na mobilnim uređajima, u ovom slučaju na Androidu i iOS-u.

3.1. Implementacija aplikacije

U implementaciji rešenja osnovni zadatak je da se razvije frontend aplikacija koja će da obezbedi implementaciju svih zahteva navedenih u specifikaciji aplikacije. Library aplikacija je sačinjena od niza komponenti grupisanih u module. Komponenta predstavlja gradivni element stranica i njena složenost zavisi od zahteva aplikacije. Modul predstavlja grupu komponenti, odnosno komponente grupisane po nekom kriterijumu. Obično to mogu biti neke zajedničke osobine ili namena, ali isto tako i područje njihove primene. Takođe, moduli mogu da ujedno čine i komponente koje su iskoristive od strane više drugih modula, što ih čini univerzalnim komponentama aplikacije. Modul takođe može da objedinjuje i druge elemente aplikacije kao što su servisi, ali isto tako postoje i moduli koji vode računa o samom rutiranju unutar aplikacije.

Glavni problem nastaje kada je potrebno učitati veliki broj modula prilikom inicijalnog prikazivanja aplikacije. Što u značajnoj meri utiče na same performanse aplikacije i učitavanje postaje znatno duže. Da bi se ovo sprečilo module organizujemo po stranicama same aplikacije i sve komponente stranice svrstavamo u jedan modul. Inicijalno je potrebno učitati modul samo stranice koja se prikazuje dok ostali moduli mogu ostati ne učitani. Da bi se ovaj sistem učitavanja ostvari koristimo odloženo učitavanje (eng. lazy loading).

Pored navedenog imamo i modul za rutiranje koji vodi računa o rutiranju i inicijalno će ce učitati samo početni ekran sa svim definisanim bibliotekama. modul za rutiranje takođe ima opciju da uz pomoć zaštitnika rute (eng. Guard) za svaki od modula koristeći funkciju canActivate navede da li se određenoj ruti može pristupiti sa ili bez autentifikacije. Zaštitnici ruta se mogu implementirati tako da u slučaju da se pristupi nekoj ruti koja zahteva autentifikaciju, korisnika usmeri na stranicu za logovanje, a zatim redirektuje nazad na željenu stranicu. Library aplikacija ima implementiran dodatak za učitavanje sadržaja (eng. interceptor), koji vodi računa i presreće sve zahteve poslate ka bekend serveru, vodeći računa o postavljanju odgovarajućih zaglavlja (eng. header) zahteva koji su potrebni da bi server znao da se radi o autentifikovanom korisniku. Takođe, pored navedenog vodi računa o svim potencijalnim greškama koje server može da vrati i blagovremeno korisnika obaveštava o nastalom problemu. Dobar korisnički interfejs u svakom trenutku korisnika mora da obavesti o svim dešavanjima unutar aplikacije, tako da korisnik nikada ne ostane u nedefinisanom stanju.

3.2. Integracija sa Ionicom i kreiranje PWA

Cilj aplikacije je da prikaže mogućnost da se web aplikacija bez velikih poteškoća može pretvoriti u progresivnu web aplikaciju. Integracija Ionic-a u Angular aplikaciju je veoma jednostavana, Angular CLI omogućava uvezivanje svih potrebnih biblioteka Ionica. Nakon toga je potrebno inicijalizovati postojeći projekat sa Ionicom pozivanjem *ionic init* komande, koja od korisnika zahteva unos imena projekta, a zatim izvršava inicijalizaciju. Angular aplikacija postaje ujedno i Ionic aplikacija, sa mnogo više mogućnosti a sledeći korak je konfiguracija potrebna za kreiranje progresivne web aplikacije.

Dve glavne karakteristike progresivnih web aplikacija su: service worker i web manifest. Angular omogućava instaliranje biblioteke koja ima mogućnost automatizacije njihovog dodavanja.

Pokretanjem komande *ng add @angular/pwa* biće modifikovano nekoliko fajlova i biće dodata dva nova fajla: *ngsw-config.json* i *manifest. webmanifest*. Fajl *ngsw-config.json* predstavlja konfiguracioni JSON fajl service workera dok fajl *manifest. webmanifest* predstavlja konfiguracioni JSON fajl koji se koristi da odredi izgled aplikacije koja se prikazuje.

Nakon završenog lokalnog testiranja, poslednja verzija aplikacija je postavljena (eng. Deploy) na Firebase server. Kao prvi korak potrebno je napraviti nalog na Firebase-u.

Zatim je potrebno izvršiti autentifikaciju u terminalu pokretanjem komande *firebase login*, a potom pokretanjem *firebase init* komande potrebno je napraviti novi projekat i dati mu ime. Po inicijalizaciji Firebase projekta, potrebno je izvršiti dodatnu produkcisku konfiguraciju nakon čega se vrši postavljanje aplikacije na Firebase server. Po uspešnom objavljivanju aplikacije ona postaje dostupna na odabranom domenu. Sa URL-om aplikacije uz pomoć Lighthouse alata omogućeno je analiziranje aplikacije na prosleđenoj putanji. Po završenom analiziranju otvara se panel u kome je prikazana statistika sajta, kao i sugestije kako da se svaki segment unapredi [7].

Takođe, s obzirom da se radi o progresivnoj web aplikaciji, ona omogućava da se na mobilnim telefonima sačuva prečica na početnom ekranu, tako da joj se kasnije može pristupati kao nativnoj aplikaciji. Dodavanjem aplikacije na početni ekran, ona će postati lako dostupna, poput aplikacija koje su instalirane uz pomoć prodavnice aplikacija. Nakon toga, u bilo kom trenutku je omogućen pristup aplikaciji bez kucanja adrese u internet pretraživaču.

4. ZAKLJUČAK

Povećano korišćenje mobilnih telefona kao posledicu ima potrebu za sve većim brojem mobilnih aplikacija. Zbog prednosti koje mobilne aplikacije pružaju postale su nezaobilazan i ključan faktor u komunikaciji sa krajnjim korisnikom. Ova promena na tržištu dovela je do toga da web aplikacije više nisu dovoljne, te da svaka od njih zahteva i svoje mobilno rešenje. Pravljenje više od jedne aplikacije zahteva više vremena, više finansijskih sredstava, a ujedno i adekvatno obučene programere za ovaj poduhvat. Nastale promene podstakle su programere da razmisljaju o jedno-aplikacijskom rešenju, jednoj aplikaciji koja će predstavljati i mobilno i web rešenje. U tom svetlu pojava progresivnih web aplikacija usledila je kao uzročno posledična potreba za rešenjem ovog problema.

Predstavljena je velika potreba za što boljim korisničkim interfejsom, sa ciljem poboljšanja korisničkog iskustva. Dizajn mobilnih aplikacija postao je ključan element u svetu web-a, gde je veliki teret pao na leđa dizajnera koji su postali primorani da se prilagode najmanjim ekranima, a da pritom pruže najbolje moguće korisničko iskustvo, uz održavanje jednostavnosti i efikasnosti aplikacije. Nedostaci koji danas postoje kod progresivnih web aplikacija predstavljaju ključnu tačku za razvoj boljeg i optimalnijeg puta za razvoj nekih budućih jezika i radnih okvira. Ključan element u razvoju budućih aplikacija uvek predstavlja korisničko iskustvo i nedostaci postojećih rešenja.

5. LITERATURA

[1] https://www.divante.com/pwabook/chapter/02-the-history-of-pwas, Divante eCommerce Software House, *" The history of PWA development*" (pristupljeno u martu 2023.)

[2] https://medium.com/progressivewebapps/history-of-progressive-web-apps-4c912533a531, ScandiPWA, "*History of progressive web applications*" (pristupljeno u martu 2023.)

[3] https://xd.adobe.com/ideas/process/ui-design/what-is-mobile-first-design, Justin Morales, "*Mobile First Design Strategy: The When, Why and How*" (pristupljeno u martu 2023.)

[4] https://inoxoft.com/blog/benefits-of-progressive-web-apps-pwa-advantages-and-disadvantages, Nazar Kvartalnyi, "*Benefits of PWA*" (pristupljeno u martu 2023.)

[5] https://www.divante.com/pwabook/chapter/01-introduction-to-pwa-technology, Divante eCommerce Software House, "*Introduction to Progressive Web Apps*" (pristupljeno u martu 2023.)

[6] https://medium.com/@tysonpaul89/what-i-learned-about-progressive-web-app-pwa-part-2-4074137301a7, Tyson Paul, "*Progressive Web App (PWA)*" (pristupljeno u martu 2023.)

[7] https://ionicthemes.com/tutorials/the-complete-guide-to-progressive-web-apps-with-ionic4, Agustin Haller, "*The Complete Guide To Progressive Web Apps with Ionic Framework*" (pristupljeno u martu 2023.)

Kratka biografija:

Dejan Predojević rođen je 16.06.1996. godine u Vrbasu. Završio je Srednju tehničku školu „Mihajlo Pupin" u Kuli 2015. godine. Fakultet tehničkih nauka u Novom Sadu je upisao školske 2015/2016. godine. Osnovne akademske studije završio je 2019. godine. Iste godine upisao je master akademske studije, modul Softversko inženjerstvo. Kontakt: dejanpredojevic11@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/23BE31Vucaj

SISTEM ZA OBRADU I VIZUALIZACIJU VELIKOG SKUPA TELEMETRIJSKIH PODATAKA IZ TRKA FORMULE 1

SYSTEM FOR BATCH PROCESSING AND VISUALIZATION OF FORMULA 1 BIG DATA TELEMETRY

Aleksa Vučaj, *Fakultet tehničkih nauka, Novi Sad*

Oblast - ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj - *U ovom radu predstavljen je sistem za paketnu obradu i vizualizaciju podataka telemetrije Formule 1. Cilj ovog rada je omogućiti lakšu analizu podataka telemetrije Formule 1 kao i njihovu vizuelizaciju. Da bi se ovo postiglo, projektovan je i implementiran sistem namenjen za paketnu obradu podataka. Glavne karakteristike ovog sistema su da je sistem samostalan zahvaljujući orkestratoru poslova koji dobavlja i procesira podatke nakon svake trke, kao i vizuelizacija obrađenih podataka.*

Ključne reči: *Paketna obrada velikih skupova podataka, Vizualizacija rezultata obrade podataka, Formula 1, Apache Spark, Veliki skupovi podataka*

Abstract - *In this paper, we present a system for batch processing and visualization of Formula 1 telemetry data. The goal of this system is to facilitate easier processing and visualization of Formula 1 telemetry data. To achieve this, a system designed for batch processing was implemented. Main features of the system are that the system can work independently with the help of a job orchestrator, which fetches and processes data after every race, as well as visualization of processed data.*

Keywords: *Big data batch processing, Data visualization, Formula 1, Apache Spark, Big data.*

1. UVOD

Sport, kao jedna od najznačanijih razonoda u svetu, sve više zavisi od prikupljanja podataka i analize istih. Odluke u sportovima, poput američkog fudbala, se već neko vreme najviše zasnivaju na obrađivanju podataka.

Ukoliko se u sport doda da pored samog sportiste na rezultate utiču i karakteristike mašine kojom upravlja takmičar, jasno je da je zavisnost od podataka postaje obavezna i neraskidiva. Formula 1 sigurno jeste jedan od najrazvijenijih i najzavisnijih sportova od različitih parametara koji utiču na performanse bolida i vozača. Osim što podešavanje bolida vozaču omogućava trenutne performanse na stazi, podaci su od još većeg značaja timovima inženjera koji imaju za zadatak da bolide pripreme na optimalan način za sve trke u toku sezone.

Inženjeri prikupljene podatka koriste kako bi pomoću svojih alata mogli da simuliraju milione trka u specijalizovanim softverskim rešenjima, kako bi utvrdili koje postavke su najbolje za njihov tim. S tim u vezi, može se primetiti da je u F1 sve veća prisutnost kompanija kojima se deo delatnosti nalazi i u domenima skladištenja i obrade velike količine podataka. Kompanije poput *Oracle*-a, *Google*-a, *Microsoft*-a i *Amazon*-a sastavni su deo ovog tehničko-tehnološkog sporta, te imaju obostranu korist da u zamenu za pružanje pomoći timovima u analizi podataka, dodatno promovišu svoj brend u jednom od najprestižnijih takmičenja.

Cilj ovog rada jeste omogućiti lakšu analizu podataka telemetrije Formule 1 kao i njihovu vizuelizaciju.

Pod terminom telemetrije se, u ovom radu i u daljem tekstu, smatraju podaci o bolidu koji se sakupljaju u toku njihove vožnje na stazi. Informacije koje su dostupne u okviru ovog rada su informacije o brzini bolida, broju obrtaja motora, trenutnom stepenu prenosa, iks, ipsilon i cet koordinatama bolida na stazi, udaljenosti od starta, udaljenosti od bolida koji je ispred i slično [1].

2. PREGLED TRENUTNOG STANJA U OBLASTI

Interesovanje za telemetriju i Formulu 1 ubrzano raste, broj alata za analizu ne prati taj tempo. Timovi Formule 1 su zapravo jedini u Formuli 1 kojima su ovi podaci neophodni kako bi se sa što više uspeha takmičili u ovom sportu. Telemetrijski podaci i njihovo tumačenje i razumevanje stvara razliku između najboljeg i najgoreg tima. Ovi podaci se koriste u razne svrhe poput simulacija koje koriste timovi kako bi pripremili bolide za predstojeće trke. Svi javno dostupni servisi za pružanje informacija o telemetriji većinski se oslanjaju na *Python* biblioteku *FastF1* kako bi dobavili svoje podatke. Sa druge strane, sami grafikoni širem auditorijumu gledaoca ne znače ništa bez dodatnog pojašnjenja i skretanja pažnje na određene detalje, pa je samim tim broj ljudi zainteresovanih da se bave ovom temom veoma mali.

U trenutku pisanja rada moguće je pronaći svega dve internet stranice na kojoj se mogu interaktivno birati željena telemetrija i više sajtova koji se bave analizom unapred pripremljenih grafikona i telemetrije. Sajtovi sa interktivnim odabirom telemetrije mogu se naći na f1-telemetry.net i f1-tempo.com. Oba sajta svoje podatke dobijaju putem *FastF1 Pyhton* biblioteke. Važno je napomenuti da niti jedan od ova dva sajta nema mogućnost uvida u telemetriju za više od jednog kruga.

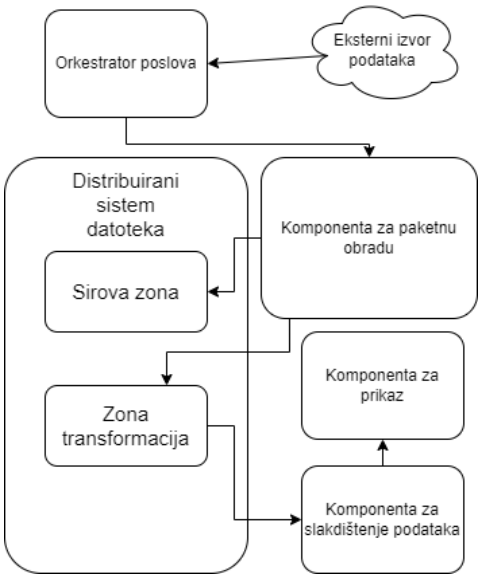
NAPOMENA:
Ovaj rad proistekao je iz master rada čiji je mentor bio dr Vladimir Dimitrieski, docent.

Dakle, analiza na nivou jedne cele trke, više trka ili pak cele sezone nije moguća.

Analiza postojećih alata koji poseduju timovi koji su učesnici F1 nije moguća zbog toga što ne postoji pristup niti uvid u njihove mogućnosti ili arhitekturu.

3. ARHITEKTURA SISTEMA

Zadatak ovog sistema je da podatke o telemetriji Formule 1 iz spoljnih izvora prikupi, izvrši različita izračunavanja i agregacije, skladišti u bazi podataka i na kraju prikaže pomoću alata za pravljenje grafikona i vizualizaciju podataka. Na Slici 1 je prikazana arhitektura sistema, dok će u nastavku teksta biti opisani prikazani moduli.



Slika 1. *Arhitektura sistema*

Podaci se preuzimaju sa jednog od eksternih izvora podataka. Nakon toga, orkestrator poslova izdaje zadatak komponenti za paketnu obradu zadataka da skladišti ove pridošle podatke u *CSV* formatu u sirovu zonu podataka na distribuiranom sistemu datoteka. Iz sirove zone podataka se podaci učitavaju u komponentu za paketnu obradu koja ih obrađuje i transformiše i tako transformisane ih čuva u *CSV* formatu u direktorijumima zone transformacija. Poslednji korak u ovom sistemu je upisivanje transformisanih podataka u trajno skladište podataka.

3.1 Orkestrator poslova

Zadatak orkestratora tokova poslova je da zadaci koji su definisani budu izvršeni u zakazano vreme. Ove zadatke definiše korisnik i mogu biti *Python* ili beš skripte, kao i beš komande.

Apache Airflow kao tehnologija namenjena za orkestraciju tokova poslova ima mnogobrojne alternative. Neke od najpopularnijih tehnologija pored *Airflow*-a jesu *Luigi*, *Apache NiFi*, *AWS Step Functions* i *Prefect*. U uži odabir za korišćenje tehnologije za izvršavanje tokova poslova ušli su *Airflow* i *Luigi*, te će u narednom delu teksta biti dato poređenje ove dve tehnologije.

Suštinski najbitnija karakteristika kod ove komponente je način i mogućnost za zakazivanje zadataka i tokova poslova. Distribuirana priroda *Airflow* omogućava i distribuirano izvršavanje zadataka koje je potrebno podesiti na izvršiocu nakon čega je dovoljno prepustiti zakazivaču koji je sastavni deo *Airflow* da sam pokreće zadatke. *Luigi* ne podržava mogućnost da sam pokreće zadatke, već je moguće namestiti kron posao (engl. *Cron Job*) putem komandne linije kako bi on pokrenuo izvršavanje poslova. Iako distribuirano izvršavanje u ovom slučaju ne igra ključnu ulogu, sposobnost *Airflow*-a da se zakaže i izvrši je ključna u odabiru ove tehnologije. Osim toga, *Airflow* ima mogućnost da pokrene izvršavanje sledećeg zadatka iako se njegov prethodnik i dalje nije potpuno završio. Ovo preklapanje ubrzava izvršavanje tokova, ali je kompleksno za implementirati

3.2 Distribuirani sistem datoteka

Distribuirani sistem datoteka u ovom sistemu služi za skladištenje podataka u sirovom i transformirsanom obliku. Podaci se čuvaju u *CSV* formatu i u zoni transformacije i u sirovoj zoni. U sirovu zonu se upisuju tek preuzeti, neobrađeni podaci. Komponenta za paketnu obradu učitava sirove podatke i nad njima vrši transformacije. Nakon trasformisanja, podaci se čuvaju u zoni transformacije, odakle se prepisuju u bazu podataka.

Kao rezultat pretrage alternativa za distribuirani sistem datoteka *Hadoop* najčešći odgovor na koji se nailazi je *HBase*. *HBase* je nerelacioni sistem za upravljanje bazom podadataka usmerenom na kolone (engl. *Column-oriented*). *HBase* kao svoju osnovu ima *HDFS* i namenjen je za skladištenje i upravljanje velikom količinom strukturiranih i polu strukturiranih podataka, a takođe svoju otpornost na greške ekspoatiše iz *HDFS*-a.

HDFS je sistem namenjen za skladištenje velikih skupova podataka na klasteru izgrađenom od kućnih uređaja i namenjen je za paketnu obradu podataka, dok je sa druge strane *HBase NoSQL* baza podataka. *HBase* je optimizovan za nasumičan pristup pisanju i čitanju velikih skupova podataka i može da podrži veći broj korisnika istovremeno.

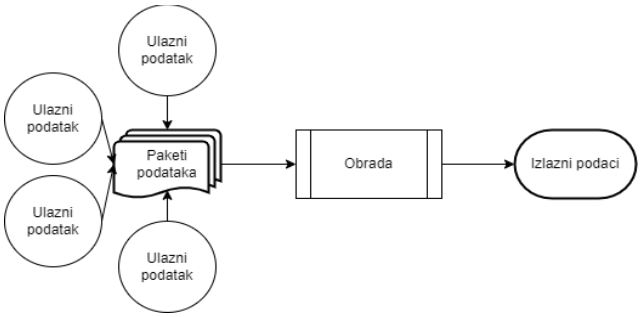
HDFS skladišti podatake na osnovu hijerarhijske strukture datoteka pri čemu su te datoteke izdeljene u blokove podataka i distribuirane u klasteru. Za razliku od *HDFS*-a, *HBase* podatke organizuje po redovima i kolonama unutar sebe.

3.3. Paketna obrada podataka

Paketna obrada podataka istovremeno obrađuje veliku količinu podataka koji se prvo skladište, a zatim se čitaju u paketima [2]. Specifikacija veličine paketa zavisi od implementacije sistema, te veličina može biti ograničena na različite načine kao što su broj podataka, vremenski okvir u kom su proizvedeni, ograničenja sistema i slično. Paketi podataka se zajedno šalju obrađivaču podataka. Za dobijanje krajnjih rezultata obrade potrebno je da se svaki zasebno obrađeni zapis izvrši uređivanje i sumiranje rezultata obrade. Na Slici 2 konceptualno je prikazana paketna obrada.

Tehnologija izabrana za implementaciju modula koji se bavi paketnom obradom podataka jeste *Apache Spark*. Kao jedna od mogućih alternativa u kojoj je moguće implementirati paketnu obradu podataka je *Apache Flink*. *Spark* i *Flink* su radni okviri otvorenog koda namenjeni za procesiranje velikih skupova podataka i jedni su od vodećih kada je u pitanju njihova upotreba. Obe ove tehnologije namenjene su za obradu velikih skupova podataka u paralelnom režimu pritom pružajući podršku putem prog-

ramskog interfejsa tokom analize i manipulacije podacima.



Slika 2. *Konceptualni prikaz paketne obrade podataka*

Spark je više optimizovan za paketnu obradu podataka i koristi se u slučajevima gde je za poslove koji vrše paketnu obradu prihvatljivo da imaju nešto višu latenciju u odnosu na obrađivače u realnom vremenu [3]. Upravljanje memorijom kod *Spark*-a oslanja se na model upravljanja memorijom poznat kao *RDD Caching*. *RDD* kešing omogućava skladištenje rezultata međukoraka u toku obrađivanja podataka što ovom sistemu veoma pogoduje jer je potrebno imati više međurezultata tokom agregiranja telemetrijskih podataka. Model obrade podataka (engl. *Data Processing Model*) koji koristi *Spark* je distribuirani skup podataka otporan na otkaze (engl. *Resilient Distributed Dataset, RDD*). *RDD* model predstavlja kolekciju elementa otpornih na otkaze koje je moguće obrađivati u paralelnom režimu.

Glavna prednost *Spark*-a je to da su performanse u poređenju sa *Hadoop*-om i do 100 puta brže za skladištenje pomoćnih rezultata u memoriji, kao i 10 puta kada se podaci čuvaju na disku [4] Prethodno napisana osobina je u ovom slučaju od velikog značaja jer u toku svake iteracije kreiraju privremene rezultate koje koristimo u toku transformisanja telemetrijskih podataka.

3.4 Komponenta za vizuelizaciju

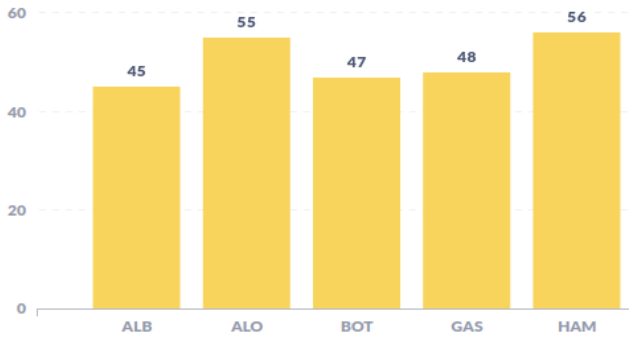
Postoji mnogo alternativa koje je moguće koristiti za vizuelizaciju podataka poput *Tableau*, *Looker*, *Power BI*, *Superset* i mnogih drugih.

Jedan od kriterijuma koji je bio bitan pri odabiru alata za vizuelizaciju svakako mora biti izgled samog korisničkog interfejsa. *Metabase* je alat koji je namenjen za korišćenje „obe" vrste korisnika. Prilagođen je i ljudima koji ne dolaze iz sveta informacionih tehnologija, te nemaju nikakvo predzanje o upitima nad bazama podataka. Koristeći alate za pravljenje upita i već gotovih grafikona, ovaj alat mogu da koriste svi koji imaju minimalno potrebno predznanje u korišćenju kompjutera. S druge strane, korisnici koji imaju iskustvo u pravljenju upita nisu uskraćeni za mogućnost da svoje znanje iskoriste za dobijanje grafikona.
Metabase interfejs sastoji se iz 3 glavne celine kada su u pitanju podaci. Podaci mogu biti predstavljeni u klasičnom tabelarnom prikazu, pomoću različitih grafikona ili u vidu skupa grafikona na jednom mestu, to jest kontrolnih tabli. Osim toga, *Metabase* pruža sigurnost podataka koristeći pristup podacima na osnovu korisnikove uloge u sistemu (engl. *Role-Based Access Control, RBAC*).

4. PRIKAZ IZGLEDA SISTEMA

Implementiran sistem omogućava interaktivnu analizu prikupljenih telemetrijskih podataka. U okviru ovog poglavlja cilj je prikazati moguće izglede aplikacije i njene mogućnosti.

Na Slici 3 prikazan je stubičasti grafikon koji prikazuje koji je najveći broj promena stepena prenosa na Velikoj nagradi Monaka za 5 vozača. Ovaj grafikon je moguće dobiti u svega 4 koraka koristeći *Metabase*-ov editor za pravljenje upita nad bazom podataka.



Slika 3. *Broj promena stepena prenosa na VN Monaka*

Pomoću ovog sistema moguće je veoma lako generisati različite grafikone koristeći telemetriju ne samo iz jedne sesije, već i iz cele sezone, što je glavno ograničenje svih trenutno dostupnih alata. Koristeći ugrađeni vizuelni editor za upite moguće je brzo i lako vizuelizovati unapred pripremljene podatke.

U sklopu *Metabase*-a moguće je i sačuvati generisane grafikone za kasnije korišćenje i grupisati ih u kontrolne table. Kontrolne table su veoma pogodne za korišćenje u slučajevima kada je potrebno brzo pristupiti različitim tipovima grafikona. Dodatna funkcionalnost koju *Metabase* pruža svojim korisnicima je da se pretplate na neku od kontrolnih tabli i da pretplaćenim korisnicima pošalje nove obaveštenje kada se novi grupa grafika formira.

5. ZAKLJUČAK

Formula 1 je specifičan sport iz više razloga. Prvi razlog jeste da za dobar rezutlat nije dovoljan samo sposoban takmičar, već i performantan bolid kojim takmičar upravlja. Drugi je da za razliku od ostalih sportova u kom se tim ili pojedinac takmiče u izolovanim događajima koji se na kraju odražavaju na finalno stanje u šampionatu, u Formuli 1 se svi takmičari takmiče u isto vreme sa svim ostalim protivnicima tokom svih događaja.

Poboljšavanje bolida dugo je zavisilo isključivo od sposobnosti vozača da prenese svoje utiske mehaničarima. Sada, inženjeri pretežno se oslanjaju na rezultate analiza i obrada podataka koje skupljaju tokom slobodnih treninga i trka kako bi bili u mogućnosti da shvate šta se tačno događa sa bolidom. Time, sakupljanje, obrada i tumačenje rezultata čine neizostavne činioce svakog tima koji ima ambicije za uspesima u ovom sportu.

Ovaj rad usmeren je na razvijanje sistema koji gledaocima omogućava da dublje zađu u inače nedostupnu analizu događaja koje je moguće videti tokom 3 dana programa

trkačkog vikenda. Subjektivni osećaji koji se dobija putem prenosa trka na televiziji ili tokom boravka na stazi mogu biti potpuno različiti od onoga šta pokazuju podaci. Zbog toga je potrebno da se prilikom svake diskusije ili analize onoga što je viđeno na trci uključe i nepobitne činjenice kako bi se dobila potpuna slika.

Realizovani sistem sastoji se od nekoliko ključnih delova. Prvi deo je za dobavljanje podataka sa izvora podataka bez kojih vršiti bilo kakvu analizu. Dalje sledi sposobnost sistema da prikupljene podatke učita, obradi, agregira i prirpemi za vizualizaciju. Poslednji, ali možda najbitniji deo, jeste čuvanje i prikaz tih podataka. Bez dobre vizualizacije nema ni analize koja bi mogla dublje dočarati šta se tačno desilo na stazi.

Smer u kom ide dalji napredak ovog sistema fokusiran je na razvijanje striming aspekta obrade podataka. Obrada podataka u realnom vremenu dala bi potpuno novu dimenziju analize podataka jer bi gledaoci mogli dobiti poređenja za manje od minute od trenutka dešavanja na stazi. Dalji napredak ovog sistema mogao bi se ogledati i u povezivanju postojećih informacija o telemetriji sa podacima o vremenskim uslovima na i oko staze. Ovime bi se mogao dobiti dublji uvid u to koji bolidi i vozači se kako ponašaju pri različitim vremenskim uslovima. Primer ove analize mogao bi dovesti do pravljenja relacija vezane za kakve performanse ima određeni tim na stazama koji su na višoj nadmorskoj visini poput Meksiko Sitija ili kako se koji vozač nosi sa visokom vlažnosti vazduha.

6. LITERATURA

[1] theOehrl. (2022). *FastF1 Documentation* [Onlajn]. Dostupno na: https://theoehrly.github.io/Fast-F1/

[2] Packt. (2022). *Distributed batch processing* [Onlajn]. Dostupno na: https://subscription.packtpub.com/-book/big-data-and-business-intelligence/9781784391409/1-/ch01lvl1sec13/distributed-batch-processing.

[3] ProjectPro. (2023). *Apache Flink vs Spark - Will one overtake the other?* [Onlajn] Dostupno na: https://www.projectpro.io/article/apache-flink-vs-spark-will-one-overtake-the-other/282.

[4] Vaidja, N. (2022). *Apache Spark Architecture - Spark Cluster Architecture Explained* [Onlajn]. Dostupno na: https://www.edureka.co/blog/spark-architecture/

Kratka biografija:

Aleksa Vučaj rođen je 8. septembra 1998. godine u Novom Sadu, Vojvodina. Pohađao je gimnaziju „Isidora Sekulić”. Fakultet tehničkih nauka upisao je 2017. godine na smeru Računarstvo i automatika, da bi 2021. diplomirao. Master studije na studijskom programu Računarstvo i automatika usmerenje Računarstvo visokih performansi upisuje 2021. i polaže sve planom i programom predviđene predmete.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 681
DOI: https://doi.org/10.24867/23BE32Bratic

PROJEKTOVANJE UPRAVLJAČKE PLOČE ZA KONTROLU DC MOTORA

DESIGN OF MAIN BOARD FOR DC MOTOR CONTROL

Nikolina Bratić, Vladimir Rajs, *Fakultet Tehničkih Nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljen je razvoj PCB modula koji se koristi kao kontrolni modul za upravljanje rada motora, sa komunikacionim interfejsima (Ethernet, USB 2.0, CAN Bus, I2C, UART), Stereo DAC 24-bitni AD konvertor.. Cilj ovog rada bio je da se opiše na najbolji mogući način celokupan proces projektivanja višeslojne štampane pločice od idejne šeme, realizacije (šematski dizajn) i rutiranja pločice do generisanja izlaznig fajlova za izradu samog PCB-a. Izabran je mikrokontroler iz STM32 familije, jer je razvojno okruženje intuitivno i lako za korišćenje.*

Ključne reči: *STM32, Ethernet, USB 2.0, CAN Bus, I2C, UART, Stereo DAC 24-bit AD konvertor*

Abstract – *In this paper the development of a PCB module which can be used as a control module for managing the operation of the engine, with communication interfaces (Ethernet, USB 2.0, CAN Bus, I2C, UART), Stereo DAC 24-bit AD converter. The goal of this paper was to describe in the best possible way the entire process of designing a multi-layer printed circuit board from the conceptual scheme, realization (schematic design) and routing of the circuit board to the generation of output files for the production of the PCB itself. A microcontroller from the STM32 family was chosen, because the development environment is intuitive and easy to use.*

Keywords: *STM32, Ethernet, USB 2.0, CAN Bus, I2C, UART, Stereo DAC 24-bit AD converter*

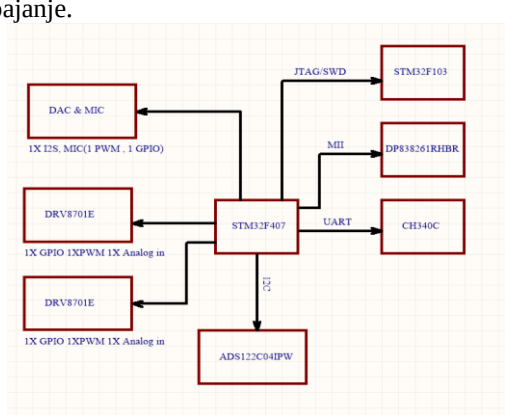
1. UVOD

U ovom radu realizovana je višeslojna štampana pločica za kontrolu upravljanja BDC motora.
Korišten je mikrokontroler STM32F407 sa debugger-om STM32F103, a od komunikacionih interfejsa: Ethernet (brzina 2MB/s), USB 2.0, CAN Bus (brzina 1MB/s), UART i I2C.
Prvo se pristupilo realizaciji šeme, zatim postavljanja komponenata za rutiranje i pravila za rutiranje. Posle rutiranja pločice urađena je provera Design rule check, i izvezeni su fajlovi za fabrikaciju (poručivanje PCB).

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

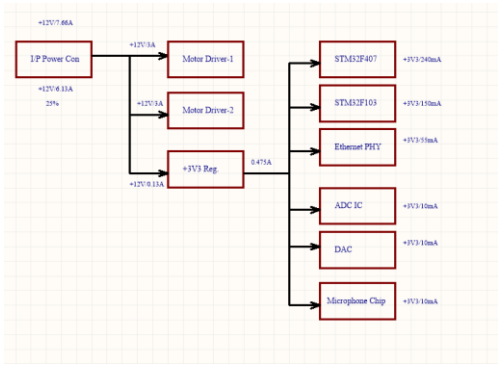
2. RAZVOJ HARDVERA

Šematski i *PCB* dizan modula rađen je u programskom paketu „*Altium Designer*”. Blok šema modula data je na slici 1, a sastoji se od sledećih podblokova: Schematic Block, Power Budget, STM32F103 Debugger, Ethernet PHY, Motor Driver 1, UART, USB interfejs, ADC interfejs, DAC I MIC, MCU - STM32F407, Motor Driver i Napajanje.



Slika 1. *Blok šema modula*

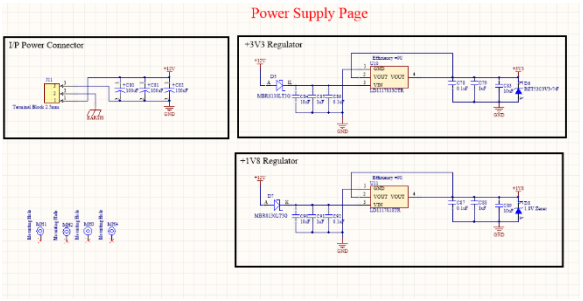
U samom projektovanju vrlo je bitno uraditi analizu budžeta za napajanje (engl.power budget), kako bismo imali podatke o strujnim i naponskim zahtevima/ograničenjima (slika 2).



Slika 2. *Budžet za napajanje*

2.1. Napajanje

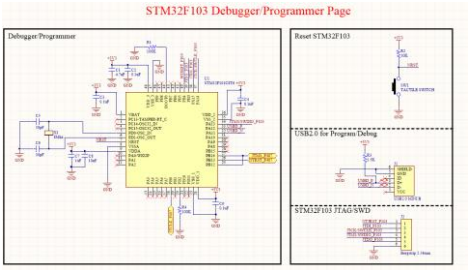
Za napajanje mikrokontrolera 3.3V korišten je linearni regulator LD1117S33CTR, ulazni napon može da bude maksimalno 15V i izlazna struja 800mA, dok je za napajanje 1.8V korišten linarni regulator LD1117S18TR sa istim karakteristikama kao i LD1117S33CTR. Električna šema napajanja prikazana je na slici 3.



Slika 3. *Električna šema bloka napajanja*

2.2. STM32F103 Debugger

Za progarmator je izabran STM32103 debager, koji ima SWD i JTAG pinove. Šematski prikaz programatora je prikazan na slici 4.

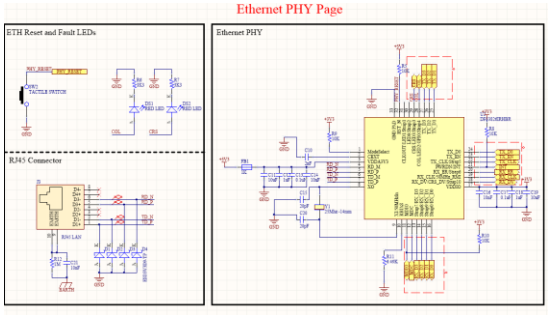


Slika 4. *Električna šema programatora*

2.3. Ethernet PHY

Ethernet je najrasprostranjenija mrežna tehnologija koja se koristi kod LAN-ova. Uobičajena bitska brzina kod prenosa podataka na ovoj mreži je 10 Mbps, a noviji standardi dozvoljavaju brzinu prenosa od 100 Mbps. Najčešće korišćeni standardi za Ethernet su Ethernet 2.0 i IEEE 802.3. U oba slučaja kao medijum za prenos se koristi deljiva magistrala po kojoj, u datom trenutku, samo jedan čvor može da prenosi (šalje) podatke. Podaci se prenose u formi okvira koji sadrži MAC (eng. media access control) izvorišnu i odredišnu adresu predajnog i prijemnog čvora, respektivno.Ethernet karticu čine dve sledeće logičke celine: interfejs fizičkog medijuma - prema standardu, logika ovog interfejsa odgovara PLS-u i PMA-u. Interfejs je odgovoran za prenos signala na električnom nivou, a čine ga sledeće dve celine: primopredajnik; i konvertor koda koji kodira/dekodira podatke.

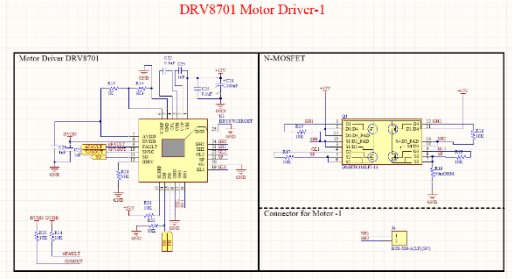
Sa ciljem da ne dođe do refleksije signala na prenosnom medijumu UTP treba da bude završen karakterističnom impedansom od 100 Ω, a koaksijalni sa 50 Ω. Izgled električne šeme Ethernet PHY dat je na slici 5.



Slika 5. *Električna šema Ethernet PHY*

2.4. Motor Driver 1/2

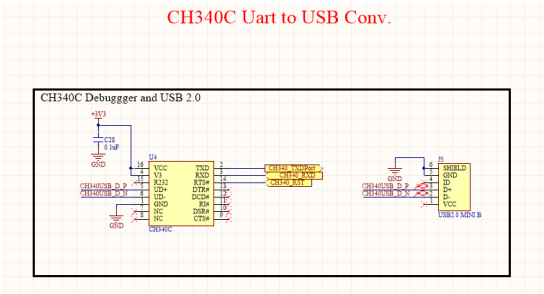
Električna šema Motor Driver data je na slici 6. DRV8701 je H most drajver sa integrisanim FET drajverom za kontrolu 4 eksterna mosfeta. Uređaj može da se napaja sa napajanjem 5.9V do 45V. Unutrašnja zaštita koja je prisutna: zaštita od niskog napona, charge pump greške, gašenje pri visokoj struji, greške pred drajvera, i zaštita od visoke temperature.



Slika 6. *Električna šema Motor Driver-a*

2.5. UART, USB interfejs

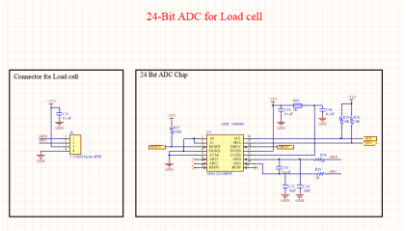
CH340 je USB konvertorski čip koji pretvara USB u serijski port. U UART režimu, CH340 obezbeđuje standardne MODEM signale, koji se koriste za proširenje serijskog porta za računare ili direktno nadogradnju sa normalnog serijskog uređaja na USB magistralu. Električna šema je prikazana na slici 7.



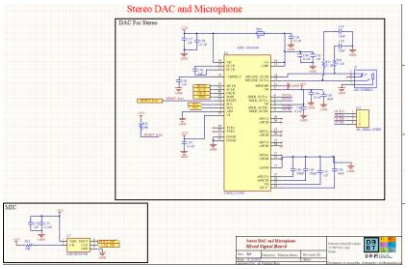
Slika 7. Električna šema UART i USB

2.6. ADC interfejs, Stereo DAC i Microphone

Električna šema *ADC* bloka data je na slici 8, dok je električna šema za Stereo DAC i Microphone prikazana na slici 9.



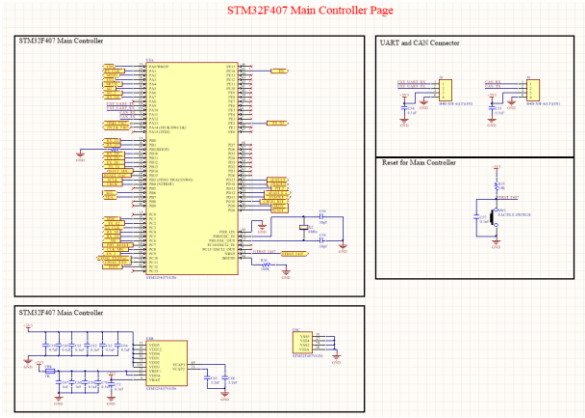
Slika 8. *ADC blok*



Slika 9. *Električna šema Stereo DAC i Microphone*

2.7. Mikrokontrolerska jedinica

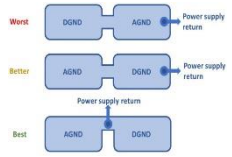
Kontroler koji ima glavnu funkciju je STM32F407VGT6 *[1]*. Električna šema MCU je data na slici 10.



Slika 10. *Električna šema MCU*

3. Pravila za rutiranje

Prilikom rutiranja štampane pločice veoma je bitno podesiti pravila u skladu sa proizvođačem. Ova pravila nisu kompletna, jer kompleksna PCB pločica može imati stotine pravila. Neka od njih su: Pločice sa uzemljenom ravni su bolje jer dozvoljavaju rutiranje signala u microstrip i stripline konfiguraciji. Takodje, značajno smanjuju impedansu uzemljenja, pa time i šum uzemljenja; Signali visoke frekvencije se trebaju rutirati na medjunivoima. Na ovaj način uzemljene ravni se ponašaju kao štit, i štite od elektromagnetne radijacije uzrokovane visokim frekvencijama; Sloj signala mora uvek biti susedan ravni; Povećavanje broja uzemljenih ravni ima veliku prednost, jer smanjuju impedansu uzemljenje plocice, kao i elektromagnetno zračenje; Napajanje-Napajanje i Uzemljenje-Uzemljenje ravni moraju biti strogo spregnute; Konfiguracije bi trebale biti simetrične. Na primer, na osmoslojnoj pločici, ako je sloj 2 ravan, sloj 7 bi isto trebao biti ravan; Slojeve sa signalima je potrebno staviti pored slojeva ravni (napajanja ili uzemljenja). Povratna struja tako može teći na susjednoj ravni, smanjujući induktansu povratne konture na minimum; Da bi se dodatno popravio šum i elektromagnetna interferencija, može se dodatno smanjiti izolacija između signalnog sloja i susjednog sloja ravni; Važna stvar koju treba imati na umu je debljina pojedinih signalnih slojeva. Kada se bira materijal, treba obratiti pažnju na njegove električne, mehaničke i termalne karakteristike; Uzemljenje-odvajanja analogne i digitalne mase (slika 11)



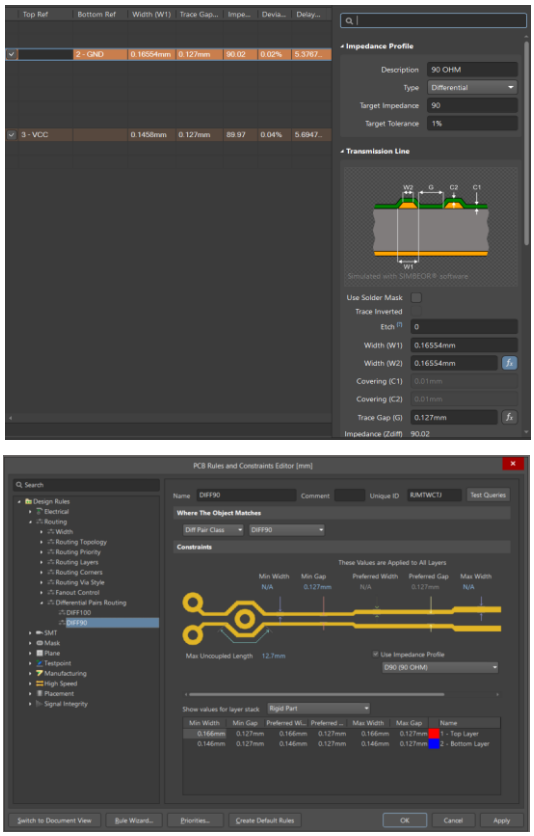
Slika 11. *Odvajanje analogne i digitalne mase*

3.1. Impedansa

Impedansa se može shvatiti kao otpornost u AC režimu. Takođe se meri u omima, kao i otpornost, ali se znatno razlikuje. Omi otpornosti igraju ulogu u DC režimu, dok omi impedanse su u AC režimu, konkretko u kolima koja imaju neku frekvenciju. Kako bi bili sigurni da se vod za signal ispravno projektovao, kao i da kvalitet signala kroz njega ne opada, impedansa se mora pažljivo kontrolisati, pošto je impedansa vodova PCB pločice inače nekontrolisana.

3.2.Važnost izjednačavanja impedansi u PCB dizajnu

Kontrola impedanse u PCB dizajnu se odnosi na izjednačavanje karakteristika materijala podloge, sa dimenzijama i pozicijom vodova, kako bi se obezbedilo da se impedansa signala nekog voda nalazi u tolerisanom opsegu. Pločice sa kontrolisanom impedansom pružaju ponovljive visokofrekventne performanse. Ispod su navedeni argumenti za kontrolu impedanse PCB dizajna. Na visokim frekvencijama, signal na vodu pločice nije samo veza, nego se ponaša i kao komponenta. Karakteristike se mogu razlikovati čak i na istom vodu, na različitim mestima. Kada god se signal propagira dalje kroz vod, različita impedansa stvara refleksiju signala koja zavisi od razlike neizjednačenih impedansi. Veće razlike impedansi stvaraju veću refleksiju i utiču na integritet signala. Refleksija signala ofsetuje stvarni ili primarni signal, i takav signal se često pogrešno tumači od strane visokofrekventnih AD konvertora i ostalih visokofrekventnih kola, rezultujući neuspešnim dekodovanjem stvarnog signala. Primer izjednačavanja impedansi sa diferencijalnim signalima prikazan je na slici 12.



Slika 12. Postavljanje pravila za diferencijalne parove

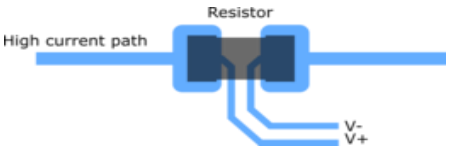
U zavisnosti od vrste signala i brzine prenosa podataka ili frekvenciji, u Tabeli 1. mogu se videti standardne vrednosti impedanse.

Standardni signal	Standardna impedansa (Ω)	Toleransa [%]
USB	90	+/- 15
HDMI	95	+/- 15
IEEE 1394	108	+/- 2
Displayport	100	+/- 20
VGA	75	+/- 5
DVI	95	+/- 15
Ethernet Cat.5	100	+/- 5

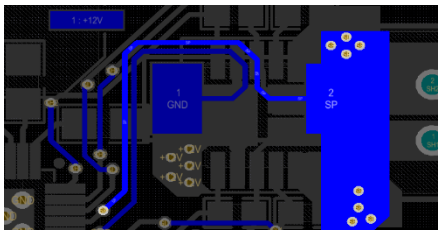
Tabela 1.*Standardne vrednosti impedanse za dizajn PCB-a*

3.3.Kelvinova veza -otpornik za strujni senzor

Kelvinove senzor linije bi trebale da se vežu direktno na otporničke senzor priključke. Vodovi trebaju biti simetrični, sa identičnom dužinom i debljinom (slika 13.a i slika 13.b). Bilo kakve greške vezane za otpornost vodova, kontakata i/ili temperaturnog koeficijenta su eliminisane na taj način. Kelvinova veza za otpornik sa četiri pristupa je neophodna za precizno merenje struje.



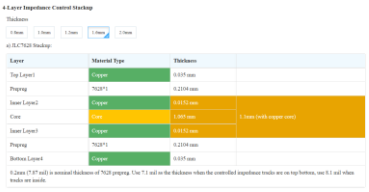
Slika 13.a) Kelvinova veza (engl.*Kelvin connection) [2]*



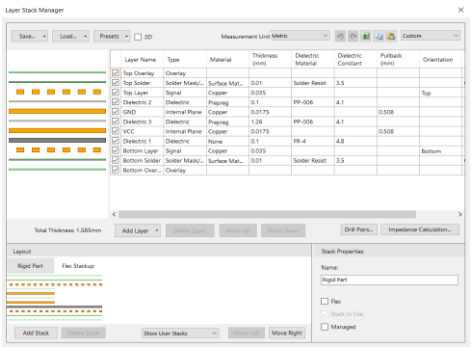
Slika 13.b) *Prikaz rutiranja senzor otpornika*

3.4. Stack up

Preporuka za četveroslojnu pločicu, stack-up od strane proizvođača štampanih pločica JLCPCB prikazana je na slici 14, dok je na slici 15. prikazan stack up projekta.



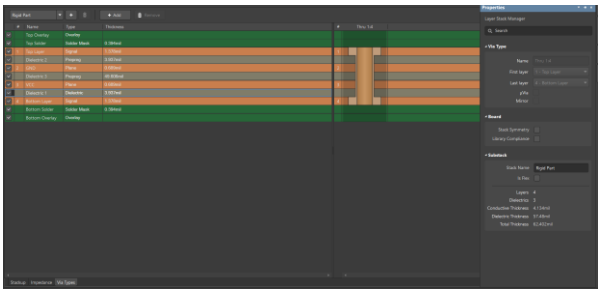
Slika 14. *Preporuka za stack-up [3]*



Slika 15. *Stack up projekta*

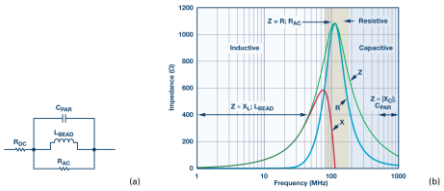
Preporučene vrednosti za VIA od strane proizvođača date su na slici 16, na slici 17. je to realizovano u projektu.

Slika 16. *Preporuka za via*



Slika 17. *Dimnezije via u projektu*

3.5.Ferrite bead



Slika 18. *Karakteristika ferrite bead [4]*

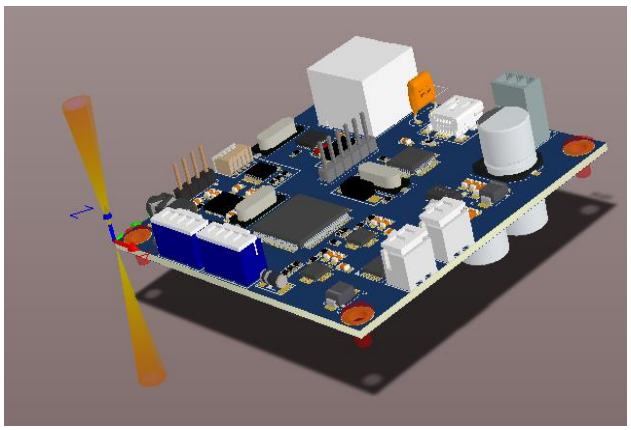
Iz razloga što su impedanse Ferrite Bead-ova induktivne, ferrite bead induktori se koriste da guše visokofrekventne signale u elektronskim komponentama (slika 18). Kada se Ferrite Bead prigušivac postavi na vod za napajanje koji napaja neki elektronski uređaj, on otklanja suvišan visokofrekventan šum koji je prisutan, ili kojeg je generisao izvor DC napajanja. Primenjivanje ferrit-a kao filtera elektromagnetne interferencije ili kao prigušivaca šuma izvora napajanja, najčesce dolazi sa pragom DC struje koji se mora ispoštovati. Struje veće od zadate vrednosti praga mogu oštetiti komponentu. Problematična stvar je da ovaj prag drastično zavisi od temperature. Kako temperatura raste, prag maksimalne struje brzo opada. Maksimalna struja takodje utiče na impedansu ferita. Kako DC struja raste, ferrite bead će se "zašititi" i izgubiti induktansu.

Na relativno visokim strujama, saturacija može smanjiti impedansu ferrite-a i do 90%. U ovom projektu korišten je ferrite bead MPZ1608B471ATA00 (470 Ohms @ 100 MHz, 1A) kod Ethernet-a.

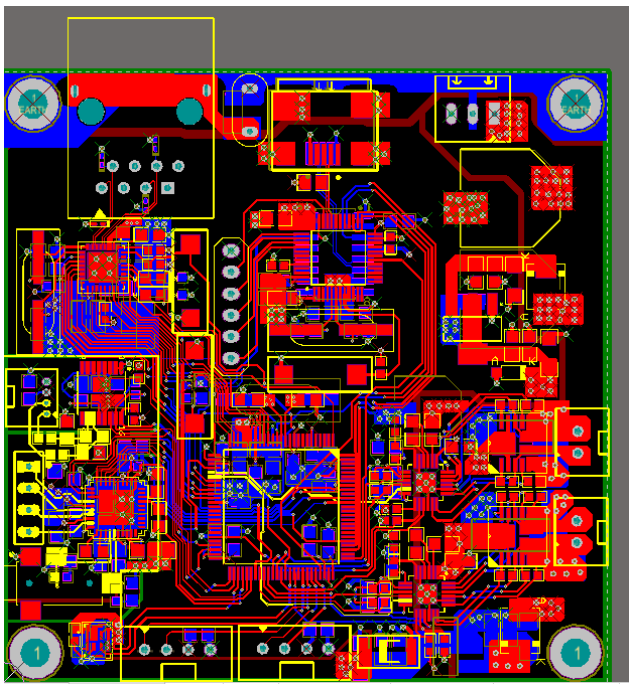
4. Štampana ploča

Štampana ploča je izrađena u četveroslojnoj tehnologiji, dimenzija 71x71 mm. Takođe su dodati bušeni otvori za ugradnju odstojnika radi lakšeg montiranja.

Izgled 3D pločice prikazan je na slici 19. dok je izgled 2D prikazan na slici 20.



Slika 19. *Izgled 3D štampane ploče*

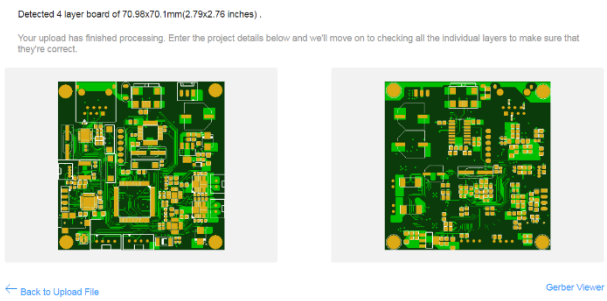


Slika 20.2D *prikaz štampane ploče*

4.2.Fajlovi za fabrikaciju - izlazni fajlovi

NC drill fajl sadrži informacije o bušenju. Gerber format je otvoreni ASCII vektorski format za dizajn štampanih ploča (PCB).

To je standard koji koristi softver industrije PCB-a da opiše slike štampanih ploča: slojeve bakra, maska za lemljenje, podaci o bušenju itd. Izgled Gerber fajlova je prikazan na slici 21.



Slika 21.*Prikaz Gerber fajlova*

Tabela 2. *BOM – Bill of Materials*

Lista komponenata koje se koriste u projektu (podaci o proizvođaču, MPN, količini, kućištu, dezignatoru) se izvozi kada se uradi kompletak šematik za projekat. Prikaz BOM liste je dat u tabeli 2.

5. ZAKLJUČAK

U ovom radu uspešno je projektovana kontrolna ploča. Data su pravila prilikom rutiranja višeslojne štampane pločice, kao i smernice za bolji stack-up. Izvezeni su fajlovi za fabrikaciju, NC Drill, Gerber i BOM. Rad se može nadograditi tako da se dodaju još neke periferije (enkoderi), i da se izvuku eksterni pinovi od mikrokontrolera.

5. LITERATURA

[1] Internet sajt, specifikacija STM32F407VGT6J STM32F407VGT6J STMicroelectronics | Integrated Circuits (ICs) | DigiKey, septembar 2022.

[2] Internet sajt, a Kelvin Connection
What is a Kelvin Connection and a Shunt Resistor? | Alp Electronics, avgust 2022.

[3] Internet sajt, preporuka za Stackup
Controlled Impedance PCB Layer Stackup- JLCPCB, septembar 2022.

[4] Internet sajt, Feririte beads
https://resources.altium.com/p/how-do-ferrite-beads-work-and-how-do-you-choose-right- one, septembar 2022.

Kratka biografija:

Nikolina Bratić rođena je u Trebinju 1996. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva -Primenjena elektronika odbranila je 2019.god.

Vladimir Rajs rođen je 1982. godine u Apatinu. Diplomirao je 2007, a doktorirao 2015. godine na Fakultetu tehničkih nauka u Novom Sadu. Od 2016. godine je bio zaposlen kao docent, od 2021. kao vanredni profesor na Departmanu za elektroniku, energetiku i telekomunikacije FTN-a.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/23BE33Stevanovic

ФАЗНА РЕГУЛАЦИЈА ДРАЈВЕРА ЗА ИНВЕРТОР ЗА БЕЖИЧНИ ПУЊАЧ ЗА ЕЛЕКТРИЧНА ВОЗИЛА

PHASE REGULATION FOR INVERTER DRIVERS FOR WIRELESS CHARGERS FOR ELECTRICAL VEHICLES

Никола Стевановић, Владимир Рајс, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду је приказан један могући начин управљања драјверима инвертора. Идеја је да се гломазан и компликован систем сачињен од рачунара, каблова, функцијског генератора и извора електричне енергије замијени микроконтролером са дисплејом осјетљивим на додир и омогући интеракција са крајњим корисником. Пројекат је базиран на развојном систему Микромедиа 7 са микроконтролером STM32F746ZG и IPS 7-инчном дисплеју резолуције 800x480 пиксела осјетљивим на додир.*

Кључне ријечи: *Микроконтролер, фазно кашњење, PWM сигнал, инвертор.*

Abstract – *The paper presents one possible way of controlling the inverter drivers. The idea is to replace the bulky and complicated system consisting of a computer, cables, function generator and power source with a single microcontroller with a touch-sensitive display and enable interaction with the end user in order to raise the flexibility of the device to a higher level. The project is based on development system Mikromedia 7 with STM32F746ZG microcontroller and an IPS 7-inch touch-sensitive display with a resolution of 800x480 pixels.*

Keywords: *Microcontroller, phase delay, PWM signal, inverter.*

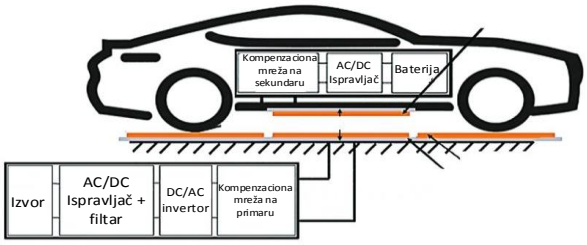
1. УВОД

Електрични аутомобили су тренутно предмет истраживања многих водећих компанија у аутомобилској индустрији, стога како напредује развој електричних возила, тако се ради и на развијању пуњача. Бежични пуњачи постају интересантни и у аутомобилској индустрији. Бежични пуњачи не захтијевају каблове, прикључке, механички контакти не постоје, па је једноставност и робусност оваквих уређаја знатно већа.

На слици 1.1 приказани су сви функционални блокови бежичног пуњача: извор електричне енергије из мреже, исправљач, инвертер, компензационо коло на примару и секундару бежичног преноса енергије, исправљач и на крају потрошач, односно батерија која се пуни.

Тема овог рада јесте адекватна контрола DC/AC инвертора у циљу остваривања жељене снаге пуњења.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Владимир Рајс, ванр. проф.



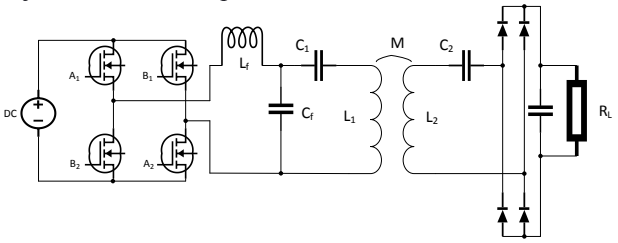
Слика 1.1 Функционални блокови бежичног пуњача

Количина пренесене енергије са примара на секундар директно зависи од контроле инвертора.

2. АНАЛИЗА ПРОБЛЕМА

Захтјев пројекта јесте да се омогући пуњење жељеном снагом за дате параметре, као и унос и приказ свих параметара. Контрола снаге пуњења се врши фазним кашњењем између контролних сигнала на инвертору реализованом као потпуни мост (*full bridge inverter*). Контролни сигнали су импулсно-ширинско модулисани сигнали (PWM) [1]. То значи да је један од захтијева да контролни уређај може генерисати два PWM сигнала на фреквенцијама у опсегу 40kHz-90kHz (највећа ефикасност бежичног преноса је у опсегу 80kHz-90kHz [2], али због испитивања је пожељно да може и ниже од тога).

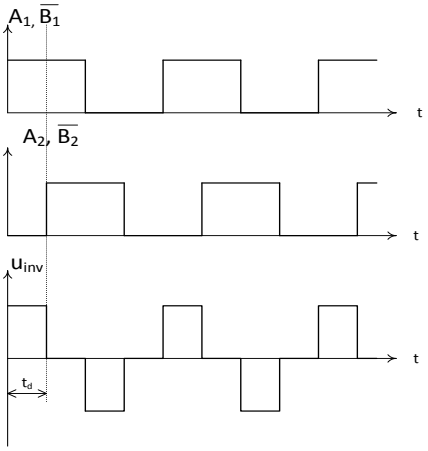
На слици 2.1 се види електрична шема бежичног пуњача. Четири транзистора, прикључена на једносмјерни напон $U_{DC,}$ чине инвертор којим управљају сигнали A_1, A_2, B_1, B_2, које је потребно генерисати помоћу контролера. Управљачки сигнали као и жељени излазни сигнал су приказани на слици 2.2. Пошто су сигнали $A_1 = \overline{B_1}$ и $A_2 = \overline{B_2}$, потребно је генерисати само два управљачка сигнала, док ће инвертовање и прилагођење бити реализовано на драјверским плочицама. Радну фреквенцију инвертора подешава корисник уносом жељене вриједности преко екрана осјетљивог на додир.



Слика 2.1 Електрична шема бежичног пуњача

Улазни једносмјерни напон се такође уноси преко екрана и мора се посебно обратити пажња да се

унесени напон који служи за прорачун фазног кашњења у потпуности слаже са стварним напоном на који је инвертор прикључен.

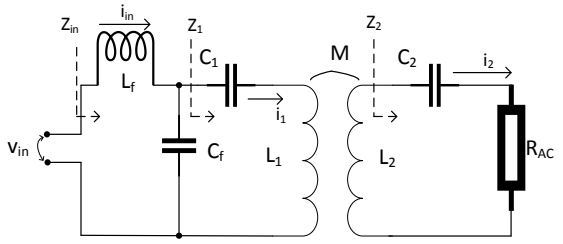


Слика 2.2 Управљачки сигнали које треба генерисати [3]

Калем L_f и кондензатор C_f чине нископропусно филтерско коло другог реда које пропушта само радну фреквенцију инвертора. L_1, C_1 и L_2 и C_2, чине компензационе мреже на примару и секундару. Енергија на примару се прелива из калема у кондензатор, па се тиме добија велика брзина промјене струје кроз калем (зависно од радне фреквенције), а путем спреге два намотаја се енергија преноси из једног калема у други. M је коефицијент спреге, односно међуиндуктивност између примарног и секундарног намотаја. Колико ће максимално енергије бити пренесено зависи од међуиндуктивности M, као и радне фреквенције инвертора. Оптимална радна фреквенција је око 85kHz [2]. Микроконтролер не може директно управљати снажним мосфетима због недовољних струјних и напонских могућности, па њима управљају драјверска кола која немају улогу на прорачун контролних сигнала, па неће бити узета у разматрање.

2.1. Прорачун фазног помјераја

Једначине којима се долази до фазног кашњења између два контролна сигнала биће представљене у наставку. Електрично коло које треба ријешити је дато на слици 2.2.1.



Слика 2.2.1 Електрична шема кола које треба ријешити

Једначина од које се полази и у којој се крије информација о фазном помјерају дата је у наставку (2.1) [3].

$$v_{in} = \frac{2\sqrt{2}}{\pi} U_{DC} \cos\left(180° - \frac{\alpha}{2}\right) \Rightarrow$$
$$\alpha = 180° - 2 * \arccos\left(\frac{v_{in_{RMS}} * \pi}{2\sqrt{2} U_{DC}}\right) \quad (2.1)$$

Уколико се v_{in_RMS} изрази преко познатих параметара, добиће се позната вриједност фазног помјераја.

$$v_{Lmax} = \sqrt{2}\sqrt{P_L R_{AC}} \quad (2.2)$$

Познавајући жељену снагу коју треба остварити и отпорност потрошача (2.2), може се израчунати максималан напон на излазу који је потребан за остваривање задате снаге. Пошто је коефицијент спреге пројектован тако да буде позната међуиндуктивност, може се наћи веза између улазног и излазног напона и улазне и излазне струје. Када се добије та веза, потребно је само уврстити прорачунати улазни напон у једначину 2.1 и добија се жељени резултат.

$$i_2 = \frac{j\omega M i_1}{R_{AC} + jX_2};\ i_1 = \frac{v_1}{Z_1};\ v_1 = \frac{jX_{C_f} Z_1}{Z_{in}(jX_{C_f} + Z_1)} v_{in} \quad (2.3)$$

Увршавањем v_1 у израз за i_1 и увршавањем струје i_1 у израз за струју i_2 , добија се једначина 2.4.

$$i_2 = \frac{j\omega M}{R_{AC} + jX_2} \frac{jX_{C_f} Z_1}{Z_{in}(jX_{C_f} + Z_1) Z_1} v_{in} = \frac{v_L}{R_{AC}} \quad (2.4)$$

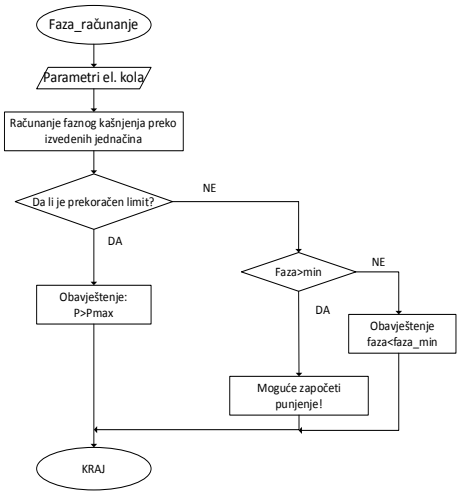
Рјешавањем израза 2.4, добија се једначина којом је могуће израчунати максималан улазни напон на инвертору, што је и потребно за израчунавање фазног кашњења.

$$v_{in\_max} = \left| -\frac{Z_{in}(jX_{C_f} + Z_1) Z_2}{R_{AC} X_{C_f} \omega M} v_L \right|;\ v_{in\_RMS} = \frac{v_{in\_max}}{\sqrt{2}} \quad (2.5)$$

2.2. Алгоритам за одређивање фазног кашњења

Познавајући једначине потребне за израчунавање фазног кашњења, формиран је алгоритам рада који је дат на слици 2.1.1.

Фазни помјерај може бити нека непозната вриједност, јер функција *arccos* може имати за аргумент неку вриједност већу од 1, што није могуће израчунати. Ово се дешава уколико се унесе жељена снага већа од максималне за дату поставку. Проблем може бити и уколико је унесена фаза мања од минималне вриједности. Превише кратак фазни помјерај би нарушио мртво вријеме које уносе драјвери. На овакве случајеве је потребно упозорити корисника и онемогућити почетак пуњења.



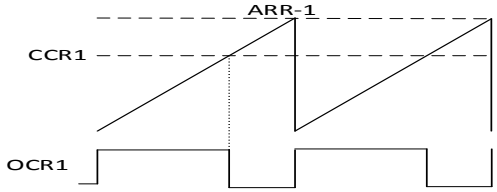
Слика 2.1.1 Алгоритам функције за израчунавање фазе

3. ГЕНЕРИСАЊЕ УПРАВЉАЧКИХ СИГНАЛА ПРЕКО КОМБИНОВАНОГ РЕЖИМА РАДА PWM ГЕНЕРАТОРА

За подешавање и иницијализацију свих периферија и модула одабраног контролера, користи се софтверски

алат STM32CubeMX. То је алат који омогућава да се графички одаберу модули и периферије који се желе користити, па алат сам генерише код који ће иницијализовати одређене регистре. Програмски код је куцан у програмском окружењу STM32CubeIDE компатибилним са конфигуратором кода [4].

Подешавање фреквенције PWM-а и фактора испуне се врши преко ARR і CCR регистара. Принцип генерисања сигнала дат је на слици 3.1 на којој се види да периоду сигнала дефинише ARR регистар, а фактор испуне CCR регистар. Излазни сигнал OCR1 је на високој вриједности све док вриједност бројача не достигне вриједност уписану у CCR регистар, а ниска вриједност траје док се бројач не ресетује, односно док не изброји до вриједности у ARR регистру. У једначинама 3.1 и 3.2 се види како се израчунавају фреквенција и фактор испуне.

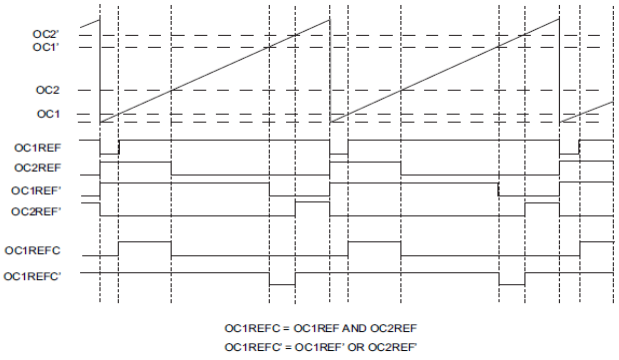


Слика 3.1 Принцип генерисања PWM сигнала

$$f_{PWM} = \frac{f_{clk}}{(ARR+1)*(Prescaler+1)} \quad (3.1)$$

$$Faktor\_ispune[\%] = \frac{CRRx}{ARR+1} * 100\% \quad (3.2)$$

Комбиновани режим рада омогућава потпуну манипулацију над PWM сигналима. Помоћу овог режима могуће је подесити два сигнала произвољног фактора испуне са произвољним фазним кашњењем међу њима. Комбиновани режим рада формира излазни сигнал тако што прави логички збир (енг. *OR*) или логички производ (енг. *AND*) два PWM-а. Детаљније ће бити објашњено у наставку и на слици 3.2. На слици 3.2 су OC1REF і OC2REF два излаза који формирају коначан OC1REFC излаз. OC1REFC је настао као логичка И функција претходна два.



Слика 3.2 Начин генерисања PWM сигнала у комбинованом режиму [4]

Да ли ће излаз настати као логички збир или производ зависи од конфигурације канала који га формирају. Тајмер 3 има четири канала, канал 1 и канал 2 (*CH1 CH2*) формирају један комбиновани излаз на било ком од та два канала, а канал 3 и канал 4 формирају други комбиновани излаз на било ком од та два канала. Ако се излаз жели форимрати на каналу 3 као логички производ канала 3 и канала 4, онда је потребно канал 3 подесити као комбиновани мод 2, а канал 4 у обичан PWM мод 1. У комбиновани мод се ставља онај канал на коме је потребно излаз формирати као комбинацију два PWM-а.

3.1. Принцип формирања управљачких сигнала

Улазни параметар је прорачунато фазно кашњење дато као угао (степени). Први корак је скалирање фазног помјераја у број тактова, односно колико пута треба бројач да окине да би се добио жељени фазни помјерај. Сљедећи корак је одлучивање да ли се ради о проблему када закашњели сигнал прелази преко краја периоде високом или ниском вриједности, јер од тога зависи да ли излазни сигнал треба формирати помоћу *AND* или *OR* комбинације сигнала. Када се одлучи којом комбинацијом се формира коначан сигнал, онда је потребно још само подесити CCR регистре. Када се регистри подесе, PWM сигнал је спреман за генерисање.

Још једно ограничење јесте подешавање фреквенције. ARR регистар је цјелобројног типа, па и фреквенција може бити тачна једино ако се у ARR регистар уписује цјелобројни податак (ако је такт за тајмер дјељив производом жељене фреквенције и прескалера). Пошто се фреквенција подешава у опсегу 40kHz-90kHz, а прескалер је подешен на вриједност 1, онда је опсег вриједности за ARR регистар дат једначином 3.3. Пошто је ARR регистар цјелобројне вриједности, онда би вриједност 533.33 била заокружена на 533, па би подешена фреквенција била 90.056kHz, што је грешка од само 56Hz, тако да је ово рјешење сасвим довољно тачно.

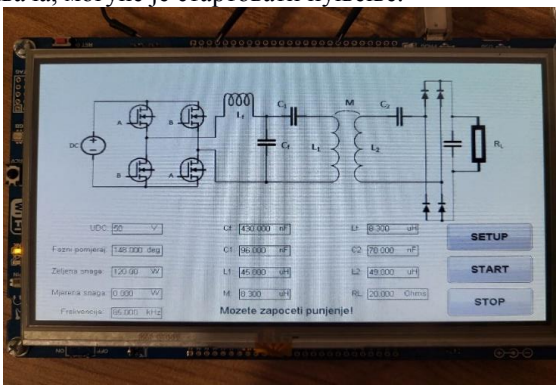
$$ARR = \frac{f_{clk}}{(f_{PWM})*(Prescaler+1)} = \frac{96MHz}{2*f_{PWM}} \quad (3.3)$$

$$\frac{96MHz}{2*90kHz} = 533.33 < ARR < \frac{96MHz}{2*40kHz} = 1200 \quad (3.4)$$

4. ИЗГЛЕД РЕАЛИЗОВАНОГ УРЕЂАЈА И КОНТРОЛНИХ СИГНАЛА

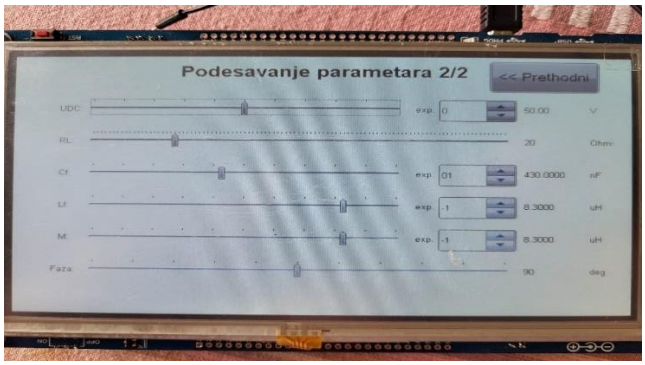
Циљ пројекта јесте да се омогући манипулација снаге пуњења возила преко екрана осјетљивог на додир за различите потребе корисника и различите типове уређаја (различита возила, параметри пуњача...). На сликама 4.1, 4.2 и 4.3 биће приказани редом прозори: почетни екран (испис свих параметара и контролна дугмад), први прозор за подешавање параметара (омогућено подешавање одређеног броја параметара пуњача), други прозор за подешавање параметара (омогућено подешавање остатка параметара).

У случају да су унесене добре вриједности свих параметара и могуће је израчунати фазни помјерај и уколико дати фазни помјерај неће пореметити рад пуњача, могуће је стартовати пуњење.



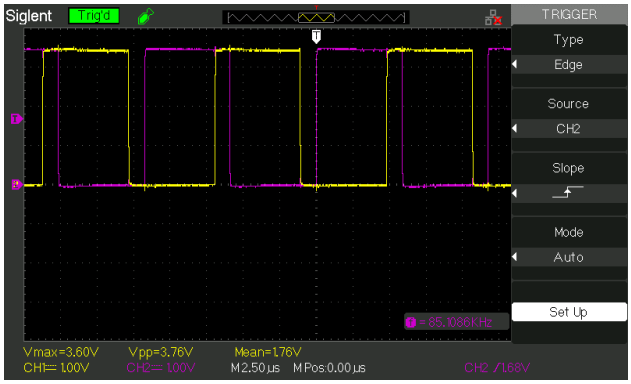
Слика 4.1 Почетни прозор

Слика 4.2 Први прозор за подешавање параметара



Слика 4.3 Други прозор за подешавање параметара

За једну такву ситуацију, гдје је прорачунати фазни помјерај 148º, контролни сигнали изгледају као на слици 4.4.



Слика 4.4. Изглед контролних сигнала за вриједност фазног кашњења од 148º

5. МОГУЋА ПРИМЈЕНА УРЕЂАЈА У ИНДУСТРИЈИ

Као што је речено у уводу, све више индустрија тежи ка развоју електричних аутомобила. Различите аутомобилске компаније праве различита рјешења по питању дистрибуције електричне енергије у аутомобилу. Овакав начин реализовања бежичног пуњача омогућава безбиједно пуњење жељеном снагом за различите типове возила, са различитим карактеристикама параметара на пријемној страни, без икакве хардверске промјене. Довољно је унијети другачије параметре путем дисплеја осјетљивог на додир и тиме ће се фазно кашњење прорачунати на одговарајући начин и омогућити адекватно пуњење.

6. ЗАКЉУЧАК

Циљ је био да се пројектује адекватна контрола снаге за бежичне пуњаче за електрична возила. а сами параметри морају имати могућност да се унесу од стране корисника. На одабраном развојном систему је успјешно реализован комплетан пројекат. Комплетан систем је јако флуидан у раду, прије свега због јако моћног контролера, али и јако удобан за рад због великих димензија дисплеја. Математички прорачун се у великој мјери поклапа са практично реализованим производом. Највећа предност оваквог рјешења јесу димензије и једноставност имплементације у комплетан уређај.

Мана оваквог система јесте што се комплетан систем заснива на математичким једначинама, односно не постоји никаква контрола снаге пуњења (повратна спрега). На дисплеју се налази податак „*mjerena snaga*” који је и даље празан, јер то није имплементирано. Највећи напредак би се добио имплементацијом повратне спреге и фином регулацијом фазног помјераја.

7. ЛИТЕРАТУРА

[1] Afarulrayi, M. Yarafi, W. M. Utomo and A. Zar. FPGA Implementation of Unipolar SPWM for Single Phase Inverter. ICCAIE. 2010. December 5-7.
[2] Varikkottil, S.; Daya, F. Estimation of Optimal Operating Frequency for Wireless EV Charging System under Misalignment. Chennai Campus. 2019.
[3] Guo, Y. Zhang, Y. Li, S. Tao, C. Wang, L. Load Parameter Joint Identification of Wireless Power Transfer Szstem Based on DC Input Current and Phase Shift Angle.
[4] Stlife.augmented. RM0385 Reference Manual. STM32F75xxx and STM32F74xxx advanced Arm-based 32-bit MCUs. STMicroelectronics. June 2018
[5] ElGhanam, E.; Hassan, M.; Osman, A. Design of a High Power, LCC-Compensated, Dynamic, Wireless Electric Vehicle Charging System with Improved Misalignment Tolerance. Energies 2021. 2021.

Кратка биографија:

Никола Стевановић рођен је у Бијељини 1997. године. Дипломирао на Факултету техничких наука из области електротехнике и рачунарства (примјењена електроника) 2020. године. Радио као сарадник у настави двије године.

Владимир Рајс рођен је 1982. године у Апатину. Дипломирао је 2007, а докторирао 2015. године на Факултету техничких наука у Новом Саду. Од 2016. године запослен је као доцент, а од 2020. као ванредни професор на Департману за енергетику, електронику и телекомуникације ФТН-а. Области интересовања су му електроника и примјењена електроника.

U realizaciji Zbornika radova Fakulteta tehničkih nauka u toku 2022. godine učestvovali su sledeći recenzenti:

Aco Antić
Aleksandar Anđelković
Aleksandar Kovačević
Aleksandar Kupusinac
Aleksandar Ristić
Aleksandar Selakov
Aleksandra Radulović
Aleksandra Pešterac
Andraš Anderla
Andrija Rašeta
Atila Zelić
Bojan Batinić
Bojan Matić
Bojan Tepavčević
Borislav Savković
Branislav Atlagić
Branislav Stevanov
Branka Nakomčić
Branko Milosavljević
Branko Škorić
Damir Đaković
Danijela Ćirić
Danijela Gračanin
Danijela Lalić
Darko Čapko
Darko Reba
Darko Stefanović
Dejan Ecet
Dejan Lukić
Dejan Reljić
Dejan Jerkan
Dejan Movrin
Dejan Ubavin
Dejana Nedučin
Dragan Ivanović
Dragan Ivetić
Dragan Jovanović
Dragan Pejić
Dragan Ružić
Dragana Konstantinović
Dragana Šarac
Dragoljub Šević
Drago Žarković
Duško Bekut
Đorđe Vukelić
Goran Jeftenić
Goran Savić
Goran Sladić
Goran Švenda
Goran Tepić
Gordan Stojić
Gordana Ostojić
Igor Dejanović
Igor Peško
Iva Šiđanin
Ivana Mihajlović
Igor Maraš
Ivan Prokić
Ivana Katić
Ivana Maraš
Ivana Miškeljin
Jasmina Dražić
Jelena Atanacković Jeličić
Jelena Borocki
Jelena Demko Rihter
Jelena Ivetić
Jelena Radonić
Jelena Slivka
Jelena Spajić
Kalman Babković
Lazar Kovačević
Lidija Krstanović
Ljiljana Popović
Ljubica Duđak
Magdolna Pal
Maja Turk Sekulić
Maja Petrović
Marija Silađi
Marinko Maslarić
Marko Lazić
Marko Marković
Marko Todorov
Marko Vekić
Maša Bukurov
Matija Stipić
Mijodrag Milošević
Milan Delić
Milan Gavrić
Milan Marinković
Milan Mirković
Milan Rapajić
Milan Rackov
Milan Trivunić
Milan Vidaković
Milena Krklješ
Milica Kostreš
Milica Miličić
Miloš Simić
Milovan Lazarević
Milja Simeunović
Miodrag Milutinov
Miodrag Žigić
Mirjana Malešev
Miroslav Zarić
Mirko Borisov
Mirko Raković
Miro Govedarica
Miroslav Kljajić
Miroslav Popović
Miroslav Zarić
Mitar Jocanović
Mladen Tomić
Mladen Radišić
Nataša Samardžić
Nebojša Brkljač
Nebojša Radović
Nebojša Ralević
Neda Milić Keresteš
Nemanja Kašiković
Nemanja Sremčev
Nemanja Tasić
Nenad Grahovac
Nenad Simeunović
Nikola Vojnović
Petar Mirković
Platon Sovilj
Radivoje Dinulović
Radomir Kojić
Ratko Obradović
Sandra Dedijer
Saša Medić
Slavica Mitrović
Senka Bajić
Slobodan Morača
Slobodan Šupić
Srđan Popov
Srđan Vukmirović
Stevan Gostojić
Stevan Grabić
Stevan Milisavljević
Stevan Stankovski
Strahil Gušavac
Svetlana Bačkalić
Svetlana Nikoličić
Tamara Ćeranić
Veran Vasić
Vesna Stojaković
Višnja Žugić
Vladimir Ilić
Vladimir Katić
Vladimir Mučenski
Vlastimir Radonjanin
Vuk Bogdanović
Vuk Vranjkovic
Zdravko Tešić
Zoran Čepić
Zoran Jeličić
Zoran Papić
Željen Trpovski