УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

ЗБОРНИК РАДОВА
ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке – зборници
Година: XL
Број: 4/2025

Нови Сад

Едиција: „Техничке науке – Зборници“
Година: XL Свеска: 4
Издавач: Факултет техничких наука Нови Сад

Главни и одговорни уредник: проф. др Борис Думнић, декан Факултета техничких наука у Новом Саду

Савет за библиотечку и издавачку делатност ФТН, проф. др Селена Самарџић Цвијановић, председник.

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

CIP-Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

378.9(497.113)(082)
62

ЗБОРНИК радова Факултета техничких наука / главни и одговорни уредник
Борис Думнић. – 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 четврта 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».

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

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

Зборник излази у два облика – електронском на веб-страници Факултета техничких наука (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 17. 9. 2024. до 30. 9.2024. 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 ФТН-а.

У свесци са редним бројем 4, објављени су радови из области електротехнике и рачунарства.

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.

Континуираним радом и унапређењем квалитета часописа, план је да часопис постане препознатљив међу ауторима, чиме ће значајно допринети да се оствари мото Факултета техничких наука:

„Високо место у друштву најбољих“

Уредништво

Садржај

Електротехника и рачунарство

Зборник радова Факултета техничких наука, Нови Сад

UDK: 4.9
DOI: https://doi.org/10.24867/30BE11Lekic

ПРЕДИКЦИЈА ПОПУЛАРНОСТИ YOUTUBE TRENDING ВИДЕО СНИМКА

PREDICTING THE POPULARITY OF A YOUTUBE TRENDING VIDEO

Душан Лекић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Са порастом популарности друштвених мрежа и платформи за дељење видео садржаја, предвиђање популарности видео записа постало је важно за ауторе садржаја, људе који се баве маркетингом као и оглашиваче. У овом раду истражујемо различите факторе који утичу на популарност YoyTубе trending видео снимка, као што су број лајкова, број дислајкова, број и сентимент коментара, наслов и слика самог видео снимка, као и карактеристике аутора, односно његовог YouTube канала.*

Кључне речи: *предикција популарности, Youtube, неуронске мреже, NLP, Computer Vision*

Abstract – *With the rise in popularity of social networks and video sharing platforms, predicting video popularity has become important for content creators, marketers and advertisers alike. In this paper, we investigate various factors that influence the popularity of a YouTube trending video, such as the number of likes, the number of dislikes, the number and sentiment of comments, the title of the video itself, the image of the video, as well as the characteristics of the author or his YouTube channel.*

Keywords: *popularity prediction, Youtube, neural networks, NLP, Computer Vision*

1. УВОД

Од свог настанка 2005. године, *Youtube* је постао синоним за онлине видео садржај. Иако су се појавиле друге платформе попут *Vimeo*, *TikTok*-а и *Instagram Reels*-а, *Youtube* остаје најпопуларнија са 2 милијарде активних корисника месечно, захваљујући огромној количини садржаја, функцијама претраге, персонализованим препорукама и могућности претплате на канале. *Youtube*-ри могу монетизовати садржај од 2007. године, а додатно зарађивати путем спонзорстава, маркетинга и продаје производа, што је многима постало основни извор прихода. Популаран видео има велики број прегледа и интеракција, док трендинг видео нагло добија много прегледа у кратком периоду. У овом раду представљамо студију о предвиђању популарности *Youtube trending* видеа коришћењем техника машинског учења, анализирајући карактеристике видеа и коментара. Коришћени су подаци о трендинг садржају из САД, укључујући сентимент анализу коментара.

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

Након инжењеринга својстава и експлоративне анализе, предикција је извршена класификационим моделима и комбинованом архитектуром неуронских мрежа где су анализирани нумерички, текстуални и визуелни подаци.

2. ПРЕДИКЦИЈА ПОПУЛАРНОСТИ

Предикција популарности онлајн садржаја укључује више корака и техника машинског учења. Прво, прикупљају се подаци о видеима, укључујући наслове, описе, тагове, број прегледа, лајкова, коментара и дељења. Ови подаци се затим обрађују и трансформишу у одговарајући формат за машинско учење. Један од важних корака је инжењеринг својстава, где се генеришу нове карактеристике као што су сентимент анализа коментара и анализа кључних речи у насловима и описима.

Затим се модели машинског учења, као што су регресија, класификација или неуронске мреже, тренирају на овим подацима како би научили да предвиђају популарност садржаја. Ови модели могу користити технике као што су *gradient boosting trees*, *random forests* и *support vector machines* за анализу нумеричких података и рекурентне и конволуцијске неуронске мреже за анализу текстуалних и визуелних података. На крају, модели се евалуирају и оптимизују како би пружили што тачније предикције, што омогућава креаторима садржаја и маркетиншким стручњацима да боље разумеју који садржај ће вероватно привући највише пажње и ангажовања публике.

3. КОРИШЋЕНЕ ТЕХНИКЕ И МОДЕЛИ

3.1 SVM

Support Vector Machine (SVM) је напредни алгоритам машинског учења за класификацију и регресију који проналази оптималну хиперравнину која раздваја различите класе података са највећом могућом маргином. SVM је ефикасан у решавању линеарно одвојивих проблема и може се проширити на нелинеарне коришћењем кернел метода. Кернел трик омогућава трансформацију података у веће димензије, олакшавајући њихово раздвајање. SVM-ови су примењени у областима као што су здравство, обрада природног језика и препознавање слике. Алгоритам је дизајниран за бинарну класификацију, али се може прилагодити за проблеме више класа. Једна од главних предности SVM-а је избегавање прекомерне усклађености избором оптималне хиперравнине.

3.2 Стабла одлучивања

Стабла одлучивања су интуитиван алгоритам машинског учења за класификацију и регресију који деле податке на подскупове, креирајући структуру сличну стаблу са чворовима који представљају одлуке на основу одређених карактеристика. Сваки чвор представља атрибут података, док гране представљају исходе тих одлука, водећи до коначних листова који представљају класе или континуиране вредности. Главна предност стабала је једноставност и лакоћа интерпретације, као и могућност рада са нумеричким и категоријалним подацима без много претходне обраде. Међутим, склона су прекомерној усклађености, нарочито када су дубока и комплексна. Овај проблем се често решава обрезивањем грана. Иако су моћна, стабла одлучивања највећу снагу показују у комбинацији са алгоритмима као што су Random Forest и XGBoost.

3.3 Random Forest

Random Forest је алгоритам машинског учења који комбинује више стабала одлучивања како би побољшао тачност и стабилност предикција. Ова техника креира више стабала одлучивања користећи различите подскупове података и узорке карактеристика, а коначна предикција се добија гласањем или просеком предикција свих стабала. *Random Forest* је отпоран на прекомерно усклађивање, јер различита стабла смањују варијанс и повећавају општу тачност модела. Овај алгоритам обрађује велике скупове података са великим бројем карактеристика без значајне претходне обраде. Такође пружа мере важности карактеристика, корисне за разумевање утицаја различитих карактеристика на предикцију.

3.4 XGBoost

XGBoost (*Extreme Gradient Boosting*) је напредни алгоритам машинског учења који користи градијентно појачавање за стварање високоефикасних модела за класификацију и регресију. Овај алгоритам ради у фазама, при чему свака нова фаза покушава да исправи грешке претходних фаза. XGBoost користи ансамбл стабала одлучивања, где сваки нови модел минимизира резидуалне грешке претходних модела. Овај приступ резултира моделима високе прецизности и перформанси. XGBoost је познат по брзини и ефикасности, захваљујући паралелној обради и оптимизацијама у управљању меморијом. Алгоритам пружа регуларизацију, која спречава прекомерну усклађеност и побољшава генерализацију модела.

3.5 Дубоке неуронске мреже

Дубоке неуронске мреже (DNN) су тип неуронских мрежа са више скривених слојева који могу аутоматски учити и представљати високе нивое апстракције у подацима. Оне се састоје од низа слојева повезаних неуронима, где сваки неурон прима улаз, обрађује га и прослеђује следећем слоју. DNN су примењене у компјутерском виду, обради природног језика и препознавању говора. Њихова обука захтева велике скупове података и значајну рачунарску моћ, што је омогућено развојем графичких процесорских јединица (GPU). Иако су веома моћне, дубоке неуронске мреже су склоне проблемима као што су превелика усклађеност и дуго време обуке. За решавање ових проблема често се користе технике као што су регуларизација и dropout.

3.6 Рекурентне неуронске мреже

Рекурентне неуронске мреже (RNN) су дизајниране за рад са секвенцијалним подацима и способне су да задрже информације о претходним стањима кроз унутрашње петље. Ово их чини идеалним за анализу временских серија, обраду природног језика и друге задатке где је контекст битан. RNN-ови памте претходне улазе кроз своју рекурентну архитектуру, омогућавајући коришћење информација из претходних корака за предвиђање будућих. Проблем нестајања или експлодирања градијената током обуке отежава учење дугорочних зависности, што је делимично решено развојем *Long Short-Term Memory* (LSTM) и *Gated Recurrent Unit* (GRU) архитектура. RNN-ови се користе у машинском преводу, препознавању говора и генерисању текста.

3.7 Конволуцијске неуронске мреже

Конволуцијске неуронске мреже (CNN) су оптимизоване за обраду структурираних података попут слика и видео снимака. CNN-ови користе конволуционе слојеве који примењују филтере на улазне податке како би екстраховали релевантне карактеристике као што су ивице, текстуре и обрасци. Састоје се од више конволуционих и пуллинг слојева, који су затим повезани са једним или више потпуно повезаних слојева. Главна предност CNN-ова је њихова способност да аутоматски уче и екстрахују значајке из слика, смањујући потребу за ручним инжењерингом карактеристика.

4. МЕТОДОЛОГИЈА

Методологија решења описаног у овом раду реализована је кроз следеће кораке: прикупљање података, експлоративна анализа података, инжењеринг својстава, претпроцесирање података, сентимент анализа коментара, обучавање и оптимизација класификационих модела, анализа текста рекурентном неуронском мрежом и формирање комплексне неуронске мреже која комбинује све наведене типове улаза за класификацију видео снимка. У наставку овог поглавља детаљније су описане све претходно наведене фазе методологије.

Поред основног скупа података потребно је било и прикупљање, где смо за сваки појединачни видео снимак помоћу Selenium алата преузели 50 најрелевантнијих коментара са видео снимка, слика(*thumbnail*), као и броја претпланика и прегледа које је аутор снимка акумулирао. Коментари нам могу омогућити дубљи залаз у њихов утицај на популарност видео снимка. Ми смо овде анализирали да ли сентимент коментара доприноси прегледима, тј. да ли видео који има добру повратну информацију од гледаоца је инхерентно популарнији због тога.

Информације од канала може се обогатити почетни скуп података и побољшати метрике примењиваних алгоритама машинског учења, што и јесте основна идеја укључивања информација о каналу у овом рад.

Након експлоративне анализе смо закључили да су током година најпопуларније категорија видеа у области музике, забаве, спорта и игрица.Такође смо закључили да је корелација између броја лајкова и броја прегледа доста већа од корелације броја прегледа са бројем коментара и бројем дислајкова. Међутим, ниједна од добијених вредности није занемарљива и има свој допринос у процесу тренирања модела, па ниједна нумеричка колона није избачена.

Инжењерингом својстава користимо постојећи скуп података како би га проширили и добили неко ново знање и значење од њега. Ово се може постићи на мноштво начина. На нашем примеру смо увели податке као што су: временски период који је био потребан да видео постане *trending*, имена категорија спајањем идентификатора са именима како би се добила прегледнија експлоративна анализа као и број негативних, неутралних и позитивних коментара за сваки видео снимак.

Претпроцесирање у овом раду се састоји из више корака који нам омогућују лакши рад са подацима и који дозвољавају да модел има прикладне податке као улазе. Неки од корака су: претварање индетификатора категорије видео снимка у њено име, претварање података кад је видео објављен и кад је ушао у *trending* у податак колико је видео снимку требало да уђе у *trending*, замена *outlier*-а са границама користећи *upper/lower bound* технику, замена недостајућих вредности са вредношћу медијана као и мноштво осталих.

Анализа сентимента прикупљених коментара је урађена уз помоћ две *Python* библиотеке, VADER [1] и *TextBlob* [2]. Резултат семантичке анализе једног конкретног коментара је поларитy и изражен је у опсегу од -1 до 1. Где доња граница -1 представља да је коментар негативан док горња граница 1 представља да је коментар позитиван. Неутралним коментаром сматрамо онај чија апсолутна вредност поларитy не прелази 0.25.

Да би класификација видео снимка била могућа, неопходно је креирање категоричког атрибута који представља дискретне вредности броја прегледа. Ово је постигнуто дискретизацијом континуалне вредности броја прегледа у две групе балансиране по величини. По узору на постојећу литературу коришћени су следећи класификациони модели: *Support Vector* класификатор, *Random Forest* ансамбл класификационих стабала, ансамбл класификационих стабала у *Extreme Gradient Boosting* конфигурацији и вишеслојна мрежа перцептрона.

Како би се предвидела популарност видео снимака, може се применити неуронска мрежа, која је способна да научи сложене обрасце у подацима и да их користи за предвиђање циљне променљиве - популарности видео снимка. Поред нумеричких података, у овом моделу имамо могућност да интегришемо и текстуалне податке као што је наслов видео снимка. Ово ће нам показати колико утицај могу имати ови подаци и колико цлицкбаит култура са насловом и погађање кључних речи описом може да утиче на популарност видео снимка. Мрежа комбинује још један тип података, а то је слика (*thumbnail*) која представља видео визуелно и игра велику улогу у његовој популарности. Желимо да сазнамо да ли квалитет ове слике којој доста аутора дају пажњу и придодају успех видеа има утицај на број прегледа видеа. Мрежа се састоји од три гране која прима одређене улазе, који пролазе кроз неколико слојева, који су специфични по грани, и на крају се конкатенирају како би добили излаз, тј. предикцију модела. Улаз у првој грани је слика, у другој наслов, а у трећој сви остали подаци који су представљени нумериком.

5. ЕКСПЕРИМЕНТИ

Скуп података који се користи је *Youtube Trending Video Datase*t са *Kaggle*-а који се ажурира дневно и може се преузети са [4]. За овај пројекат је задњи пут преузет 28.01.2023 како би имали конзистентан скуп података. Други део скупљања скупа података је прикупљање Youtube коментара које је урађено преко Selenium-а. Неке од значајних колона одабраног скупа су: наслов видео снимка, број лајкова и дислајкова, број коментара, опис, линк ка слици видео снимка и број прегледа. Поред и оригиналних додати су и атрибути канала и анализе сентимента.

Експерименти су се сводили на тестирање тачности модела додавањем и одузимањен одређених атрибута скупа података као што су резултати анализе сентимента и тагови видео снимка. Класификациони модели тестирани су на тест скупу података, издвојеном у почетној фази пројекта (20% од целог скупа података) насумичним поступком. Валидациони скуп није био потребан пошто библиотека са којом се радила унакрсна валидација има уграђено издвајање и евалуирање валидационих података. Извршен је и преглед прошлих експеримената, како би се видела експериментација са подацима и како су они утицали на резултате.

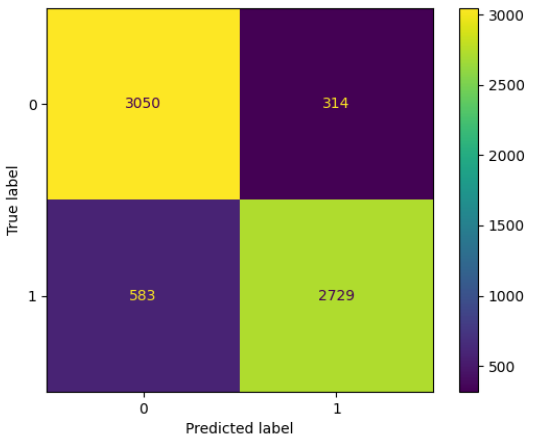
Класификациони модели евалуирани су употребом метрике тачности, AUC, MAE и F-score. Метрике су изабране у складу са проблемом, имајући у обзир да је циљ бинарна класификација, као и да је број позитивних и негативних класа добро избалансиран у датом скупу података. Коришћена је и матрица конфузије, да би се уочило на којим класама модел више греши. Поред матрице конфузије извршена је и додатна анализа грешака како би могли да уочимо зашто наши модели греше и како можемо да побољшамо њихове перформансе. Урађена је и анализа тагови и анализа сентимента утичу на класификационе моделе.

6. РЕЗУЛТАТИ И ДИСКУСИЈА

Табела 1 приказује резултате евалуације свих коришћених модела. Најбоље резултате нумеричке анализе је дао *Random Forest* модел. Видимо како су се, у односу на вишеслојну мрежу перцептрона, мере евалуације са анализом текстуалних елемената смањиле, док се при убацивању визуелних елемента прецизност повећала. Међутим када уместо тренирања наше конволутивне мреже имамо *ResNet50* перформансе се погоршају. Можемо да закључимо да визуелни атрибути имају већи и позитиван утицај у односу на текстуалне са нашим ограниченим ресурсима за тренирање. Такође, видимо да наше конволутивна мрежа има боље перформансе од *ResNet50* која је претходно обучавана. Можемо да претпоставимо да је разлог за то велики корпус података и самим тим слика, где је наша конволутивна мрежа имала довољно ресурса да естрахује смислене атрибуте и да те обрасце примени у разликовању популарних и непопуларних видео снимака.

Табела 1. *Евалуација без тагова са анализом сентимента и оптимизованим хиперпараметрима*

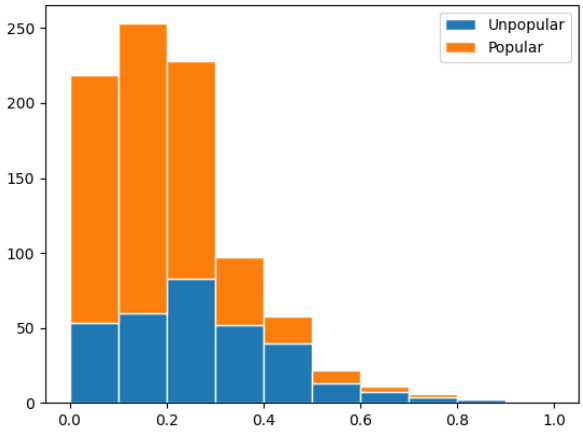
Класификациони модели	Метрике [%]			
	Тачност	AUC	F1	MAE
SVM	86.56	86.53	85.88	13.43
Random Forest	87.60	87.57	87.13	12.40
XGBoost	87.52	87.47	87.01	40.75
MLP	85.77	85.73	84.90	14.23
MLP са LSTM	85.38	85.34	84.46	14.62
MLP са LSTM i CNN	85.82	85.81	85.17	14.21
MLP са LSTM i ResNet50	85.31	85.25	84.03	14.69



Слика 1. *Матрица конфузије* SVM *модела*

Матрице конфузије SVM-а је приказана на слици 1 Матрице модела су у сличним односима као и ова. Видимо да се највише грешака јавља код видео снимака који су популарни а модел их препознаје као непопуларне.

Претпостављамо да се ово може објаснити чињеницом да је теже класификовати популарне снимке који имају мање цифре кад се тиче најважнијих атрибута за класификацију. Ово се може приметити на слици 2 где имамо хистограм лајкова код погрешних предикција, и где видимо да модел греши код малих броја лајкова и то највише код популарних видео снимака.



Слика 2. *Хистограм лајкова код погрешних предикција у односу на популарне и непопуларне видео снимке*

7. ЛИТЕРАТУРА

[1] VADER https://vadersentiment.readthedocs.io/en/latest/

[2] TextBlob. https://textblob.readthedocs.io/en/dev/

[3] YouTube Trending Video Dataset (updated daily): RISHAV SHARMA. Available: https://www.kaggle.com/datasets/rsrishav/YouTube-trending-video-dataset

Кратка биографија:

Душан Лекић рођен је у Краљеву 2000. год. Студент на мастер академским студијама Факултета техничких наука, у оквиру Универзитета у Новом Саду. Након завршене средње школе, уписује 2018. године Факултет техничких наука, смер Рачунарство и аутоматика. У трећој години се усмерава на модул Примењене рачунарске науке и информатика. Након завршених основних студија, уписује 2022. године мастер академске студијена Факултету техничких наука, смер Рачунарство и аутоматика, модул Интелигентни системи.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/30BE13Santrac

PREDIKCIJA ŽANRA FILMA UPOTREBOM MAŠINSKOG UČENJA

MOVIE GENRE PREDICTION USING MACHINE LEARNING

Stefan Santrač, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – *Prilikom izrade novih filmova često dolazi do curenja informacija. Takve informacije su nepotpune i na osnovu njih možemo saznati ko radi na filmu, kada se film premijerno prikazuje, naslov filma, itd. Da bi se formirala puna slika o filmu potrebno je poznavati i kom žanru film pripada. U ovom radu vrši se predikcija žanra filma korišćenjem Multilabel klasifikatora (Multilabel k Nearest Neighbours, Classifier chains, Binary relevance) na osnovu informacija o osobama koje rade na filmu, koji posao te osobe obavljaju, naslova filma, godini premijernog prikazivanja i informacije da li je film namenjen za decu. Rezultati predikcija su evaluirani pomoću Micro-average i Macro-average metoda evaluacij.*

Ključne reči: — *film; predikcija žanra; multilabel klasifikacija; mašinsko učenje*

Abstract – *When creating new films, information often leaks. Such information is incomplete and can reveal details like who is working on the film, when it will premiere, the film's title, etc. To get a complete picture of the film, it's necessary to know the genre it belongs to. In this study, genre prediction is performed using Multilabel classifiers (Multilabel k-Nearest Neighbors, Classifier Chains, Binary Relevance) based on information about the people working on the film, their roles, the film's title, the year of its premiere, and whether the film is intended for children. The prediction results are evaluated using Micro-average and Macro-average evaluation methods.*

Keywords: *film; genre prediction; multilabel classification; machine learning*

1. UVOD

Žanr je koncept koji se koristi u filmskim studijama i teoriji filma za opisivanje sličnosti između grupa filmova zasnovanih na estetskim ili širim društvenim, institucionalnim, kulturnim i psihološkim aspektima. Filmski žanr ima sličnosti u formi i stilu, temi i komunikativnoj funkciji. Filmski žanr se stoga zasniva na skupu konvencija koje utiču kako na produkciju pojedinačnih dela u okviru tog žanra, tako i na očekivanja i iskustva publike. Žanrovi se koriste u industriji, u proizvodnji i marketingu filmova, od strane filmskih analitičara i kritičara u analizi filma, i kao okvir za publiku u odabiru i doživljaju filmova.

Svaki film može da pripada grupi više žanrova, što utiče na njegovu popularnost, kao i na grupu ljudi kojima je namenjen i koji će ga pogledati. Žanr filma u velikoj meri zavisi od glumaca koji se u njemu pojavljuju, režisera, scenarista, godine prikazivanja, itd. Živimo u svetu društvenih mreža i široko rasprostranjenih medija, gde svakodnevno procure informacije o budućoj saradnji poznatih glumaca i režisera, godini u kojoj će film premijerno biti prikazan, kao i mnoge druge informacije vezane za film. Ljubiteljima filmova širom sveta je cilj da saznaju i sam žanr filma, kako bi znali da li im taj film upada u sferu interesovanja i da bi odlučili da li će pratiti njegov razvoj.

Upravo ovim problemom se i bavi ovaj rad. U njemu će biti prikazano više različitih rešenja za predikciju žanra filma na osnovu glumačkog kadra, producenta, režisera, scenariste, kompozitora muzike za film, kinematografa, da li je film namenjen za decu, naslova filma i godine kada je film premijerno prikazan. Sličan problem je rešavan u drugim naučnim radovima, gde je rađena predikcija žanra na osnovu različitih parametara, kao što su filmski posteri, radnja filma ili kratki reklamni prikaz filma (eng. *trailer*). Za *multilabel* klasifikaciju korišćene su metode *Multilabel k Nearest Neighbours*, *Binary Relevance* i *Classifier Chains*.

U ovom radu korišćeno je više metoda predikcije i one će biti poređene pomoću *Micro-average* i *Macro-average* metoda evaluacije. Najbolje rezultate su dale *Classifier Chains* metoda na osnovu Micro-average evaluacije i *Multilabel k Nearest Neighbours* metoda na osnovu *Macro-average* evaluacije. Rezultati predikcija su definisani i rangirani i te informacije se mogu iskoristiti u budućim radovima koji se bave sličnim temama.

2. METODOLOGIJA

U ovom poglavlju je predstavljena implementacija sistema za predikciju žanra filma na osnovu ulaznih podataka kao što su tip/format sadržaja (npr. film, serija, video...), naslov sadržaja koji su tvorci koristili u promociji filma, da li je sadržaj namenjen za decu, godina kada je sadržaj premijerno prikazan, osobe koje rade na filmu i posao koji te osobe obavljaju. Očekivani izlaz sistema predstavlja skup do najviše 3 žanra filma.

Pre same primene metoda za predikciju žanra filma bilo je potrebno izvršiti izmene nad skupom podataka kako bi podaci bili spremni za obuku i testiranje modela. Pošto su se podaci za polja *primaryTitle*, *isAdult*, *startYear*, *genres*, *category*, *primaryName* u skupu podataka nalazila

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

u tekstualnom obliku izvršena je njihova konverzija u numeričke vrednosti.

Term frequency-inverse document frequency (TF-IDF) je tehnika pronalaženja informacija koja meri frekvenciju termina (TF) i njegovu inverznu frekvenciju dokumenta (IDF). Svaka reč ili termin koji se pojavljuje u tekstu ima svoj TF i IDF rezultat. TF predstavlja broj pojavljivanja tražene reči u svim dokumentima, što pokazuje koliko je određeni termin bitan. Dok IDF predstavlja logaritam količnika ukupnog broja dokumenata i broja dokumenata u kojima se ovaj termin pojavljuje, on smanjuje važnost termina koji se često pojavljuju u dokumentima. Konkretno u ovom radu TF-IDF je jedna od metoda korišćena za konverziju ulaznih podataka iz tekstualnog formata u numeričke vrednosti pre prosleđivanja u model.

Druga tehnika korišćena za konverziju tekstualnih podataka u vektore numeričkih vrednosti je *Doc2Vec*. On predstavlja generalizaciju *Word2Vec* tehnike. *Word2Vec* može da napravi procene visoke tačnosti o značenju reči na osnovu njihovog pojavljivanja u tekstu. Ove procene daju asocijacije reči sa drugim rečima, na primer, reči poput „kralj“ i „kraljica“ bile bi veoma slične jedna drugoj.

Word2Vec reprezentacija je kreirana korišćenjem 2 algoritma: *Continuous Bag-of-Words model* (CBOW) i *Skip-Gram model*. *Continuous Bag-of-Words* predviđa trenutnu reč na osnovu konteksta, odnosno na osnovu okolnih reči. Korišćenjem *Continuous Bag-of-Words* algoritma se koriste reči „mačka”, „je”, „sela” za predviđanje reči „na”.

Skip-gram algoritam radi suprotno od CBOW algoritma: umesto da se svaki put predviđa jedna reč, koristi se jedna reč da bi se predvidele sve okolne reči, odnosno kontekst.

Cilj *Doc2Vec-a* je da kreira numerički prikaz dokumenta, bez obzira na njegovu dužinu. Dokumenti nisu iste logičke strukture kao reči, algoritam koji rešava ovaj problem je proširenje CBOW modela. Umesto korišćenja samo reči za predviđanje naredne reči, dodaje se još jedan vektor obeležja (Paragraf id), koji je jedinstven za dokument. Ovaj metod se naziva *Distributed Memory version of Paragraph Vector* (PV-DM).

Kao i kod *Word2Vec-a*, može se koristiti drugi algoritam, sličan *Skip-gramu*, koji se naziva *Distributed Bag of Words version of Paragraph Vector* (PV-DBOW).

Dodatno za *Doc2Vec* je prvo potrebno prikupiti sve reči iz labela i proslediti ih *Doc2Vec* modelu kako bi mogao da se napravi rečnik, zatim se taj model obučava i koristi se *infer_vector* metoda kako bi se reči iz skupa podataka transformisale u vektore numeričkih vrednosti.

Multilabel embedding tehnike [1] pojavile su se kao odgovor na potrebu da se nosi sa velikim prostorom za labele, ali sa razvojem računara postale su metod za poboljšanje kvaliteta klasifikacije. U ovom radu korišćen je *LabelNetwork Embeddings*.

S obzirom da film može pripadati grupi više žanrova, u ovom radu će se koristiti *Multilabel k Nearest Neighbours* (ML-KNN) algoritam, *Binary Relevance* algoritam i *Classifier Chains* algoritam za *Multilabel* klasifikaciju. Pomoću njih će se vršiti klasifikacija filmova na više žanrova. Dobijeni rezultati klasifikacije će se evaluirati i potom međusobno porediti.

Multilabel k Nearest Neighbours (ML-KNN) je *lazy learning* algoritam. Kao što mu ime implicira, ML-KNN je izveden iz popularnog *algoritma k Nearest Neighbours* (KNN) [2]. Na početku se identifikuju k najbližih suseda u trening skupu. Zatim, prema statističkim informacijama dobijenim iz skupova labela ovih susednih instanci se određuje skup labela za test instancu. Nakon transformacije podataka iz tekstualnog formata u numeričke vrednosti, korišćenjem neke od gore navedenih metoda za word embedding, prosleđuju se ML-KNN metodi u obliku *dense* matrice i vrši se predikcija.

Binary relevance [3], transformiše problem klasifikacije sa N labela u N *single-label* odvojenih problema binarne klasifikacije koristeći zadati binarni klasifikator. U ovom radu *Binary relevance* metodi prosleđen je SVC kao binarni klasifikator. *Binary relevance* metod zahteva da se ulazni podaci nalaze u obliku *sparse* matrice. Rečenice pretvorene u numeričke vrednosti pomoću gore navedenih *word embedding* metoda će se nalaziti u obliku *dense* matrice, zato je potrebno transformisati je u *sparse* oblik.

Porodica metoda poznata kao *Classifier chains* [4] postala je popularan pristup *Multilabel learning* problemima. Ovaj pristup uključuje povezivanje standardnih binarnih klasifikatora u lančanu strukturu, tako da predviđanja klase labela postaju obeležja za druge klasifikatore. *Classifier chains-u* je zadat SVC kao binarni klasifikator. Kao što je slučaj i kod *Binary relevance* metode i *Classifier chains* metoda zahteva dodatnu konverziju podataka u *sparse* oblik.

Micro-average preciznost je zbir svih pozitivnih rezultata predviđanja i deli se sa zbirom svih pozitivnih i negativnih rezultata predviđanja. U osnovi to je količnik broja tačno identifikovanih predviđanja sa ukupnim brojem predviđanja.

Macro-average precinznost predstavlja prosečnu preciznost sistema nad različitim skupovima.

Macro-average će izračunati metriku nezavisno za svaku klasu, a zatim će uzeti prosečnu vrednost i tako tretirati sve klase podjednako, dok *će Micro-average* agregirati doprinose svih klasa za izračunavanje prosečne metrike.

Cilj ovog rada je predikcija žanra filma na osnovu prosleđenih parametara. Poređeni su rezultati predikcija metoda *Multilabel k Nearest Neighbours*, *Classifier chains* i *Binary relevance* korišćenjem *Micro-average* i *Macro-average* metoda evaluacije. Nakon dobijenog finalnog skupa podataka urađen je *shuffle* podataka kako bi se obezbedilo da modeli ostanu opšti i da se izbegne *overfit*. Nakon toga nad njim je izvršeno čišćenje podataka. Popunjene su nedostajuće vrednosti, uklonjeni su razmaci, dijakritici i zagrade. Pored TF-IDF i *Doc2Vec*, za konverziju reči u numeričke vektore korišćene su i metode Label encoder i *Multilabel binarizer*.

U *Multilabel* klasifikaciji, *Micro-average* je poželjniji ukoliko postoji neuravnoteženost klasa, tj. ima mnogo više primera jedne klase nego drugih klasa. Na osnovu

postojanja disbalansa žanrova filmova u finalnom skupu podataka, zaključeno da je *Micro-average* bolji metod evaluacije u ovom slučaju.

3. OPIS SKUPA PODATAKA

Inicijalni skupovi podataka su preuzeti sa zvaničnog sajta "imdb.com." [5] U ovom radu se koriste tri skupa podataka, svaki skup podataka se nalazi u *tab-separated values* (tsv) formatu. Podaci su filtrirani na osnovu vrednosti polja titleType, gde su uzeti u obzir samo podaci čija je vrednost tog polja *'movie'*. Kako je atribut originalTitle napisan u izvornom jeziku i manje je zastupljen od primaryTitle-a odlučeno je da se ovaj atribut zanemari. Prethodno je navedeno da skup podataka sadrži samo filmove, pa atribut endYear postaje suvišan. Kako je cilj ovog rada predviđanje žanra još neobjavljenog filma atribut runtimeMinutes neće biti poznat, pa će iz tog razloga biti uklonjen iz finalnog skupa podataka.

Kako u finalnom skupu podataka već imamo osobe koje učestvuju u izradi filma, polja birthYear i deathYear ne utiču na predikciju žanra filma, pa su uklonjena. Atribut primaryProfession postaje suvišan iz razloga što se u atributu category već navodi zanimanje kojim se osoba bavila u izradi filma. KnownForTitles je polje koje navodi najpoznatija kinematografska dela u čijoj je izradi određena osoba učestvovala, a za predikciju žanra filma veću važnost ima broj ostvarenih uloga u filmovima određenog žanra što je već prikazano u skupu podataka, iz tog razloga atribut knownForTitles će biti uklonjen.

Nakon filtriranja, tri skupa podataka se spajaju u finalni skup podataka. Spajanje se vrši na osnovu atributa tconst, odnosno nconst, koji će nakon spajanja biti izbačeni jer predstavljaju jedinstvene identifikatore. Finalni skup podataka se sastoji od atributa:

- titleType – tip/format sadržaja (npr. film, serija, video...)
- primaryTitle – naslov sadržaja koji su tvorci koristili u promociji filma
- isAdult – da li je sadržaj namenjen za decu
- startYear – godina kada je sadržaj premijerno prikazan
- genres – skup do najviše 3 žanra filma
- category – kategorija posla osobe na filmu
- primaryName – ime osobe koja radi na filmu

Ciljno obeležje predstavlja atribut genres, gde se njegova vrednost predviđa na osnovu vrednosti ostalih podataka u skupu. Atribut genres predstavlja skup do najviše 3 žanra filma i može uzimati vrednosti iz skupa: *action*, *adventure*, *animation*, *biography*, *comedy*, *crime*, *documentary*, *drama*, *family*, *fantasy*, *film-noir*, *history*, *horror*, *musical*, *music*, *mystery*, *romance*, *sci-fi*, *sport*, *thriller*, *war*, *western*.

4. REZULTATI I DISKUSIJA

Prilikom kreiranja *Doc2Vec* modela optimizovani su sledeći parametri: 'vector_size' = 64, 'window' = 2, ' min_count' = 1, ' workers' = 8, 'epochs' = 20. Finalni skup podataka nakon transformacija je podeljen na trening i test skup u odnosu 80/20, pri čemu je iz test skupa izbačena vrednost ciljne labele.

Kao što je već navedeno u metodologiji, zbog disbalansa žanrova u finalnom skupu podataka zaključeno je da je Micro-average bolji metod evaluacije u ovom slučaju, a na osnovu rezultata iz tabele 1 optimizovana vrednost broja suseda u *Multilabel k Nearest Neighbours* metodi iznosi k = 7.

ML-KNN	Micro-average	Macro-average
k = 3	0.32	0.11
k = 4	0.33	0.13
k = 5	0.38	0.08
k = 6	0.37	0.06
k = 7	**0.42**	0.06
k = 8	0.39	0.10

Tabela 1. *Rezultati Mulitlabek k Nearest Neighbours metode*

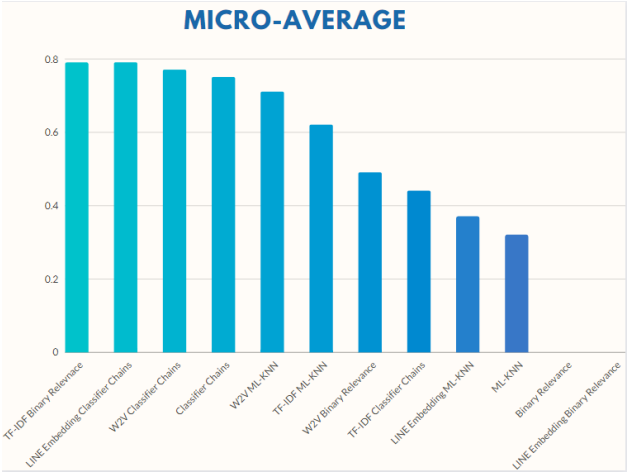
Optimizovane vrednosti za Label Network Embeddings su: 'weighted' = True, 'include_self_edges' = False, 'batch_size' = 1000, 'order' = 3, korišćen je LINE *embedding* metod ('LINE'), 'dimension' = 5, 'aggregation_function' = 'add', 'normalize_weights' = True, kao *regresor* je korišćen *RandomForestRegressor* sa parametrom 'n_estimators' = 10.

Rezultati dobijeni korišćenjem metodologija opisanih u poglavlju II prikazani su u tabeli 2.

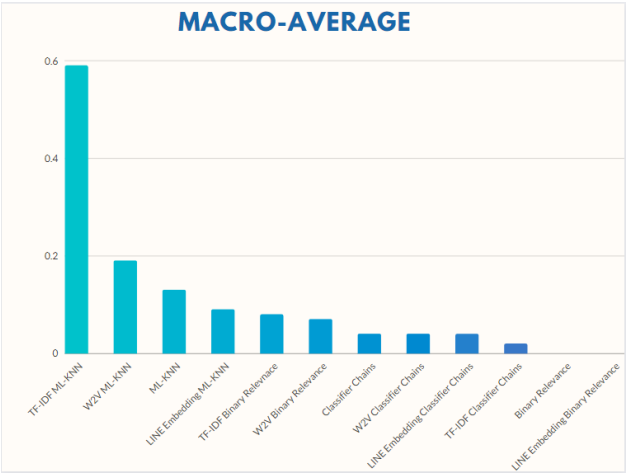
Classifiers	Micro-average	Macro-average
ML-KNN	0.32	0.13
Binary Relevance	0.00	0.00
Classifier chains	0.75	0.04
TF-IDF ML-KNN	0.62	**0.59**
TF-IDF Binary Relevance	**0.79**	0.08
TF-IDF Classifier chains	0.44	0.02
W2V ML-KNN	0.71	0.19
W2V Binary Relevance	0.49	0.07
W2V Classifier chains	0.77	0.04
LINE embedding ML-KNN	0.37	0.09
LINE embedding Binary Relevance	0.00	0.00
LINE embedding Classifier chains	0.79	0.04

Tabela 2. *Rezultati predikcije*

Na osnovu slike 1 se uočava da se *Classifier chains* metoda u proseku najbolje pokazala za *Micro-average* evaluaciju. Jedino odstupanje od očekivanog rezultata ostvarila je metoda *TF-IDF Binary Relevance* sa tačnošću od 0.79 i pokazala se kao najbolja metoda za TF-IDF *embedding*. Dok je korišćenjem drugih *word embedding* tehnika ostvarila najlošije rezultate, kao što je i očekivano jer je ovaj metod najjednostavniji ali ujedno i najbrži.



Slika 1. *Rezultati predikcije evaluirani Micro-average metodom*



Slika 2. *Rezultati predikcije evaluirani Macro-average metodom*

Sa slike 2 vidljivo je da se ML-KNN metoda najbolje pokazala za *Macro-average* evaluaciju i TF-IDF ML-KNN je ostvarila ubedljivo najbolju tačnost od 0.59. Metode predikcije daju znatno slabije rezultate evaluacijom pomoću *Macro-average* metode u odnosu na evaluaciju *Micro-average* metodom, što je i očekivano ponašanje zbog disbalansa žanrova u finalnom skupu podataka.

Kao što je već navedeno, u ovom radu najbolji rezultati za ML-KNN metodu su ostvareni kada je broj suseda k = 7. U skupu podataka su postojala ograničenja u vidu nedostajućih vrednosti za žanr i godinu premijernog prikazivanja, pa su torke sa tim nedostajućim vrednostima izbačene iz finalnog skupa podataka.

4. ZAKLJUČAK

Problem čijim se rešavanjem bavi ovaj rad je predviđanje žanrova filma na osnovu dostupnih podataka o filmu kao što su: osobe koje rade na filmu, godine premijernog prikazivanja filma, da li je film preporučljiv za decu, naslov filma. Motivacija za rešavanje ovog problema je sve veće interesovanje publike za buduće filmove i njihova želja da saznaju što više informacija o filmu kako bi mogli da znaju da li ti filmovi ulaze u njihovu sferu interesovanja. Često se desi da informacije o filmovima koji su u fazi planiranja ili izrade dospeju u javnost i od takvih informacija nastaju novinski članci o pojedinostima filma. Rešavanje ovog problema bi takođe olakšalo novinarima spekulacije o žanru filma.

U prethodnim sekcijama opisan je postupak vršenja predikcije žanra filma korišćenjem tehnika mašinskog učenja. Prvi korak ka rešavanju ovog problema bio je spajanje tri skupa podataka u finalni skup podataka. Finalni skup podataka je bilo potrebno ograničiti i očistiti od nepotrebnih i nevalidnih podataka. Zatim je izvršena transformacija tekstualnih podataka u numeričke vrednosti. Nakon toga su kreirani predikcioni modeli (*Multilabel k Nearest Neighbours*, *Classifier chains* i *Binary relevance*) i izvršena je njihova optimizacija. Poslednji korak u rešavanju ovog problema je bila evaluacija metoda korišćem Micro-average i *Macro-average* evaluacija. Kao najbolja rešenja pokazali su se *Binary Relevance* korišćenjem TF-IDF word embedding-a sa tačnošću od 0.79 i *Classifier chains* korišćenjem LINE embedding-a takođe sa tačnošću od 0.79.

Takođe ovaj rad prilikom rešavanja problema predikcije žanra filma koristi više *Multilabel* klasifikatora i poredi ih. Rezultati poređenja mogu biti analizirani, da bi se izdvojilo najbolje rešenje koje će biti iskorišćeno u budućim radovima koji se bave rešavanjem sličnog problema.

4. LITERATURA

[1] Piotr Szymański, Tomasz Kajdanowicz (2017) Multi-label embedding-based classification Available: http://scikit.ml/multilabelembeddings.html[datum pristupa 10.07.2024.]
[2] D.W. Aha, Special AI review issue on lazy learning, Artif. Intell. Review 11
[3] Zhang, Min-Ling, et al. "Binary relevance for multi-label learning: an overview." Frontiers of Computer Science 12.2 (2018): 191-202.
[4] Read, Jesse, et al. "Classifier chains for multi-label classification." Machine learning 85.3 (2011): 333-359.
[5] [Online]Available: https://www.imdb.com/interfaces/ [datum pristupa 10.07.2024.]

Kratka biografija:

Stefan Santrač rođen je u Novom Sadu 1998. god. Master rad na Fakultetu tehničkih nauka iz oblasti Računarstva i automatike – Primenjene računarske nauke i informatika odbranio je 2024.god.
kontakt: santrac.stefan@gmail.com

Зборник радова Факултета Техничких наука, Нови Сад

UDK: 4.8
DOI: https://doi.org/10.24867/30BE21Radojicic

ПРОЈЕКТОВАЊЕ АДАПТИВНИХ РЕГУЛАТОРА ПРИМЕНОМ ППО АЛГОРИТМА УПОТРЕБОМ ПРОГРАМСКОГ ПАКЕТА МАТЛАБ

DESING OF ADAPTIVE CONTROLLERS BY MEANS OF PPO ALGORITHM USING MATLAB

Вељко Радојичић, *Факултет техничких наука, Нови Сад*

Област – **ЕЛЕКТРОТЕХНИКА И РАЧУ-НАРСТВО**

Кратак садржај – *У овом раду истражена је могућност употребе учења потркепљењем, кон-кретно Proximal Policy Optimization алгоритма, за проблеме управљања континуалним динамич-ким системима. Изабрано је неколико репрезен-тативних, линеарних и нелинеарних система са континуалном динамиком, а за решење пробле-ма управљања, трениран је одговарајући агент, и то користећи MATLAB-ово софтверско окружење Reinforcement Learning Designer. За приказ резул-тата и симулацију, коришћен је Simulink.*

Кључне речи *PPO, политика, агент, регулатор*

Апстракт - *This paper focuses on exploring of using Reinforcement learning's Proximal Policy Optimization algorithm for problems of control of continual dynamic systems. Reinforcement learning agent has been trained on relatively simple linear and non-linear exaples of systems, using MATLAB's Reinforcemet Learning Designer application, while for result and simulation, Simulink has been used.*

Keywords: *PPO, policy, agent, controller*

1. УВОД

Proximal Policy Optimization (у наставку *PPO*), као један од новијих алгоритама учења потрекљењем (енг. *reinforcement learning*), представља једно уна-прeђење *Policy gradient* алгоритама, као што су *Natural Policy Gradient*, *REINFORCE*, *Trust Region Policy Optimizaion* итд. *PPO* се, између осталог, из-двоја и због своје *actor-critic* структуре, која пружа баланс између ескплорације и експоатације. Делат-ник (енгл. *actor*) јесте агент или регулатор (контро-лер). То је ентитет који на основу мерења одређу-је наредну акцију коју треба извршити. Оценитељ (енгл. *critic*) је ентитет који оцењује текуће поноша-ње система у затвореној спрези и даје информације на основу којих регулатор ажурира политику одлу-чивања. С обзиром да је та терминологија уобича-јена у литератури на енглеском језику, у наставку овог рада наставићемо да називамо овакву архитек-туру термином *actor-critic*, и тако ћемо је и наводи-ти, у оригиналу, на енглеском језику, латиницом. Овакву структуру карактерише специфичан израз за апроксимацију градијента критеријума оптимал-ности

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{\substack{i=1..N \\ t=0..T}} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \mathrm{adv}_{\pi,\phi,i,t}\,, \quad (1)$$

где смо "унапређење" (енгл. *advantage*), односно разлику очекиване добити након једне акције ($r(s_{i,t}, a_{i,t}) + \gamma\hat{v}_{\pi,\phi}(s_{i,t+1})$) и полазне очекиване до-бити ($\hat{v}_{\pi,\phi}(i, s_t)$) означили са

$$A_{\pi,\phi,i,t} = r(s_{i,t}, a_{i,t}) + \gamma\hat{v}_{\pi,\phi}(s_{i,t+1}) - \hat{v}_{\pi,\phi}(i, s_t)\,, \quad (2)$$

где су са θ и ϕ означени параметри одговарајућих неуронских мрежа делатника и оценитеља. Увође-њем новог, ограниченог критеријума оптималности, спречавају се нагле ажурирања полике, што може имати дестабилишући ефекат у процесу учења [2]. Напомињемо да се унапређење може посматрати као разлика апостериорне и априорне естимације добити. Политике се код овог алгоритма ажурира-ју на следећи начин

$$\Delta\theta^* = \arg\max_\theta \underset{s,a \sim \pi_{\theta_k}}{\mathrm{E}} [L(s, a, \theta_k, \theta)] \quad (3)$$

Критеријум оптималности дат је у облику

$$L(s, a, \theta_k, \theta) = \min\left(\rho_\theta(a|s) A_{\pi_{\theta_k}}(s, a),\ g(\epsilon, A_{\pi_{\theta_k}}(s, a)) \right), \quad (4)$$

при чему је

$$\rho_\theta(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\theta+\Delta\theta}(a|s)}\,. \quad (5)$$

У претходним једначинама, са је означена функ-ција унапређења (*advantage*-функција), док је g де-финисано са

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A\,, & A \geq 0 \\ (1-\epsilon)A\,, & A < 0 \end{cases}. \quad (6)$$

Параметар ϵ стоји као ограничавајући параметар за ажурирање политике. У случају да је A-функција

НАПОМЕНА: Овај рад је проистекао из мастер рада чији је ментор др Милан Рапајић, редовни професор.

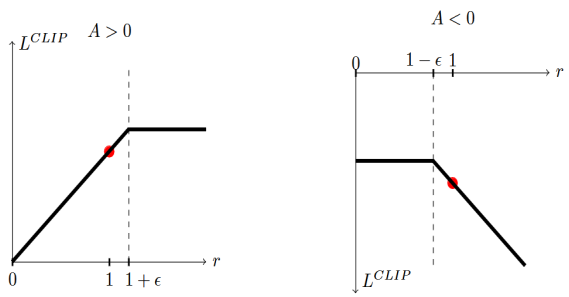
позитивна, критеријум оптималности своди се на облик

$$\mathcal{L}(s,a,\theta_k,\theta) = \min\left(\rho_\theta(a|s),(1+\epsilon)\right)A_{\pi_{\theta_k}}(s,a)\ , \tag{7}$$

и тада критеријум оптималности расте уколико $\pi_\theta(s|a)$ расте. Ипак због оператора min, онда када промена политике $\rho_\theta(s|a)$ достигне вредност $1+\epsilon$, $\mathcal{L}$ престаје да расте. Другим речима, ограничено је колико наредна политика може да одступа од претходне. У случају да A-функција има негативну вредност, критеријум оптималности постаје

$$\mathcal{L}(s,a,\theta_k,\theta) = \max\left(\rho_\theta(a|s),(1-\epsilon)\right)A_{\pi_{\theta_k}}(s,a)\ . \tag{8}$$

Критеријум оптималлности расте уколико и $\pi_\theta(s|a)$ опада, тј. уколико акција постаје мање вероватна. Оператор max спречава промену $\mathcal{L}$ након $1-\epsilon$, тако да промена политике такође ограничена. За фактор ϵ, емпиријски је показано да се вредности блиске $\epsilon = 0.2$ дају најбоље резултате [4].



Слика 1: *Зависност критеријума оптималности од односа вероватноћа текуће и и наредне политике. Слика преузета са [5]*

2. ОПИС ПРОБЛЕМА

Истраживање могућности примене *PPO* алгоритма на проблеме управљана динамичким системима изискује потребу за адекватним софтверским окружењем, што због лакше имплементације и тренирања самог алгоритма, што због могућности симулације. *Simulink*, симулационо окружење програмског пакета *MATLAB*, заједно са интегрисаном *Reinforcement Learning Designer* апликацијом, пружа неопходне алате за овакав подухват. Конкретно, ограничили смо се на неколико репрезентативних динамичких система, а као циљ је постављено пројектовање *RL*-агента који би у класичној повратној спрези заменио неки од конвенционалних регулатора (нпр. ПИД регулатор), као и анализу добијених резултата.

2.1. Избор агента

Апликација омогућава олакшано прављење агента. У случају да корисник не жели да га имплементира кроз MATLAB, и прихвата подразумевану топологију неуронских мрежа за actor-a и critic-a, агент се прави секвенцијалним бирањем опција и хиперпараметара. Објекат агента такође се може направити коришћењем методе `rlPPOAgent`. Предност овакве имплементације је у томе што се агент може формирати независно од окружења, док је предуслов за прављење агента кроз апликацију постојање одговрајућег, претходно направљеног *enviroment*-а. Агент је у *simulink*-у представљен блоком *RL Agent*. Као улазни сигнали блока, дефинисани су вектор обсервација, сигнал награде и сигнал који означава да је излаазна величина изван опсега. На излазу блока је сигнал акције, управљачки сигнал.

2.2. Избор окружења

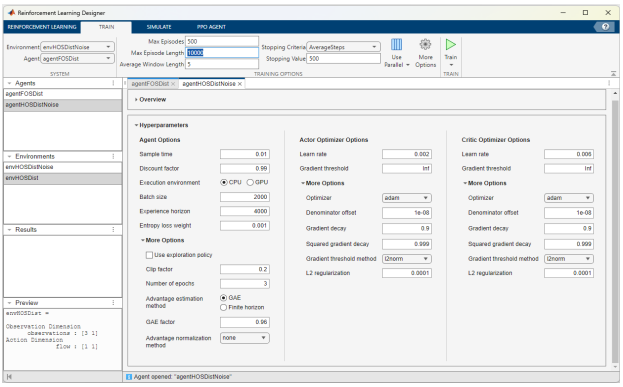
Корисник може одабрати неки од постојећих *enviroment*-а, а помоћу опције *Import*, уводи се тзв. *custom* окружење. За потребе овог рада, адекватно окружења имплеметирана је кроз *Simulink*, а потом, употребом метода *rlSimulinkEnv* и *createIntegratedEnv*. Излазни сигнал ограничен је на вредности из опсега $(-1000, 1000)$, а вектор обсервација сачињен је од сигнала грешке (разлика између сигнала референце и излазног сигнала) и интеграла сигнала грешке. Функција награде формирана је тако да строго "кажњава" излазак из опсега, велике вредности сигнала грешке и нагле промене управљачког сигнала. Рачунање награде врши се по формули

$$r = -e^2 - (u-u_p)^2\ , \tag{9}$$

где је r сигнал награде, e сигнал грешке, а u и u_p управљачки сигнали у текућем и претходном тренутку.

2.3. Обука

Тренирање агента у одговарајућем окружењу извршено је кроз *Reinforcement Learning Designer*, одабиром опције *Train* и подешавањем хиперпараметара.



Слика 2: *Тренирање агента*

Тренинг се може посматрати и кроз апликацију, где су приказане награде и вредности стања и усређене вредности стања, или кроз *Simulink*, где је

приказано понашање агента (регултора) током епизода.

2.4. Симулација

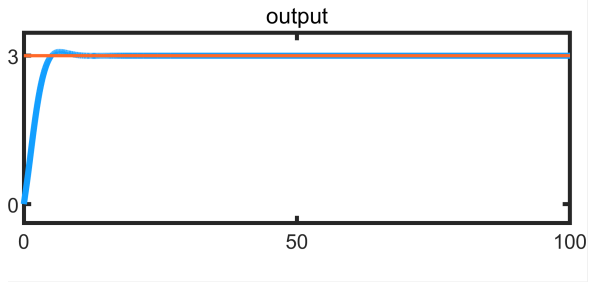
Симулирање агента (регулатора) такође се може извршити кроз апликацију, где је омогућено и чување добијених резултата, али без директног увида у понашање регулатора. Из тог разлога симулација је извршена је у оригиналном *Simulink* моделу. Име објекат агента се тада прослеђује уграђеном блоку *RL Agent* који се налази у *Simulink* моделу.

3. РЕЗУЛТАТИ

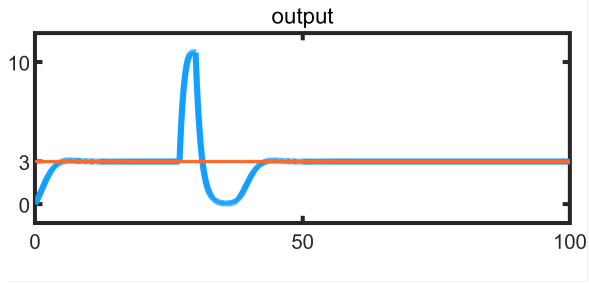
Агент (регулатор) намењен је за управљање у затвореној повратној спрези. Системи од интереса су систем првог реда описан функцијом преноса $\frac{1}{s+1}$, систем другог реда описаном функцијом преноса $\frac{1}{(s+1)^2}$, а за репрезент нелиенарног систем одабран је модел елктричног кола са потрошачем константне снаге.

3.1. Систем првог реда

Тренирање *PPO RL*-агента за почетак је извршено за управљање системом првог реда који је описан функцијом преноса $W(S) = \frac{1}{s+1}$. За референтни сигнал у току тренирања, одабрана је секвенца Хевисајдових сигнала променљиве вредности. Такође, агент је у трениран са присуством степ-поремећаја амплитуде 10 у трајању 3% времена симулације од 100 секунди, као и у случају када поремећај не постоји.



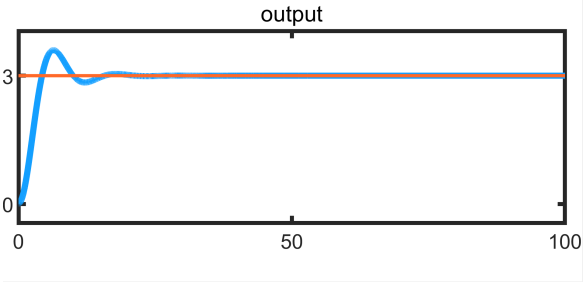
Слика 3: *Степ одзив система првог реда без присуства поремећаја*



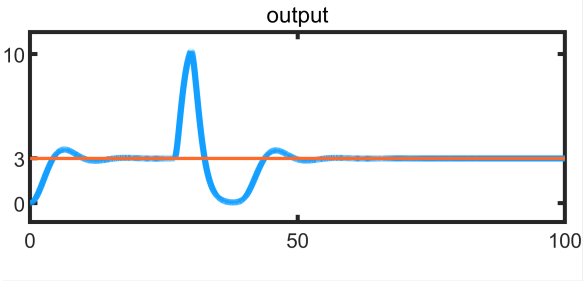
Слика 4: *Степ одзив система првог реда са присуством поремећаја*

3.2. Систем другог реда

Као репрезентативан динамички систем вишег реда, одабран је систем описан функцијом преноса $W(s) = \frac{1}{(s+1)^2}$. И у овом случају, агент је тренирам тако да је референца степ-сигнал променљиве висине. Поново, изложена су два засебна резултата симулације, један без присуства поремећаја, а други у присуству поремећаја истог типа као и случају система првог реда.



Слика 5: *Степ одзив система другог реда без присуства поремећаја*



Слика 6: *Степ одзив система другог реда са присуством поремећаја*

3.3. Нелинеарни систем

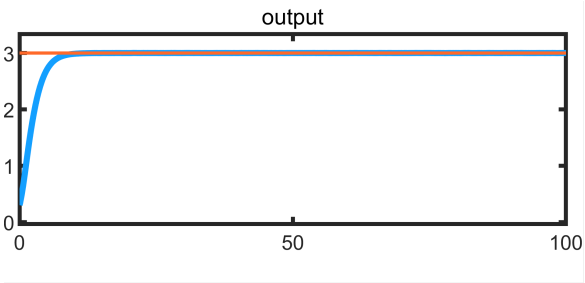
За пример нелинеарног система, узет је математички модел електричког кола са потрошачем константне снаге, где су са v_c и i_c означени напон на кондензатору и струја потрошача, респективно. Математички модел да је у облику

$$\begin{aligned} \dot{v}_c &= \frac{1}{C_f R_f}\left(u - v_c - R_f i_p\right) \\ \dot{i}_p &= \frac{1}{L_v}\left(v_c - R_v i_p - \frac{P}{i_p}\right) \\ y &= v_c \end{aligned} \tag{10}$$

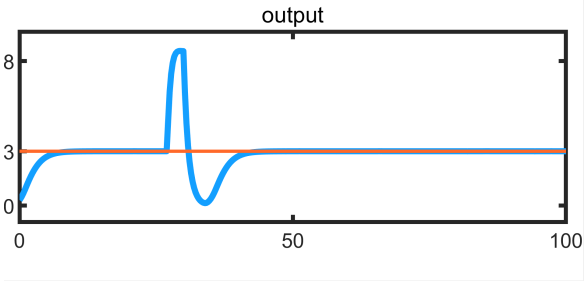
Имајући у виду да је акценат рада на понашању алгоритма, а не на самим моделима, али и због једноставности симулације и нумеричких ограничења, за све параметре кола узете су јединичне вредности, па се модел своди на

$$\begin{aligned} \dot{x}_1 &= x_1 - x_2 + u \\ \dot{x}_2 &= x_1 - x_2 - \frac{1}{x_2} \\ y &= x_1 \end{aligned} \tag{11}$$

Узлазним сигналом сматра се сигнал напонског генератора u, а излазним сингалом напон на кондензатору v_c, односно стање x_1 Почетне вредности стања су $x_1 = 0.3$ и $x_2 = 0.3$.



Слика 7: *Степ одзив нелинеарног система без присуства поремећаја*



Слика 8: *Степ одзив нелинеарног система са присуством поремећаја*

4. ЗАКЉУЧАК

У овом раду, приказана је употреба *Proximal Policy Optimization* алгоритма на проблеме управљања континуалним динамичким системима, на неколико репрезентативних примера. *RL*-агент, који у оваквој управљачкој шеми мења неки од стандардних регулатора (нпр. ПИД регулатор), показао се као могуће решење, макар на приказаним моделима. Перформансе овог и конценционалних регулатора, у различитим случајевима не разликују се значајно, барем не у покривеним случајевима. Ипак, треба имати у виду да понашање агената који су засновани на алгоритмима са овако комплексном структуром итекако зависи од подешавања хиперпараметара, што такође представља изазов, и може се сматрати засебним проблемом.

Литература

[1] Richard S. Sutton, Andrew G. Barto "Reinforcement Learning: An Introduction, Second edition", Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 2014.-2015.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, "Proximal Policy Optimization Algorithms", `https://arxiv.org/abs/1707.06347`, 2017.

[3] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, Pieter Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation", ICLR, 2016.

[4] Nai-Chieh Huang, Ping-Chun Hsieh, Kuo-Hao Ho, I-Chen Wu "PPO-Clip Attains Global Optimality: Towards Deeper Understandings of Clippin", Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan, `https://arxiv.org/abs/2312.12065`, 2024.

[5] Wouter van Heeswijk, "Policy Gradients In Reinforcement Learning Explained" Medium, 2022.

Кратка биографија:

Вељко Радојичић рођен је у Новом Саду 2000. год. Дипломски рад на Факултету техничких наука из области Електротехнике и рачунарства на тему "Пројектовање струјне и напонске петље са спрежним филтром LCL тип" одбранио је у јулу 2023. год.
Контакт: radojicic.veljko@uns.ac.rs

Зборник радова Факултета техничких наука, Нови Сад

UDK: 4
DOI: https://doi.org/10.24867/30BE28Buljevic

CLOUD ОРИЈЕНТИСАНО РЕШЕЊЕ ЗА РЕЗЕРВАЦИЈУ СМЕШТАЈА УПОТРЕБОМ AWS ПЛАТФОРМЕ

A CLOUD-ORIENTED ACCOMMODATION BOOKING SOLUTION USING AWS PLATFORM

Александар Буљевић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Рад се бави развојем апликације за резервацију смештаја употребом AWS Cloud платформе и serverless архитектуре. У раду су описани сви коришћени AWS сервиси као и сама архитектура и дизајн решења. Један од задатака рада је пролазак и анализа једног примера апликације за резервације смештаја, и пролазак кроз комплетан развој апликације, од дизајна, до саме објаве веб апликације. Описане су и све технологије и алати који су коришћени у изради решења.*

Кључне речи: *AWS, cloud, резервације, смештај, AWS CDK*

Abstract – *The paper deals with the development of an accommodation booking application using the AWS Cloud platform and serverless architecture. The paper describes all the AWS services used as well as the architecture and design of the solution. One of the tasks of the work is to go through and analyze an example of an accommodation booking application, covering the complete development of the application, from design to the actual release of the web application. All the technologies and tools used in the creation of the solution are also described.*

Keywords: *AWS, cloud, serverless, reservations, accommodation, AWS CDK*

1. УВОД

У данашњем дигиталном добу, технолошка иновација непрекидно обликује начин на који обављамо свакодневне активности. Индустрија туризма и угоститељства није изузетак.

Са све већим бројем путника који користе Интернет за тражење смештаја, поставља се потреба за развојем ефикасних, скалабилних и поузданих система за резервацију смештаја. Путници данас очекују брзо, једноставно и интуитивно искуство приликом резервације смештаја, са могућношћу прегледа и упоређивања различитих опција, прилагођених њиховим потребама и преференцијама.

У овом контексту, cloud технологија постаје кључан играч. Cloud технологија омогућава приступ рачунарским ресурсима и услугама путем Интернета, без потребе за физичким присуством на локацији. Ово пружа не само флексибилност и скалабилност у управљању инфраструктуром, већ и омогућава иновације и агилност у развоју софтвера.

2. ЗАХТЕВИ

Дефинисање функционалних и нефункционалних захтева пре почетка имплементације пројекта је есенцијална јер омогућава јасно разумевање шта систем треба да ради (функционални захтеви) и како треба да ради (нефункционални захтеви) [1]. Ово осигурава да развојни тим има прецизне смернице за дизајн и имплементацију, смањујући ризик од грешака и неспоразума.

2.1. Функционални захтеви

Одабрани функционални захтеви су резервације смештаја, приказ свих смештаја, приказ карактеристика ресторана и кампа, страница галерије, контакт страница и подршка за више језика.

2.2. Нефункционални захтеви

Одабрани нефункционални захтеви су употребљивост, скалабилност, расположивост и безбедност.

3. AWS CLOUD ПЛАТФОРМА И ЊЕНИ СЕРВИСИ

3.1. Увод у Cloud

Cloud, или „облак“, представља модел за омогућавање приступа рачунарским ресурсима путем Интернета, пружајући корисницима флексибилност, скалабилност и ефикасност без обзира на њихову физичку локацију. Ова парадигма мења начин на који се IT инфраструктура конфигурише, управља и користи [2].

Класични модели пословања захтевају изградњу, одржавање и надоградњу физичке инфраструктуре, што може бити скупо, сложено и ограничавајуће. Cloud технологија трансформише овај приступ, омогућавајући компанијама да изнајмљују ресурсе према потреби, уместо да их купују и одржавају. Овај модел плаћања према потрошњи, познат и као "pay-as-you-go" или "pay-per-use", омогућава компанијама да ефикасно користе своје ресурсе и избегну трошкове неискоришћених капацитета [3].

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

Поред финансијских бенефита, cloud технологија доноси и оперативне предности. Флексибилност и скалабилност омогућавају компанијама да брзо реагују на промене у потражњи и пословним захтевима, скалирајући своје ресурсе према потреби.

С друге стране, прелазак на cloud технологију може изазвати одређене изазове и препреке. На пример, компаније морају пажљиво размотрити питања приватности и усклађености са регулативама када је у питању чување и обрада осетљивих података у облаку [4].

У сваком случају, cloud технологија представља неизбежан корак ка дигиталној трансформацији и кључну компоненту за модернизацију пословања у данашњем дигиталном добу.

3.2. Amazon Web Services и коришћени сервиси

3.2.1. Amazon Web Services

AWS је водећа платформа за пружање услуге рачунарства у облацима, која нуди широк спектар услуга и решења које омогућавају организацијама да буду агилније, иновативније и исплативије. AWS је постао синоним за cloud инфраструктуру због своје свеобухватне понуде која укључује рачунарске ресурсе, складиштење података, мрежне услуге, базе података, аналитичке алате, алате за развој апликација и управљање, као и бројне друге услуге које омогућавају развој и имплементацију софистицираних апликација и услуга [5].

3.2.2. AWS Lambda

AWS Lambda је serverless сервис који омогућава извршавање кода без потребе за управљањем серверима [6]. Serverless рачунарство представља модел рачунарства у облаку у коме провајдер алоцира ресурсе динамички, на захтев, водећи бригу о серверима уместо корисника.

Коришћење Lambda функција ослобађа програмере од потребе да размишљају о инфраструктури и омогућава им да се фокусирају на развој функционалности. Lambda функције се извршавају само када су активиране, аутоматски се скалирају у складу са захтевима и наплаћују се само за време извршавања кода.

Кроз овај сервис, програмери могу да пишу код користећи широк спектар подржаних програмских језика, укључујући Node.js, Python, Java, C# и Go. Lambda функције могу бити повезане са различитим AWS сервисима, што омогућава аутоматизацију и интеграцију са остатком архитектуре система.

3.2.3. DynamoDB

DynamoDB је потпуно управљан (енгл. fully managed) сервис NoSQL базе података који омогућава брзо и еластично чување и извођење података. Овај сервис је изузетно погодан за апликације које захтевају високе перформансе и скалирање, као и за апликације које раде са великим обимом података [7].

Она нуди флексибилност у дизајнирању шеме података и могућност управљања различитим типовима података.

Једна од највећих предности DynamoDB-а је његова скалабилност. Такође, DynamoDB обезбеђује репликацију података у више AWS региона, што обезбеђује сигурност и доступност података у случају катастрофе или проблема.

3.2.4. CloudFront

CloudFront је управљани сервис за доставу садржаја (CDN - Content Delivery Network) који обезбеђује брзо и сигурно достављање веб садржаја крајњим корисницима широм света. Овај сервис игра кључну улогу у убрзавању учитавања веб страна, смањењу кашњења и побољшању корисничког искуства [8].

CloudFront функционише као распрострањена мрежа сервера који се налазе на различитим локацијама широм света. Ови сервери, познати као Edge локације, сачувани су на стратешким тачкама близу крајњих корисника и сачувају копије веб садржаја. Када корисник захтева одређени садржај, CloudFront аутоматски преноси тај захтев до најближе Edge локације и пружа садржај са најбољим перформансама.

3.2.5. Simple Storage Service (S3)

Amazon S3 је сервис за чување података који обезбеђује издржљиво, скалабилно и сигурно чување података у облаку. Овај сервис пружа флексибилност и доступност за различите потребе [9].

Amazon S3 омогућава креирање и управљање бакетима (енгл. bucket) за чување података. Бакети представљају контејнере у којима се чувају објекти, као што су слике, текстуални фајлови, видео записи и други подаци.

Једна од највећих предности Amazon S3 је његова издржљивост и доступност. Овај сервис реплицира податке у више AWS региона и обезбеђује 99.999999999% доступности података.

3.2.6. Simple Email Service (SES)

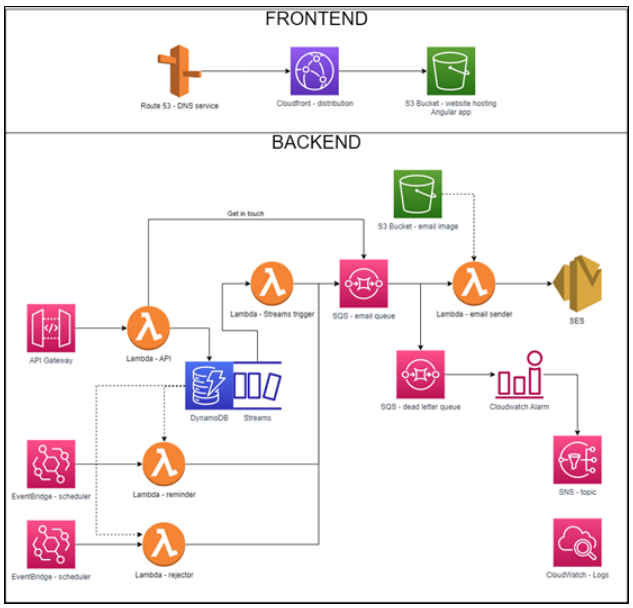
Amazon SES је сервис за слање е-поште који омогућава стабилну и скалабилну доставу електронских порука. Овај сервис игра кључну улогу у комуникацији са корисницима путем е-поште и обезбеђује сигурно и ефикасно слање порука [10].

Amazon SES омогућава креирање и слање различитих типова е-порука, укључујући текстуалне поруке, HTML поруке и поруке са прилозима. Сервис обезбеђује механизме за аутентификацију и шифровање порука, што осигурава доставу порука на сигуран начин.

4. АРХИТЕКТУРА СИСТЕМА И ДИЗАЈН

4.1. Приказ архитектуре

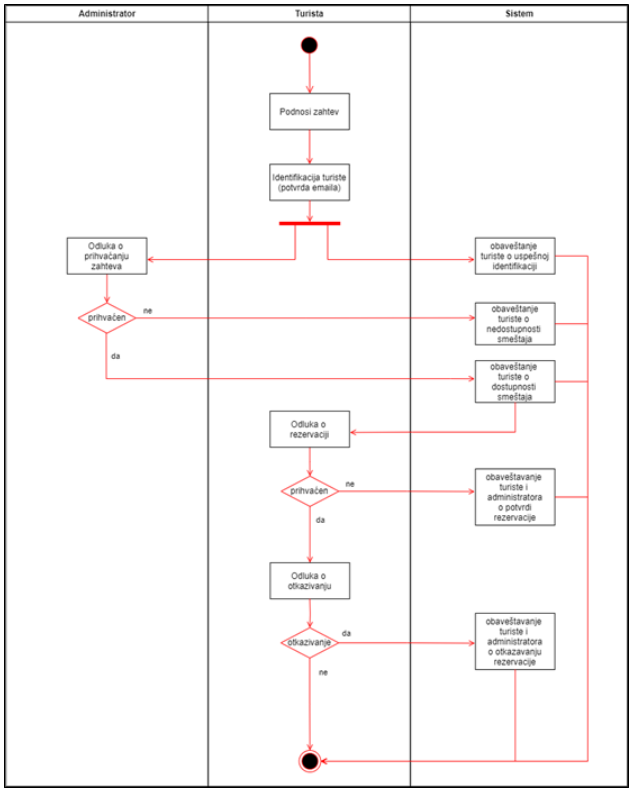
На слици 4.1. се може уочити комплетна архитектура решења и она је детаљније описана у наставку.



Слика 4.1. Архитектура решења.

4.2. Дијаграм активности

Од тренутка подношења захтева до његове крајње обраде, неопходна је вишеструка интеракција како корисника тако и администратора са софтверским решењем. Ток електронске поште, тј. интеракције корисник-администратор је могуће видети на слици 4.2.



Слика 4.2. Дијаграм активности.

4.3. Дизајн решења

У изради дизајна сајта, коришћен је Figma, популаран алат за дизајн корисничког интерфејса. Циљ дизајна је био креирање интуитивног, визуелно привлачног и респонзивног сајта који ефикасно преноси информације и побољшава корисничко искуство.

Figma је моћан и широко коришћен алат за дизајн корисничког интерфејса који омогућава дизајнерима да креирају и уређују дизајн у облаку [11].

5. ИМПЛЕМЕНТАЦИЈА РЕШЕЊА

5.1. Имплементација бекенда

Серверска страна апликације је написана у Java програмском језику, користећи Spring Boot радни оквир (енгл. framework).

Представља више-модуларни пројекат који се састоји од неколико модула. Сваки модул има специфичну улогу и доприноси свеукупној функционалности система.

5.1.1. Spring Boot

Spring Boot је популаран радни оквир за развој апликација у програмском језику Јава. Он је део већег Spring екосистема и осмишљен је са циљем да поједностави и убрза процес развоја апликација, укључујући веб апликације, микро услуге и пословне апликације. [12].

Предности коришћења су брзо постављање и развој, микро услуге, смањење boilerplate кода, активни екосистем и заједница, функције спремне за продукцију.

5.2. Имплементација фронтенда

У овој секцији ће бити детаљно описана имплементација клијентске стране апликације, која је развијена коришћењем JavaScript програмског језика и Angular радног оквира.

5.2.1. Angular

Angular је модеран, клијентски радни оквир за развој веб апликација који је развио и одржава Google [13]. Он пружа многе функционалности и алате који олакшавају развој сложених и интерактивних веб апликација. Кључне карактеристике Angular-a су компонентна архитектура, двострана веза (енгл. Two-way data binding), рутирање и сервиси и зависности.

5.3. Имплементација инфраструктуре

У овој секцији ће бити описана имплементација инфраструктуре апликације, укључујући испоручивање кода на AWS Cloud и спецификацију инфраструктуре коришћењем AWS CDK (Cloud Development Kit).

5.3.1. AWS CDK

AWS Cloud Development Kit (CDK) је радни оквир отвореног кода (енгл. Open-source) који омогућава програмерима да дефинишу cloud инфраструктуру користећи познате програмске језике. AWS CDK трансформише ове дефиниције у AWS CloudFormation шаблоне, што олакшава управљање и имплементацију инфраструктуре као кода (IaC - Infrastructure as Code) [14]. Кључне карактеристике су

да подржава разне познате програмске језике, конструкције (Constructs) различитих новоа, повратне петље, екстензије и библиотеке, CDK CLI: Командна линија (CLI).

5.3.2. Пример коришћења и примена

За бекенд, коришћен је CDK који је везан за Java програмски језик, док је за фронтенд коришћена верзија написана у TypeScript језику. Листинг 5.6. приказује како је коришћен CDK да би био генерисан ред за обраду електронске поште:

```
private Queue generateEmailQueue(DeadLetterQueue
deadLetterQueueConfiguration, String emailQueueName) {
    return Queue.Builder
            .create(this, emailQueueName)
            .queueName(emailQueueName)
            .deadLetterQueue(deadLetterQueueConfiguration)
            .visibilityTimeout(Duration.seconds(30))
            .build();
}
```

Листинг 5.6. Функција за прављење реда за електронску пошту користећи AWS CDK.

6. ЗАКЉУЧАК

Овај рад је демонстрирао како употреба AWS cloud платформе може значајно унапредити систем за резервацију смештаја кроз примену serverless архитектуре. Коришћењем различитих AWS сервиса као што су AWS Lambda, API Gateway, DynamoDB, и други, постигнуте су висока скалабилност, поузданост и ефикасност система. Примена концепта инфраструктура-као-код (енгл. Infrastructure as Code) путем AWS Cloud Development Kit-a, омогућила је аутоматизовано управљање и конфигурацију инфраструктурних ресурса.

Овај приступ доприноси брзој и лакој имплементацији промена, што је од кључног значаја за модерне софтверске системе који се морају брзо прилагођавати променљивим условима на тржишту.
Модел плаћања по потрошњи омогућава флексибилно управљање буџетом и ресурсима, чиме се смањују оперативни трошкови и омогућавају иновације.
Коришћењем најновијих технологија и пракси у облаку, овај приступ представља важан корак ка модернизацији и унапређењу услуга у индустрији туризма и угоститељства.

7. ЛИТЕРАТУРА

[1] „Functional and Nonfunctional Requirements: Specification and Types,“ [На мрежи]. Available: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/.

[2] „What Is Cloud Computing? Definition, Benefits, Types, and Trends,“ [На мрежи]. Available: https://www.spiceworks.com/tech/cloud/articles/what-is-cloud-computing/.

[3] „What is pay-as-you-go Cloud Computing (PAYG)?,“ [На мрежи]. Available: https://www.digitalocean.com/resources/article/pay-as-you-go-cloud-computing.

[4] „Pros and Cons of Cloud Storage,“ [На мрежи]. Available: https://www.securestorageservices.co.uk/article/11/pros-and-cons-of-cloud-storage.

[5] „Overview of Amazon Web Services,“ [На мрежи]. Available: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html.

[6] „What is AWS Lambda?,“ [На мрежи]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

[7] „What is Amazon DynamoDB?,“ [На мрежи]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

[8] „What is Amazon CloudFront?,“ [На мрежи]. Available: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html.

[9] „What is Amazon S3?,“ [На мрежи]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

[10] „What is Amazon SES?,“ [На мрежи]. Available: https://docs.aws.amazon.com/ses/latest/dg/Welcome.html.

[11] „What is Figma?,“ [На мрежи]. Available: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma.

[12] „What is Java Spring Boot?,“ [На мрежи]. Available: https://www.ibm.com/topics/java-spring-boot.

[13] „What is Angular?,“ [На мрежи]. Available: https://angular.dev/overview.

[14] „What is the AWS CDK?,“ [На мрежи]. Available: https://docs.aws.amazon.com/cdk/v2/guide/home.html.

Кратка биографија:

Александар Буљевић је рођен 15. октобра 1999. године у Новом Саду. У школској 2018/19. уписује студијски програм Софтверско инжењерство и информационе технологије на Факултету техничких наука у Новом Саду. Након завршених основних студија, 2022. године, уписује мастер студије на истом студијском програму, смер Електронско пословање.

контакт:
abuljevic8@gmail.com

UDK: 4
DOI: https://doi.org/10.24867/30BE28Buljevic

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/30BE29Jevtic

GLASOVNO I TEKSTUALNO UPRAVLJANJE INFORMACIONIM SISTEMIMA ZASNOVANO NA KORIŠĆENJU VELIKIH JEZIČKIH MODELA VEŠTAČKE INTELIGENCIJE

USING AI LARGE LANGUAGE MODELS FOR VOICE AND TEXT BASED MANAGEMENT OF INFORMATION SYSTEMS

Jovana Jevtić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO I INFORMACIONE TEHNOLOGIJE

Kratak sadržaj - *U radu su predstavljeni interfejsi prirodnog jezika (eng. Natural Language Interfaces) kao i prednosti koje se dobijaju njihovim korišćenjem. Opisana je obrada prirodnog jezika i veliki jezički modeli koji se mogu iskoristiti za realizovanje informacionog sistema sa glasovnim i tekstualnim upravljanjem. Takođe, prikazana je softverska arhitektura jednog takvog sistema i data studija aplikacije za upravljanje restoranom kao primer.*

Ključne reči: *interfejsi prirodnog jezika, obrada prirodnog jezika, veliki jezički modeli*

Abstract – *The paper presents natural language interfaces and the advantages gained from their use. It describes natural language processing and large language models that can be utilized for implementing systems with voice and text control. Additionally, it shows the software architecture of such a system and provides a case study of a restaurant management application as an example*

Keywords: *Natural Language Interfaces, Natural Language Processing, Large Language Models*

1. UVOD

Informacioni sistemi predstavljaju skup softvera, hardvera, podataka, ljudi i procedura koji su objedinjeni tako da generišu informacije koje omogućavaju aktivnosti neke organizacije. Pod tim imenom počeli su da se razvijaju sredinom 20. veka i od tada je postojalo više različitih načina za njihovu upotrebu. Prva era je era komandne linije kada su se računari koristili tako što su korisnici unosili tekstualne komande. Nakon toga, da bi se postigla lakoća korišćenja počinju da se razvijaju grafički korisnički interfejsi. Nakon komandne linije prešlo se na korišćenje desktop aplikacija koje su obično bile specifične za operativne sisteme. Sa razvojem interneta, postaju popularne veb aplikacije za čiju upotrebu su bili neophodni veb pretraživači. Kako su veb aplikacije na početku bile spore, pojavljuju su Client Side Rendering i Single Page Applications, kao njihova optimizacija. Pored veb aplikacija danas su u velikoj upotrebi i mobilne aplikacije. Ipak, jedna stvar se može izdvojiti kao zajednička za prethodno opisane vrste aplikacija, a to je da korisnik interaguje sa njima preko dobro struktuiranih formi koje mora da popuni na specifičan način da bi dobio željeni odgovor.

Obrada prirodnog jezika [1] je grana veštačke inteligencije kojoj je cilj da omogući da računari mogu da razumeju, obrade i generišu prirodni jezik, dok veliki jezički modeli [2] predstavljaju njene napredne modele koji su trenirani nad velikim količinama podataka. Razvoj ove grane veštačke inteligencije i njenih modela prethodnih godina, daju nam mogućnost da razvijamo sistem sa glasovnim i tekstualnim upravljanjem. Korisnik bi prirodnim jezikom mogao da iskaže šta želi da dobije od sistema. Takav način upotrebe bi doneo veću efikasnost, brže upravljanje i smanjeno vreme obuke. Upravo to daje nam motivaciju za razvijanje informacionog sistema sa glasovnim i tekstualnim upravljanjem koji je opisan u ovom radu.

2. STANJE U OBLASTI

Interfejsi prirodnog jezika [3] su tehnologije koje omogućavaju korisnicima da komuniciraju sa računarom koristeći prirodni jezik, kako u pisanoj, tako i u govornoj formi. Njihov cilj je da pojednostave komunikaciju između ljudi i računara, čineći je priodnom i intuitivnom. Razvoj ovakvih interfejsa uveliko zavisi od obrade prirodnog jezika čiji će osnovni koncepti i relevantni modeli biti opisani u nastavku poglavlja.

Obrada prirodnog jezika (eng. Natural Language Processing – NLP) je grana veštačke inteligencije koja se odnosi na računarsku obradu jezika ljudi. Tu se ubrajaju algoritmi kojima je ulaz tekst ili govor nastao od strane čoveka, ali i oni kojima je cilj da proizvedu takav izlaz. NLP ima mnoštvo zadataka koje rešava, a neki od njih su: prepoznavanje govora, pretvaranje teksta u govor, prevođenje, izdvajanje informacija, pretraga informacija, ogovaranje na pitanja, tekstualna klasifikacija. Iako NLP uspešno rešava pomenute zadatke, postoji mnoštvo izazova koji su morali da se prevaziđu, ali i onih koji se i dalje prevazilaze. Neki od uobičajenih izazova su kontekstualne reči i fraze, sinonimi, homofoni, homonimi, različite gramatičke strukture, sarkazam i ironija, ali i sve ostale dvosmislenosti, nejasnoće i ne tako dobro definisani delovi ljudskog jezika.

Na početku, prirodni jezici obrađivani su pomoću sistema koji su zasnovani na semantičkoj analizi i rasčlanjivanju. Oni su doveli do određenog uspeha, ali su bili ograničeni velikom složenošću jezičkih pojava. Zbog toga su počeli

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila prof. dr Gordana Milosavljević.

da se koriste modeli koji se zasnivaju na mašinskom učenju kao što su konvolucione i rekurentne neuronske mreže, da bi se i oni prevazišli velikim jezičkim modelima zasnovanim na arhitekturi transformatora. Njihova arhitektura biće opisana u sledećem potpoglavlju.

2.1. Transformator modeli

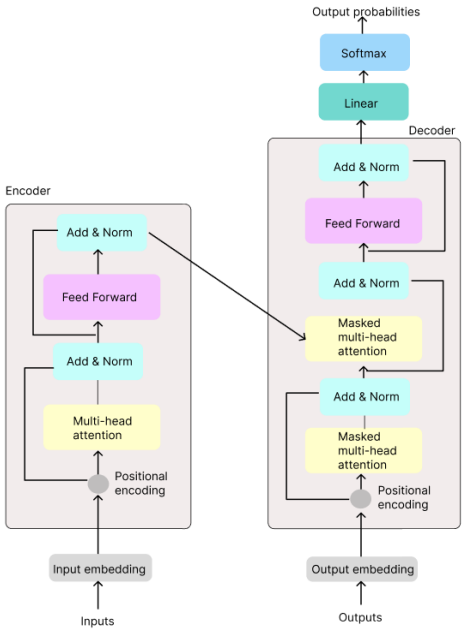
Transformator modeli [4] razlikuju se od svojih prethodnika po tome što uspešno rešavaju problem mogućnosti obraćanja pažnje na određene reči bez obzira koliko su one udaljene jedna od druge i po tome što imaju bolje performanse. Ovi modeli imaju sposobnost da prevaziđu oba koristeći prednosti mehanizma pažnje. Glavna karakteristika arhitekture ovih modela je da sadrže koder i dekoder. Pojednostavljeno, moglo bi se reći da koder uzima ulaz i daje matričnu reprezentaciju tog ulaza, dok dekoder preuzima tu matričnu reprezentaciju i iterativno generiše izlaz. Koder i dekoder su zapravo stek sa više slojeva, ali tako da oba imaju isti broj slojeva. Svi koderi imaju istu strukturu, a ulaz ulazi u svaki od njih i prosleđuje se sledećem koderu u nizu. Takođe, svi dekoderi imaju istu strukturu i dobijaju ulaz od poslednjeg kodera i prethodnog dekodera. Radni tok kodera može se podeliti na sledeće korake:

- ugrađivanje ulaza - dešava se samo u prvom koderu i predstavlja pretvaranje ulaznih tokena u vektore. Ova ugrađivanja obuhvataju semantička značenja leksema i pretvaraju ih u numeričke vrednosti. Svi koderi dobijaju listu vektora, ali u svakom koderu sem u prvom to bi bio izlaz kodera direktno ispod njega.
- poziciona kodiranje – dodaju se ulaznim ugrađenim elementima da bi pružili informacije o poziciji svakog tokena u nizu. Ovo transformer modelima omogućava da razumeju poziciju svake reči u rečenici. To se postiže upotrebom različitih sinusnih i kosinusnih funkcija.
- mehanizam pažnje - ovaj mehanizam može da se nazove i samopažnja i on omogućava da se svaka reč iz ulaza poveže sa drugim rečima. Rezultat pažnje se izračunava na osnovu upita, ključa i vrednosi. Upit je vektor koji predstavlja određenu reč ili token iz ulazne sekvence. Ključ je, takođe, vektor koji odgovara svakoj reči ili tokenu iz ulazne sekvence. Vrednost je povezana sa ključem i koristi se za konstruisanje izlaza pažnje. Kada se ključ i upit dobro podudaraju, što znači da imaju visoku ocenu pažnje, odgovarajuća vrednost se naglašava na izlazu. Vrednosti dobijene izračunavanje pažnje se nakon toga normalizuju kako bi se dobije težine pažnje, koje se koriste za generisanje izlaznog vektora.
- potpuno povezana mreža (normalizacija) - svaki izlaz podsloja se dodaje svoj ulazu (preostala veza) kako bi se ublažio problem nestajanja gradijenta. Ovaj proces će se ponoviti i nakon sledećeg koraka.
- neuronska mreža sa propagacijom unapred – kombinacija linearnih slojeva sa ReLu aktivacionom funckojim smeštenom između njih.
- izlaz kodera – skup vektora od kojih svaki predstavlja ulaznu sekvencu sa bogatim kontekstualnim razumevanjem.

Što se tiče dekodera, njegov radni tok može se podeliti na sledeće korake:

- ugrađivanje izlaza - deo isti kao kod kodera, dakle ulaz prvo prolazi kroz ugrađivanje.
- poziciono kodiranje - baš kao i kod kodera, ulaz prolazi kroz sloj pozicionog kodiranja.
- maskirani mehanizam samopažnje - ima jednu ključnu razliku u odnosu na isti sloj u koderu, a to je da sprečava pozicije da prisustvuju narednim pozicijama, što znači da svaka reč u nizu nije pod uticajem budućih tokena. Ovo maskiranje osigurava da predviđanja za određenu poziciju mogu zavisiti sam od poznatih izlaza na pozicijama pre nje.
- koder-dekoder pažnja više glava – ovaj sloj omogućava dekoderu da pronađe i nagalsi najrelevantnije delove ulaze kodera.
- neuronska mreža sa propagacijom unapred – svaki sloj dekodera uključuje potpuno povezanu mrežu za prosleđivanje, primenjenu na svaku poziciju posebno.
- linearni klasifikator i softmax za generisanje izlaznih verovatnoća - ovo je konačni linearni sloj koji funkcioniše kao klasifikator.

Opisana arhitektura kodera i dekodera može se videti na slici Slika 2.1.

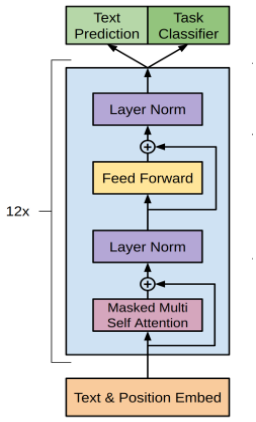


Slika 2.1 Arhitektura kodera i dekodera transformator modela

Modeli zasnovani na arhitekturi transformatora su Generative Pre-trained Transforme (GPT) modeli razvijeni od strane OpenAI kompanije. Tačnije, GPT koristi neizmenjeni dekoder deo transformatora, s tim što mu nedostaje deo vezan za pažnje dobijene koderom. Arhitektura ove porodice modela može se videti na slici Slika 2.2.

Porodicu modela čine mnogobrojne verzije GPT, GPT-2, GPT-3 i GPT-4 modela. U ovom radu, baziraćemo se na GPT-3 modele. Model ima 175 milijardi parametara koji predstavljaju težine na osnovu kojih model donosi odluke prilikom obrade i generisanja teksta. Treniran je nad više od 570 GB podataka koji uključuju vesti, knjige, vikipediju i mnoge druge izvore teksta. Može da se koristi za generisanje teksta koji je relevantan za zadatu temu, da razume zadati kontekst i koristi ga za generisanje odgovora, može prevodi tekstove sa jednog jezika na drugi, da vrši sumarizaciju teksta, generište tekstove i slično. Konkretan model koji se koristio za potrebe demonstracije sistema sa glasovnim i tekstualnim

upravljanjem je GPT-3.5-turbo [6], unapređena verzija GPT-3 modela, dizajnirana tako da bude jeftinija i brža od prethodne verzije.



Slika 2.2 Arhitektura GPT modela[5]

Još jedan model koji ima arhitekturu transformatora koji je, takođe, iskorišćen za potrebe demonstracije sistema sa glasovnim i tekstualnim upravljanjem je Whisper [7]. To je sistem za automatsko prepoznavanje govora obučen nad 680 hiljada sati višejezičkog materijala. Što se tiče arhitekture, implementiran je kao potupni koder-dekoder transformator.

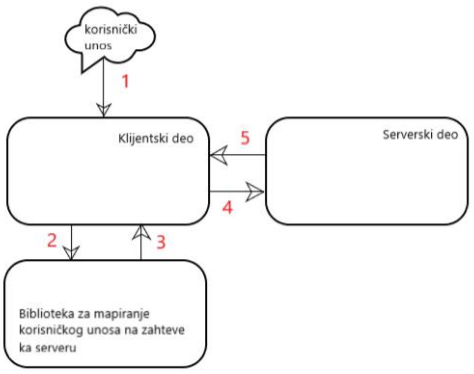
3. SOFTVERSKA ARHITEKTURA SISTEMA SA GLASOVNIM I TEKSTUALNIM UPRAVLJANJEM

Informacioni sistem sa glasovnim i tekstualnim upravljanjem, pored toga što bi se koristio na drugačiji način, drugačije bi se i razvijao. U nastavku ovog poglavlja biće prikazana moguća arhitektura jednog takvog veb sistema.

Sistem bi se sastojao od serverskog dela, koji opciono može da bude skup više manjih servisa i baza podataka i od klijentskog sloja. Na klijentskom sloju se uvodi razlika u odnosu na klasične veb sisteme. Pošto će ulaz u sistem biti prirodni jezik, klijentski sloj će morati da ga obradi pre nego što ga prosledi serverskom sloju. U tu svrhu, nastala je biblioteka kojoj se prosleđuje korisnički unos na osnovu kog ona vraća informacije o tome kakav je zahtev potrebno proslediti na serverski deo. Sam sistem i tok informacija kroz njega prikazani su na slici Slika 3.1. Iako je biblioteka za obradu korisničkog unosa, radi lakšeg prikaza, predstavljena kao posebna komponenta, ipak je potrebno naglasiti da je ona samo jedna zavisnost klijentskog sloja. Dakle, klijentski sloj će korisnički unos proslediti biblioteci koja će obradom da izdvoji putanju ka krajnjoj tački servera, HTTP metodu i telo zahteva. Nakon toga klijentski sloj će proslediti zahtev ka serveru, a kada primi odgovor, iskoristiće neku od svojih generičkih komponenti da taj odgovor prikaže.

Pomenuta biblioteka za mapiranje korisničkog zahteva na zahtev ka serveru, realizovana je za potrebe ovog rada, kao Node Package Manager (npm) paket [8]. Deo paketa koji se tiče komunikacije sa velikim jezičkim modelom implementiran je pomoću OpenaAI biblioteke [9]. Konkretno korišćene su dve njene funkcije:

- *openai.chat.completions.create* - funkcija koja vraća odgovor jezičkog modela na prosleđena pitanja. Bitno je napomenuti da postoji ograničenje u broju tokena koje jezički model može da obradi.



Slika 3.1 Arhitektura sistema sa glasovnim i tekstualnim upravljanjem

- *openai.audio.transcriptions.create* - funkcija koja obrađuje prosleđeni audio fajl tako da generiše tekst na osnovu njega.

Ono što je neophodno proslediti pomenutoj biblioteci pre same obrade zahteva, je Swagger [10] specifikaciju serverskog dela kao i podatke o velikom jezičkom modelu koji će se koristiti za obradu tekstualnih podataka. Pomenuta biblioteka, nudi dve funkcije koje se mogu pozvati:

- *processRequest* – prima tekstualni zahtev korisnika, a vraća putanju do krajnje tačke, HTTP merodu i opciono telo zahteva. U pozadini, biće poslato nekoliko zahteva jezičkom modelu. Prvo će se zahtevati krajnja tačka i HTTP metoda, a nakon toga u zavisnosti od tipa HTTP metode i telo zahteva. Ukoliko postoji telo zahteva, zahtevaće se i njegova validacija od strane jezičkog modela.
- *processRequestFromAudio* – prima audio zahtev korisnika, dok je povratna vrednost ista kao za prethodno opisanu funkciju. Ova funkcija ima jedan dodatni korak pre izvršavanja svih koraka prethodne funckije, a to je generisanje teksta iz prosleđenog audio fajla.

4. STUDIJA SLUČAJA APLIKACIJE ZA UPRAVLJANJE RESTORANOM

Za demonstraciju sistema sa glasovnim i tekstualnim upravljanjem izabran je informacioni sistem restorana koji je dovoljno raznovrstan da pokrije tipične scenarije ali sužen tako da se ne bavimo detaljima koji nisu relevantni za temu.

Aplikacija je namenjena za korišćenje, isključivo, od strane zaposlenih restorana. Prema tome postoji nekoliko tipova korisnika koji mogu da je koriste: sistem administratori, menadžeri, kuvari, konobari i šankeri. Što se tiče funkcionalnosti sistema, moguće je upravljati jelovnicima, stavkama jelovnika, podacima o zaposlenima i njihovim platama, porudžbinama i stavkama porudžbina, i rezervacijama stolova. Takođe, moguće je generisati izveštaje o poslovanju u određenom vremenu.

U nastavku potpoglavlja biće prikazani korisnički zahtevi izraženi prirodnim jezikom i rezultati obrade biblioteke za mapiranje korisničkog zahteva na zahtev ka serveru.

- Ukoliko korisnik želi da dobavi informacije o svim konobarima, dovoljno je da uputi zahtev “list all waiters”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:

{ “httpMethod”: “GET”, “path”: “/api/v1/waiter”}.

- Ukoliko korisnik želi da vidi sve porudžbine određenog konobara dovoljno je da uputi zahtev sličan “find all orders that have waiter with id 7”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:
{ “httpMethod”: “GET”, “path”: “/api/v1/orders/byWaiter/7” }.

- Ukoliko korisnik želi da vidi sve porudžbine određenog konobara koje imaju status “Poručeno”, dovoljno je da uputi zahtev “filter orders with status Poručeno and waiter id 7”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:
{ “httpMethod”: “GET”, “path”: “/api/v1/orders/filter/7/Poručeno” }

- Ukoliko korisnik želi da doda novog konobara u sistem potrebno je da uputi zahtev u kom se nalaze svi potrebni podaci za dodavanje novog konobara. Zahtev bi trebao da bude sličan “Create a new waiter. His first name is Marko, his last name is Markovic, his email is mare@example.rs, his phone number is 1234, his account number is 123”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:

```
{ “httpMethod”: “POST”,
“path”: “/api/v1/waiters”,
“payload”: {
   "name": "Marko",
   "lastName": "Markovic",
    "phoneNumber": "1234",
     "email": "mare@example.rs",
      "accountNumber": "123"
} }.
```

Ukoliko korisnik prilikom slanja zahteva izostavi neki od obaveznih podataka, biblioteka za mapiranje će to prepoznati i korisniku će biti prikazana poruka o tome. Primer poruke za prethodni zahtev je:
“In order for the request body to be valid for creating a waiter, we need to include the following information:
- First name of the waiter (name)
- Last name of the waiter (lastName)
- Phone number of the waiter (phoneNumber)
- Account number of the waiter (accountNumber)
Without this information, the request body would be considered invalid.“

5. ZAKLJUČAK

U današnjem svetu, računari imaju sveprisutnu i ključnu ulogu u gotovo svim aspektima života. Upravo zbog toga, postaje sve bitniji način na koji se odvija interakcija između korisnika i računarskih sistema. Interfejsi prirodnog jezika omogućavaju korisnicima da komuniciraju sa računarima koristeći isključivo prirodni jezik i time pružaju intuitivnije i pristupačnije korisničko iskustvo. To je dalo motivaciju za razvijanje sistema sa glasovnim i tekstualnim upravljanjem koji je opisan u ovom radu. Mogućnost implementacije ovakvih sistema omogućio je napredak obrade prirodnog jezika, a najviše veliki jezički modeli zasnovani na arhitekturi transformatora.
U radu je opisana softverska arhitektura sistema sa glasovnim i tekstualnim upravljanjem. Posebna pažnja posvećena je biblioteci koja mapira korisnički zahtev na zahtev ka serveru i koja je implementirana u svrhe ovog rada. Takođe, prikazano je nekoliko primera upotrebe na slučaju aplikacije za upravljanje restoranom.
Prikazani primeri upotrebe dovode nas do zaključka da implementirani sistem uspešno obrađuje tipične zahteve korisnika restoranske aplikacije i time mu olakšava upotrebu. Ipak, bitno je pomenuti i ograničenja koja se ogledaju u nepodržavanju obrade kompleksnijih upita, brzini odgovora, ali i broju tokena koje veliki jezički model može da obradi u određenom vremenskom periodu. Ono što bi doprinelo još većoj uspešnosti ovih sistema je svakako rad na razvijanju velikih jezičkih modela kako bi se uklonila pomenuta ograničenja. Takođe, ovakva vrsta upravljanja mogla bi da se oproba i u drugim tipovima aplikacija, kao što su one za upravljanje pametnim kućnim uređajima.

LITERATURA

[1] Nadkarni, Prakash M., Lucila Ohno-Machado, and Wendy W. Chapman. "Natural language processing: an introduction." *Journal of the American Medical Informatics Association* 18.5, 2011

[2] Chang, Yupeng, et al. "A survey on evaluation of large language models." *ACM Transactions on Intelligent Systems and Technology* 15.3, 2024

[3] Ogden, William C., and P. Bernick. "Using natural language interfaces." *Handbook of human-computer interaction*. North-Holland, 1997.

[4] Radford, Alec. "Improving language understanding by generative pre-training." (2018).

[5] Brown, Tom B. "Language models are few-shot learners." *arXiv preprint arXiv:2005.14165* (2020).

[6] GPT-3.5 Turbo https://platform.openai.com/docs/models/gpt-3-5-turbo [datum pristupa 25.08.2024.]

[7] Introducing Whisper https://openai.com/index/whisper [datum pristupa 25.08.2024.]

[8] npm https://www.npmjs.com/ [datum pristupa 28.08.2024.]

[9] OpenAI https://platform.openai.com/docs/api-reference/introduction [datum pristupa 28.08.2024.]

[10] Swagger https://swagger.io/ [datum pristupa 29.08.2024.]

Kratka biografija:

Jovana Jevtić rođena je 22.06.1999. godine u Zvorniku. Završila je osnovnu školu „Branko Radičević“ u Malom Zvorniku, a nakon toga gimnaziju u srednjoj školi Mali Zvornik. Godine 2018. upisala je Fakultet tehničkih nauka u Novom Sadu, smer Softversko inženjerstvo i informacione tehnologije. Sve ispite polaže i studije završava u roku 2022. godine. Iste godine upisuje master akademske studije na istom smeru koje završava 2024. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4
DOI: https://doi.org/10.24867/30BE30Ruvceski

PERFORMANSE MIKROSERVISNE APLIKACIJE U NODE.JS I .NET OKRUŽENJU POSTAVLJENO NA AWS-U

PERFORMANCES OF MICROSERVICE APPLICATION IN NODE.JS AND .NET ENVIRONMENT DEPLOYED ON AWS

Teodora Ruvčeski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Sa rastom upotrebe mikroservisne arhitekture u razvoju softverskih aplikacija, postaje važno razumeti kako različite tehnologije, kao što su Node.js i .NET, utiču na performanse ovakvih sistema. Ovaj rad istražuje i upoređuje performanse mikroservisnih aplikacija razvijenih u Node.js i .NET tehnologijama, kroz analizu tri specifična toka podataka. Merenja su izvršena na identičnim aplikacijama, koje implementiraju iste funkcionalnosti, ali koriste različita tehnološka okruženja. Rezultati pokazuju razlike u efikasnosti i brzini odziva između ove dve platforme, pružajući uvide u njihove prednosti i slabosti u realnim uslovima mikroservisnih okruženja.*

Ključne reči: *.NET, Node.js, Mikroservisi, AWS, Performanse*

Abstract – *With the increasing adoption of microservice architecture in software development, it becomes important to understand how different technologies, such as Node.js and .NET, impacts the performance of such systems. This paper explores and compares the performance of microservice applications developed in Node.js and .NET technologies through the analysis of three specific data flows. The measurements were conducted on identical applications, implementing the same functionalities but using different technological environments. The results highlight differences in efficiency and response speed between these two platforms, providing insights into their strengths and weaknesses in real-world microservice environments.*

Keywords: *.NET, Node.js, Microservices, AWS, Performances*

1. UVOD

Savremeni razvoj softverskih rešenja sve češće se oslanja na mikroservisnu arhitekturu, koja omogućava modularni pristup kreiranju aplikacija kroz male, nezavisne servise. Ovaj pristup donosi brojne prednosti, uključujući veću skalabilnost i fleksibilnost, ali i jednostavnije održavanje sistema. Međutim, pri izboru tehnologije za razvoj mikroservisa, performanse igraju ključnu ulogu.

U ovoj studiji analizirane su performanse aplikacija razvijenih u dva popularna okruženja – Node.js i .NET. Oba tehnološka rešenja imaju široku primenu u industriji, ali se razlikuju u pristupu obradi zadataka i upravljanju resursima. Node.js [1] se oslanja na asinhroni model koji omogućava visoku skalabilnost u realnim vremenskim aplikacijama, dok .NET pruža robustan okvir za izgradnju visoko performantnih rešenja sa naglaskom na stabilnost i integraciju sa različitim sistemima.

Cilj istraživanja jeste da se kroz seriju testova proceni efikasnost i brzina odziva aplikacija razvijenih u ova dva okruženja. Testovi su uključivali različite scenarije, poput interakcije sa bazom podataka, komunikacije sa eksternim servisima i vremena potrebnog za pokretanje mikroservisa. Ova analiza pruža uvid u specifične prednosti i izazove svake od tehnologija, čime doprinosi boljem razumevanju njihovih performansi u kontekstu realnih zahteva softverskih sistema.

2. KORIŠĆENE TEHNOLOGIJE

U ovom istraživanju korišćeno je nekoliko ključnih tehnologija kako bi se omogućilo testiranje i poređenje performansi aplikacija. Aplikacije su razvijene u mikroservisnom okruženju, arhitekture koja omogućava modularan pristup razvoju softverskih rešenja [5]. Svaki mikroservis predstavlja nezavisnu komponentu zaduženu za specifične funkcionalnosti, što olakšava skaliranje i održavanje aplikacija. Ovakva arhitektura omogućava bolju distribuciju resursa i bržu obradu zahteva, a koristi se u modernim aplikacijama koje zahtevaju visoku efikasnost i fleksibilnost.

Za implementaciju mikroservisa korišćen je .NET, tehnologija koja omogućava razvoj robustnih, sigurnih i visoko performantnih aplikacija. Jedna od glavnih prednosti .NET platforme je njena sposobnost da se lako integriše sa širokim spektrom sistema i tehnologija, što je posebno važno u kompleksnim poslovnim okruženjima. .NET podržava *multithreading* i paralelno izvršavanje zadataka [3], čime se postiže optimizovano upravljanje resursima i smanjuje vreme izvršavanja zahteva. Osim toga, .NET pruža napredne alate za praćenje performansi, debugovanje i profilisanje aplikacija, što olakšava održavanje sistema u produkcionim uslovima [4].

S druge strane, Node.js je korišćen zbog svoje sposobnosti da efikasno upravlja velikim brojem istovremenih konekcija zahvaljujući asinhronom modelu [2] izvršavanja. Ova tehnologija je poznata po svojoj brzini i skalabilnosti, što je čini idealnom za aplikacije

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

koje zahtevaju visok nivo interakcije sa korisnicima i brze odgovore. Node.js koristi *event-driven* arhitekturu [6], što omogućava neblokirajuće operacije, idealne za aplikacije u realnom vremenu i one koje često komuniciraju sa eksternim API-jevima. Za razvoj API-ja unutar Node.js okruženja korišćen je Express, lagan i fleksibilan framework koji značajno ubrzava proces razvoja, omogućavajući jednostavno rukovanje HTTP zahtevima i integraciju sa različitim servisima.
Aplikacije su postavljene na AWS EC2 instancama, pružajući *cloud* okruženje za pokretanje i testiranje performansi u identičnim uslovima. Za upravljanje bazama podataka korišćena je Supabase platforma, zajedno sa PostgreSQL bazom podataka, koja je omogućila efikasno skladištenje i pristup podacima za potrebe aplikacija.
Testiranje zahteva i performansi aplikacija izvršeno je korišćenjem alata Postman, koji je korišćen za simulaciju stvarnih *HTTP* zahteva prema aplikacijama, omogućavajući preciznu analizu vremena odgovora i opterećenja aplikacija.
Ove tehnologije su odabrane zbog njihove fleksibilnosti, skalabilnosti i mogućnosti da podrže moderne aplikacije u realnom vremenu, što ih čini idealnim za ovo istraživanje performansi.

3. ARHITEKTURA APLIKACIJE

Aplikacija korišćena za testiranje dve tehnologije, .NET i Node.js, predstavlja složenu web aplikaciju u formi više različitih API-ja koji čine jedan mikroservis (*Application Programming Interface*). Ovi API-ji omogućavaju komunikaciju između klijentskih aplikacija i servera putem definisanih pristupnih tačaka (*endpoints*), pružajući standardizovanu razmenu podataka. API se može posmatrati kao jedan od načina primene principa Crne kutije (*Black Box*), gde klijent koristi funkcije koje sistem pruža, bez potrebe da zna kako su te funkcije implementirane unutar sistema. API pristup se često koristi kako bi se unifikovala komunikacija između različitih sistema i svih klijentskih aplikacija. Struktura i arhitektura aplikacije biće detaljnije opisano u nastavku rada, sa posebnim fokusom na tehničke aspekte implementacije API-ja i testiranja u oba tehnološka okruženja.

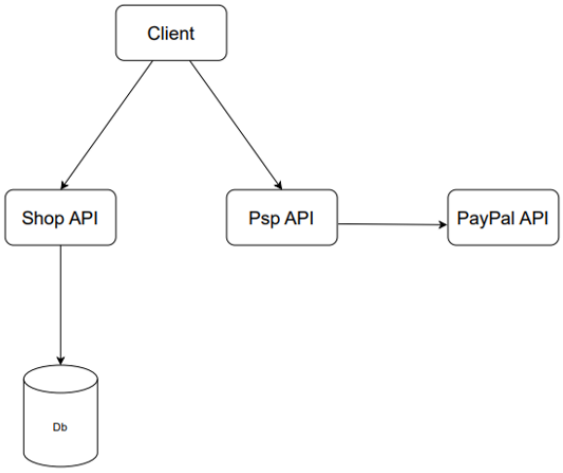
3.1 API

API (Programski interfejs za aplikacije) omogućava komunikaciju između softverskih sistema definišući pravila interakcije sa funkcijama i podacima sistema. Kroz pristupne tačke (*endpoints*), API unifikuje komunikaciju između različitih sistema, bez potrebe da klijent poznaje detalje implementacije. U dokumentaciji su opisane pristupne tačke, parametri i očekivani odgovori, čineći API ključnim za efikasnu razmenu podataka.

3.2 Test Aplikacija

Testirana aplikacija je platforma za potragu za poslom, omogućavajući kompanijama objavu oglasa i kandidatima da kreiraju profile i postavljaju CV-jeve. Ključna funkcionalnost je kupovina premium paketa, uz podršku za sigurne online transakcije putem PayPal-a. Aplikacija je implementirana kao sistem mikroservisa (*shop-api, psp-api, paypal-api, bank-api*), uz NGINX koji orkestrira komunikaciju.

Back-end aplikacije razvijen je u oba okruženja, .NET i Node.js, sa identičnim *endpoints*-ima i logikom, što omogućava direktno poređenje performansi i skalabilnosti ova dva tehnološka *stack-a*.



Slika 1. *Dijagram komponenti sistema*

3.3 Deployment aplikacije na AWS

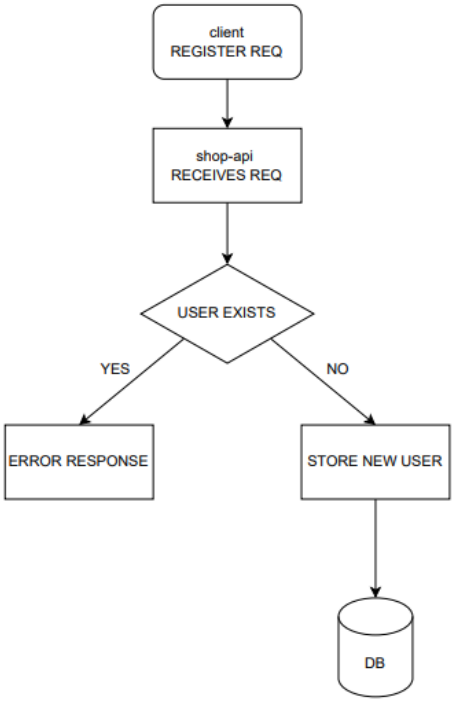
Proces deploymenta mikroservisne aplikacije na AWS infrastrukturu izveden je korišćenjem EC2 instanci, kako bi se omogućilo kompetentno poređenje performansi između Node.js i .NET aplikacija. Oba okruženja koriste *t2.micro* instance sa istim konfiguracijama, uključujući 1 vCPU, 1 GB RAM-a i 8 GB EBS skladišta.

Za Node.js aplikaciju, EC2 instanca sa *Amazon Linux 2 AMI* je konfigurisana za pokretanje aplikacije, uz instalaciju Node.js, NGINX-a i *deployment* putem SCP-a. Za .NET aplikaciju, EC2 instanca sa Windows Server 2019 je korišćena, uz instalaciju IIS-a i .NET Core Runtime-a, te deployment unutar IIS-a. Nakon *deploymenta*, oba sistema su testirana kako bi se osiguralo ispravno funkcionisanje aplikacija, omogućavajući poređenje njihovih performansi u identičnim *cloud* okruženjima.

4. REZULTATI TESTIRANJA

U narednim poglavljima biće prikazani rezultati testiranja aplikacija u različitim scenarijima kako bi se detaljno analizirale performanse oba tehnološka okruženja. Testovi su sprovedeni kako bi se procenila efikasnost aplikacija u ključnim aspektima kao što su brzina obrade zahteva, komunikacija sa eksternim servisima i vreme pokretanja sistema.
Testiranje je vršeno merenjem vremena potrebnim za izvršenje svakog pojedinačnog zahteva do trenutka dobijanja odgovora od servera. Zahtevi prema API-ju slati su putem *Postman* aplikacije. Za svaki scenario merenje je izvođeno više puta zatim je računata srednja vrednost merenja, devijacija vrednosti i distribucija izmerenih vremena.
Ova analiza pruža jasne uvide u mogućnosti i ograničenja korišćenih tehnologija, omogućavajući bolje razumevanje njihovih performansi u realnim uslovima.

4.1 Interakcija sa bazom podataka – registracija korisnika

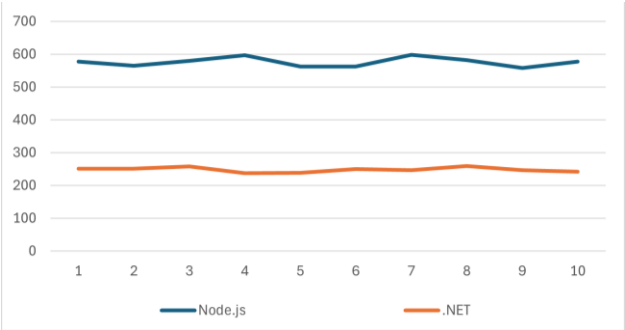


Slika 2. *Dijagram toka podataka registracije korisnika*

Prvi test obuhvata interakciju sa bazom podataka tokom registracije korisnika, pri čemu su merenja vršena za vreme potrebno da korisnički zahtev bude obrađen i podaci upisani u bazu. U oba okruženja izvršeno je 10 merenja.

Prosečno vreme izvršavanja za Node.js iznosi 575 ms, uz standardnu devijaciju od 14.46 ms. S druge strane, za .NET aplikaciju prosečno vreme izvršavanja iznosi 248 ms, sa standardnom devijacijom od 7.55 ms.

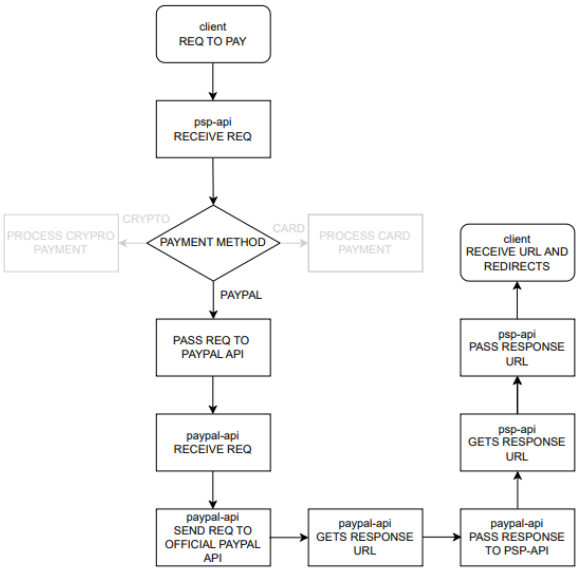
Rezultati pokazuju da je .NET značajno brži u obradi zahteva i upisivanju podataka u bazu, što ukazuje na efikasniji način upravljanja resursima tokom I/O operacija u ovom okruženju.



Slika 3. *Distribucija izmerenih vremena za registraciju*

4.2 Komunikacija sa eksternim servisom – plaćanje putem PayPal-a

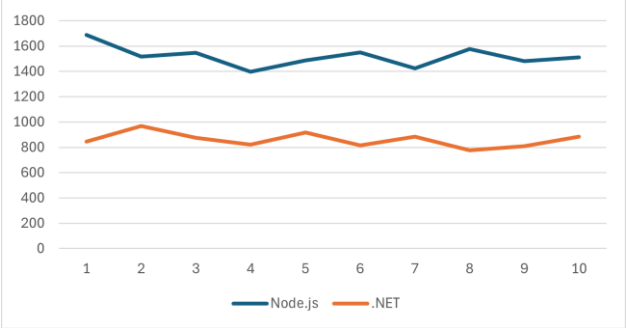
U drugom testu mereno je vreme komunikacije sa PayPal API-jem prilikom obrade plaćanja. I u ovom slučaju je izvršeno 10 merenja za oba okruženja.



Slika 4. *Dijagram toka podataka paypal plaćanja*

Prosečno vreme izvršavanja za Node.js je 1517 ms, dok je standardna devijacija 81.39 ms, što ukazuje na varijacije u merenjima. Za .NET aplikaciju, prosečno vreme iznosi 859 ms, uz standardnu devijaciju od 57.26 ms, što pokazuje nešto manju varijabilnost u odnosu na Node.js.

Ovde .NET takođe pokazuje prednost, sa bržim vremenom odgovora u odnosu na Node.js, što je važno za aplikacije koje zahtevaju brzu i sigurnu obradu plaćanja.



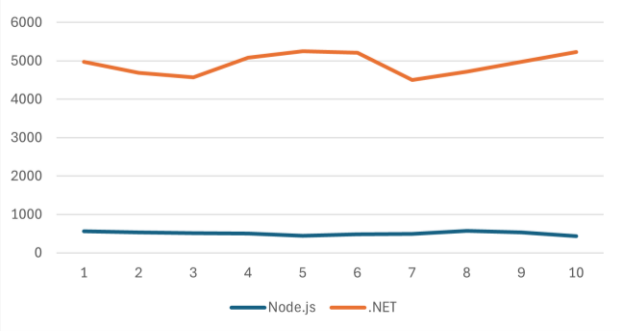
Slika 5. *Distribucija izmerenih vremena za paypal plaćanje*

4.3 Inicijalizacija i pokretanje mikroservisa

Treći test fokusirao se na vreme potrebno za kompletno pokretanje mikroservisa, uključujući inicijalizaciju resursa i povezivanje sa bazama podataka. Test je obavljen sa 10 merenja u oba okruženja.

Prosečno vreme izvršavanja za Node.js iznosi 507 ms, uz standardnu devijaciju od 45.03 ms. S druge strane, za .NET aplikaciju, prosečno vreme je značajno veće, iznoseći 4919 ms, sa standardnom devijacijom od 282.19 ms, što ukazuje na veću varijabilnost u merenjima.

Node.js pokazuje veliku prednost u brzini pokretanja, zahvaljujući svojoj jednostavnijoj arhitekturi i brzini JavaScript izvršavanja, dok .NET zahteva značajno duže vreme za inicijalizaciju zbog složenijih procesa prevođenja i optimizacije aplikacije za dugotrajno izvršavanje.



Slika 6. *Distribucija izmerenih vremena za pokretanje servisa*

Ovi rezultati pokazuju kako se performanse Node.js i .NET razlikuju u zavisnosti od specifičnog scenarija, pri čemu .NET nudi bolju efikasnost u operacijama koje zahtevaju obradu podataka, dok Node.js briljira u brzini pokretanja i upravljanju istovremenim konekcijama.

5. ZAKLJUČAK

Zaključak ove studije pruža dublji uvid u performanse aplikacija razvijenih u dva vodeća tehnološka okruženja – Node.js i .NET – i pokazuje kako se svaka od ovih tehnologija ponaša u specifičnim realnim scenarijima. Testovi su obuhvatili ključne aspekte performansi mikroservisnih aplikacija, uključujući interakciju sa bazama podataka, komunikaciju sa eksternim servisima i inicijalizaciju sistema.

Rezultati pokazuju da je .NET ostvario superiorne rezultate u oblastima kao što su brzina obrade podataka i stabilnost prilikom komunikacije sa eksternim servisima, što je naročito važno za aplikacije koje zahtevaju visoku pouzdanost i procesorsku efikasnost. .NET koristi *multi-thread* model, koji omogućava paralelnu obradu zadataka, čime postiže bolje rezultate u situacijama koje uključuju intenzivne operacije sa bazom podataka i zahtevaju visoku efikasnost u radu sa resursima. Ova karakteristika čini .NET pogodnim za aplikacije koje zahtevaju brzu obradu velikog broja zahteva i stabilnu integraciju sa složenim ekosistemima.

Sa druge strane, Node.js se istakao svojom efikasnošću kada je reč o brzom pokretanju aplikacija i upravljanju velikim brojem istovremenih konekcija, zahvaljujući svojoj asinhronoj prirodi i *event-driven* arhitekturi. Node.js koristi jedinstveni *event loop*, što omogućava lako skaliranje aplikacija i efikasno rukovanje brojnim zahtevima koji dolaze od klijenata. Ova tehnologija se posebno pokazala efikasnom u scenarijima gde je brzina pokretanja sistema ključna, što je čini pogodnom za aplikacije koje zahtevaju agilnost i brzu reakciju na zahteve korisnika.

Sveobuhvatna analiza pokazuje da oba tehnološka okruženja imaju svoje specifične prednosti, koje dolaze do izražaja u zavisnosti od zahteva projekta. .NET se pokazao kao bolji izbor za aplikacije koje zahtevaju stabilnost, sigurnost i efikasnost u radu sa bazama podataka i eksternim servisima, dok Node.js pruža prednosti u aplikacijama koje zahtevaju brzo startovanje i jednostavno skaliranje. Ova razlika u performansama ukazuje na to da izbor tehnologije ne treba da bude vođen isključivo preferencijama, već da mora biti pažljivo prilagođen specifičnim potrebama i ciljevima projekta.

Ovi rezultati mogu poslužiti kao vodič za inženjere softverskih sistema pri odlučivanju o tehnologijama koje će koristiti u budućim projektima. U zavisnosti od zahteva u pogledu performansi, brzine odziva, pouzdanosti i fleksibilnosti, Node.js i .NET pružaju različite prednosti.

Buduća istraživanja u ovoj oblasti mogla bi se fokusirati na dodatnu analizu skalabilnosti i energetske efikasnosti oba tehnološka okruženja.

Takođe, dodatno istraživanje moglo bi obuhvatiti evaluaciju sigurnosnih karakteristika svake platforme, kao i analizu ponašanja ovih okruženja u produkcionim uslovima pod velikim opterećenjem. Uvođenje novih tehnologija, poput *serverless* arhitektura i *edge computing-a*, u ova okruženja moglo bi doneti nove mogućnosti za optimizaciju i unapređenje performansi sistema, što predstavlja plodno tlo za dalja istraživanja.

6. LITERATURA

[1] Mario Casciaro, *"Node.js Design Patterns"*, 2016.

[2] Lakshmi Prasanna Chitra, Ravikanth Satapathy, *"Performance comparison and evaluation of Node.js and traditional web server"*, International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies, 2017.

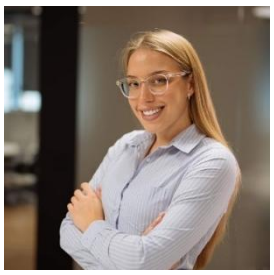
[3] Adam Freeman, "*Pro ASP.NET Core 6*", 2021.

[4] Iris Classon, Kenneth Yagen, *"Microservices in .NET Core: With Examples in Nancy"*, Apress, 2017.

[5] Sam Newman, *"Building Microservices: Designing Fine-Grained Systems"*, O'Reilly Media, 2021.

[6] David Mark Clements, *"Node Cookbook: Discover solutions, techniques, and best practices for server-side web development with Node.js 14"*, Packt Publishing, 2020.

Kratka biografija:



Teodora Ruvčeski rođena je u Novom Sadu 1999. godine. Osnovne akademske studije na Fakultetu tehničkih nauka Univerziteta u Novom Sadu upisala je 2018. godine. Diplomirala je 2022. godine na smeru Primenjeno softversko inženjerstvo i iste godine upisala master akademske studije na smeru Računarstvo i informatika.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/30BE31Aleksic

PRIMENA MAŠINSKOG UČENJA U PREDIKCIJI ŽIVOTNOG VEKA LJUDI NA OSNOVU SOCIO-EKONOMSKIH I DEMOGRAFSKIH KARAKTERISTIKA

APPLICATION OF MACHINE LEARNING IN PREDICTING HUMAN LIFE EXPECTANCY BASED ON SOCIO-ECONOMIC AND DEMOGRAPHIC CHARACTERISTICS

Lenka Isidora Aleksić, Fakultet tehničkih nauka, Novi Sad

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – *Ovaj rad analizira primenu različitih metoda mašinskog učenja u predikciji životnog veka ljudi koristeći socio-ekonomske i demografske karakteristike. Kroz obradu podataka iz skupa ‚World Health Statistics 2020' Svetske zdravstvene organizacije, razvijeni su modeli koji koriste algoritme poput Decision Tree, XGBoost, Random Forest i neuronskih mreža. Rad opisuje proces preprocesiranja podataka, treniranje modela i evaluaciju performansi, sa posebnim fokusom na upotrebu Python biblioteka kao što su NumPy, Pandas, TensorFlow i Scikit-learn za implementaciju rešenja.*

Ključne reči: *HALE; polu-nadgledano učenje; metode mašinskog učenja; životni vek; baza podataka SZO.*

Abstract – *This paper analyzes the application of various machine learning methods in predicting human life expectancy using socio-economic and demographic characteristics. Through data processing from the 'World Health Statistics 2020' dataset, models have been developed using algorithms such as Decision Tree, XGBoost, Random Forest, and neural networks. The paper describes the data preprocessing process, model training, and performance evaluation, with a particular focus on the use of Python libraries such as NumPy, Pandas, TensorFlow, and Scikit-learn for implementing the solutions.*

Keywords: *HALE; semi-supervised learning; machine learning methods; life-expectancy; WHO-dataset;*

1. UVOD

Tema životnog veka je od važna za javnozdravstvene ustanove, jer omogućava bolje razumevanje faktora koji utiču na dugovečnost ljudi, a time i razvoj politika koje mogu poboljšati zdravstvene ishode u različitim populacijama. Za potrebe ovog istraživanja korišćeni su podaci Svetske zdravstvene organizacije iz skupa World Health Statistics 2020 (WHS2020), koji obuhvata informacije za preko 200 zemalja sveta. Cilj rada je primeniti različite modele mašinskog učenja radi predikcije očekivanog životnog veka na osnovu gorenavedenih karakteristika. Poseban akcenat stavljen je na optimizaciju modela i njihovu sposobnost predikcije na globalnom nivou. Metode poput *Decision Tree*-a, *XGBoost*-a, i dubokih neuronskih mreža primenjene su kako bi se omogućilo otkrivanje skrivenih obrazaca u podacima. Rad se takođe osvrće na izazove rada sa podacima, kao što su nepotpune vrednosti i normalizacija. Ovo istraživanje nastoji da ponudi dublji uvid u predikciju životnog veka korišćenjem savremenih tehnologija mašinskog učenja, što može imati značajan uticaj na različite discipline, uključujući ekonomiju, sociologiju i javne politike.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Dunja Vrbaški, asis. prof.

2. SKUP PODATAKA

Skup podataka za implementaciju rešenja sadrži 39 zasebih fajlova o različitim temama koji mogu uticati na zdravlje ljudi. Glavni fokus podataka je na indikatorima kao što su očekivani životni vek, stope mortaliteta, zdravstvene usluge, i socio-ekonomski faktori. Svaka tabela unutar skupa podataka sadrži informacije o zemljama, godinama, demografskim promenljivim, kao i o zdravstvenim pokazateljima, omogućavajući sveobuhvatnu analizu.

Health Adjusted Life Expectancy (HALE) kombinuje dužinu života sa kvalitetom života, pružajući dublji uvid u zdravlje stanovništva i predstavlja glavni indikator za metode mašinskog učenja, za koji je ovaj skup podataka posebno prilagođen. Time se omogućava analiza velikih i složenih zdravstvenih trendova putem regresije, klasifikacije i klasterizacije. Međutim, jedan od najvećih izazova pri radu sa ovim podacima jeste prisustvo nedostajućih vrednosti i neujednačenost u formatima, što zahteva pažljivo pretprocesiranje, uključujući normalizaciju i imputaciju nedostajućih vrednosti. Uprkos izazovima, primena ovih tehnika omogućava stvaranje robusnog skupa podataka koji može služiti za precizniju analizu zdravstvenih trendova i predikciju životnog veka na globalnom nivou.

3. KORIŠĆENE TEHNOLOGIJE ZA ANALIZU PODATAKA

Prilikom rada na analizi i modeliranju podataka u okviru predikcije životnog veka, *Python* biblioteke predstavljaju alat za efikasno upravljanje podacima, statističku analizu i primenu mašinskog učenja. One omogućavaju obradu velikih skupova podataka, vršenje kompleksnih matematičkih operacija i vizualizaciju rezultata, što je neophodno za savremeni pristup analizi zdravstvenih

podataka. Kako zajedno čine jedinstvenu osnovu za obradu i analizu podataka u okviru ovog projekta, u sledećim poglavljima biće opisane njihove ključne funkcije.

3.1. *NumPy* biblioteka

NumPy je osnovna *Python* biblioteka koja omogućava efikasnu manipulaciju višedimenzionalnim nizovima podataka. Njena ključna struktura je *ndarray*, koja ubrzava operacije na velikim setovima podataka zbog mogućnosti skladištenja podataka u kontinuiranom bloku memorije. Ova organizacija omogućava brzo izvršavanje složenih matematičkih operacija poput matričnih operacija, Fourierovih transformacija, kao i rešavanje sistema linearnih jednačina. Jedna od najvažnijih karakteristika *NumPy*-a je vektorizacija, koja omogućava operacije na celim nizovima podataka bez potrebe za eksplicitnim iteracijama. To čini rad sa velikim podacima efikasnim i ubrzava analizu, posebno u aplikacijama mašinskog učenja. U ovom radu, *NumPy* je bio presudan za pripremu podataka i statističku analizu, osiguravajući brzinu i efikasnost u procesu obrade velikih numeričkih skupova podataka.

3.2. *Pandas* biblioteka

Pandas služi za rad sa tabelarnim podacima, posebno korisna za manipulaciju i analizu strukturiranih podataka kroz strukturu *DataFrame*. Ova struktura omogućava skladištenje podataka različitih tipova u redovima i kolonama, omogućavajući efikasno baratanje velikim datasetovima. Pandas nudi funkcije za rad sa nedostajućim vrijednostima, poput *fillna()* i *dropna()*, što je od velike važnosti u radu sa stvarnim podacima. Biblioteka takođe pruža alate za grupisanje i agregaciju, olakšavajući složene analize po kategorijama. U ovom radu, *Pandas* je bio ključan u pripremi podataka za modele mašinskog učenja, omogućavajući spajanje i transformaciju različitih datasetova, te analiziranje vremenskih serija koje su bile za istraživanje demografskih i zdravstvenih trendova.

3.3. *Scikit-Learn* biblioteka

Scikit-learn je osnovna *Python* biblioteka za mašinsko učenje. Veoma je korisna jer nudi širok spektar algoritama za klasifikaciju, regresiju, klasterizaciju, kao i alate za smanjenje dimenzionalnosti. Posebno se ističe zbog svoje jednostavnosti upotrebe i standardizovanog procesa treniranja modela.

U ovom radu, *Scikit-learn* je korišćen za implementaciju različitih modela mašinskog učenja poput *Decision Tree*, *Random Forest*, i *XGBoost*. Uz pomoć funkcija kao što su *train_test_split()* i *cross_val_score()*, proces treniranja i evaluacije modela je bio olakšan, omogućavajući lako poređenje performansi. Biblioteka je takođe omogućila napredne tehnike za pretprocesiranje podataka, skaliranje i enkodiranje kategorijalnih promenljivih. Evaluacija modela je vršena uz metrike poput *Mean Squared Error* (MSE) i R^2 *Score*.

3.4. *TensorFlow* i *Keras* biblioteke

TensorFlow i *Keras* su korišćeni za izradu i treniranje dubokih neuronskih mreža, omogućavajući fleksibilan rad sa složenim podacima. *Keras* je API koji olakšava kreiranje i treniranje modela, dok *TensorFlow* pruža infrastrukturnu podršku i ubrzanje uz pomoć GPU-a. Ove biblioteke omogućavaju dodavanje slojeva i podešavanje hiperparametara modela, što je bilo presudno u optimizaciji performansi neuronskih mreža korišćenih u ovom radu. Funkcije poput *fit()* i *evaluate()* olakšale su proces treniranja, dok su napredne tehnike, kao što su *Recurrent Neural Networks* (RNNs) i *Convolutional Neural Networks* (CNNs), korišćene za analizu vremenskih serija i prepoznavanje obrazaca u podacima.

4. METODE OBUČAVANJA

U ovom poglavlju akcenat će biti na modelima obučavanja koji su korišćeni u ovom projektu kako bi se dobili optimalni rezultati. Sposobnost mašinskog učenja da uči iz podataka i donosi odluke na osnovu tih znanja čini ga jednim od najmoćnijih alata u današnjem svetu informacionih tehnologija.

4.1. *Decision Tree* model obučavanja

Decision Tree je osnovni algoritam mašinskog učenja koji se koristi za rešavanje problema klasifikacije i regresije. Ovaj model oponaša način na koji ljudi donose odluke kroz hijerarhijsku strukturu drveća, gde svaka grana predstavlja odluku na osnovu ulaznih podataka, a svaki čvor predstavlja karakteristiku ili atribut. *Decision Tree* je jednostavan za interpretaciju i vizualizaciju, zbog čega se često koristi u oblastima kao što su medicina i finansije, gde je objašnjivost modela ključna. Ipak, jedan od njegovih najvećih nedostataka je sklonost ka prekomernom učenju (*overfitting*), naročito kada model postane previše složen ili dubok. Da bi se smanjila ova sklonost, koriste se tehnike poput orezivanja stabla (*pruning*) ili ograničenja na maksimalnu dubinu stabla.

4.2. Neuronske mreže

Neuronske mreže su napredan alat u mašinskom učenju, inspirisan radom bioloških neurona, i omogućavaju rešavanje složenih zadataka poput prepoznavanja obrazaca, obrade slika i prirodnog jezika. Ove mreže se sastoje od slojeva veštačkih neurona koji su povezani u mrežu i mogu učiti složene i nelinearne relacije u podacima. U ovom radu, neuronske mreže su korišćene za predikciju životnog veka na osnovu demografskih i zdravstvenih podataka. Zbog svoje sposobnosti da modeliraju složene odnose, neuronske mreže su postale ključne u mnogim oblastima, ali njihova primena zahteva veliku količinu podataka i vremena za treniranje.

4.3. XGBoost u mašinskom učenju

XGBoost je napredna implementacija algoritma gradijentnog boostinga, koja je poznata po visokoj brzini i preciznosti. Ovde *XGBoost* je korišćen je kao još jedan model jer pruža veoma precizne rezultate. Ključni princip *XGBoost*-a je kombinovanje više jednostavnih modela, poput *Decision Tree*-a, kako bi se formirao snažan ansambl. Algoritam koristi različite tehnike optimizacije, poput regulacije (L1 i L2), kako bi se izbegao problem pretreniranosti i poboljšale performanse modela.

4.4. *Random Forest* model obučavanja

Random Forest je ansambl metoda koja kombinuje više *Decision Tree*-a kako bi se poboljšala stabilnost i tačnost

predikcija. Prednost ovog modela je u tome što primenjuje tehniku *bagging*, koja koristi različite podskupove podataka za generisanje svakog stabla, čime se smanjuje rizik od pretreniranosti. Konačna predikcija se zasniva na glasanju svih stabala u ansamblu, čime se postiže visoka tačnost i otpornost na šum u podacima.

4.5. ARIMA model u analizi vremenskih serija

ARIMA (*Autoregressive Integrated Moving Average*) model je jedan od najpoznatijih modela za analizu vremenskih serija. Ovaj model se koristi za predikciju vremenski zavisnih podataka, kao što su ekonomski indikatori ili vremenske promene. Sastoji se od tri glavna dela: autoregresivnog (AR), integrisanog (I) i pokretnog proseka (MA), gde svaka komponenta modeluje određene aspekte vremenskih serija. ARIMA model je korišćen za predikciju životnog veka jer omogućava hvatanje obrazaca u podacima i preciznu prognozu budućih vrednosti na osnovu istorijskih podataka.

5. REZULTATI I DISKUSIJA

Gore navedeni modeli testirani su pomoću nadgledanog i polu-nagledanog učenja, s ciljem da se identifikuju najbolje performanse i ključni faktori uticaja na predikcije.

Jedan od ključnih aspekata analize bio je procena uticaja vremenske kolone, koja predstavlja period u kojem su prikupljeni podaci, na performanse modela. RMSE i R^2 metrike su korišćene za evaluaciju tačnosti modela. Na osnovu rezultata, modeli kao što su *Random Forest* i *XGBoost* pokazali su se superiornima u poređenju sa *Decision Tree* modelom i potpuno povezanom neuronskom mrežom. Takođe, polu-nadgledano učenje nije značajno poboljšalo performanse u većini slučajeva. Samo kod *Decision Tree* modela i neuronske mreže bez period kolone uočeni su pozitivni rezultati polu-nadgledanog učenja. Kada je period kolona uključena, neuronske mreže su pokazale poboljšanja u predikciji. Najbolje performanse su primećene kod *XGBoost* i *Random Forest* modela, sa stabilnim rezultatima u različitim varijacijama podataka.

Rezultati RMSE i R^2 metrika pokazali su konzistentnost predikcija kod modela sa period kolonom, dok su modeli bez ove kolone pokazali manju tačnost. Pored toga, rezultati su pokazali da je vremenska komponenta značajan faktor u predikciji očekivanog životnog veka, posebno u modelima koji uključuju vremenske serije.

Generalno, performanse modela su bile osetljive na kvalitet podataka, sa boljim rezultatima kada je dostupna veća količina podataka. Primena naprednih tehnika kao što su vektorizacija i regulacija parametara, značajno su doprinele poboljšanju preciznosti predikcija.

6. ZAKLJUČAK

U radu su istražene metode mašinskog učenja i veštačke inteligencije za predikciju očekivanog životnog veka na osnovu podataka iz World Health Statistics 2020. Kroz primenu modela kao što su Decision Tree, XGBoost, Random Forest, i potpuno povezana neuronska mreža, analizirani su socio-ekonomski i demografski faktori koji utiču na HALE metriku. Rezultati pokazuju da faktori poput obrazovanja, prihoda i zdravstvenog stanja značajno utiču na očekivani životni vek, dok su pol i mesto stanovanja takođe važni prediktori. Primena semi-supervised metoda donela je dodatne uvide, ali postoji mogućnost za dalje unapređenje modela uključivanjem novih podataka, kao što su zdravstvene navike i genetske predispozicije. Optimizacija hiperparametara i napredne tehnike obrade podataka mogu doprineti boljoj generalizaciji modela. Zaključno, ovaj rad naglašava potencijal mašinskog učenja u predikciji zdravstvenih ishoda i pruža osnovu za buduće alate koji će unaprediti donošenje odluka u zdravstvu i javnim politikama.

7. LITERATURA

[1] Wolfson, M.C., 1996. Health-adjusted life expectancy. Health Reports-Statistics Canada, 8, pp.41-45.

[2] Luy, M., Di Giulio, P., Di Lego, V., Lazarevič, P. and Sauerberg, M., 2020. Life expectancy: frequently used, but hardly understood. Gerontology, 66(1), pp.95-104.

[3] Tanha, J., Van Someren, M. and Afsarmanesh, H., 2017. Semi-supervised self-training for decision tree classifiers. International Journal of Machine Learning and Cybernetics, 8, pp.355 370.

[4] Hill, K. and Choi, Y., 2006. Neonatal mortality in the developing world.Demographic research, 14, pp.429-452.

Kratka biografija:

Lenka Isidora Aleksić rođena je 14.10.1999. godine u Zrenjaninu. Smer računarstvo i automatika na Fakultetu tehničkih nauka u Novom Sadu upisala je 2018. godine. Osnovne studije završila je u septembru 2022. godine. Od oktobra 2022, nakon upisa na master studije, kreće redovno da radi na fakultetu kao saradnik u nastavi, a ubrzo i u struci kao *Data* inženjer.

Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.9
DOI: https://doi.org/10.24867/30BE32Zaric

АЛГОРИТАМ ЗА ТРАНСФОРМЕР ЗА ПРОБЛЕМ ПРЕВОЂЕЊА СА ЕНГЛЕСКОГ НА СРПСКИ ЈЕЗИК

IMPLEMENTING TRANSFORMER ALGORITHM FOR THE TRANSLATION PROBLEM FROM ENGLISH TO SERBIAN LANGUAGE

Драган Зарић, *Факултет техничких наука, Нови Сад*

Област − ЕЛЕКТРОТЕХНИКА И РАЧUНАРСТВО

Кратак садржај − *Тема овог рада је имплементирање алгоритма за трансформер за проблем превођења са енглеског на српски језик и формира-ње претренираног модела који се може користити за овај и остале задатке обраде природних језика. Структура која је имплементирана овим радом је предложена у раду [1].*

Кључне речи: *Обрада природних језика, Трансформер, Енкодер-декодер модели, Превођење тек-ста*

Abstract - *This paper topics is the implementation of a transformer algorithm for the problem of translation from English to Serbian language and the formation of a pretrained model that can be used for this and other natural language processing tasks. The structure implemented in this paper was proposed in the paper [1].*

***Keywords*:** *Natural language processing, Transformer, Sequence-to-sequence model, Translation task*

1. УВОД

Обрада природног језика је научна подобласт која користи знање из вештачке интелигенције, информатике и лингвистике. Она се бави способношћу машина да интерпретирата, манипулише и разуме људски језик онако како је написан или изговорен. Данас се за задатке обраде природних језика најче-шће користе претренирани (енг. *pretrained*) модели тј. модели који су претходно обучени над великим скупом података. Затим се такви модели фино подешавају (енг. *fine-tuning*) за специфичан задатак користећи пренесено учење (енг. *transfer learning*). За фино подешавање потребан је мањи број пода-така, самим тим и мање времена за обучавање и мање ресурса се троши. Циљ овог рада је да се кре-ира претрениран модел за превођење са енглеског на српски који касније може бити коришћен за решавање разних проблема обраде природних језика.

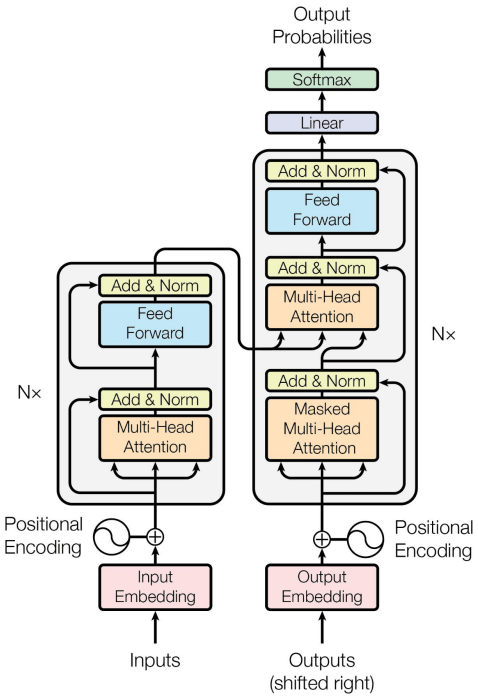
НАПОМЕНА:
Овај рад проистекао је из мастер рада чији је ментор био др Дарко Чапко, ред. проф.

2. СКУП ПОДАТАКА

Први корак у решавању датог проблема је проналажење прикладног скупа података. Трансформе-ри се састоје од великог броја подесивих параме-тара па је за њихово ваљано обучавање потребан велики број подтака којим ће модел бити похрањен приликом обучавања. Након прикупљених сирових података (преузетих са сајта [4]) потребно је кон-вертовати текст у бројеве које модел може да ра-зуме. Тај поступак назива се токенизација. У овом раду је улазни скуп података подељен тако да свака реч чини један токен. Вокабулар који је добијен је садржао 30.000 токена. Поред токена из скупа по-датака додати су токени *[SOS]* за почетак секвенце,*[EOS]* за крај секвенце, *[UNK]* за непознате токене и *[PAD]* токени који не носе никакво значење него служе да свака улазна секвенца буде исте дужине. Осим тога, уз сваку секвенцу приложена је и маска за пажњу (енг. *attention mask*) која се на њу односи. Она говори који део секвенце носи информацију а који не (*[PAD]* токени) [2].

3. ТРАНСФОРМЕР

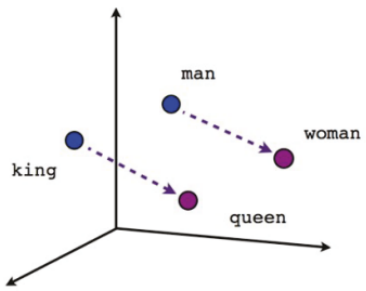
Трансформер је тип модела машинског учења ко-ји се користи за задатке обраде природних језика (енг. *NLP*). Први пут је представљен у раду [1] и од тада се искључиво он користи за решавање про-блема из поменуте области. Ако бисмо посматрали трансформер за проблем превођења, он је као *black box* који на улазу добија реченицу на енглеском језику а на излазу даје адекватан превод те речени-це на српски језик. Унутрашња структура транс-формера се састоји од низа енкодера и декодера. Енкодер служи за претварање улазних података у нумеричку репрезентацију који модел може да ра-зуме а декодер из тих нумеричких репрезентација добија излаз односно преведену реченицу. Унутар сваког енкодера и декодера се налазе механизам па-жње (енг. *attention mechanism*) и неуронска мрежа са пропагацијом сигнала унапред (енг. *feed forward neural network*). Механизам пажње служи да модел схвати контекст и однос између речи из улазне се-квенце а неуронска мрежа за генерално разумевање реченице.



Слика 1: *Структура трансформера [1]*

3.1. Уградња речи (енг. *word embedding*)

Уградња речи је кључна техника за обраду природ-ног језика јер омогућава моделима да претворе дис-кретне језичке ентитете (као што су речи и токени) у континуиране векторске репрезентације које но-се богате информације о значењу и контексту речи. Речи се представљају као вектори у високодимензи-оналном простору. Идеја је да се ухвати семантичко значење речи, тако да речи са сличним значењима буду ближе једна другој у векторском простору. То омогућава да се речи третирају као нумерички вектори, што омогућава извођење математичких опе-рација над њима, на пример њихово поређење преко косинусне сличности или неке друге метрике. Када представимо речи у простору можемо и визуелно тумачити њихово значење и однос неких речи. Са слике 2 можемо закључити да је разлика између ре-чи *"краљ"* и *"краљица"* слична разлици између речи *"мушкарац"* и *"жена"*.



Слика 2: *Визуелно репрепрезентоване речи се могу тумачити [5]*

Уграђивање токена је метода која се користи при-ликом рада са трансформерима: за сваки токен се у табели везује *embedding* вектор који представља ње-гово значење. Димензија *embedding* вектора се на-зива d_{model} и може имати вредности: 256, 512, 768, 1024, 1536, 3072 итд. Већа димензионалност најче-шће значи бољи квалитет енкодовања, али и дуже тренирање. У овом раду је $d_{model} = 512$.

3.1.1. *Positional embedding*

Много о значењу неке речи можемо закључити на основу њеног односа са другим речима у оквиру јед-не реченице. Позициона уградња служи да модел не види улазни вектор као само колекцију токена већ колекцију токена који су у неком односу тј. редосле-ду. Сваком токену у улазној секвенци се придружу-је додатни *positional embedding* вектор који садржи информацију о раздаљини токена са сваким дру-гим токеном. Тај вектор је фиксан, једном се изра-чуна и користи се такав приликом тренира- ња и закључивања и исти је за сваку реченицу. С друге стране, *embedding* вектор се мења приликом трени-рања. Да се не би памтила два вектора, positional embedding и input embedding вектори се сабирају и такви улазе у модел (што значи да морају би-ти истих димензија тј. d_{model}). С обзиром на то да се *positional embedding* вектор не мења он се може посматрати као пристрасност (енг. *bias*). У изради овог рада је коришћена следећа формула [1] за рачунање *positional embedding* вектора:

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right),$$

где је *pos* позиција, а *i* је димензија. Свака ди-мензија позиционог кодирања одговара синусоид-ној функцији. Дакле, коначни улаз у трансфор-мер представља збир *positional embedding* вектора и *input embedding* вектора.

3.2. Механизам пажње (*attention layer*)

3.2.1. *Single-head attention*

Механизам пажње служи за разумевање улазне се-квенце тј. контекст токена у односу на све остале токене и омогућава моделу да се фокусира на ра-зличите делове улазне секвенце док обрађује подат-ке. У случају превођења са једног језика на други битно је схватити које речи су релевантне у извор-ној и преведеној реченици, то ради тако што се фо-кусира на различите токене унутар сваке секвенце додељујући им различите тежине на основу њихо-ве релевантности и важности за тренутни задатак. Пажња се израчунава по следећој формули [1]

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

где су Q, K и V вектори упита, кључа и вредно-сти који се добијају множењем улазног вектора са матрицама упита, кључа и вредности чији се пара-метри добијају током тренинга. *Softmax* функцију користимо јер на излазу даје вредности у опсегу (0,1) док је сума свих вредности 1. Захваљујући то-ме, излаз из механизма пажње можемо посматрати

као вероватноћу да је сваки од токена тачан излаз.

3.2.2. *Multi-head attention*

Уколико би механизам пажње рачунао само је-дан контекст за сваку реч постоји велика шанса да би погрешно интерпретирао реченицу. *Multihead attention* дели *embedding* вектор на одређени број *"глава"* и за сваки део вектора посебно рачуна ме-ханизам пажње и на крају те векторе конкатени-ра. Свака *"глава"* садржи читаву улазну реченицу (скуп токена) али гледа другачији аспект сваког то-кена. *Multihead attention* се рачуна по формули [1]

$$MultiHead(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = Concat(head_1, \ldots, head_\mathrm{h})\mathrm{W}^{\mathrm{O}},$$

где је

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V).$$

3.3. *Feed forward neural netword*

Механизам пажње омогућава моделу да се фокуси-ра на релевантне делове секвенце, али сама пажња не обрађује детаљно све информације које је доби-ла. Након ње долази *FFN* која омогућава транс-формеру да разуме сложене односе између подата-ка. *FFN* примењује нелинеарне функције на сваку позицију у секвенци независно. На тај начин, свакој позицији у секвенци даје богатију и комплекснију репрезентацију, што помаже у бољем разумевању целокупног текста. Састоји од две линеарне транс-формације са *ReLU* активационом функцијом из-међу [1]

$$FFN(\mathrm{x}) = max(0, \mathrm{x}\mathrm{W}_1 + \mathrm{b}_1)\mathrm{W}_2 + \mathrm{b}_2.$$

3.4. *Layer normalization*

Након сваког прорачунавања потребно је вршити нормализацију да се све вредности улазног вектора ограниче на опсег од 0 до 1. Додатно, постоје па-раметри γ и β за скалирање и померање тог опсега који служе да појачавају и смањују вредности ради што бољег предвиђања. Поменути параметери нису фиксни и уче се током обуке. Следећа формула [3] која се примењује на свим векторима

$$LayerNorm(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2,$$

где су: x улазни вектор, μ средња вредност улаза, σ^2 варијанса улаза, ϵ мала константа додата ради нумеричке стабилности, γ и β параметри који се уче. Укупан број подесивих параметара модела је 35 милиона.

3.5. Енкодер

Енкодер на улазу добија вектор који представља збир позиционог и улазног уграђивања, затим ти вектори пролазе кроз слојеве енкодера. Сваки енкодер саджи слој пажње и *Feed Forward* мрежу. Излаз сваког слоја енкодера је улаз у следећи слој енкоде-ра осим последњег. Излаз последњег слоја енкодера је улаз у сваки од декодер слојева и представља век-торе упита и кључа приликом њиховог подешавања тежина. Главна улога енкодера је разумевање улазних се-квенци и њихово претварање у апстрактнију пред-ставу која ће бити прослеђења декодеру или сама генерисати предикцију.

3.6. Декодер

Декодер служи за генерисање излаза. Док енкодер прима текст на одређеном језику (у нашем случају енглеском), декодер прима одговарајучу секвенцу на жељеном језику (у нашем случају српском). Обу-чавање декодера се своди на то да се предвиђа сле-дећи токен у односу на улазни. Механизам пажње током обуке може приступити само речима позици-онираним пре у реченици. Након првог механизма пажње, у декодеру постоји и унакрсни механизам пажње (енг. *cross attention*) где су вектори упита и кључа излаз из енкодера а вектор вредности се добија из механизма пажње самог декодера. При-ликом прослеђивања одређене реченице коју жели-мо да преведемо излаз из енкодера се рачуна само у првој итерацији јер се улазна секвенца није про-менила. Улаз у декодер се мења односно додаје се токен по токен, почевши од *[SOS]*. Декодер пред-виђа следећи токен у свакој итерацији. Када се на улазу нађе последња реч из секвенце поступак је завршен.

3.7. Слој пројекције и закључивање

Слој пројекције (енг. *projection layer*) служи да вектор, који је излаз из декодера, преведе у конкретне излазне токене (слова, подречи или речи). Након тога, вектори пролазе кроз *softmax* функцију која служи да се излаз из слоја пројекције (енг. *logit*) мапира на вероватноће да је неки од вектора сле-дећи који треба да буде изабран.

Након добијених вероватноћа за сваки токен из во-кабулара потребно је одредити који ће токен би-ти генерисан узимајући онај са највећом верова-тноћом или неку другу технику. Изабрани токен се детокенизује у стварну реч или карактер и добија-мо читљив текст на излазу. Излаз из декодера који је прошао кроз слој пројекције се пореди за жеље-ним излазом (*label*) и рачуна се колико је модел до-бро предвидио користећи одређену метрику (нпр. функцију губитка односно *loss funtion*). Затим се пропагирају промене тежина уназад и процес се по-навља све док се не обради читав скуп података. За ажурирање параметара користи се *Adam* алго-ритам. Због променљивог корака учења који се при-лагођава градијенту и корекције бајаса овај алго-ритам је стабилан и брзо конвергира. Уз то, *Adam* треба да памти само два покретна просека по пара-метру што га чини ефикасним за меморију, што је

посебно важно за велике неуронске мреже и велики број података.

4. АЛГОРИТАМ

На слици 4 је приказан псеудокод за обучавање трансформера.

```
begin
    улазни скуп података поделити на серије реченица
    иницијализуј модел насумичним параметрима
    for(i = 1 to број епоха)
        for(i = 1 to број серија реченица)
            израчунај излаз енкодера
            for(i = 1 to крај серије)
                предвиди следећи токен рачунањем излаза декодера
            израчунај губитак
            ажурирај параметаре коришћењем Адамовог алгоритма
        покрени валидацију модела
        сачувај тежине модела
end
```

Слика 3: *Псеудокод тренинг петље*

5. РЕЗУЛТАТИ

На сликама 4-8 су приказани резултати тестирања. Краће секвенце (слика 4) је модел успешно превео. У дужим секвенцама (слике 5-8) модел је грешио лица, падеже а непознате речи није попунио.

```
    SOURCE: A friend called me.
    TARGET: - Prijatelj me pozvao.
 PREDICTED: - Prijatelj me pozvao .
```

Слика 4: *Једноставне секвенце лако преводи*

```
   SOURCE: Hey,you Hey,you won what?
   TARGET: Hey,ti hey,šta si pobijedio?
PREDICTED: Hej , kako si li ti uradio ?
```

Слика 5: *Добар контекст али неадекватан превод*

```
   SOURCE: - Where else can they get in here?
   TARGET: Gde još mogu da uđu ovde?
PREDICTED: Gde bi da bi mogao ovde ?
```

Слика 6: *Добар контекст али неадекватан превод*

```
   SOURCE: This night is a night full of terror
   TARGET: Noæas æemo otpoèeti "noæ terora"!
PREDICTED: Noæas æemo " noæ terora "!
```

Слика 7: *Лош скуп ће резултирати лошим предикцијама*

```
    SOURCE: Listen, I love you, baby.
    TARGET: Slušaj, volim te, dušo.
 PREDICTED: Slušaj , volim ti , dušo .
```

Слика 8: *Погрешна лица и падежи неких речи*

6. ЗАКЉУЧАК

У овом раду је имплементиран алгоритам трансформера предложен у раду [1]. Добијен је претрениран модел за превођење са енглеског на српски језик. Тренирање трансформер модела са 35 мили-она параметара захтева значајну количину времена и ресурса. Како би се постигла задовољавајућа тач-ност потребно је користити неколико стотина хиља-да до милион парова реченица за тренинг. Типич-но, моделима ове величине треба између 10 и 50 епоха како би достигли стабилну тачност, уз напо-мену да је потребно пажљиво пратити валидациони скуп како би се избегло преобучавање. Време тре-нирања зависи од хардверских ресурса, попут гра-фичке картице и броја *CPU* језгара, али у просеку би тренирање једног модела са 35 милиона параме-тара траје око недељу дана на снажнијој *GPU* кон-фигурацији. На *CPU* ресурсима, овај процес траје знатно дуже, од неколико недеља до месеци, у за-висности од расположиве снаге.

С обзиром на то да је овај модел трениран над 10 хиљада реченица 1 дан и трансформер је успео да ухвати значење и контекст речи можемо закључити да је ово моћна структура која се може користити за разне проблеме обраде природних језика. Надоградња овог рада ће бити *„чишћење”* скупа по-датака и тренинрање модела над милион реченица.

Литература

[1] A. Vaswani, N. Shazeer and others, *Attention is All You Need*, Advances in Neural Information Processing Systems, 2017, pp. 5998-6008. https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper. pdf

[2] Daniel Jurafsky and James H. Martin, *Speech and Language Processing*, An Introduction to Natural Language Processing, 3rd edition, 2024 https://web.stanford.edu/~jurafsky/slp3/

[3] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, *Layer Normalization*, 2016 https://arxiv.org/abs/1607.06450

[4] https://data.statmt.org/opus-100-corpus/v1.0/supervised/en-sr/

[5] Слика преузета са: https://www.brainlabsdigital.com/blog/what-word2vec-means-for-seo/

Кратка биографија:

Драган Зарић рођен је у Ужицу 2000. године. Мастер рад на Факулте-ту техничких наука из области Елек-тротехнике и рачунарства одбранио је 2024. године.
Контакт: zaricdragan00@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/30BE33Kovacevic

UTICAJ MATERIJALA NA UNAPREĐENJE PERFORMANSI KAVEZNIH ASINHRONIH MAŠINA
MATERIALS INFLUENCE ON IMPROVEMENT OF SQUIRREL-CAGE INDUCTION MACHINES PERFORMANCE

Aleksandra Kovačević, Dejan Jerkan, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj *U radu je izložena metoda konačnih elemenata i mogućnosti njene primene u analizi električnih mašina. Predstavljen je model trofaznog kaveznog asinhronog motora izrađenog pomoću softverskog alata koji implementira metodu konačnih elemenata. Analizirane su statičke i dinamičke karakteristike, u smislu uticaja materijala upotreljenih za izradu rotorskog kaveza. Fokus je na upoređivanju perfomansi dve izvedbe mašine, kod kojih su upotrebljeni različiti materijali za izradu kaveznog namotaja (bakar i aluminijum).*

Ključne reči: *Asinhroni motor, Metoda konačnih elemenata, Kavezni rotor, Bakar, Alumijum*

Abstract *This paper presents the finite element method and the possibilities of its application in the analysis of electrical machines. A model of a three-phase squirrel-cage induction motor made using a software tool that implements the finite element method is presented. Static and dynamic characteristics were analyzed, in terms of the influence of the materials used to make the rotor cage. The focus is on comparing the performance of two versions of the machine, in which different materials were used to make the cage winding (copper and aluminum).*

Keywords: *Induction motor, Finite element method, Squirrel cage, Copper, Aluminium*

1. UVOD

Električne mašine su ključni elementi u svim segmentima elektroenergetskog sistema (EES): proizvodnji, prenosu, distribuciji i potrošnji električne energije. Zbog svoje jednostavnosti i ekonomičnosti, robusnosti i sigurnosti, niske potrebe za održavanjem i otpornosti na preopterećenje, asinhroni motori preuzimaju veliki udeo kao pogonski motori, a danas se u industriji, poljoprivredi, domaćinstivima i sl, pojavljuju u više od 90% slučajeva. konstantno se radi na usavršavanju asinhronih motora. Zbog široke primene, konstrukcija i izbor materijala rotora asinhronih mašina, utiču na više faktora od kojih je najizraženiji cena, a sama konstrukcija zavisna je od uslova pod kojim motor radi, vrste pogona, načinu pokretanja motora, nominlanih parametara.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dejan Jerkan, vanr.prof.

2. METODA KONAČNIH ELEMENATA

Metoda konačnih elemenata je izvedena iz opšte teorije elektromagnetnih polja, uvođenjem određenih pomoćnih vektorskih i električnih veličina koje su poznate pod nazivom potencijali elektromagnetnog polja. Strogo uzevši, potencijali elektromagnetskih polja se analitički mogu uvesti i izraziti jedino u homogenim sredinama, ali je upotrebom numeričkih metoda mogućnost njihove primene proširena na praktično sve oblasti proučavanja makroskopskih polja, koja se vrlo često uspostavljaju u sredinama koje se ne mogu smatrati niti homogenim, niti linearnim, a vrlo često ni izotropnim. Za izradu magnetskih kola električnih mašina se koriste nelinearni feromagnetski materijali, a osnovi razlog za njihovu primenu je velika relativna permeabilnost (permeabilnost feromagnetskih materijala je izuzetno složena veličina, koja je zavisna od intenziteta magnetskog polja, brzine promene izvora polja, a u slučaju anizotropnih materijala i pravca i smera magnetnog polja).

Za opisivanje elektromagnetnih polja u sredinama proizvoljnih karakteristika i proizvoljne konfiguracije izvora koriste se Maksvelove jednačine i jednačina kontinuiteta u integralnom obliku:

$$\oint_l \vec{E} \cdot \overrightarrow{dl} = -\frac{\partial}{\partial t}\int_S \vec{B} \cdot \overrightarrow{dS} \tag{2.1}$$

$$\oint_l \vec{H} \cdot \overrightarrow{dl} = \int_S \left(\vec{J}_{izv} + \vec{J} + \frac{\partial \vec{D}}{\partial t}\right) \cdot \overrightarrow{dS} \tag{2.2}$$

$$\oint_S \vec{D} \cdot \overrightarrow{dS} = \int_v \rho dv \tag{2.3}$$

$$\oint_S \vec{B} \cdot \overrightarrow{dS} = 0 \tag{2.4}$$

$$\oint_S \vec{J} \cdot \overrightarrow{dS} = -\frac{\partial}{\partial t}\int_v \rho dv \tag{2.5}$$

Primenom operatora prostornog diferenciranja na date Maksvelove jednačine u integralnom obliku (uz primenu poznatih teorema vektorske analize, kao što su Stoksova i teorema Gaus-Ostrogradskog, dolazi se do njihovog diferencijalnog oblika, datog relacijama (2.6)-(2.9).

$$\boldsymbol{rot\vec{E} = -\frac{\partial \vec{B}}{\partial x}} \quad (2.6)$$

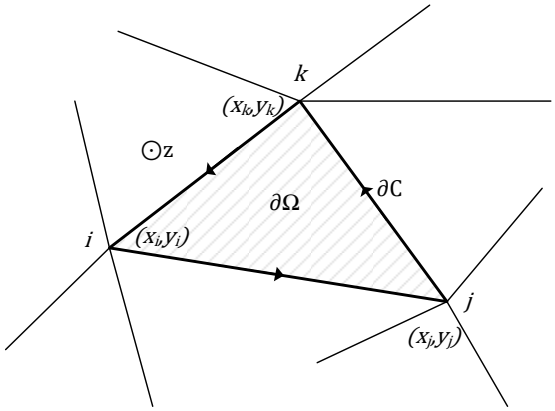
$$\boldsymbol{rot\vec{H} = \vec{J}_{izv} + \vec{J}} \quad (2.7)$$

$$\boldsymbol{div\vec{B} = 0} \quad (2.8)$$

$$\boldsymbol{div\vec{J} = 0} \quad (2.9)$$

2.1 Formulacija metode konačnih elemenata u dvodimenzionalnom domenu

Domen integracije predstavlja površ jednog od mnoštva konačnih elemenata na koje je izdeljena cela oblast od interesa. Pošto je oblast dvodimenzionalna, konačni elementi mogu biti mnogouglovi, ali je najčešći slučaj da se usvajaju segmenti trougaonog oblika. Umesto kontinualnog problema, metoda konačnih elemenata ima za cilj rešavanje magnetskog vektor potencijala u temenima, odnosno čvorovima mreže trougaonih segmenata, dok se vrednost potencijala u proizvoljnim tačkama unutar segmenta dobija interpolacijom vrednosti potencijala u čvorovima. Ako se oblast izdeli na konačne elemente dovoljno gusto, tako da se ne gubi značajno na preciznosti metode, često je dovoljno usvojiti da se vrednost potencijala unutar segmenta dobije linearnom interpolacijom potencijala čvorova**.**

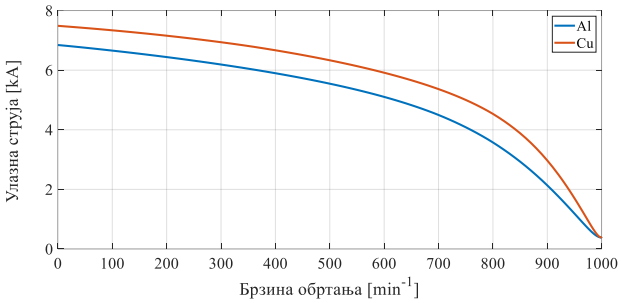


Slika 2.1 *Konačni element u obliku trougaonog segmenta*

3. UTICAJ MATERIJALA PROVODNIKA ROTORA NA PERFORMANSE MOTORA

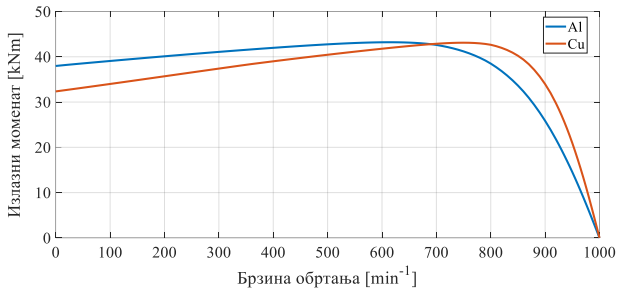
Kod asinhronih motora sa kaveznim rotorom postoji važan izbor između upotrebe jeftinijeg, kaveza izrađenog od aluminijuma u odnosu na skuplji kavez u vidu bakarnih štapova. Korišćenje neadekvatne konstrukcije rotora može nepotrebno povećati gubitke, odnosno troškove ili dovesti do kvara samog asinhronog motora.

Razmatra se trofazni šestopolni asinhroni motor, čiji je model izrađen pomoću softverskog paketa koji implementira metodu konačnih elemenata. Izlazna snaga motora iznosi $1250W$, nazivnog napona $500V$, čiji su namotaji spregnuti u trougao. Data brzina obrtanja iznosi $992\ o/min$, nominalna učestanost $50Hz$. Opterećenje je konstante snage, a radna temperatura $75℃$.



Slika 3.1 *Statička karakteristika ulazne struje motora*

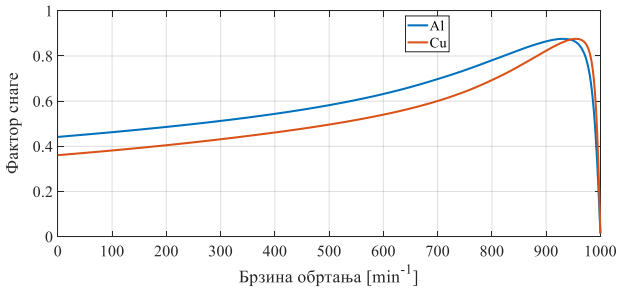
Na slici 3.1 prikazana je statička karakteristika ulazne struje motora, odakle se jasno može uočiti da je struja motora sa kaveznim rotorom izrađenog od bakarnih štapova veća u odnosu na struju motora sa kavezom izrađenog od aluminijumskih šipki. Razlog leži u znatno većoj otpornosti alumijuma u odnosu na bakar.



Slika 3.2 *Statička karakteristika izlaznog momenta motora*

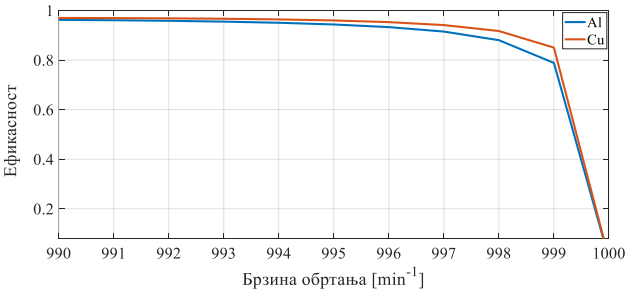
Na slici 3.2 prikazana je statička karakteristika izlaznog momenta motora, koja se može analizirati iz dve perspektive. Najpre, na prelaznom delu, kako polazni momenat zavisi od otpora, polazni momenat motora sa aluminijumskim kavezom je veći u odnosu na polazni momenat motora sa kavezom izrađenog od bakarnih štapova, što čini razliku ali ne i presudnu s obzirom da se danas mašine ne pokreću toliko često direktno iz mreže. Dakle, u pogledu izlaznog momenta u prelaznom delu, prednost bakra u odnosu na aluminijum nije ekskluzivna. Međutim, na linearnom delu karakteristike, razlike su veće i značajnije. Naime, prevalno klizanje motora sa aluminijumskim kavezom je veće od prevalnog klizanja motora sa kavezom izrađenog od bakarnih štapova. Dakle, korišćenjem bakarnih štapova prilikom izrade rotora postižu se manji gubici, kako zbog manje vrednosti klizanja, tako i zbog samih osobina bakra.

Na slici 3.3 prikazana je zavisnost faktora snage od brzine obrtanja prilikom korišćenja aluminijuma, odnosno bakra kao materijala od kog su izrađeni štapovi rotora.



Slika 3.3 *Zavisnost faktora snage motora u odnosu na brzinu obrtanja*

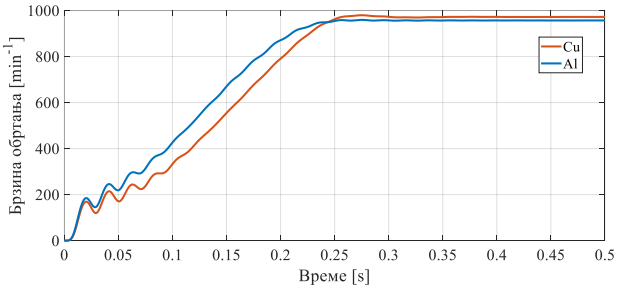
Prilikom pokretanja, faktor snage je bolji kod aluminijumskog, ali nema značajnih razlika u odnosu na rotor izrađen od bakarnih štapova.



Slika 3.4 *Zavisnost efikasnosti od brzine obrtanja*

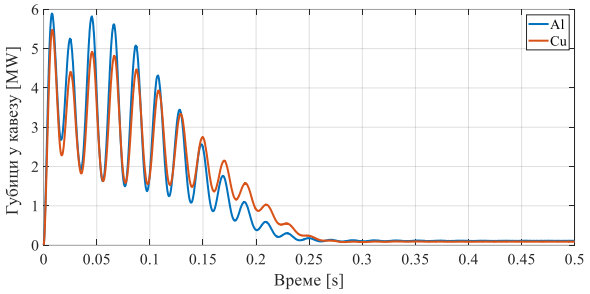
Na slici 3.4 prikazana je zavisnost efikasnosti motora od brzine, pri čemu se značajnije razlike mogu uočiti u poslednjim tačkama analiziranog intervala, gde se primećuje povećanje efikasnosti korišćenjem bakarnih štapova.

Na slikama datim u nastavku biće predstavljena uporedna analiza dinamičkih karakteristika, za rotor sa alumijumskim i za rotor sa bakarnim štapovima.



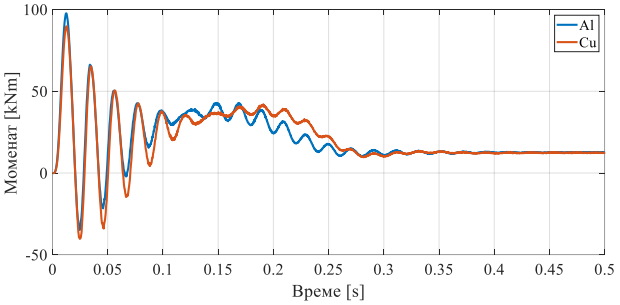
Slika 3.5 *Promena brzine tokom vremena*

Na slici 3.5 prikazana je promena brzine obrtanja motora tokom prve polovine sekunde simulacije. Sa datog grafika očitana je prevalna brzina za bakarnu izvedbu $n_{pr} = 978{,}9\ min^{-1}$, koja je veća za oko 2% od prevalne brzine koju ova mašina dostiže prilikom izrade rotorskog kaveza od aluminijuma, iz čega proističe da je prevalno klizanje u slučaju upotrebe aluminijuma duplo veće od prevalnog klizanja u slučaju upotrebe bakra kao materijala od kog je izrađen rotorski kavez, što je u saglasnosti sa činjenicom da je otpornost aluminijuma značajno veća od otpornosti bakra.

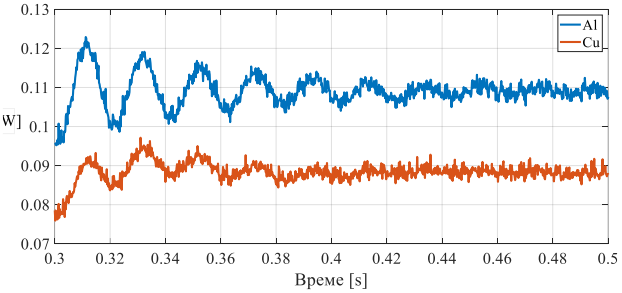


Slika 3.6 *Promena gubitaka u kavezu tokom vremena*

Na slici 3.6 predstavljena je promena snage gubitaka u kavezu tokom prvih 0,5s, sa izdvojenim detaljem ustaljenog perioda između 0,3s i 0,5s. Uočava se da su gubici u kavezu rotora veći prilikom korišćenja aluminijuma, kako tokom prelaznog perioda, tako i u ustaljenom stanju, odakle se očitava razlika između gubitaka u slučaju upotrebe ova dva materijala i iznosi oko 20,5kW na stranu aluminijuma. Dakle, prilikom odabira bakra inicijalni troškovi jesu veći, ali dugoročno gledano oni su manji zbog značajno manje snage gubitaka koja se ostvaruje u kavezu rotora prilikom korišćenja bakra, što ide u prilog energetskoj efikasnosti koju treba ostvariti.



Slika 3.7 *Promena momenta tokom vremena*



Slika 3.8 *Promena momenta tokom vremena – ustaljeni period*

Na slici 3.7 i slici 3.8 prikazane su promene momenta tokom vremena. Uočavaju se odstupanja koja postoje u dinamičkom periodu zbog odnosa otpornosti i reaktanse koji se razlikuje. U saglasnosti sa statičkom karakteristikom, prilikom upotrebe aluminijuma se prvo dostiže prevalni momenat, dok je većim delom izlazni momenat bakarne izvedbe veći u odnosu na momenat koji

se postiže upotrebom aluminijumskog kaveza, što se slaže sa činjenicom da se povećavanjem otpora rotorskog namotaja, izlazni momenat smanjuje.

4. ZAKLJUČAK

U radu je analiziran trofazni jednokavezni asinhroni motor čiji je model izrađen upotrebom softvera koji implementira metodu konačnih elemenata. U prvoj varijaciji, rotor kaveza konstruisan je od alumijumskih štapova, a u drugoj od bakarnih. Prilikom uporedne analize dobijenih karakteristika, zakljujčuje se da bakarni kavez, iako je skuplji pruža unapređenje perfomansi u pogledu većeg polaznog momenta. Kako je prevalno klizanje motora sa aluminijumskim kavezom veće od prevalnog klizanja motora sa kavezom izrađenog od bakarnih štapova zaključuje se da se korišćenjem bakarnih štapova prilikom izrade rotora postižu manji gubici, kako zbog manje vrednosti klizanja, tako i zbog samih osobina bakra. Na osnovu dobijenih rezultata i urađene analize, zaključuje se da u realizacijama gde je efikasnost ključna, bakar zbog manje snage gubitaka predstavlja bolji izbor. Zbog veće provodljivosti, izlazni momenat je veći u bakarnoj nego aluminijumskoj izvedbi. Bakar može bolje podneti veće struje bez pregrevanja zbog svoje bolje provodljivosti što je naročito bitno u aplikacijama sa visokim startnim ili radnim trenutkom. Takođe, štapovi izrađeni od bakra mogu izdržati više temperature pre nego što dođe do značajnog gubitka provodljivosti, što je ključno u uslovima visokih opterećenja

5. LITERATURA

[1] Slobodan N. Vukosavić, *„Električne mašine“*, Beograd, 2010.

[2] Tihomir Latinović, Miroslav Prša, *„Osnovi elektrotehnike i električnih mašina“*, Banja Luka, 2013.

[3] Ion Boldea, Syed A. Nasar, *„The Induction Machines Design Handbook“-second edition*, 2002.

[4] Juha Pyrhonen, Tapani Jokinen, Valeria Hrabovcova, *„Design of Rotating Electrical Machines“*, 2008.

[5] Austin Hughes, *„Electric Motors and Drives“-third edition*, 1993.

[6] Veran Vasić, Đura Oros, *„Uvod u električne mašine-skripta“*, Novi Sad, 2018.

[7] Jožef Varga, *„Analitički metod za sračunavanje karakteristika trofaznih asinhronih motora“*, Subotica.

[8] Jožef Varga, *„Električne mašine II“*, Subotica, 2006.

[9] *„Niskonaponski asinhroni motori“ – Sever-katalog*, Subotica, 1979.

Kratka biografija:

Aleksandra Kovačević rođena je u Kikindi 2001. godine. Diplomirala je na Fakultetu tehničkih nauka u Novom Sadu 2023. godine. Master rad iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine odbranila je 2024. godine.

kontakt:
aleksandra.kovacevic02@gmail.com

Dejan Jerkan je vanredni profesor na Fakultetu tehničkih nauka u Novom Sadu, na Katedri za Energetsku elektroniku i pretvarače. Oblast interesovanja su mu modelovanje i dijagnostika električnih mašina, kao i metoda konačnih elemenata.

Зборник радова Факултета техничких наука, Нови Сад

UDK: 4.7
DOI: https://doi.org/10.24867/30BE34Pruginic

АНАЛИЗА ВЕРОВАТНОЋЕ ГРЕШКЕ ПАКЕТА КОД IEEE 802.11p СТАНДАРДА

ANALYSIS OF PACKET ERROR PROBABILITY IN THE IEEE 802.11p STANDARD

Милица Пругинић, *Факултет техничких наука, Нови Сад*

Област – ЕНЕРГЕТИКА, ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ

Кратак садржај – *Тема рада брави се проучавањем бежичних комуникација, са акцентом на IEEE 802.11p стандард, његову примену и утицај који остварује у домену аутомобилских комуникација. Саобраћајну инфраструктуру чини више различитих типова сервиса који су уједињени у систем Vehicle to Everything (V2X). У MATLAB софтверском окружењу, са главним елементима који чине комуникациони систем – предајник, комуникациони канал и пријемник, остварена је практична примена на верзији софтвера R2024b. Главни мотив рада је да прикаже симулацију мерења стопе грешака пакета поређењем параметара SNR (Signal to noise ratio) и PER (Packet Error Rate).*

Кључне речи: Бежичне комуникације, *V2X, IEEE 802.11p, MATLAB, SNR, PER*

Abstract – *The subject of the study of Wireless Communications, with an emphasis on the IEEE 802.11p standard, its application and impact in the field of automotive communications. The traffic infrastructure consists of several different types of services that are united in the Vehicle to Everything (V2X) system. In the MATLAB software environment, with the main elements that include in the communication system – transmitter, communication channel and receiver. A practical application was realized on software version R2024b. The main motive of the thesis is to show the simulation of packet error rate measurement by comparing the SNR (Signal to noise ratio) and PER (Packet Error Rate) parameters.*

Keywords: *Wireless Communications, V2X, IEEE 802.11p, MATLAB, SNR, PER*

1. Увод

Стандард бежичних технологија *IEEE 802.11p* представља кључну технологију која је омогућила раст и напредак у области аутомобилских комуникација. Поменути стандард је развијен као део *IEEE 802.11* стандарда, чија је улога да допринесе стварању интелигентних транспортних система.

НАПОМЕНА:
Овај рад проистакао је из мастер рада чији ментор је била др Милица Петковић, доцент.

IEEE 802.11p стандард ради у фреквенцијском опсегу који је резервисан за аутомобилске комуникације. Само неке од предности *IEEE 802.11p* стандарда су висока поузданост у преносу података, као и брза размена прецизних информација. Главни циљ поменутог стандарда је да унапреди свакодневну вожњу и да пружи подршку са аспекта безбедности у саобраћају, без обзира на променљиво саобраћајно окружење.

2. ГЛАВНИ ЕЛЕМЕНТИ КОМУНИКАЦИОНОГ СИСТЕМА

Главни елементи који чине сваки комуникациони систем су предајник, комуникациони канал и пријемник. Протокол је основно средство за размену података и контролних информација у слојевитим архитектурама комуникационих система. За потребе размењивања информација од значаја користе се протоколи који укључују главне моделе јединица као што су физички (*PHY*) и *MAC* слој. Информације, односно, кориснички подаци се преносе између слојева у облику који је погодан за пренос, а то су сервисне јединице података (*Service Data Unit - SDU*). Облик који је погодан за пренос је *PHY SDU*, односно *PSDU***.** У оквиру јединичних модела размена података се одвија преко протоколских јединица података (*Protocol Data Units - PDU*). У *MAC* моделу се размењује *MAC PDU*, или познат као *MPDU,* са својим модулом. У оквиру *PHY* модела се размењује *PHY PDU* са својом јединицом [1].

Предајник учествује у преносу информација од значаја, контролних информација и информација које су потребне за обраду података на физичком слоју. Предајник конвертује сигнал у облик који је погодан за даљи пренос кроз комуникациони канал. Идентификација грешака у преносу се врши на начин поређења послатих *PSDU* са примљеним *PSDU.*

Комуникациони канал чине *Vehicle to Vehicle* (*V2V) Channel* и *AWGN (Additive white Gaussian noise).* Комуникациони канал за потребе комуникације између возила (*V2V Channel*) омогућава директну бежичну комуникацију потребама интелигентног транспортног система (*Intelligent transportation system - ITS*). Деле се информације о брзини, положају и променама у правцу кретања возила. Процењују се перформансе комуникације између возила приликом вожње у различитим условима. Комуникациони канал је виртуелна путања за бежичну комуникацију, која за потребе симулације, има улогу у слању пакета између возила. Главни циљ је анализа стопе грешака (*Packer*

Error Rate - PER) у различитим тачкама односа снаге корисног сигнала и шума. У комуникационом каналу се додаје адитивни бели Гаусов шум који има улогу да симулира ефекат шума за потребе симулације. Адитивни бели Гаусов шум је математички модел који утиче на анализу утицаја различитих нивоа шума, у пракси, код бежичних канала као медијума преноса података. Анализира се квалитет преноса и процењују се перформансе док се комуникација одвија у реалним условима.

Пријемник је у овом случају најсложенији саставни елемент комуникационог система. Он сам по себи ствара јединствени систем који укључује неколико блокова.

Први, обавезни блок у систему пријемника је *Packet Detect and Coarse Frequency Correction***,** односно блок који је задужен за детекцију пакета и грубу корекцију фреквенције. Овај блок има задатак да детектује почетак преноса и крај преноса пакета који се преноси што директно омогућује пријемнику да зна када може да очекује пријем информација од значаја. Његова улога је да исправи грубе промене које се јављају током преноса у фреквенцији сигнала [2].
Други, обавезни блок у систему пријемника је *Timing Synchronization and Fine Frequency Correction,* односно блок који је задужен за синхронизацију времена и фину корекцију фреквенције. Овај блок има задатак да обради примљени сигнал и да изврши тачну синхронизацију у временском домену, што укључује правилну интерпретацију временских интервала између суседних симбола. Главни продукт овог блока је прецизна корекција примљеног сигнала [2].

Наредни, обавезни блок у систему пријемника је *L-LTF Demodulator*. Улога поменутог блока је да изврши процес демодулације чиме ће издвојити корисне информације и осигурати тачну синхронизацију за даљи пренос. Такође, *L-LTF Demodulator* је блок на пријемној страни који има улогу да оптимизује пријемник чиме се доприноси поузданом и ефикасном пријему [2].

Наредни, обавезни блок у систему пријемника о коме ће бити речи је *NonHT Data Demodulation.* Главни задатак овог блока је процес обраде специфичних података који су одређени захтевом за *NonHT (Non High Throughput)* модулациони режим [2].

Наредни, обавезни блок у систему пријемника о коме ће бити речи је *NonHT Data Phase Tracking* који доприноси поузданој комуникацији и важан је елемент када је реч о бежичним аутомобилским комуникацијама, веома високе мобилности. Овај блок има задатак да ризик од губитка синхронизације сведе на минимум тако што прати фазу и ради на одржавању тачног временског интервала током процеса пријема сигнала на пријемној страни [2].

Наредни, обавезни блок у систему пријемника о коме ће бити речи је *Decision Directed Channel Tracking* који има задатак да одржи оптималну везу између предајне и пријемне стране. Поменути, обавезни блок прилагођава параметре канала у складу са условима у којима се одвија аутомобилска комуникација. Неки од ризичних сценарија који имају утицаја на квалитет комуникације су препреке које се јављају током комуникације као и временски или било који други амбијентални услови. Такође, сам положај возила значајно утиче на то каквог квалитета ће бити комуникација [2].

Наредни, обавезни блок у систему пријемника о коме ће бити речи је *NonHT Data Equalization,* чија је примарна улога да смањи утицај изобличења која су настала услед дејства карактеристика бежичног канала. Овај блок има задатак да оптимизује перформансе пријемника и да допринесе бољем квалитету примљеног сигнала [2].

Наредни, обавезни блок у систему пријемника о коме ће бити речи је *NonHT Data Decoding.* Блок *NonHT Data Decoding* има улогу декодовања и веома је важан блок за ефикасан рад пријемника. Поменути блок нам обезбеђује прецизне информације када је реч о стању система и пружа нам увид у команде и поруке које су саставни део комуникације. На излазу последњег блока, у систему пријемника, налази се *PSDU* [2].

3. СИМУЛАЦИЈА МЕРЕЊА СТОПЕ ГРЕШАКА ПАКЕТА У АУТОМОБИЛСКИМ КОМУНИКАЦИЈАМА У MATLAB СОФТВЕРСКОМ ОКРУЖЕЊУ

Симулација мерења стопе грешака пакета у аутомобилским комуникацијама у *MATLAB* софтверском окружењу је изведена у лабораторијским мерењима на верзији софтвера *R2024b*. Како је *MATLAB* сам по себи веома сложено софтверско окружење, пружа нам мноштво софтверских могућности и приказа. Симулација мерења стопе грешака пакета у аутомобилским комуникацијама, у *MATLAB* софтверском окружењу, може да се реализује имплементацијом кода. Симулација треба да прикаже реално понашање параметара за *IEEE 802.11p* стандард.

Прва целина поставља задатак конфигурације таласног облика. Када имплементирамо *PSDU (Physical Layer Service Data Unit)* кодујемо га и на тај начин формирамо таласни облик пакета који се шаље кроз комуникациони канал. У овом раду се користи *QPSK* модулациона техника. Потом прелазимо на имплементацију параметара који су специфични за *IEEE 802.11p* пренос. Затим имплементирамо објекте конфигурације за *NonHT* пренос уз помоћ функције *wlanNonHTConfig,* затим постављамо ширину канала на 10 MHz што има директног утицаја на опсег фреквенција које се користе за потребе преноса сигнала.

Следећа целина показује на који начин можемо да креирамо и да конфигуришемо комуникациони канал. У овом случају моделујемо радио канал за потребе аутомобилских комуникација. Сценарио описује урбано окружење уз присуство бројних објеката, као што су пословни и стамбени објекти, али и истакнути углови који имају утицаја на комуникацију. Приликом имплементације користимо сценарио *NLOS (Non line of sight propagation),*

односно, сценарио у ком немамо директну линију видљивости.

Од велике важности је дефиниција профила кашњења. У овом примеру се симулира профил „*Urban NLOS*“ што означава да је наше окружење у ком се комуникација одвија урбано. У урбаном окружењу имамо појаву вишеструких путева али и честа кашњења сигнала. У окружењу у ком преовладава *NLOS* сценарио можемо приметити да постоје бројне физичке препреке на путу од предајника до пријемника што директно утиче на кашњење сигнала, али и на појаву рефлексија, дифракција и расејања сигнала јер послати сигнал стиже до пријемника након пропагације кроз неколико различитих путања [2].

За сваку вредност *SNR*, односно, односа снаге корисног сигнала и шума, генерисаће се одређени број пакета који се пропушта кроз канал у ком фигурише шум. Потом ће се пакети пропустити кроз демодулатор. На самом крају процеса се одређује стопа грешке пакета за сваку вредност односа корисне снаге сигнала и шума. На тај начин ћемо проверити перформансе система са различитим нивоима *SNR* [2].

Потребно је дефинисати параметре као што су *maxNumErrors* и *maxNumPackets*. *maxNumErrors* је параметар који означава максималан број симулираних грешака пакета на једној *SNR* тачки. Када се ова граница достигне, симулација се завршава за ту тачку. Параметар *maxNumPackets* означава максималан број симулираних пакета на једној *SNR* тачки. Овај параметар ограничава дужину трајања симулације ако се граница за грешаке пакета не достигне [2].

У симулацији се имплементира и генератор случајних бројева. Главна улога генератора случајних бројева је да обезбеди да симулација буде поновљива и да добијени резултати могу да се упореде, што је од великог значаја када год имамо симулацију појединих сценарија или бројне друге експерименте [2].

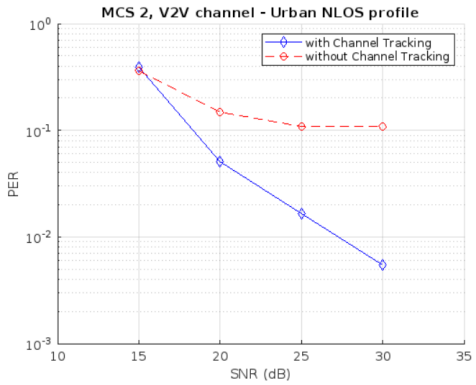
Обрада пакета је сложен процес који треба да обухвати више повезаних активности. Прво се врши детекција пакета, а затим његова груба процена. Фреквенција носиоца се коригује и потребно је извршити синхронизацију симбола. На самом крају обраде потребно је урадити фину корекцију и процену фреквенције носиоца.

Постоје два различита случаја за тестирање. Један случај укључује сценарио у ком имамо омогућено праћење канала, док у другом сценарију немамо омогућено праћење канала. У симулацији са праћењем канала чувамо вредности стопе грешака пакета. У већини симулираних сценарија, бољи резултати су постигнути у симулацији са омогућеним праћењем канала. Добијен је мањи *PER* него у симулираним сценаријима код којих немамо укључено праћење канала, а самим тим ни адаптацију пријемника на промене у каналу.

Једна од основних, обавезних активности приликом пројектовања система за потребе аутомобилских комуникација укључује обавезан поступак тестирања система. Са тестирањем система и његових карактеристика можемо унапред уочити слабе тачке. На следећих пар приложених слика, можемо уочити графички приказ различитих сценарија у којима имамо различите вредности параметара „*maxNumErrors*“ и „*maxNumPackets*“ .

Већи број послатих пакета и већи број грешака дају поузданије резултате, али и дуже време трајања симулације. Мали број постављен за параметар „*maxNumErrors*“, а велики број додељен параметру „*maxNumPackets*“ може такође дати поуздане резултате са брзом симулацијом исцртавања графика. Насупрот томе, мали број послатих пакета и мали број грешака даје брз графички приказ симулације али са малом прецизношћу. У сценарију у којем имамо мали број послатих пакета, а велики број грешака доводи до брзог графичког исцртавања али овакав сценарио може дати добре резултате само у случају када имамо високи *PER*, односно мали *SNR*.

Приказ прве симулације је дат на слици 1 и укључује параметар „*maxNumErrors*“ подешен на 147, док је параметар „*maxNumPackets*“ подешен на 548.



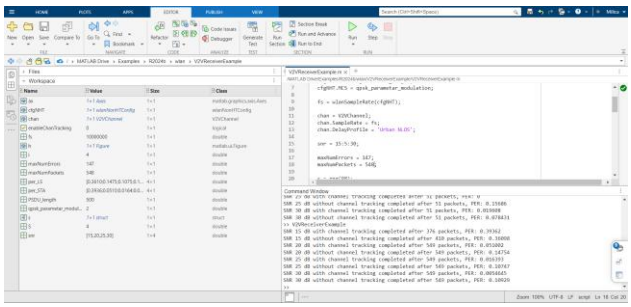
Слика 1. Резултат прве симулације

Приказ „*Comand Window*“ прозора који одговара првој симулацији је дат на слици 2.

```
>> V2VReceiverExample
SNR 15 dB with channel tracking completed after 376 packets, PER: 0.39362
SNR 15 dB without channel tracking completed after 410 packets, PER: 0.36098
SNR 20 dB with channel tracking completed after 549 packets, PER: 0.051002
SNR 20 dB without channel tracking completed after 549 packets, PER: 0.14754
SNR 25 dB with channel tracking completed after 549 packets, PER: 0.016393
SNR 25 dB without channel tracking completed after 549 packets, PER: 0.10747
SNR 30 dB with channel tracking completed after 549 packets, PER: 0.0054645
SNR 30 dB without channel tracking completed after 549 packets, PER: 0.10929
>>
```

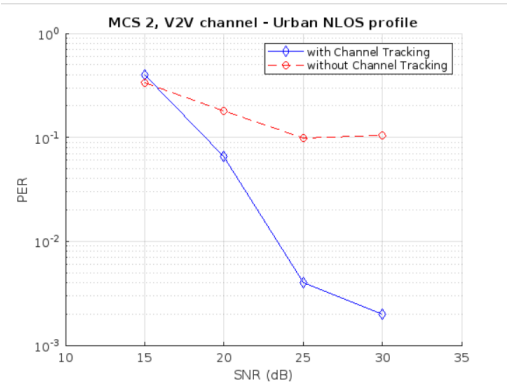
Слика 2. Comand Window прве симулације

Слика 3 има за циљ да прикаже како изгледа *MATLAB* окружење након извршене симулације. У оквиру домена „*Workspace*“ приложени су сви параметри и класе којима они припадају, као и њихове нумеричке вредности.



Слика 3. MATLAB окружење прве симулације

Приказ друге симулације је дат на слици 4 и укључује параметар „*maxNumErrors*“ подешен на 45, док је параметар „*maxNumPackets*“ подешен на 501.



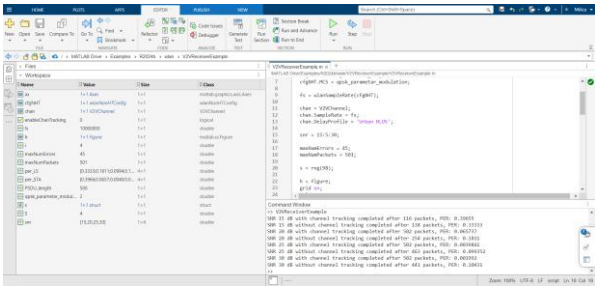
Слика 4. Резултат друге симулације

Приказ „*Comand Window*“ прозора који одговара другој симулацији је дат на слици 5.

```
>> V2VReceiverExample
SNR 15 dB with channel tracking completed after 116 packets, PER: 0.39655
SNR 15 dB without channel tracking completed after 138 packets, PER: 0.33333
SNR 20 dB with channel tracking completed after 502 packets, PER: 0.065737
SNR 20 dB without channel tracking completed after 254 packets, PER: 0.1811
SNR 25 dB with channel tracking completed after 502 packets, PER: 0.0039841
SNR 25 dB without channel tracking completed after 463 packets, PER: 0.099352
SNR 30 dB with channel tracking completed after 502 packets, PER: 0.001992
SNR 30 dB without channel tracking completed after 441 packets, PER: 0.10431
>>
```

Слика 5. Comand Window друге симулације

Слика 6 има за циљ да прикаже како изгледа *MATLAB* окружење након извршене друге симулације.



Слика 6. MATLAB окружење друге симулације

4. ЗАКЉУЧАК

Главна предност имплементације кода за *IEEE 802.11p* стандард та што се параметри могу потпуно флексибилно мењати. На тај начин можемо да допринесемо широком спектру нових симулација што ће нам детаљније пружити увид у перформансе система и показати његове како добре тако и слабе тачке. На овај начин, променом параметара, тестирамо целокупни систем и доприносимо његовом унапређењу. Сваки корак инжењерског напретка је пут ка стварању успешних пројеката који су ту да олакшају свакодневно коришћење технологија будућности.

5. ЛИТЕРАТУРА

[1] Eldad Perahia, Robert Stacey, „Next Generation Wireless LANs – 802.11n and 802.11ac“, Second Edition, Cambridge, University Press, First published 2008, Second edition 2013.

[2] „802.11p Packet Error Rate Simulation for a Vehicular Channel”, MathWorks, 802.11p Packet Error Rate Simulation for a Vehicular Channel - MATLAB & Simulink (mathworks.com), (приступила страници: 16.09.2024.)

Кратка биографија:

Милица Пругинић рођена је у Зрењанину 2000. год. Основне академске студије је уписала шк. 2019/20. године, а дипломирала 22.09.2023. године на Факултету техничких наука у Новом Саду, на смеру Енергетика, електроника и телекомуникације. усмерење: Комуникационе технологије и обрада сигнала и тиме стекла звање дипл. инг. Електротехнике и рачунарства. Мастер академске студије је уписала 2023. године на истом смеру, преусмерење: Информационо-комуникационе технологије и обрада сигнала.

контакт: pruginicm0@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

UDK: 4.8
DOI: https://doi.org/10.24867/30BE35Gluhovic

ДЕТЕКЦИЈА КРОВНИХ ПОВРШИНА СА САТЕЛИТСКИХ СЛИКА УПОТРЕБОМ РАЧУНАРСКЕ ИНТЕЛИГЕНЦИЈЕ

DETECTION OF ROOF SURFACES FROM SATELLITE IMAGES USING ARTIFICITAL INTELLIGENCE

Михаило Глуховић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТХНИКА И РАЧУНАРСТВО

Кратак садржај – *Циљ овог рада био је да се обави истраживање на пољу детекције и сегментације кровних површина са сателитских слика употребом различитих алгоритама базираних на употреби вештачке интелигенције. Рад описује поступак и проблеме креирања реалног скупа за обучавање модела, као и проблеме имплементације и тренирања различитих модела базираних на конволуционим неуронским мрежама. На крају су приказани резултати предикције модела и дискусија добијених резултата.*

Кључне речи: *детекција, сегментација, обучавајући скуп, конволуционе неуронске мреже, YOLO, SegNet, U-Net, DeepLabv3*

Abstract – *The aim of this paper was to conduct an study in the field of detection and segmentation of roof surfaces from satellite images using various artificial intelligencebased algorithms. The paper describes the process and challenges of creating a real dataset for model training, as well as the issues related to the implementation and training of different models based on convolutional neural networks. Finally, the results of the model's predictions are presented, along with a discussion of the obtained results.*

Keywords: *detection, segmentation, training dataset, convolutional neural networks, YOLO, SegNet, U-Net, DeepLabv3*

1. УВОД

Основна идеја овог рада била је да се пронађу готови, већ обучени, модели за сегментацију кровова са сателитских слика. С' обзиром да потрага за истима није била успешна, за потребе рада било је неопходно тестирати што већи број потенцијално употребљивих модела. Конкретно, имплементирани су U-Net, DeepLabv3, YOLO и SegNet. Сви ови модели се заснивају на конволуционим неуронским мрежама [1].

За обуку ових модела неопходни су обучавајући скуп и одређена процесорска снага.

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

Карактеристике коришћене радне машине су биле: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz процесор са 8 Gb RAM-а и NVIDIA GeForce GTX 1050 графичку картицу са 4 Gb RAM-а.

Скупови за обучавање се састоје од улазне сателитске слике и резултујуће бинарне маске, на којој су белим пикселима означени кровови а црном позадина. Квалитет базе података за обуку има значајан утицај на коначне перформансе модела. Међутим, поред ограниченог приступа квалитетним моделима, постоји и недостатак одговарајућих и обимних база података, са изузетком неколико ресурса.

2. ОДАБИР МОДЕЛА

Главно ограничење пројекта била је мала процесорска снага која директно доприноси дужем времену обучавања модела.

Поред дужег времена обучавања снага рачунара директно утиче и на то које моделе је уопште могуће тренирати за прихватљиво време. Постоји велики број различитих модела заснованих на конволуционим неуронским мрежама намењеним за обраду слика (нпр. YOLO, U-Net, SegNet, DeepLab, DeepMask, TensorMask, PANet...). У оквиру овог рада успешно су имплементирани следећи модели.

U-Net [2] је конволуциона мрежа дизајнирана за семантичку сегментацију са енкодер-декодер архитектуром, где енкодер примењује конволуционе операције и max-pooling да извуче карактеристике са слика и смањи их, а декодер користи up-sampling и конкатенацију са претходним слојевима за реконструкцију детаља слике.

DeepLabv3 [3] је модел за семантичку сегментацију који комбинује atrous конволуцију и просторни пирамидални pooling (енгл. Atrous Spatial Pyramid Pooling - ASPP).

YOLO [4] је модел намењен за детекцију објеката у једном пролазу преко слике, третирајући овај задатак као проблем регресије. Мрежа дели слику на мрежу ћелија и предвиђа позиције објеката, граничне оквире и класе, омогућавајући брзу и тачну детекцију објеката.

SegNet [5] је мрежа за семантичку сегментацију са енкодер-декодер архитектуром која користи max-pooling индексе за реконструкцију мапа обележја, за разлику од U-Net који памти целе мапе обележја.

3. ПРИПРЕМА ОБУЧАВАЈУЋЕГ СКУПА

Једина јавно доступна база [6] погодна за овај рад била је намењена детекцију површине крова која је повољна за постављање соларних панела. Што значи да је сваки нпр. димњак рупа у масци. Поред тога слике су биле малих димензија и лоше резолуције (слика 3.1).

Слика 3.1 Пример из пронађене базе [1]

Резултати предикције U-Net модела обученог на наведеној бази били су незадовољавајући. Разлог томе је могла бити лоша база или лош модел, оба су узета у обзир.

За потребе спремања квалитетне реалне базе неопходно је било обезбедити сателитску слику високе резолуције и пронаћи алат у коме би се ручно означили кровови.

Уз помоћ python библиотека leafmap и samgeo омогућен је приступ разним сателитима од којих се на око издвојио Google-ов. Google Satellite нуди слике резолуције од 15 м до 15 цм по пикселу. На мапи је означена површина од 300 хектара која обухвата целу Сремску Каменицу и сачувана као tif датотека. Каменица има пуно различитих објеката за детектовати и слика изнад Каменице је резолуције 15 цм по пикселу што је чини идеалном за потребе обучавајућег скупа.

Слика је заузимала 2 Gb и потребно ју је било поделити на 50 мањих слика како би могли да се обрађују у алатима за маскирање. Употребљен алат за анотирање јесте superviesly [7] са својим „mask pen tool“ који омогућава прецизно обележавање кровова. Наизглед лак задатак наилази на доста препрека, неке кровове је лако означити, док су неки прекривени крошњама, сакривени у хладу или просто веома сличне боје као околни бетон и пут (слика 3.2).

Слика 3.2 Примери проблематичних ситуација

Креирана база имала је 50 слика димензија 2281x2625 пиксела и 50 одговарајућих маски са означеним крововима. Већина модела не може да обрађује толико велике слике због тога је било потребно поново их поделити, овог пута на слике димензија 640x640 пиксела. Да би димензије свих слика биле исте слике димензија различитих од 640x640 пиксела су избачени, тако да је база на крају садржала 303 слике и 303 одговарајуће маске.

У жељи да се база прошири за што мање времена, обзиром да је за једну од 50 почетних слика било потребно око 2 сата да се цела анотира, једина преостала опција јесу аугментације, односно трансформације и модификације већ постојећих слика. Две главне врсте аугментација су коришћене: аугментације геометрије, које су у овом случају битније и аугментације боје и контраста.

Аугментације геометрије су скалирање и ротирање. Скалирање је у овом случају непотребно пошто су формиране слике истих димензија. Ротирање за „нецеле“ углове, односно ротације за углове који нису облика $k * \pi/2$ доводе до појаве mirroring, односно ивице слике се пресликавају као у огледалу како би одржале жељене димензије (слика 3.3).

Слика 3.3 Примери mirroring-а

Из тог разлога коришћене су аугментације: целе ротације уз hotizontal flip (ротирање по вертикалној оси), vertical flip (ротирање по хоризонталној оси) и комбинација та два уз аугментације боје и контраста (слика 3.3).

Слика 3.3 Пример из проширеног скупа

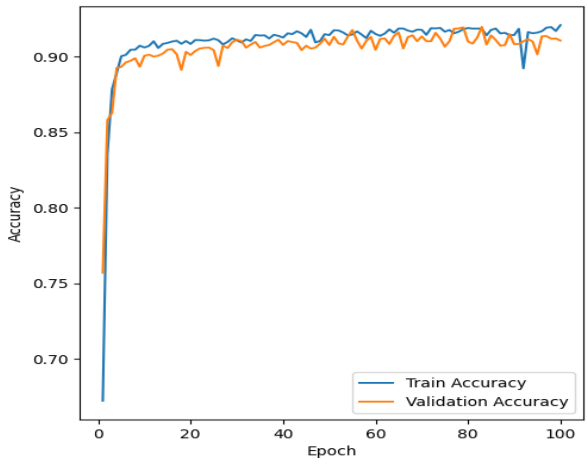
[1] Преузето из [6]

4.ОБУКА НЕУРОНСКИХ МРЕЖА

Тренирање модела је процес који захтева озбиљну количину рачунарске снаге, поготово у случају сегментације слика. Обрада графичких података (слика) за рачунар представља један од најзахтевнијих послова, јер треба да ради са великим датотекама које троше велику количину ресурса. Управо је то разлог зашто се за тренирање користе графичке картице које су оптимизоване за паралелну обраду података.

На расположивој машини која има Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz процесор са 8 Gb RAM-а и NVIDIA GeForce GTX 1050 графичку картицу са 4 Gb RAM-а, тренирање је било прилично захтевно. NVIDIA пружа софтвер CUDA [11] који је компатибилан са библиотеком pytorch помоћу које су имплементирани сви модели, и помоћу pytorch + cu118 библиотеке су тренирани. Тренирање је у одређеним случајевима било од 2 до 10 пута брже на графичкој картици него на процесору. Па шта је онда проблем? Проблем је што модел почне да се тренира и у неком моменту стане због немогућности алоцирања меморије на графичкој картици.
Један начин да се ово превазиђе јесте подешавање величина улазних слика, што је и урађено. Све слике, након подешавања су биле 640x640 пиксела и заузимале 1 Mb.

Тренирање има много параметара, од којих се у овом случају издвојио batch size, који одређује колико узорака скупа података пролази кроз мрежу у једном пролазу пре него што се изврши ажурирање тежина. Типична вредност овог параметра је од 32 па докле год машина може да изнесе. Због величине меморије коришћеног рачунара вредност овог параметра морала је бити или 2 или 4. То значи да ако 242 слике чине тренинг скуп, једна епоха тренирања ће имати 121(batch size = 2) или 61 (batch size = 4) итерацију. Мањи batch size утиче директно на време и квалитет обуке.



Слика 4.1 Промена тачности модела током обуке

Како се са слике 4.1 (тренинг SegNet модела) види модел почне да осцилује после 30те епохе. Узимајући то у обзир као и да је тренинг трајао скоро 2 дана, остали модели су тренирани на 30 до 40 епоха и трајањем у просеку 10-12 сати за U-Net, SegNet и DeepLabv3.

YOLO је модел који се најбрже тренира, али њега је и са најмањом вредности batch size параметра немогуће тренирати на графичкој картици. Архитектура YOLO алгоритма је дизајнирана за брзину, користи мање слојева и мањи број параметара од осталих модела. YOLO је уједно и једини модел који није било потребе тренирати од нуле већ је могуће преузимање и дотрениравање (енгл. fine-tune) модела “yolo8v-seg.pth”.Тренирање овог модела на процесору трајало је краће него тренирање осталих модела на графичкој картици. Из тог разлога YOLO је једини модел који је трениран на оригиналном скупу и на скупу проширеном аугментованим сликама. Међутим како YOLO у току тренинга сам врши аугментације улазних података, никаква разлика није примећена.

5. РЕЗУЛТАТИ

За потребе тестирања модела, на исти начин као са Каменицом, снимљен је Телеп, део Новог Сада који релативно личи на Каменицу. Слика обухвата око 100 хектара и као таква била је превелика за моделе. Слика је подељена на делове 640x640 пиксела, делови су провучени кроз модел и затим се од резултата реконструисала маска оригиналне слике.

На слици 5.1 приказана је слика и излазна бинарна маска из сваког модела. Стварна маска (енгл. ground turth) није направљена тако да није могуће дати бројеве за тачност предикције, али се са слика види да су модели успено сегментисали већину кровова.

Оригинална слика

Резултат предикције SegNet модела

Резултат предикције U-Net модела

Резултат предикције YOLO модела

Резултат предикције DeepLabv3 модела

Слика 5.1 Резултати предикција модела

6. ЗАКЉУЧАК

Успешно су имплементирани различити модели за сегментацију кровова са сателитских слика и направљена је реална база за тренирање тих модела. Сваки модел поседује одређене добре стране, али из резултата рада се издвајају SegNet и U-Net као лошије опције за решавање датог проблема. Оба модела представљају енкодер-декодер архитектуру где U-Net боље чува мапе обележја од SegNet-а, али као што се у резултатима види долазило је до погрешних класификација где је пут класификован као кров. Ове моделе је боље користити у ситуацијама где нема великих варијација у величинама и облицима објеката за сегментацију.

YOLO је показао одличне, очекиване, резултате обзиром да је то веома популаран модел који се користи за разне проблеме детекције и сегментације.

Коначно DeepLabv3 се показао као најбољи избор за конкретни проблем, са највише детекција и без погрешних класификација. DeepLabv3 је модел са најсложенијом архитектуром и најдужим временом обуке.

Правац даљег рада је повећање тачности детекције применом захтевнијих модела на јачем хардверу и са већим скуповима обуке.

7. ЛИТЕРАТУРА

[1] Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2017). *Fundamental concepts of convolutional neural network*. Department of Computer Science, University of Gour Banga, India; A.K. Choudhury School of Information Technology, University of Calcutta, India; Department of Computer Science & Engineering, M.A.K.A.U.T., India.

2] Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional networks for biomedical image segmentation*. Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany.

[3] Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). *Rethinking atrous convolution for semantic image segmentation*. Google Inc.

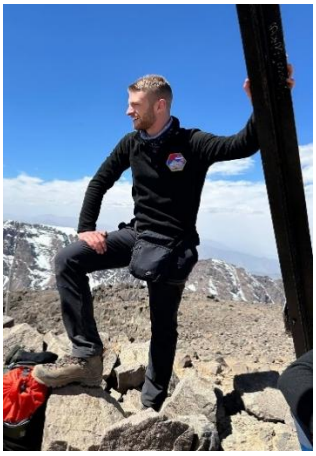
[4] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection*. University of Washington, Allen Institute for AI, Facebook AI Research.

[5] Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). *SegNet: A deep convolutional encoder-decoder architecture for image segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[6] https://drive.google.com/drive/folders/1v_VmwMEud1fcvdHRdgF8IWglBtXE8hbe

[7] https://app.supervisely.com/

Кратка биографија:



Михаило Глуховић рођен је 2001. године у Суботици. Основну школу завршио је у Новом Саду. Похађао је гимназију „J. J. Змај“. 2019. године уписује Факултет техничких наука, смер Рачунарство и аутоматика. Мастер академске студије наставља на смеру Аутоматика и управљање система. Тренутно је запослен на факултету као сарадник у настави. Контакт: gluhovic.mihailo@uns.ac.rs

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/30BE36Vidovic

REALIZACIJA PROGRAMSKOG PREVODIOCA, ASEMBLERA I EMULATORA ZA RISC-V MIKROPROCESOR

REALISATION OF COMPILER, ASSEMBLER AND EMULATOR COMPONENTS FOR RISC-V MICROPROCESSOR

David Vidović, *Fakultet tehničkih nauka, Novi Sad*

Oblast– ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisan je razvoj i realizacija programskog prevodioca C programskog jezika višeg nivoa na asemblerski jezik, implementacija asembler komponente zadužene za prevođenje asemblerskog koda u mašinski jezik i softverskog emulatora koji simulira rad RISC-V mikroprocesora sa jednom fazom obrade. Opisan je tok izrade korisničke aplikacije za grafički prikaz izlaznih podataka opisanih komponenti.*

Ključne reči: *RISC-V, programski prevodilac, asembler, emulator*

Abstract – *This paper describes the development and implementation of a high-level C programming language translator (compiler) into assembly language, the implementation of an assembler component responsible for translating assembly code into machine language, a software emulator that simulates the work of a single-cycle RISC-V microprocessor, and user application for graphical display of the output data of the described components.*

Keywords: *RISC-V, compiler, assembler, emulator*

1. UVOD

Postepenim prestankom važenja Murovog zakona došlo je drastičnih promena u svetu poluprovodnika i mikroprocesora. Postalo je sve teže za proizvođače tradicionalnih mikroprocesora da unaprede performanse uređaja, pre svega zbog fizičkih ograničenja prilikom fabrikacije komponente. Da bi se performanse izvršavanja postojećih i novih programa unapredile, pažnja je okrenuta ka načinu prevođenja programa iz programskih jezika visokog nivoa na mašinski jezik i na optimizacije korisničkog izvornog koda [1]. Takav nivo apstrakcije je suviše nizak za korisnika koji program opisuje u programskom jeziku visokog nivoa i ovaj posao pripada komponenti koja prevodi korisnički izvorni kod na skup asemblerskih instrukcija koje na najnižem nivou apstrakcije predstavljaju program za izvršavanje. U ovom radu opisan je razvoj jednostavnog programskog prevodioca koji prevodi podskup *ANSI* C programskog jezika na RV64I podskup instrukcija RISC-V arhitekture [2], asembler komponente koja generiše mašinski kod u vidu binarnih reči koje predstavljaju instrukcije, i softverskog emulatora koji simulira rad 64-bitnog mikroprocesora RISC-V arhitekture. Radovi koji pokrivaju sličnu temu već postoje, kao što su [3] i [4].

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

2. REALIZACIJA PROGRAMSKOG PREVODIOCA

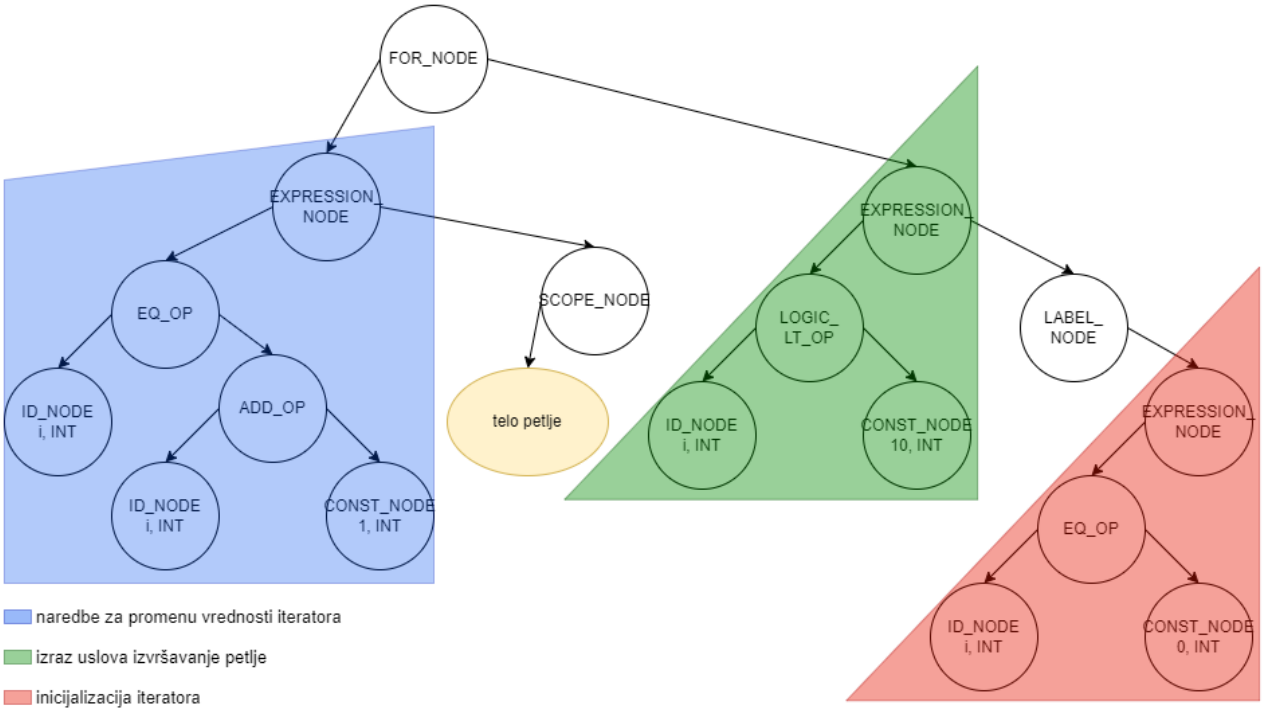
Programski prevodilac ili kompajler (eng. *Compiler*) je računarski program namenjen za prevođenje koda jednog programskog jezika u drugi programski jezik. Kod koji se prevodi zove se izvorni kod, a kod dobijen transformacijom je obično mašinski ili asemblerski kod. Struktura modernih programskih prevodilaca podeljena je na tri dela: prednji deo (eng. *front-end*), srednji deo (eng. *middle-end*) i zadnji deo (eng. *back-end*). Prednji deo je zadužen za skeniranje i parsiranje izvornog koda koji se posmatra kao ulazni tekst. Vrši se leksička analiza izvornog koda, nakon čega se proverava sintaksna ispravnost napisanog teksta (koda) kroz sintaksnu analizu i semantička ispravnost programa kroz semantičku analizu. Rezultat izvršavanja prednjeg dela prevodioca je obično međukod reprezentacija programa koja može biti tekstualna ili vidu strukture stabla, grafa ili neke druge strukture podataka. Srednji deo služi za dodatnu obradu i eventualne optimizacije generisanog međukoda, dok zadnji deo prevodioca rešava probleme kao što su alokacija registara i generisanje konačnog koda za ciljanu arhitekturu.

2.1 Realizacija leksera

Leksičku analizu izvršava lekser komponenta koja generiše tok (eng. *stream*) tokena koje prosleđuje parser komponenti. Za potrebe ovog rada korišćen je program Flex koji služi kao generator leksera napisanog u C programskom jeziku na osnovu opisa željenog leksera prateći sintaksu programa. Srž programa predstavlja deo sa pravilima za generisanje tokena. Lekser skenira ulazni tekst s leva na desno i na svako poklapanje dela teksta sa nekim pravilom izvršava skup naredbi definisanih za to pravilo i generisani token prosleđuje parseru. U najvećem broju slučajeva lekser treba samo da izbroji red i kolonu u kojoj je pronašao neko pravilo (posredstvom metode *count()* u svrhu pružanja detaljnije prijave greški korisniku) i da parseru prosledi generisani token, dok kod nekih pravila treba i da popuni polja strukture koja predstavlja vrstu podatka tokena atributima koje je pronašao u izvornom tekstu. Tako na primer za skeniran identifikator lekser u strukturi tokena *IDENTIFIER* popunjava polja sa nazivom promenjive, brojem reda i brojem kolone u kojoj je token generisan.

2.2 Realizacija parsera

Zadatak parsera jeste da na osnovu dobijenog toka tokena i opisane gramatike programskog jezika generiše stablo parsiranja. Za opis gramatike C programskog jezika i za generisanje parsera na osnovu datog opisa korišćen je program Yacc, koji generiše konačni automat u C jeziku na osnovu datog opisa gramatike. Prilikom parsiranja programa generiše se stablo parsiranja i za svako pronađeno pravilo u skupu tokena moguće je definisati skup naredbi u C jeziku koje parser treba da izvrši. Ova osobina generisanih parsera je iskorišćena u ovom radu da bi se prilikom parsiranja programa generisalo apstraktno sintaksno stablo koje predstavlja međukod ovog prevodioca.



Slika 1. *Primer podstabla FOR petlje*

Parsiranje počinje od najjednostavniji gramatičkih pravila koje zadovoljava pojedinačni token, i završava se formiranjem pravila *translation_unit* koje podrazumeva kompletan ispravno napisan C program sa definicijom bar jedne funkcije koja ima deklarisan tip podatka i telo funkcije omeđeno vitičastim zagradama.
Ukoliko parser pronađe skup tokena koji ne zadovoljavaju ni jedno pravilo generiše se greška jer izvorni kod nije sintaksno ispravan.

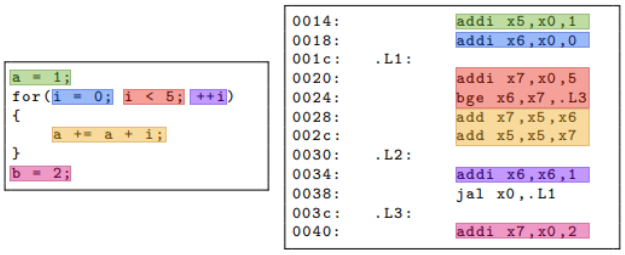
2.2.1 Tabela simbola

Veoma važna struktura podataka unutar programskog prevodioca je tabela u kojoj se čuvaju svi potrebni podaci o identifikatorima u korisničkom programu koji se prevodi. U ovom radu tabela simbola je realizovana kao heš (eng. *hash*) tabela zbog dobrih performansi prilikom pretrage tabele (što je u ovoj primeni česta pojava) i dobrog uklapanja u potrebe projekta.
Ključ na osnovu kojeg se izračunava indeks elementa u nizu jeste naziv identifikatora i za heš funkciju korišćen je algoritam FNV-1a koji koristi kombinaciju operacije XOR i množenja sa 64-bitnim prostim brojem za generisanje *heša*.
Važni atributi svaki promenjive se čuvaju u poljima strukture *ht_entry* i svaka tabela sadrži vrste koje predstavljaju pokazivače na objekte ove strukture. Za svaki domen važenja u programu kreira se posebna tabela (globalne promenjive i funkcije, i lokalne promenjive unutar svake funkcije).

2.3 Generisanje međukoda

Međukod u ovom prevodiocu predstavlja apstraktnog sintaksno stablo. Stablo je binarno i unidirekciono, što znači da svaki čvor stabla ima najviše dva naslednika. Čvorovi stabla su pokazivači na instance strukture *ASTnode* koja sa svojim poljima predstavlja kompletan opis instrukcije međukoda. Pomenuta struktura sadrži pokazivače na dva čvora stabla: *ASTnode *left* i *ASTnode *right*, koji predstavljaju njegove naslednike, odakle unidirekciona osobina stabla. Stablo se generiše u toku parsiranja, počevši od listova ka korenu, prateći unapred određena pravila za generisanje stabla, kao na primer: čvorovi identifiaktora i konstanti su uvek listovi stabla, čvorovi izraza uvek sa leve strane pokazuju na operacije unutar izraza a sa desne na prethodni izraz, čvorovi operacija uvek pokazuju na svoje operande, čvorovi funkcija sa leve strane pokazuju na čvor identifikator funkcije a sa leve strane na telo funkcije itd. Kroz stablo se prolazi rekurzivnom funkcijom i generisano je tako da rekurzivnim prolazom zagarantuje ispravan redosled programa.
Specifične su naredbe granjanja i petlje kod kojih su usvojena posebna pravila za očuvanje ispravnosti programa. Generisani čvor *IF* sa leve strane uvek pokazuje na telo petlje, a sa desne na izraz koji je potrebno izvršiti kako bi bila doneta odluka da li je potrebno da dođe do programskog skoka. Slično pravilo važi i za *IF-ELSE*, *WHILE*, *DO-WHILE* i *SWITCH* petlje. Razlikuje se samo *FOR* petlja zbog kompleksnijeg

zaglavlja, kao što je prikazano na slici 1., gde je osigurano da inicijalizacija iteratora bude prevedena u instrukcije koje su van petlje, zatim sledi provera uslova i telo petlje, a na kraju tela petlje



Slika 2. *Generisan kod za jednostavnu FOR petlju*

se vrši promena vrednosti iteratora i programski skok na proveru zadovoljenosti uslova petlje. Programski skokovi se vrše posredstvom labela u asemblerskom kodu, a pošto u trenutku generisanja naredbi granjanja kod petlji nisu poznate adrese labela programskih skokova, popunjavanje polja namenjenog za adresu labele kod ovakvih čvorova je postignuto posredstvom virtuelnog steka (eng. *stack*), odnosno LIFO strukture. Svaki put kada se generiše čvor naredbe granjanja pokazivač na njega se postavlja na vrh steka, i svaki put kada se generiše čvor labele uzima se jedan pokazivač sa vrha steka i dodeljuje mu se generisana labela. Na ovaj način je osiguran ispravan prevod ugneždenih petlji.

Ovaj programski prevodilac nad međukodom vrši i optimizaciju izvornog koda, u vidu eliminacije mrtvog koda. Mrtvi kod je kod koji se nikada neće izvršiti i obično se odnosi na deo programa koji se nalazi posle bezuslovne *return* naredbe u funkciji. Ukoliko u stablu postoji *RETURN_NODE* čvor, iz stabla se uklanjaju svi njegovi čvorovi roditelji u posmatranom telu funkcije i ovaj čvor postaje direktan naslednik čvora za telo funkcije. Na sličan način postignut je mehanizam *break* i *continue* naredbi.U toku generisanja međukoda ne vrši se alokacija registara, i mikroprocesor se tretira kao da poseduje beskonačno mnogo registara.

2.4 Generisanje asemblerskog koda

Rekurzivnim prolazom kroz apstraktno sintaksno stablo metoda *populate_IR* generiše konačni kod za ciljanu strukturu. Kod je unutar prevodioca realizovan u vidu bidirekcione spregnute liste sačinjenje od čvorova koji su pokazivači na instance strukture *IR_node* koja poseduje sve potrebne atribute za opis instrukcije. Svaki čvor liste predstavlja jednu instrukciju.

2.5 Alokacija registara

Alokacija registara u programskim prevodiocima je NP-kompletan problem [5]. Koriste se razni algoritmi za pokušaj pronalaska optimalnog rešenja za bilo koji ulazni program, kao što su analiza živih intervala ili tehnika bazičnih blokova i lineranog skeniranja [6]. U ovom radu izabran je daleko jednostavniji algoritam za implementaciju, koji ima pohlepnu osobinu. Vodi se evidencija o zauzetosti svakog dostupnog registra za lokalne promenjive, koji u RISC-V arhitekturi ukupno ima sedam, i popunjavaju se redom dok svi ne postanu puni. Od tada nadalje, kada god dođe do referenciranja promenjive čija vrednost nije u nekom registru, bira se jedan registar za prosipanje vrednosti i vrše se *load* i *store* operacije zamene vrednosti u odabranom registru. Algoritam je pohlepan jer će ostalih 6 registara uvek biti zauzeti od

```
0058: addi x5 ,x0 ,1
005c: addi x6 ,x0 ,2
0060: addi x7 ,x0 ,3
0064: addi x28 ,x0 ,4
0068: addi x29 ,x0 ,5
006 c: addi x30 ,x0 ,6
0070: addi x31 ,x0 ,7
0074: sw x6 , -44( x8)
0078: addi x6 ,x0 ,8
007c: sw x6 , -20( x8)
0080: addi x6 ,x0 ,9
0084: sw x6 , -16( x8)
0088: addi x6 ,x0 ,10
008c: sw x6 , -12( x8)
0090: addi x6 ,x0 ,11
```

Kodni fragment 1. *Primer pohlepne alokacije registara*

strane prvih 6 promenjivih u programu, kao što je slučaj u primeru prikazanom na kodnom fragmentu 1.

3. REALIZACIJA ASEMBLERA

Asembler komponenta je realizovana u C programskom jeziku, većinom posredstvom makro funkcija i metoda za dekodovanje i manipulaciju stringovima. Realizovana je kao deo programskog prevodioca, što je opravdano jer prevodilac ima samo jednu ciljanu arhitekturu. Ulazni tekst u vidu asemblerskih instrukcija obrađuje i na osnovu njega generiše izlazni fajl ekstenzije *.bin* sa mašinskim kodom programa, odnosno instrukcijama napisanom u formi binarnih reči.

4. REALIZACIJA EMULATORA

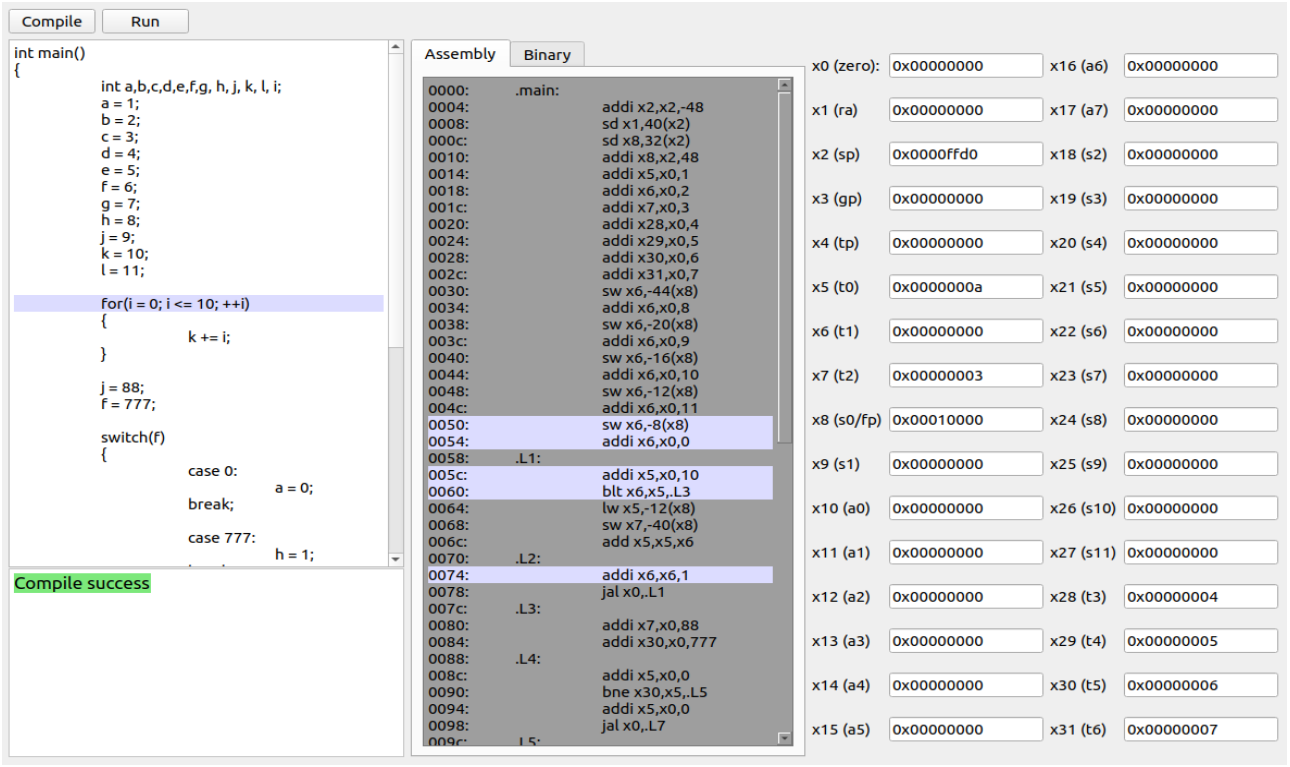
Emulator je razvijen u programskom jeziku C++ posredstvom klasa za memoriju, magistralu podataka i procesorsku jedinicu (eng. *CPU*), po uzoru na [7]. Klasa DRAM predstavlja memorijski sistem emulatora realizovan statičkim nizom u programu. Instrukcije programa za izvršavanje se čitaju i postavljaju na adresu 0x80000000 odakle počinje programska memorija. Preko klase BUS koja simulira rad magistrale podataka instrukcija sa adrese na koju pokazuje programski brojač se prenosi do procesora svaki put kada se pozove metoda *cpu_fetch*. Za simulaciju izvršavanja programa koristi se metoda *cpu_exec* koja izvršava zahvaćenu instrukciju tako što je dekoduje i vrši operaciju iz instrukcije na procesoru domaćinu. Emulator prestaje sa radom kada se iz memorije zahvati nevalidna instrukcija, što uključuje i kraj programa (memorijska lokacija nakon poslednje instrukcije u programu sadrži vrednost 0).

Pomoćne metode *dump_regs* i *dump_stack* na terminal i u izlazne fajlove ispisuju stanje svih registara i ne-nulte vrednosti na steku sistema na kraju izvršavanja programa.

5. REALIZACIJA KORISNIČKE APLIKACIJE

Korisnička aplikacija razvijena je u programskom jeziku C++ koristeći postojeći skup biblioteka za razvoj grafičkog okruženja *Qt*. Glavni i jedini prozor aplikacije

podeljen vertikalno na tri dela. Skroz levi deo sadrži tekstualni editor i tastere *Compile* i *Run*, te mali tekstualni prozor nalik terminalu za ispis potencijalni greški u programu. Pritiskom na taster *Compile* u srednjem delu generiše se asemblerski kod u jednom *tabu* i mašinski kod u drugom *tabu*, kao rezultat rada programskog prevodioca i asemblera. Pritiskom na taster *Run* u skroz desnom delu ispisuje se stanje registara procesora nakon izvršavanja programa, kao rezultat rada emulatora.



Slika 3. *Primer izgleda korisničke aplikacije*

Glavna osobina aplikacije jeste mogućnost označavanja kursorom pojedinačnih linija koda u izvornom kodu u tekst editoru, na osnovu čega aplikacija prikazuje u koje instrukcije u asemblerskom i mašinskom kodu se pretvorila označena linija izvornog koda.

6. ZAKLJUČAK

Cilj ovog projekta bilo je upoznavanje sa osnovnim tehnikama realizacije programskih prevodilaca i ostalih softverskih alata niskog nivoa. Uspešno je realizovan programski prevodilac koji prevodi podskup *ANSI* C gramatike na *RV64I* asemblerski jezik. Mana ove komponente je što ne podržava celi skup C89 standarda, kao što je rad sa strukturama, rezervisanje memorije na *heap*-u, rad sa brojevima u pokretnom zarezu, instrukcije množenja i deljenja itd. Moguća unapređenja uključuju pisanje prevodioca u C++ ili Rust jeziku zbog većeg nivoa apstrakcije i raznovrsnijih mogućnosti za realizaciju komponenti.

7. LITERATURA

[1] D. A. Patterson, J. L. Hennessy, *"Computer Organization and Design RISC-V Edition: The Hardware Software Interface"*, Morgan Kaufmann, 2017

[2] RISCV, History of RISC-V https://riscv.org/about/history/ (pristupljeno u junu 2024.).

[3] Z. Vujadzin Rakic, P. Rakic, T. Petric, *"miniC Project for Teaching Compilers Course"*, University of Novi Sad/Faculty of Technical Sciences, Novi Sad, Serbia, 2014.

[4] A. Z. Henley, *"Let's make a Teeny Tiny compiler"*, Carnegie Mellon University, 2020.

[5] G. J. Chaitin, *"Register Allocation and Spilling via Graph Coloring"*, IBM Research P.O.Box 218, Yorktown Heights, NY 10598, 1981.

[6] D. R. Koes, S. C. Goldstein, *"Register Allocation Deconstructed"*, Carnegie Mellon University Pittsburgh, PA

[7] Writing a simple RISC-V emulator in plain C, https://fmash16.github.io/content/posts/riscv-emulator-in-c.html (pristupljeno u februaru 2024.)

Kratka biografija:

David Vidović rođen je u Derventi 2000. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektronike – Embedded sistemi i algoritmi odbranio je 2023.god. od kada radi kao saradnik u nastavi na Katedri za elektroniku.
kontakt: david.vidovic@uns.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.85
DOI: https://doi.org/10.24867/30BE37Gajic

DIZAJN I IMPLEMENTACIJA MODERNOG SISTEMA ZA ELEKTRONSKO PLAĆANJE
DESIGN AND IMPLEMENTATION OF A MODERN ELECTRONIC PAYMENT SYSTEM

Mladen Gajić, Fakultet tehničkih nauka, Novi Sad

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Elektronsko plaćanje predstavlja ključnu komponentu savremenog poslovanja, omogućavajući brze, sigurne i efikasne transakcije između korisnika i finansijskih institucija. Ovaj rad istražuje dizajn i implementaciju modernog sistema za elektronsko plaćanje zasnovano na mikroservisnoj arhitekturi. Korišćene tehnologije uključuju Spring Boot, React, PostgreSQL, i Docker Compose što omogućava visoku skalabilnost, fleksibilnost i sigurnost sistema. Rad takođe analizira tehničke izazove tokom implementacije, prednosti mikroservisne arhitekture, kao i potencijalna unapređenja, uključujući prelazak sa Docker Compose na Kubernetes, proširenje funkcionalnosti i usklađivanje sa zakonskim regulativama.*

Ključne reči: *elektronsko plaćanje, mikro-servisna arhitektura, integracija sa ostalim sistemima za plaćanje, bankarski sistem, transakcije*

Abstract – *Electronic payment is a key component of modern business, enabling fast, secure and efficient transactions between users and financial institutions. This paper investigates the design and implementation of a modern electronic payment system based on a microservice architecture. Technologies used include Spring Boot, React, PostgreSQL, and Docker Compose, which enables high scalability, flexibility, and system security. The paper also analyzes the technical challenges during implementation, the advantages of microservices architecture, as well as potential improvements, including the transition from Docker Compose to Kubernetes, expanding functionality and compliance with legal regulations.*

Keywords: *electronic payment, micro-service architecture, integration with other payment systems, banking system, transactions*

1. UVOD

Digitalna transformacija donela je revolucionarne promene u svim aspektima društva, uključujući način na koji obavljamo finansijske transakcije.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Dunja Vrbaški, docent.

Elektronsko plaćanje postalo je neizostavan deo globalne ekonomije, omogućavajući trenutne, sigurne i praktične transakcije između korisnika, trgovaca i finansijskih institucija. Pojava interneta, mobilnih uređaja i digitalnih platformi stvorila je uslove za razvoj sistema koji omogućavaju obavljanje transakcija bilo gde i bilo kada, eliminisajući potrebu za fizičkim prisustvom ili gotovinom. U ovom radu je analiziran dizajn i implementacija modernog sistema za elektronsko plaćanje, koji koristi mikroservisnu arhitekturu za postizanje visoke skalabilnosti, modularnosti i sigurnosti.

2. MIKROSERVISNA ARHITEKTURA

Mikroservisna arhitektura je moderan pristup dizajnu softverskih sistema, gde se aplikacija sastoji od niza manjih, nezavisnih servisa, od kojih svaki obavlja specifičan zadatak. Ovi servisi komuniciraju međusobno putem dobro definisanih API interfejsa, što omogućava decentralizovan razvoj, testiranje i distribuciju. Prednosti mikroservisne arhitekture u odnosu na tradicionalnu monolitnu arhitekturu su brojne, uključujući skalabilnost, fleksibilnost, otpornost na greške i mogućnost brze isporuke novih funkcionalnosti [1].

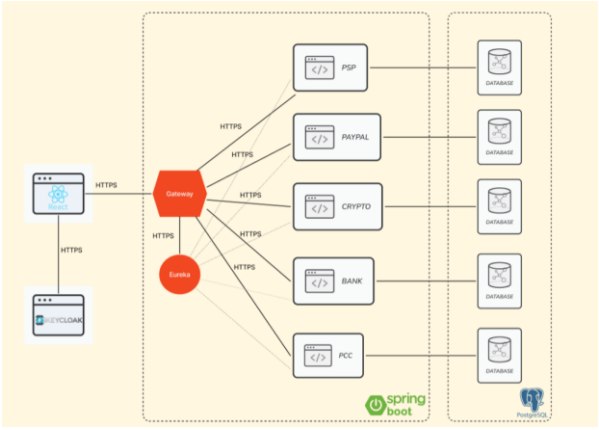
2.1. PREDNOSTI MIKROSERVISNE ARHITEKTURE

Mikroservisna arhitektura omogućava skaliranje pojedinačnih servisa po potrebi, što znači da se resursi mogu efikasno koristiti i distribuirati u zavisnosti od trenutnih opterećenja. Na primer, ako određeni servis doživljava povećan promet, moguće ga je skalirati horizontalno, dodavanjem novih instanci, bez potrebe za skaliranjem cele aplikacije. Ovaj pristup omogućava bolju performansu sistema, smanjujući rizik od zagušenja i omogućavajući optimalno korišćenje resursa. Jedna od ključnih prednosti mikroservisne arhitekture je otpornost na greške. Kvar jednog servisa ne mora nužno da utiče na funkcionisanje celog sistema, što značajno povećava dostupnost i pouzdanost usluga. Zbog izolacije servisa, sistem može ostati funkcionalan čak i ako jedan ili više servisa prestanu sa radom. Ovo je posebno važno za sisteme elektronskog plaćanja, gde je neprekidnost usluge ključna za korisničko poverenje i zadovoljstvo. Razvojni timovi mogu raditi na različitim servisima nezavisno jedni od drugih, koristeći različite tehnologije i programske jezike koji najbolje odgovaraju potrebama svakog servisa. Ovaj nivo fleksibilnosti omogućava brži razvoj i lakše

održavanje sistema, kao i brzu integraciju novih funkcionalnosti [1].

2.2. MIKROSERVISI U SISTEMU ELEKTRONSKOG PLAĆANJA

Sistem za elektronsko plaćanje razvijen u ovom radu sastoji se od nekoliko ključnih mikroservisa, od kojih svaki ima specifičnu ulogu u okviru sistema. Ilustracija sistema prikazana je na slici 1.



Slika 1. Uprošćeni prikaz mikroservisne arhitekture korišćene za izradu sistema za elektronsko plaćanje

***Gateway* servis**: Funkcioniše kao ulazna tačka u sistem, upravlja rutiranjem zahteva, autentifikacijom i autorizacijom korisnika, kao i kontrolom pristupa. Korišćenje Spring Cloud Gateway-a omogućava centralizovano upravljanje protokom podataka i primenu sigurnosnih politika.

Payment Service Provider (PSP) servis: Implementiran kao centralna tačka za integraciju sa različitim platnim platformama. Ovaj servis komunicira sa korisničkom aplikacijom i obavlja procesiranje plaćanja kroz integraciju sa eksternim platnim provajderima.

Bank servis: Simulira operacije banke unutar sistema, uključujući evidenciju klijenata, vođenje računa i obavljanje transakcija. Svaka banka u sistemu poseduje svoju instancu Bank servisa, što omogućava izolaciju podataka i specifičnu konfiguraciju za svaku instituciju.

PCC (Payment Card Clearing) servis: Obavlja funkcionalnosti klirinških kuća za platne kartice, omogućavajući autorizaciju i poravnanje transakcija između različitih banaka. Ovaj servis je ključan za obezbeđivanje sigurnih i efikasnih međubankovnih transakcija.

Crypto servis: Dizajniran za podršku plaćanja putem kriptovaluta, omogućavajući transakcije u različitim digitalnim valutama. Korišćenje Spring Boot-a za integraciju sa blockchain mrežama pruža funkcionalnosti poput generisanja adresa za uplatu, validacije transakcija i praćenja stanja na digitalnim novčanicima.

3. KORIŠĆENE TEHONOLOGIJE

Za implementaciju sistema za elektronsko plaćanje korišćene su savremene tehnologije koje omogućavaju visoku skalabilnost, fleksibilnost i sigurnost sistema. U ovom delu rada detaljno ćemo analizirati ključne tehnologije koje su korišćene u implementaciji.

3.1. Spring Boot

Spring Boot je moderan tehnološki okvir razvijen kao nadogradnja na već dobro poznati Spring Framework, namenjen za kreiranje samostalnih i proizvodnih Spring aplikacija uz značajno smanjenje složenosti konfiguracije i ubrzavanje celokupnog procesa razvoja. *Spring Boot* pruža unapred definisane konfiguracije za različite vrste aplikacija, što omogućava brži razvoj i lakše održavanje sistema. Jedna od ključnih prednosti *Spring Boot*-a je mogućnost kreiranja samostalnih aplikacija koje uključuju ugrađeni server, poput *Tomcat* servera, što eliminiše potrebu za instaliranjem i održavanjem eksternog aplikacionog servera. Ovo značajno pojednostavljuje razvoj, testiranje i distribuciju aplikacija, omogućavajući razvojnim timovima da se fokusiraju na poslovne funkcionalnosti umesto na tehničku infrastrukturu [2]. Ovaj pristup omogućio je modularan i fleksibilan razvoj sistema, koji je lako skalabilan i prilagodljiv budućim potrebama.

3.2. React

React je jedna od najpopularnijih *JavaScript* biblioteka za izradu korisničkih interfejsa, razvijena od strane *Facebook*-a (sada *Meta*). *React* omogućava kreiranje interaktivnih i dinamičnih korisničkih interfejsa putem komponentne arhitekture, koja omogućava razvoj modularnih i ponovnih komponenti. U našem sistemu, *React* je korišćen za izradu korisničke stranice, omogućavajući korisnicima jednostavan i intuitivan interfejs za obavljanje transakcija. Komponentna arhitektura *React*-a omogućila je brz razvoj svih neophodnih funkcionalnosti, dok je virtuelni *DOM* osigurao visoke performanse i responzivnost aplikacije [3].

3.3. PostgreSQL

PostgreSQL je moćan sistem za upravljanje relacionim bazama podataka, poznat po svojoj stabilnosti, pouzdanosti i bogatom skupu funkcionalnosti. *PostgreSQL* podržava *ACID*(*Atomicity*, *Consistency*, *Isolation*, *Durability*) svojstva, koja osiguravaju doslednost i integritet podataka, što je od ključne važnosti za sisteme elektronskog plaćanja. Svaki mikroservis ima svoju bazu podataka unutar iste *PostgreSQL* instance. Ova arhitektura omogućava izolaciju podataka između mikroservisa, čime se povećava sigurnost i smanjuje mogućnost konflikata između različitih delova sistema [4].

3.4. Docker

Doker je platforma za razvoj, isporuku i pokretanje aplikacija u izolovanim okruženjima poznatim kao kontejneri. Doker omogućava brzu i efikasnu kontejnerizaciju aplikacija, što olakšava upravljanje i skaliranje sistema. Korišćenje Dokera u okviru sistema za

elektronsko plaćanje omogućilo je izolaciju svakog mikroservisa, čime su minimizovani konflikti između servisa i povećana sigurnost sistema. Doker je takođe omogućio brzo skaliranje servisa, kao i jednostavno upravljanje zavisnostima i konfiguracijama aplikacija [5].

3.5. Docker Compose

Docker Compose je alat za definisanje i pokretanje višestrukih Docker kontejnera kao jedne aplikacije. U okviru ovog sistema, *Docker Compose* je korišćen za pokretanje svih mikroservisa i njihovih zavisnosti, omogućavajući jednostavno testiranje i razvoj celokupnog sistema u lokalnom okruženju [5].

4. INTEGRACIJA SISTEMA ZA ELEKTRONSKO PLAĆANJE SA OSTALIM APLIKACIJAMA

Integracija sistema za elektronsko plaćanje sa postojećom ostalim aplikacijama predstavlja ključni korak ka unapređenju korisničkog iskustva i povećanju efikasnosti poslovanja. Elektronsko plaćanje je postalo standard u modernom poslovanju, omogućavajući brže i sigurnije transakcije. Proces integracije obuhvata nekoliko važnih elemenata, uključujući strategiju implementacije plaćanja i tehničke aspekte registracije korisnika i trgovaca u bankarskom sistemu. Takođe u sistemu elektronskog plaćanja razlikujemo nekoliko procesa plćanja, koji su opisani u narednim delovima ovog rada.

5. PROCES NAPLATE KARTICOM

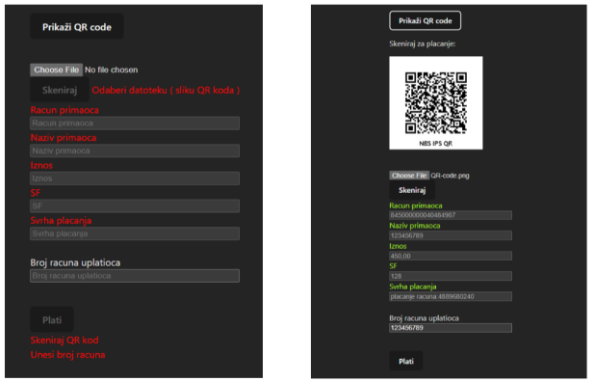
Elektronsko plaćanje karticom u sistemu elektronskog plaćanja obuhvata nekoliko faza:

- *Inicijalizacija transakcije*: Kupac potvrđuje kupovinu u aplikaciji i bira način plaćanja karticom. Nakon toga, sistem ga preusmerava na stranicu *PSP (Payment Service Provider)* gde unosi podatke o kartici.
- *Autentifikacija i validacija:* Podaci o kartici se prosleđuju *PSP* servisu, koji obavlja validaciju kod banke izdavaoca kartice. Banka proverava tačnost podataka i raspoloživost sredstava, te autorizuje transakciju.
- *Realizacija transakcije:* Ako je transakcija odobrena, sredstva se rezervišu na računu kupca i prenose na račun prodavca. Kupac dobija povratnu informaciju o uspehu transakcije.

Postoji jednostavniji i kompleksniji scenario koji može da se dogodi u zavisnosti u kojim bankama kupac i prodavac imaju račune. Jednostavni scenario jeste kada kupac i prodavac imaju račune u istoj banci, što omogućava bržu realizaciju transakcije jer se sve operacije odvijaju unutar istog sistema. Složeniji scenario jeste taj kada kupac i prodavac imaju račune u različitim bankama. U ovom slučaju, proces uključuje PCC (Payment Clearing and Settlement) servis, koji koordinira komunikaciju između banaka kako bi se transakcija uspešno realizovala.

6. PLAĆANJE IPS QR KODOM

Plaćanje putem IPS QR koda predstavlja moderan i brz način digitalne transakcije. Korisnik skenira QR kod putem mobilne aplikacije svoje banke, a aplikacija automatski popunjava sve potrebne podatke za transakciju, što ubrzava proces i smanjuje mogućnost greške. Ovaj metod je posebno koristan u maloprodajnim objektima i restoranima gde je brzina ključna za obavljanje što većeg broja transakcija u kratkom vremenu [6]. Implementacija *IPS* plaćanja u sistemu elektronskog plaćanja pruža korisnicima dodatnu mogućnost, omogućavajući im izbor između tradicionalnog plaćanja karticom i bržeg, jednostavnijeg plaćanja QR kodom, koji je simuliraln otpremanjem slike QR koda, kao što je prikazano na slici 2.



Slika 2. Primer izgleda stranice pre i posle otpremanja QR koda

7. INTEGRACIJA SA DRUGIM PLATNIM PROVAJDERIMA

Jedna od važnih funkcionalnosti sistema za elektronsko plaćanje jeste integracija sa različitim platnim provajderima, što omogućava korisnicima fleksibilnost u izboru načina plaćanja. Sistem podržava integraciju sa *PayPal* servisom za plaćanje, kao i plaćanje kriptovalutama čime se omogućava globalna pokrivenost i jednostavno obavljanje transakcija.

7.1. INTEGRACIJA SA PAYPAL SERVISOM ZA PLAĆANJE

PayPal je jedan od najpoznatijih globalnih provajdera elektronskih plaćanja, koji omogućava korisnicima da obavljaju transakcije koristeći njihove *PayPal* naloge. U okviru ovog sistema, PayPal servis je razvijen kao poseban mikroservis, koji omogućava integraciju sa *PayPal* API interfejsom za obavljanje transakcija, povrat sredstava i verifikaciju stanja na računima [7]. Integracija sa *PayPal* servisom omogućava korisnicima jednostavan i siguran način plaćanja, dok sistemu omogućava dodatno proširenje funkcionalnosti u skladu sa potrebama tržišta.

7.2. INTEGRACIJA SA KRIPTOVALUTAMA

Kriptovalute, kao što su *Bitcoin* i *Ethereum*, postale su popularan način plaćanja u poslednjih nekoliko godina, posebno u industrijama gde se vrednuju decentralizacija i anonimnost transakcija. U okviru ovog sistema, *Crypto*

servis omogućava korisnicima da obavljaju transakcije koristeći kriptovalute, što uključuje generisanje adresa za uplatu, validaciju transakcija i praćenje stanja na digitalnim novčanicima. *Bitcoin* je prva kriptovaluta, lansirana 2009. godine, dok je *Hedera Hashgraph*, inovacija iz 2018, koja koristi efikasniji algoritam za obezbeđivanje visokog nivoa sigurnosti i brzine. U našem sistemu, *Hedera Hashgraph* omogućava brzo generisanje i validaciju transakcija uz minimalne troškove i energetsku efikasnost. Integracija sa alatima poput *HashPack* i *HashScan* dodatno osigurava transparentnost i pouzdanost u obradi kripto transakcija [8,9].

Korišćenje kriptovaluta kao načina plaćanja pruža korisnicima dodatnu fleksibilnost, dok sistemu omogućava prilagođavanje savremenim trendovima u finansijskom sektoru.

8. ZAKLJUČAK

Ovaj rad predstavlja sveobuhvatan pristup dizajnu i implementaciji savremenog sistema za elektronsko plaćanje, koristeći mikroservisnu arhitekturu i savremene tehnologije kao što su *Spring Boot, React, PostgreSQL* i *Docker*. Sistem je dizajniran da bude fleksibilan, skalabilan i siguran, sa fokusom na integraciju sa različitim platnim provajderima i prilagođavanje savremenim trendovima u finansijskom sektoru. Sistem za elektronsko plaćanje razvijen u ovom radu ima visoku fleksibilnost i skalabilnost, ali postoje i dodatna unapređenja koja bi mogla poboljšati performanse i konkurentnost sistema.

8.1. POTENCIJALNA UNAPREĐENJA

Jedno od glavnih unapređenja je prelazak sa *Docker Compose* na *Kubernetes* platformu, što bi omogućilo bolju automatizaciju, skalabilnost i upravljanje resursima. Kubernetes pruža napredne mogućnosti za orkestraciju kontejnera, kao što su automatsko skaliranje, upravljanje opterećenjem i otpornost na greške, što bi dodatno poboljšalo performanse i pouzdanost sistema [10].

Kako bi sistem bio usklađen sa zakonskim regulativama u različitim državama, neophodno je proučiti i implementirati relevantne zakonske zahteve. Ovo uključuje usklađivanje sa zakonima o zaštiti podataka, kao što je *GDPR* u Evropi, kao i sa standardima za bezbednost platnih kartica (*PCI DSS*). Implementacija zakonskih regulativa osigurava da sistem ostane u skladu sa zakonskim okvirima, čime se smanjuje rizik od pravnih posledica i povećava poverenje korisnika [11,12].

Kako bi sistem ostao konkurentan na dinamičnom tržištu elektronskih plaćanja, neophodno je kontinuirano proširivati funkcionalnosti i prilagođavati se novim tehnologijama i trendovima.

9. LITERATURA

[1] Blog Instituta za Matematiku i Informatiku u Kragujevcu, https://blog.imi.pmf.kg.ac.rs/spring-cloud-mikroservis-arhitektura/, pristupano 09.09.2024.

[2] Spring Boot Documentation, https://docs.spring.io/spring-boot/index.html , pristupano 07.09.2024.

[3] React Documentation, https://react.dev , pristupano 08.09.2024.

[4] Worsley, J., & Drake, J. D. (2002).”Practical PostgreSQL”. " O'Reilly Media, Inc."

[5] Docker Documentation, https://docs.docker.com/ , pristupano 09.09.2024.

[6] IPS sistem plaćanja Narodne Banke Srbije, https://www.nbs.rs/sr/ciljevi-i-funkcije/platni-sistem/nbs-operator/ips-nbs/index.html , pristupano 12.09.2024.

[7] Paypal integration for Spring-Boot backend https://medium.com/@lsampath210/paypal-integration-for-spring-boot-backend-243e71c89a74 , pristupano 09.09.2024.

[8] Osnove Crypto tehnologije, https://www.coinbase.com/learn/crypto-basics, pristupano 12.09.2024

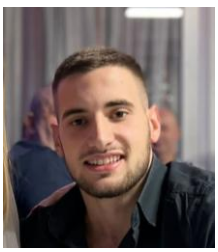
[9] Hashing it out with Hedera Hashgraph https://www.messari.io/report/hashing-it-out-with-hedera-hashgraph pristupano 08.09.2024.

[10] Kubernetes Documentation, https://kubernetes.io/docs/home/ , pristupano 10.09.2024.

[11] Zakon o zaštiti podataka EU, https://gdpr-info.eu/ , pristupano 14.09.2024

[12] Payment Card Industry Security Standards Council, https://www.pcisecuritystandards.org/ , pristupano 12.09.2024.

Kratka biografija:



Mladen Gajić rođen 2000. godine u Loznici, odrastao je u Malom Zvorniku. Srednju tehničku školu “Ivan Sarić” u Subotici završio je 2019. godine. Fakultet Tehničkih Nauka u Novom Sadu upisuje iste te 2019 godine, i postaje student jednog od najprestižnijih fakulteta u Srbiji.

kontakt: mladengajic00@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.7
DOI: https://doi.org/10.24867/30BE38Stjepanovic

JEDNO REŠENJE SIMULACIJE SOME/IP KOMUNIKACIJE NA ELEKTRONSKOJ KONTROLNOJ JEDINICI (ECU)

ONE SOLUTION FOR SOME/IP COMMUNICATION SIMULATION ON ELECTRONIC CONTROL UNIT (ECU)

Jelena Stjepanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – OBRADA SIGNALA

Kratak sadržaj – *Ovaj rad se bavi kreiranjem servera i klijenta u okviru jedne elektronske kontrolne jedinice (Electronic Contol Unit - ECU) i uspostavom SOME/IP (Scalable Service-Oriented MiddlewarE over IP) komunikacije između istih. U uvodu je rečeno nešto više o elektronskoj kontrolnoj jedinici i samoj temi rada. U drugom poglavlju opisana je standardizovana softverska arhitektura u automobilskoj industriji, dok je celo treće poglavlje posvećeno SOME/IP komunikaciji. U četvrtom poglavlju se nudi koncept rešenja, dok peto i šesto poglavlje opisuju njegovu realizaciju i verifikaciju – respektivno. U sedmom poglavlju izveden je zaključak, a u osmom je dat pregled literature.*

Ključne reči: *ECU, SOME/IP, komunikacija, server, klijent, protokol*

Abstract – *This paper deals with the creation of a server and a client within one electronic control unit (Electronic Control Unit - ECU) and the establishment of SOME/IP (Scalable Service-Oriented Middleware over IP) communications between them. In the introduction, something more was said about the electronic control unit and the topic of the work itself. The second chapter describes the standardized software architecture in the automotive industry, while the entire third chapter is devoted to SOME/IP communication. The fourth chapter offers the concept of the solution, while the fifth and sixth chapters describe its implementation and verification - respectively. In the seventh chapter, a conclusion is drawn, and in the eighth, an overview of the literature is given.*

Keywords: *ECU, SOME/IP, communication, server, client, protocol*

1. UVOD

Elektronska kontrolna jedinica (*electronic control unit* – ECU), kao jedan od najvažnijih delova automobila, predstavlja računar koji izvršava sve operacije vezane za pravilan rad i funkcionisanje vozila.
U vozilu postoji više jedinica koje imaju različite funkcije. Samostalna elektronska kontrolna jedinica, koja nije povezana sa ostalim jedinicama nema veliki značaj niti može ostvariti bilo kakvu funkciju. Pravi smisao ECU se postiže tek uspostavom komunikacije, odnosno uvezivanjem svih ECU datog vozila u komunikacionu mrežu gde će promene svake od njih biti redovno i pravovremeno ažurirane, a njihove funkcionalnosti ostvarene kroz međusobno primanje ili slanje poruka i odgovarajuće ponašanje u skladu sa istim.

Kada je u pitanju komunikacija, u automobilskoj industriji se sve više potencira SOME/IP (*Scalable Service-Oriented MiddlewarE over IP*) protokol kao najadekvatniji i po mnogim merilima najfleksibilniji. S obzirom na to da popularnost ovog protokola sve više raste, interesantno je ispitati i implementirati neke od njegovih funkcionalnosti.

Ovaj rad se bavi kreiranjem servera i klijenta u okviru jedne ECU i uspostavom SOME/IP komunikacije između istih. Inovativnost koju bi donela implementacija navedenog ogleda se u činjenici da se SOME/IP protokol uglavnom koristi za komunikaciju između dve različite ECU. S obzirom na to da se u okviru date ECU uglavnom nalazi više softverskih komponenti, korisno je znati da se komunikacija između datih komponenti može ostvariti uz sve mogućnosti SOME/IP komunikacije, pa u tom smislu nije potrebno izmeštati softverske komponente na različite ECU, niti pristajati na kompromis u smislu korišćenja nekog vida interne komunikacije.

2. Softver za automobile – AUTOSAR

AUTOSAR (*AUTomotive Open System ARchitecture*), predstavlja otvorenu, standardizovanu softversku arhitekturu za automobilsku industriju. AUTOSAR je standardizovao dve softverske platforme – *Classic* i *Adaptive*. Danas je AUTOSAR *Classic* u širokoj upotrebi i takođe je prvi izbor za namenske elektronske kontrolne jedinice sa visokim standardima u pogledu sigurnosti i determinističkog izvršenja. AUTOSAR *Classic Platform* arhitektura se raspoznaje na najvišem nivou apstrakcije između tri softverska sloja koja se pokreću na mikrokontroleru: aplikacija, izvršno okruženje i osnovni softver [1].

3. SOME/IP

SOME/IP je razvijen od strane BMW grupe 2011. godine. Kao što ime govori, reč je o rešenju u vidu komunikacionog srednjeg sloja koje se oslanja na TCP/IP (*Transmission Control Protocol*) stek. Dizajniran je da se prilagodi uređajima različitih veličina i različitim

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Dejan Vukobratović.

operativnim sistemima, takođe, podržava funkcije *Infotainment* domena kao i druge domene u vozilu, što mu omogućava da se koristi kao zamena za MOST (*Media Oriented Systems Transport*), kao i zamena za tradicionalna CAN (*Controller Area Network*) scenarija.

3.1. Ključne funkcionalnosti SOME/IP protokola

Neke od glavnih funkcionalnosti SOME/IP protokola su [2]:

• Obezbeđuje servisno orijentisanu komunikaciju unutar mreže

• Podržava širok opseg funkcionalnosti kao posredni softver: *Remote Procedure Call* (RPC), *Service Discovery* (SD), *Publish/Subscribe (Pub/Sub)*

• Segmentacija UDP (*User Datagram Protocol*) poruke (omogućavajući transport velikih SOME/IP poruka preko UDP-a bez potrebe za fragmentacijom)

• Može biti implementiran na različitim operativnim sistemima ili čak na ugrađenim sistemima koji nemaju operativni sistem

• Koristi se za klijent/server serijalizaciju unutar ECU

3.1.1 SOME/IP kao vrsta RPC-a

RPC protokol olakšava komunikaciju između umreženih računara. Program na jednom računaru na mreži koristi RPC da bi došao do programa na drugom računaru na mreži bez poznavanja detalja mreže.

Servis koji nudi SOME/IP može biti sačinjen od nula (dozvoljen je prazan servis) ili više metoda (*methods*), događaja (*events*), ili polja (*fields*) [3].

• Metode donose mogućnost klijentu da izdaje RPC koji se izvršava na strani servera. Razlikuju se *Request/Response* metode i *Fire&Forget* metode.

Request/Response predstavlja jedan od najčešćih obrazaca komunikacije. Jedan učesnik komunikacije (klijent) šalje poruku zahteva na koju odgovara drugi učesnik komunikacije (server). *Fire&Forget* predstavlja oblik komunikacije zasnovan na upućenom zahtevu serveru od strane klijenta bez povratne poruke, odnosno odgovora od strane servera.

• Događaji predstavljaju podatke koji se šalju ciklično ili prilikom svake promene od provajdera ka pretplatniku. Uvek se javljaju u grupama (*eventgroup*).

• Polja su kombinaciju jednog ili više od ponuđenih: *notifier, getter, setter.*

3.1.2 Servisno orjentisana komunikacija

Komunikacija zasnovana na signalu (*Signal-based communication*) već se dugo koristi u poznatijim komunikacionim protokolima. Ovakva komunikacija pretpostavlja da softver neće biti modifikovan. Takođe, podaci se šalju mrežom kad god se vrednosti podataka ažuriraju. Pošiljaoca ne brine da li čvor u mreži zahteva podatke, a ovakav pristup može opteretiti čvorove podacima koji im možda nikada neće biti potrebni [3].

U slučaju servisno orijentisane arhitekture (*Service-based*), pošiljalac šalje podatke samo ako su primaocu potrebni. Stoga, u takvom aranžmanu, server mora biti obavešten o prijemnicima koji čekaju podatke.

3.1.2.1 *Service Discovery*

U pogledu servisno orijentisane komunikacije, presudna uloga SOME/IP protokola se ogleda u *Service Discovery* (SD) modulu. Ovaj modul se koristi da eksplicitno signalizuje [4]:

• Status servisne instance (dostupna ili ne). Ukoliko je dostupna, kako doći do nje.

• Mehanizam pretplate i objave (klijent se pretplaćuje na željeni sadržaj). Nakon što se klijent pretplatio na željeni sadržaj zadatak SD-a je birati događaje/polja (*events/fields*) koji su potrebni klijentu.

3.2. SOME/IP *Binding*

ara::com predstavlja apstraktni C++ API komunikacionog posrednog softvera u okviru AUTOSAR platforme. U slučaju kada se SOME/IP koristi kao transportni sloj u arhitekturi ara::com-a uočljiva su tri dela: ara::com *frontend* koji je vidljiv korisnicima posrednog softvera uključujući *skeleton* i *proxy*. Drugi deo jeste *backend* i odnosi se na implementaciju same konekcije između modula *(binding-specific*). Treći deo jeste SOME/IP *daemon* [5].

Prva dva dela se integrišu u svaku aplikaciju posebno, dok je SOME/IP *daemon* odvojen i može biti korišćen od strane više različitih aplikacija. SOME/IP *daemon* je odlika ECU i predstavlja posrednika za svu komunikaciju.

SOME/IP *binding* ima neke od sledećih karakteristika [5]:

• Aplikacija koja koristi SOME/IP *binding* zahteva da se SOME/IP *daemon* pokrene

• SOME/IP *daemon* ne pruža podršku za monitoring mreže, pa se može desiti da *Service Discovery* uđe u inicijalnu fazu čekanja *(Initial Wait Phase*) čak i ako je veza prekinuta

• Za sve SOME/IP servise, komunikacija se obavlja preko IP soketa. Ovo znači da SOME/IP *daemon* uvek koristi kompletni mrežni stek za komunikaciju. IP poruke će biti poslate i primljene nazad od strane SOME/IP *daemon*-a. Sa konfiguracijske tačke gledišta, u slučaju servisa koji koriste SOME/IP, ne bi trebalo da postoje razlike bilo da su aplikacije u istoj ili u različitim ECU.

• SOME/IP *daemon* dodeljuje memoriju na zahtev.

4. KONCEPT REŠENJA

S obzirom na to da je cilj ovog rada simulirati SOME/IP komunikaciju na B2 ploči koja predstavlja elektronsku kontrolnu jedinicu, koncept rešenja je sadržan u sledećem:

• Izabrati dve od postojećih softverskih komponenti u datom steku i u okviru jedne napisati novu klijentsku, a u okviru druge novu serversku aplikaciju u skladu sa već postojećim servisima

• Napraviti biblioteke i integrisati ih sa datim softverskim komponentama, a zatim integrisati biblioteke u postojeći projekat

• Obezbediti podršku za komunikaciju sa pločom

• Rešenje verifikovati razmenom poruka između klijenta i servera i to ispisivanjem poruka na konzoli

5. REALIZACIJA REŠENJA

U skladu sa servisima koji već postoje u steku, napravljen je novi servis - *NewService* sa ulogom klijenta i *NewServerService* sa ulogom servera.

5.1. Konfiguracija ARXML-fajlova

S obzirom na to da se svi servisi koji se nalaze u datom steku, generišu na osnovu ARXML-fajlova koji detaljno opisuju dati servis, u svrhu dodavanja klijenta i servera, prethodno je bilo potrebno napraviti za svaki od njih novi ARXML-fajl, pri čemu su ovi fajlovi napravljeni ručno po uzoru na ARXML-fajlove drugih servisa. Podešavanje parametara u okviru ARXML-fajlova vrši se na taj način da klijent i server budu kompatibilni, odnosno da se međusobno prepoznaju i komuniciraju, u čemu se i ogleda kompleksnost zadatka. Kada se govori o prepoznavanju klijenta i servera, u slučaju SOME/IP protokola tu ulogu izvršava *Service Discovery* modul.

• Na serverskoj strani postoje tri parametra koja definišu različite faze SOME/IP *Service Discovery* protokola: *Initial Wait Phase, Repetition Wait Phase, Main Phase.*

•Ukoliko su klijent i server kompatibilni, odnosno, ukoliko klijent pronađe servera, pretplaćuje se na njegov sadržaj (*SubscribeToEvent*) i prima podatke koje je server poslao još u početnoj fazi.

Prilikom konfiguracije klijenta i servera, parametri od ključnog značaja su bili: *service id, instance id, major version* i *minor version*. Da bi servis koji nudi server odgovarao onom koji klijent traži, navedeni parametri treba da zadovoljavaju uslove *matches_service* (1) funkcije, gde su servisi čija se kompatibilnost proverava označeni sa self i other:

```
def matches_service(self, other: Service) -> bool:

    if self.service_id != other.service_id:
        return False
    if (
        self.instance_id != 0xFFFF
        and other.instance_id != 0xFFFF
        and self.instance_id != other.instance_id                    (1)
    ):
        return False
    if (
        self.major_version != 0xFF
        and other.major_version != 0xFF
        and self.major_version != other.major_version
    ):
        return False
    if (
        self.minor_version != 0xFFFFFFFF
        and other.minor_version != 0xFFFFFFFF
        and self.minor_version != other.minor_version
    ):
        return False

    return True
```

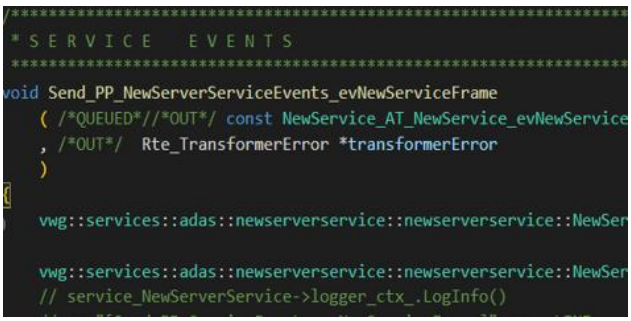
Na osnovu navedenog može se zaključiti da će klijent i server biti kompatibilni ako imaju jednake vrednosti *service id* parametra i ukoliko su vrednosti za *instance id, major version* i *minor version* ili jednake ili, bilo na klijentskoj, bilo na serverskoj strani postavljene na neku od džoker vrednosti – kada je u pitanju *instance id* to je 0xFFFF, za *major version* to je 0xFF, a za *minor version* to je 0xFFFFFFFF. U Tabeli 1. prikazane su vrednosti koje su dodeljene klijentu i serveru. Treba napomenuti da se do prethodnih zaključaka došlo eksperimentalnim putem, odnosno, primenjene su različite kombinacije vrednosti konfiguracionih parametara pre nego što je dobijena odgovarajuća kombinacija. Pri istraživanju je korišćena brojna dokumentacija iz AUTOSAR i ara::com standarda.

Tabela 1. *Konfiguracioni parametri*

POLJE	KLIJENT	SERVER
service id	24748	24748
instance id	65535	1
major version	255	1
minor version	4294967295	0
event id	32769	32769
event group id	32768	32769
udp port	42809	42810
TTL	3	3

5.2. Pravljenje klijentske i serverske aplikacije

Zadatak je osmišljen tako da se preko SOME/IP protokola šalje *event*. U tu svrhu, na serverskoj strani su napisane sve neophodne funkcije koje služe za ponudu i slanje *event*-a (Slika 1). Analogno tome, na klijentskoj strani su napisane neophodne funkcije koje služe za potražnju servisa, pretplaćivanje na servis i primanje sadržaja. U funkcije su dodati ispisi u cilju praćenja toka komunikacije kada se aplikacije spuste na ploču i mogućnosti ispravljanja eventualnih nepravilnosti.



Slika 1 *Definicija Send funkcije na serverskoj strani koja služi za slanje event-a*

5.2.1. Verifikacija koda

Nakon što su server i klijent napravljeni, neophodna je verifikacija napisanog koda. To se postiže pravljenjem biblioteka i njihovom integracijom sa softverskim komponentama.

Dve dobijene biblioteke potrebno je integrisati u postojeći projekat i obezbediti podršku za komunikaciju sa pločom, nakon čega se može pratiti tok razmene poruka.

6. VERIFIKACIJA REŠENJA

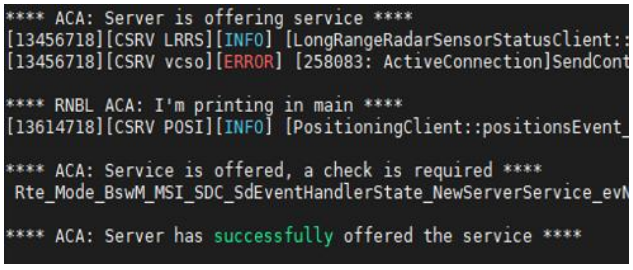
Nakon što su izvršena sva podešavanja vezana za ploču, moguće je preko *MobaXterm* konzole pratiti tok izvršavanja svih segmenata komunikacije između klijenta i servera.

Ispisi sa zvezdicama su iz testnih funkcije (jedna za klijenta, jedna za servera), dok ispisi bez zvezdica predstavljaju ispise iz samog steka.

Testne funkcije su osmišljene na taj način da se iz datog *case*-a prelazi u sledeći tek ako je ispunjen dati uslov (klijent prima servis tek ukoliko prođe proveru da se uspešno pretplatio). Na serverskoj strani testna funkcija sadrži dva *case*-a. U okviru prvog se vrši *offer*-ovanje servisa, a potom, ukoliko je ispunjen uslov da je servis uspešno *offer*-ovan, u sledećem *case*-u se vrši slanje *event*-a.

Na klijentskoj strani nulti *case* se uvek izvršava i podrazumeva traženje servera preko funkcije *StartFindService*(). U sledeći *case* se prelazi pod uslovom da je pronađen server, i u tom slučaju vrši se *subscribe*-ovanje. Ukoliko se klijent uspešno *subscribe*-ovao, prelazi se u naredni *case* gde klijent prima *event* preko funkcije *Receive*().

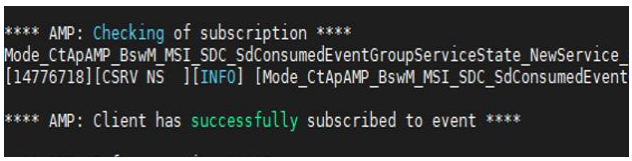
Tok dešavanja na serverskoj strani prikazan je na Slici 2. Kao što je prikazano, server je uspešno ponudio servis. Nakon toga je poslao podatke, čekajući klijenta koji će moći da ih primi. Treba napomenuti da se prethodno navedeno dešava u više ciklusa, odnosno, server konstantno nudi servis čekajući klijenta koji će da se pretplati i primi podatke.



Slika 2 *Server nudi servis preko Offer funkcije*

Sa druge strane, klijent kreće u potragu za serverom. Vrši se ciklična provera da li je klijent uspeo da pronađe server ili nije. Nakon uspešnog pronalaska servera, na konzoli se ispisuje poruka: *Server is found.*

Nakon što pronađe server, klijent pokušava da se pretplati. Poruka *Client has successfully subscribed to event* označava da se klijent uspešno pretplatio (Slika 3).



Slika 3 *Klijent se uspešno pretplatio na event koji je poslao server*

Nakon uspešnog *subscribe*-ovanja, klijent može da primi podatke od servera pozivajući funkciju *Receive*(). Da bi se ispitala funkcionalnost komunikacije potrebno je inače praznu strukturu koja predstavlja *event*, popuniti podacima čiji je tip definisan u *.typedef* fajlu.

7. ZAKLJUČAK

SOME/IP komunikacija kao relativno novo rešenje se ne može svrstati u oblasti koje počivaju na konačnim i strogim principa kao što bi to bio slučaj sa nekim starijim i duže vremena primenjivanim rešenjima iz bilo koje oblasti, jer u slučaju ovog rešenja tek mnogo toga treba da se istraži i postoji dosta prostora za otkrivanje i improvizaciju.

Sama činjenica da je u ovom radu dokazano da je moguće koristiti SOME/IP protokol za uspostavljanje komunikacije u okviru iste ECU iako se ovaj protokol uglavnom koristi za komunikaciju između dve različite ECU, daje podstrek da se ovom rešenju u budućnosti pokloni još više pažnje i da se ispitaju sve druge mogućnosti. Takođe, dobijeni rezultati pokazuju da i u okviru steka koji je korišćen postoji dosta prostora za razvoj i unapređivanje, pa s tim u vezi, može se reći da je glavni cilj ovog rada postignut, pri čemu se ostavlja prostora za dalje unapređivanje performansi, kako SOME/IP komunikacije, tako i steka.

8. LITERATURA

[1] V.I. Utkin, “Variable structure control systems with sliding modes”, *IEEE Trans. Automat. Control*, Vol. AC-22, pp. 210-222, April 1977.

[2]https://www.scribd.com/document/450637862/SOME-IP-Intro [Accessed 04 07 2021]

[3] https://www.embitel.com/blog/embedded-blog/how-some-ip-enables-service-oriented-architecture-in-ecu-network [Accessed 05 07 2021]

[4]https://www.autosar.org/fileadmin/user_upload/standards/foundation/10/AUTOSAR_RS_SOMEIPServiceDiscoveryProtocol.pdf [Accessed 29 08 2021]

[5] Technical Reference Communication Management, Vector [Accessed 10 10 2021]

Kratka biografija:

Jelena Stjepanović rođena je u Zvorniku 1997. god. *Bachelor* rad na Fakultetu tehničkih nauka iz oblasti Telekomunikacije i obrada signala odbranila je 2020.god.
kontakt: jstjepanovic1111@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38.049.77
DOI: https://doi.org/10.24867/30BE39Popadic

ПРОЈЕКТОВАЊЕ УНИВЕРЗАЛНОГ БИКВАДРАТНОГ Gm-C ФИЛТРА

DESIGN OF UNIVERSAL MULTIMODE Gm-C BIQUAD FILTER

Горан Попадић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду је пројектован универзални биквадратни MISO (енгл. Multiple Input Single Output) филтар у 0,13 μm CMOS UMC технологији. Филтар, који има три улаза и један излаз, може да ради у више режима: напонском, транскондуктансном, струјном и трансрезистивном. Пројектовани филтар има граничну (централну) фреквенцију око 251 MHz и потрошњу око 4,5 mA.*

Кључне речи: *универзални Gm-C филтар, операциони транскондуктансни појачавач, коло за поларизацију, фактор доброте, транскондуктанса, корнер анализе.*

Abstract *– In this paper an universal biquadratic MISO (Multiple Input Single Output) filter is designed in 0,13 μm CMOS UMC technology. The filter, having three inputs and one output, can operates in multiple modes: voltage, transconductance, current, and transresistance. The designed filter has a cut-off (center) frequency of about 251 MHz and a current consumption of about 4.5 mA.*

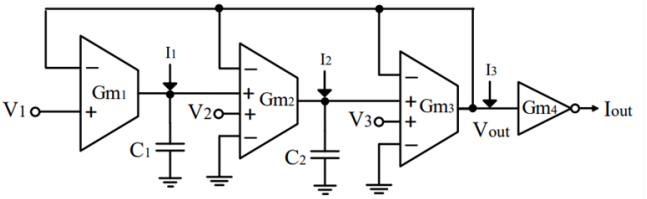
Keywords: *universal Gm-C filter, Operational Transconductance Amplifier (OTA), biasing circuit, quality factor, transconductance, corner analyses.*

1. УВОД

Електрични филтри су кола која обликују фреквенцијски спектар улазног електричног сигнала на основу прописаних захтева. У овом раду је пројектован филтар заснован на операционим транскондуктансним појачавачима (енгл. *Operational Transconductance Amplifier – OTA*). Предност употребе ових појачавача је лако подешавање параметара филтара, што их чини веома погодним за примену у интегрисаним колима. OTA-C или Gm-C филтри користе само појачаваче и кондензаторе па се једноставно реализују.

Њихове предности, попут мале површине, потрошње и цене, им обезбеђују широку примену. На пример, могу се пронаћи у аудио опреми, у мрежи са три канала за распоређивање фреквенцијских домена у три опсега (ниски, средњи и високи) која садржи три филтра (нископропусник, високопропусник и пропусник опсега фреквенција) [1].

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



Слика 1. *Предложена топологија универзалног биквадратног Gm-C филтра*

2. АНАЛИЗА УНИВЕРЗАЛНОГ Gm-C ФИЛТРА

На слици 1 приказана је топологија предложеног филтра који се састоји од четири степена. Први степен представља OTA коло са два улаза. Други и трећи степен представљају OTA кола са четири улаза [2]. У транскондуктансном и струјном режиму рада се користи инвертујући појачавач (инвертор) на излазу. Дати филтар има могућност реализације свих филтарских функција: пропусник ниских фреквенција НФ (енгл. *Low Pass* – LP), пропусник опсега фреквенција ПО (енгл. *High Pass* – HP), пропусник високих фреквенција ВФ (енгл. *Band Pass* – BP), непропусник опсега фреквенција НО (енгл. *Band Stop* – BS) и пропусник свих опсега фреквенција ПФ (енгл. *All Pass* – AP).

Преносна функција филтра у **напонском режиму** [2]:

$$V_{out} = \frac{Gm_1Gm_2(V_1) + SC_1Gm_2(V_2) + S^2C_1C_2(V_3)}{D(s)}. \quad (1)$$

Преносна функција филтра у **струјном режиму** [2]:

$$I_{out} = -\frac{Gm_2Gm_3Gm_4(I_1) + SC_1Gm_3Gm_4(I_2) + S^2C_1C_2(I_3)}{D(s)}. \quad (2)$$

Преносна функција филтра у **трансрезистивном режиму** [2]:

$$Vout = \left(\frac{1}{Gm_3}\right)\frac{Gm_2Gm_3(I_1) + SC_1Gm_3(I_2) + S^2C_1C_2(I_3)}{D(s)}. \quad (3)$$

Преносна функција филтра у **транскондуктансном режиму** [2]:

$$I_{OUT} = -\frac{Gm_1Gm_2Gm_4(V_1) + SC_1Gm_2Gm_4(V_2) + S^2C_1C_2Gm_4(V_3)}{D(s)}. \quad (4)$$

Полином у имениоцу је престављен једначином:

$$D(s) = S^2C_1C_2 + SC_1Gm_2 + Gm_1Gm_2. \quad (5)$$

Фактор доброте је дат једначином [2]:

$$Q = \sqrt{\frac{C_2Gm_1}{C_1Gm_2}}. \quad (6)$$

Централна фреквенција је дата једначином [2]:

$$\omega_0 = \sqrt{\frac{Gm_1Gm_2}{C_1C_2}}. \quad (7)$$

У напонском и транскондуктансном режиму рада струје I_1, I_2 и I_3 су једнаке нули. Напони V_1, V_2 и V_3 су једнаки нули у струјном и трансрезистивном режиму. Реализација одзива филтра је приказана у табели 1.

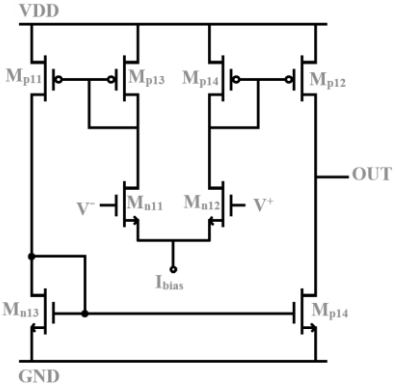
Табела 1. *Добијање различитих одзива филтра*

Тип филтра (неинвертујући)	***Напонски режим***			***Трансрезистивни режим***		
	V_1	V_2	V_3	I_1	I_2	I_3
НФ	V_{in}	0	0	I_{in}	0	0
ПО	0	V_{in}	0	0	I_{in}	0
ВФ	0	0	V_{in}	0	0	I_{in}
НО	V_{in}	0	V_{in}	I_{in}	0	I_{in}
ПФ	V_{in}	V_{in}	V_{in}	I_{in}	I_{in}	I_{in}
Тип филтра (инвертујући)	***Струјни режим***			***Транскондуктан-сни режим***		
	I_1	I_2	I_3	V_1	V_2	V_3
НФ	I_{in}	0	0	V_{in}	0	0
ПО	0	I_{in}	0	0	V_{in}	0
ВФ	0	0	I_{in}	0	0	V_{in}
НО	I_{in}	0	I_{in}	V_{in}	0	V_{in}
ПФ	I_{in}	I_{in}	I_{in}	V_{in}	V_{in}	V_{in}

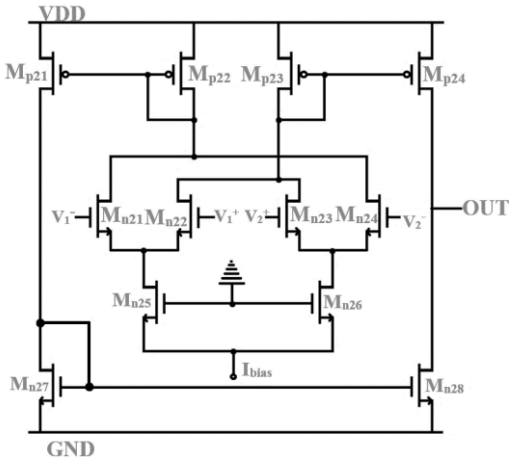
3. ПРОЈЕКТОВАЊЕ УНИВЕРЗАЛНОГ БИКВАДРАТНОГ Gm-C ФИЛТРА

Предложени филтар је пројектован у 0,13 μm UMC CMOS технологији. Коришћени су RF модели транзистора који раде на напону од 1,2 V и имају минималну дужину канала 120 nm. На слици 2 дат је шематски приказ пројектованог двоулазног OTA кола који представља први степен пројектованог филтра [3].

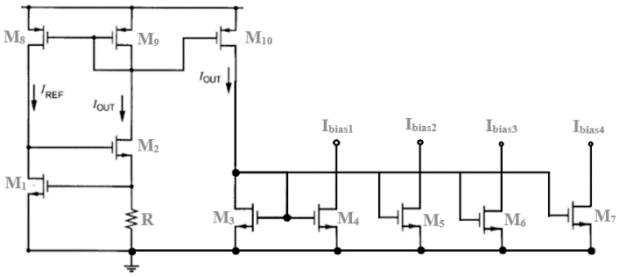
На слици 3 приказано је четвороулазно OTA коло које чини други и трећи степен пројектованог филтра [2]. Такође је потребно имати добро пројектован струјни извор, како би се смањила промена струје (и тиме обезбедиле мање варијације транскондуктансе и централе фреквенције филтра) при процесним, напонским и температурним променама. Дужина канала транзистора струјног извора је већа од минималне и износи 200 nm, како би се повећала прецизност (смањила систематска грешка). На слици 4 је приказана електрична шема струјног извора [4]. Вредност отпорника је 5,6 kΩ. Са терминала I_{bias1} се доводи струја за први, са I_{bias2} за други и са I_{bias3} за трећи степен филтра. Улога терминала I_{bias4} је поменута касније у раду. У табели 2 су приказане димензије транзистора пројектованих транскондуктансних појачавача.



Слика 2. *OTA коло са два улаза*



Слика 3. *OTA коло са четири улаза*



Слика 4. *Струјни извор [4]*

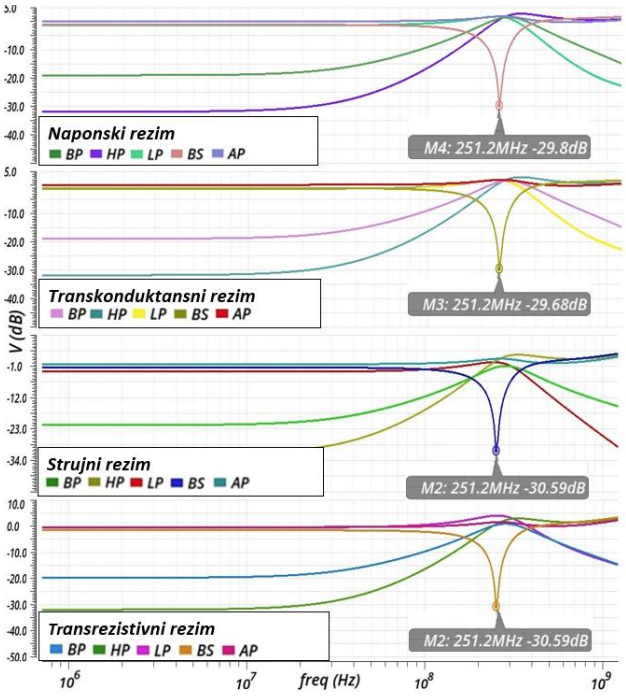
Табела 2. *Димензије транзистора OTA кола*

Транзистор	**Број прстију**	**Број транзистора**	**Ширина канала [μm]**
M_{p11}, M_{p12}	4,0	6,0	2,0
M_{p13}, M_{p14}	4,0	2,0	2,0
M_{n11}, M_{n12}	4,0	4,0	0,9
M_{n13}, M_{n14}	4,0	1,0	3,5
M_{p21}, M_{p24}	4,0	8,0	2,0
M_{p22}, M_{p23}	4,0	2,0	2,0
M_{n21}, M_{n22}, M_{n21}, M_{n24}	4,0	4,0	3,5
M_{n25}, M_{n26}	4,0	8,0	2,0
M_{n27}, M_{n28}	4,0	1,0	1,1
$pmos_{invertor}$	4,0	1,0	2,4
$nmos_{invertor}$	4,0	1,0	2,0
M_1, M_2	4,0	4,0	2,0
M_3	4,0	2,0	2,0
M_4	4,0	6,0	2,0
M_5, M_6, M_7	4,0	12,0	2,0
M_8, M_9, M_{10}	4,0	1,0	2,0

4. РЕЗУЛТАТИ СИМУЛАЦИЈА И ДИСКУСИЈА

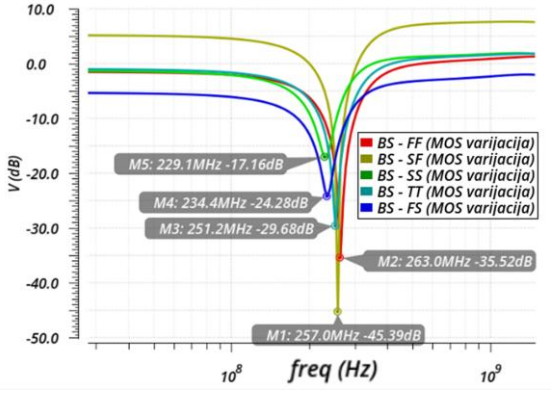
Централна фреквенција пројектованог филтра је око 251 MHz, док је фактор доброте приближно једнак јединици. Вредности транскондуктанси износе $G_{m1} = G_{m2} \approx 4$ mS, а вредности капацитивности су $C_1 = C_2 = 2{,}6$ pF. Напајање је симетрично и износи ±0,6 V. Укупна потрошња кола је 4,5 mA. Струја I_{bias1} је 150 μA, док су струје I_{bias2} и I_{bias3} једнаке 300 μA. На слици 5 су приказани резултати анализа за различите режиме рада филтра.

Утицај промена процеса је анализиран коришћењем доступних модела са максималним одступањем технолошких параметара процеса (корнер модела). За транзисторе је посматрано следећих пет случајева:



Слика 5. *Резултати симулација за све режиме рада*

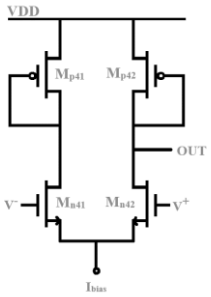
TT – типични (номинални) корнер (енгл. *Typical NMOS – Typical PMOS*), **SS** – спор NMOS - спор PMOS (енгл. *Slow NMOS – Slow PMOS*), **FS** – брз NMOS -спор PMOS (енгл. *Fast NMOS – Slow PMOS*), **SF** – спор NMOS - брз PMOS (енгл. *Slow NMOS – Fast PMOS*), **FF** – брз NMOS - брз PMOS (енгл. *Fast NMOS – Fast PMOS*). Приликом корнер анализа се може уочити промена појачања за SF и FS корнере у транскондуктансном и струјном режиму рада. Ово понашање је очекивано и јавља се као последица промене заједничког (DC) напона на излазу инвертора, због тога што је увек један транзистор доминантнији и вуче напон на своју страну, слика 6.



Слика 6. *Транскондуктансни режим – инвертор*

5. УНАПРЕЂЕЊЕ ТОПОЛОГИЈЕ Gm-C ФИЛТРА

Како би се избегле значајне промене карактеристика филтра услед промене заједничког напона на излазу инвертора за транскондуктансни појачавач на излазу филтра је употребљен диференцијални појачавач са диодним транзисторима као оптерећењем (уместо инвертора). Електрична шема кола је приказана на слици 7. У табели 3 приказане су димензије транзистора пројектованог диференцијалног појачавача. На терминал I_{bias} се доводи струја са прикључка I_{bias4} са слике 4. Ова струја има велику вредност од 300 μA како би се постигао заједнички напон од 0 V на излазу. На позитиван улаз се доводи излаз из трећег степена (заједнички напон од 0 V), док се негативан улаз повезује на масу. Са терминала *OUT* се очитавају карактеристике за транскондуктансни и струјни режим рада.

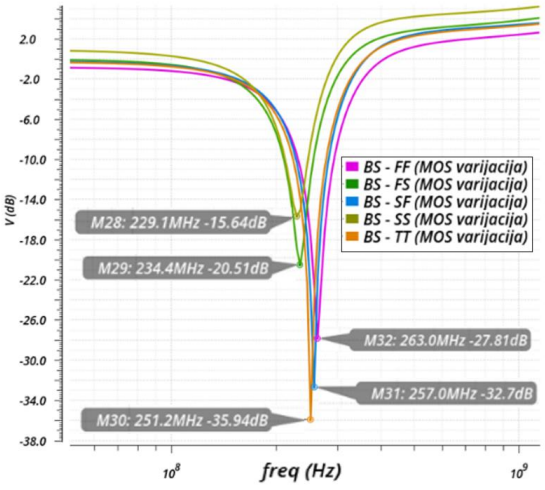


Слика 7. *Диференцијални појачавач са диодно везаним транзисторима у оптерећењу*

Табела 3. *Димензије транзистора диференцијалног појачавача*

Транзистор	**Број прстију**	**Број транзистора**	**Ширина канала [μm]**
M_{p41}, M_{p42}	4	1	2
M_{n41}, M_{n42}	4	2	2

Резултати симулација за све режиме рада су доста слични већ приказаним резултатима, уз знатно мања одступања карактеристика у корнер анализама (нпр. појачање у пропусном опсегу). Вредност централне фреквенције је око 251 MHz, док се слабљење у непропусном опсегу разликује за транскондуктансни режим и износи -35 dB, док за струјни износи -28 dB. На слици 8 приказани су резултати корнер анализа за транскондуктансни режим рада филтра након замене инвертора са диференцијалним појачавачем.



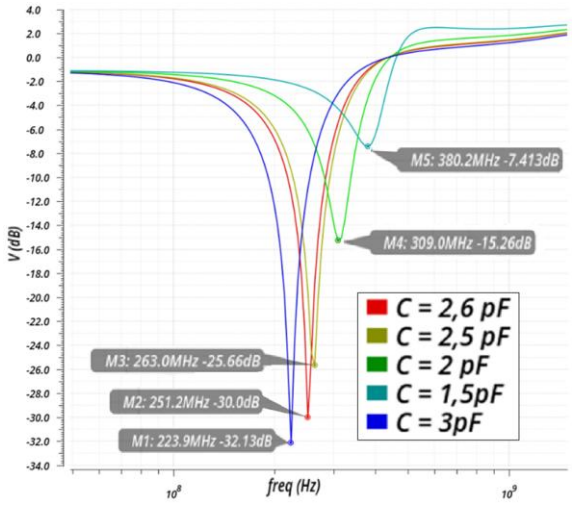
Слика 8. *Транскондуктансни режим – диференцијални пар*

У табели 4 дати су резултати корнер анализа при процесним, напонским и температурним променама: вредности централне фреквенције (f_0) и слабљења (α_N,

α_{TK}, α_S, α_{TR} редом за напонски, транскондуктансни, струјни и трансрезистивни режим рада) на централној фреквенцији у непропусном опсегу.

Табела 4. *Резултати корнер анализа*

Процесна варијација транзистора					
	TT	**SS**	**FS**	**SF**	**FF**
f_0[MHz]	251,2	229,1	234,4	257	263
α_N[dB]	-30	-16,9	-20,2	-51,8	-35,4
α_{TK}[dB]	-35,9	-15,6	-20,5	-32,7	-27,8
α_S[dB]	-28,4	-13,3	-17,4	-37,5	-27,27
α_{TR}[dB]	-30,9	-15,7	-19,18	-39,3	-27,3
Варијација температуре					
[°C]	**125**	**85**	**50**	**27**	**0**	**-40**
f_0[MHz]	204,2	229,1	245	251,2	263	275
α_N[dB]	-24,8	-24,9	-25,7	-30	-33,3	-30,8
α_{TK}[dB]	-27,6	-27,4	-27,6	-35,9	-30,7	-26,3
α_S[dB]	-22,5	25,3	-27,9	-28,4	-38,2	-30,5
α_{TR}[dB]	-23	-26,2	-19,2	-30,9	-38,1	-31,5
Варијација отпорника					
	MIN	**TYP**	**MAX**		
f_0[MHz]	281,8	251,2	229,1		
α_N[dB]	-29,2	-30	-30,8		
α_{TK}[dB]	-35,9	-35,9	-36,5		
α_S[dB]	-26,7	-28,4	-29.9		
α_{TR}[dB]	-29,3	-30,9	-32,4		
Варијација кондензатора					
	MIN	**TYP**	**MAX**		
f_0[MHz]	288,4	251,2	223,9		
α_N[dB]	-19,6	-30	-33,2		
α_{TK}[dB]	-19,9	-35,9	-27,4		
α_S[dB]	-19,1	-28,4	-28,9		
α_{TR}[dB]	-20,8	-30,9	-30,5		
Варијација напона напајања					
[mV]	**540**	**600**	**660**		
f_0[MHz]	239,9	251,2	263		
α_N[dB]	-38,7	-30	-26,04		
α_{TK}[dB]	-38,5	-35,9	-28,5		
α_S[dB]	-32	-28,4	-27,8		
α_{TR}[dB]	-32	-30,9	-29,7		



Слика 9. *Зависност централне фреквенције од вредности кондензатора*

Табела 5. *Зависност централне фреквенције од вредности кондензатора*

C1=C2 [pF]	**1,5**	**2**	**2,5**	**2,6**	**3**
f_0	**380,2**	**309**	**263**	**251,2**	**223,9**
α	**-7,41**	**-15,26**	**-25,7**	**-30**	**-32,1**

Потребно је још анализирати зависност централне фреквенције филтра од вредности капацитивности C_1 и C_2. На основу резултата приказаних на слици 9 и у табели 5, ово коло може да ради максимално до 300 MHz јер се смањује слабљење у пропусном опсегу са порастом фреквенције.

5. ЗАКЉУЧАК

У овом раду описан је поступак пројектовања универзалног биквадратног Gm-C филтра који може да ради у четири различита режима рада. Анализирана је топологија са инвертујућим појачавачем и диференцијалним појачавачем на излазу кола. Мана предложеног решења са диференцијалним појачавачем на излазу јесте већи број компоненти (већа заузета површина), док је предност мања осетљивост на процесне и напонске варијације. Одступања минимална вредност централне фреквенције од типичне вредности су: 8,76% (процесне варијације транзистора), 18,72% (температурне варијације), 8,76% (процесне варијације отпорника) и 11,16% (процесне варијације кондензатора). Одступања максималне вредност централне фреквенције од типичне вредности су: 4,78% (процесне варијације транзистора), 9,56% (температурне варијације), 11,95% (процесне варијације отпорника) и 14,74% (процесне варијације кондензатора). Варијације напона изазивају 4,78 % одступања у оба случаја.

5. ЛИТЕРАТУРА

[1] Hua-Pin Chen, Yi-Zhen Liao, Wen-Ta Lee, "Tunable mixed-mode OTA-C universal filter", *Analog Integrated Circuits and Signal Processing*, Vol. 58, pp. 135-141, Nov 2008.

[2] Mostafa Parvizi, "Design of a new low power MISO multi-mode universal biquad OTA-C filter", *International Journal of Electronics*, Vol. 106, No. 3 pp. 440-454, Nov 2018.

[3] Jean-Paul Eggennont, Denis De Ceuster, Denis Flandre, Bernard Gentinne, Paul G. A. Jespers, and Jean-Pierre Colinge, "Design of SOI CMOS Operational Amplifiers for Applications up to 300˚C", *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 2, pp. 179-186, February 1996.

[4] Vlad Anghel, Gheorghe Brezeanu, "Low Current References With Supply Insensitive Biasing", *Annals of the Academy of Romanian Scientists Series on Science and Technology of Information*, Vol. 3, No. 2/2010, pp. 7-22, 2010.

Кратка биографија:

Горан Попадић рођен је у Новом Саду 1997. године. Мастер рад на Факултету техничких наука одбранио је 2024. године.

контакт: popadicgoran97@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: ()
ДОИ: ()

АНАЛИЗА И ОПТИМИЗАЦИЈА Wav2vec 2.0 МОДЕЛА ЗА ПРЕПОЗНАВАЊЕ ГОВОРА НА СРПСКОМ ЈЕЗИКУ

ANALYSIS AND FINE-TUNING OF THE Wav2vec 2.0 MODEL FOR SERBIAN SPEECH RECOGNITION

Тамара Бановац, Владо Делић, *Факултет техничких наука, Нови Сад*

Област – ЕНЕРГЕТИКА, ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ

Кратак садржај – *Овај рад се бави применом Wav2vec 2.0 модела за препознавање говора на српском језику. Рад обухвата анализу оригиналног модела, обученог техником самонадгледаног учења на нелабелираним подацима, и fine-tuning фазу на српском језику. Оригинална имплементација, са 53000 сати нелабелираних и 10 минута лабелираних података, постигла је WER од 4.8/8.2, што потврђује ефикасност коришћених метода. Циљ је испитати применљивост ових метода на аудио подацима на српском језику. Постигнути резултати на српском језику показују WER од 10.3% и CER од 3.4%, уз могућност даљих унапређења.*

Кључне речи: *Препознавање говора, самонадгледано учење, српски језик, Wav2vec модел, трансформер*

Abstract – *This paper focuses on the application of the Wav2vec 2.0 model for speech recognition in Serbian. It includes an analysis of the original model, trained using a self-supervised learning technique on unlabeled data, and the fine-tuning phase for Serbian. The original implementation, with 53,000 hours of unlabeled and 10 minutes of labeled data, achieved a WER of 4.8/8.2, demonstrating the effectiveness of the applied methods. The goal is to explore the applicability of these methods to Serbian audio data. The results for Serbian show a WER of 10.3% and a CER of 3.4%, with room for further improvements.*

Keywords: *Speech recognition, self-supervised learning, Serbian language, Wav2vec model, transformer*

1. Увод

Надгледано учење је техника која користи унапред лабелиране податке, где свака улазна вредност има одговарајућу ознаку. Иако даје одличне резултате у областима попут препознавања говора, често је тешко обезбедити довољну количину лабелираних података за тренинг напредних модела. Самонадгледано учење (енг. *self-supervised learning*, *SSL*) решава овај проблем тако што омогућава моделима да се тренирају на великим количинама нелабелираних података, чиме се генеришу репрезентације података.

НАПОМЕНА:
Овај рад настао је из мастер рада чији ментор је био проф. др Владо Делић.

Један од најпознатијих *SSL* модела је *Wav2Vec 2.0* [1], који је трениран на нелабелираним сировим аудио подацима, а касније дообучен за препознавање говора користећи релативно мало лабелираних података. Оригинална имплементација модела, са 53000 сати нелабелираних и 10 минута лабелираних података, постигла је *WER* од 4.8/8.2.

У овом раду истражује се ефективност ових метода на српском језику. Коришћена су два *Wav2Vec 2.0* модела, са основном и проширеном архитектуром, а они су доступни у оквиру *torchaudio* библиотеке. Модели су дообучени на три базе података: *Common Voice Corpus 18.0* [2] (7 сати валидираних аудио података), Јужне Вести [3] (50.55 сати са одређеним непоклапањима између транскрипта и снимака) и *Serbian ParlaSpeech-RS* [4] (896 сати аудио снимака, такође са одређеним непоклапањима).

У наставку је описана архитектура *Wav2Vec 2.0* [1] модела и представљен је и процес тренинга, који обухвата фазу претренирања на великој количини нелабелираних података, као и фазу дообуке. Посебно је анализиран процес дообуке модела за препознавање говора на српском језику, као и коришћене базе података. Након тога, су приказани резултати које модел постиже, уз анализу метрика као што су стопа грешке у препознавању карактера и речи.

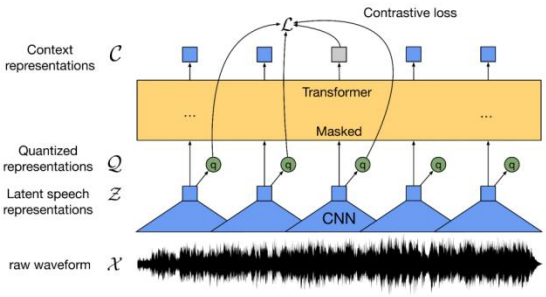
2. Архитектура *WAV2VEC 2.0* модела

Wav2Vec 2.0 модел се састоји из неколико основних целина које заједно омогућавају ефикасно учење репрезентација података погодних за различите задатке, укључујући и препознавање говора. Модел на свом улазу прима сирове аудио податке у временском облику. Прва целина је кодер обележја који се заснива на седмослојној конволуционој неуронској мрежи, а њена главна функција је да смањи димензионалност аудио података. Овај кодер пресликава скуп улазних података X у скуп вектора обележја Z. Скуп Z се састоји од T вектора $z_1, z_2, \ldots, z_T$, где је T број фрејмова који се налазе у улазном аудио фајлу, а сваки вектор z_t је повезан са по једним временским фрејмом x_t.

Након тога се примењује маскирање, где се одређени вектори обележја z_t постављају на нулу и затим шаљу на улаз трансформера [5] чија је функција да предвиди маскиране векторе. Трансформер генерише контекстуалне репрезента $c_1, c_2, \ldots, c_T$ које представљају предвиђања за маскиране векторе. Како

би се омогућило учење модела, вектори обележја z_t се квантизују путем квантизационог модула, који непрекидне вредности вектора $z_1, z_2, \ldots, z_T$ пресликава у дискретне јединице $q_1, q_2, \ldots, q_T$.

Квантизоване вредности $q_1, q_2, \ldots, q_T$ се користе за израчунавање контрастивног губитка (енг. *contrastive loss*) [6], који је кључан за обуку модела. Овај губитак пореди предвиђања трансформера са квантизованим векторима и мери степен сличности између њих, чиме модел учи репрезентације података. На слици 1 дат је приказ архитектуре модела.



Слика 1. Архитектура *Wav2vec 2.0* модела.

3. ТРЕНИНГ МОДЕЛА

Тренинг *Wav2Vec 2.0* модела за аутоматско препознавање говора се састоји из 2 фазе. Прва фаза обуке, позната и као претренирање, има за циљ учење општих репрезентација података које су довољно информативне да се могу користити за решавање различитих задатака машинског учења. Да би се ово постигло модел се тренира тако да решава контрастивни задатак (L_m) [6] уз коришћење неозначених података. Поред контрастивног губитка користи се и додатни губитак L_d (*diversity loss*), који подстиче модел да равномерно користи све елементе из кодних књига. Финална функција губитка која се минимизује у овом делу обуке модела има следећи облик:

$$L = L_m + \alpha L_d, \qquad (1)$$

где је α хиперпараметар.

Након претренирања, модел пролази кроз фазу дообуке (енг. *fine-tuning*), где се на врх трансформера додаје на случајно иницијализовани линеарни слој. Трансформер на свом излазу генерише матрицу димензија $N \times 768$ (односно $N \times 1024$ за проширену архитектуру), где је N број фрејмова у улазном аудио сигналу, а 768 представља димензију контекстуалних репрезентација за сваки фрејм. Линеарни слој који се додаје има димензије $768 \times C$, где је C број класа у речнику. Речник обухвата сва слова одговарајућег језика, у овом раду српског језика, размак и специјалне токене. Постоји само један такав линеарни слој, и исти скупови тежина примењују се паралелно на све векторе из матрице трансформера. Након примене линеарног слоја, сваки вектор димензије 768 се трансформише у вектор димензије C, што представља логит вредности за сваку класу у речнику. Затим се на ове логит вредности примењује *softmax* функција, која их претвара у вероватноће за сваку класу. Обука додатог слоја се изводи минимизацијом CTC губитка (*Connectionist Temporal Classification loss*) [7], која користи транскрипте како би модел био прилагођен за препознавање говора.

Након почетне обуке додатог линеарног слоја, препоручљиво је дообучити и све остале слојеве модела са малом брзином учења, такође користећи CTC губитак. Изузетак је кодера обележја који није потребно додатно обучавати. Овим процесом се постиже фино прилагођавање целокупног модела за конкретан задатак препознавања говора на изабраном језику.

4. ДООБУКА МОДЕЛА ЗА ПРЕПОЗНАВАЊЕ ГОВОРА НА СРПСКОМ ЈЕЗИКУ

У овом поглављу детаљно су описане коришћене базе података на српском језику, кораци предобраде података и параметри тренинга модела за препознавање говора на српском језику. Такође су приказани су резултати кроз метрике као што су *CER* (*Character Error Rate*) и *WER* (*Word Error Rate*).

4.1. Базе података

За дообуку модела за препознавање говора на српском језику коришћене су три различите базе. Прва база је *Common Voice Corpus 18.0* [2], која садржи 7 сати валидираних аудио података на српском језику, при чему се подаци одлично поклапају са транскриптима, друга је Јужне Вести [3], која има 50.55 сати аудио података и трећа и највећа база је S*erbian ParlaSpeech-RS 1.0* [4], која садржи 896 сати аудио снимака. У друге две базе постоје одређена непоклапања између транскрипта и аудио снимака, што отежава тренирање модела. У зависности од тога који модел је коришћен (модел са основном архитектуром или вишејезички *Wav2vec* модел), предобрада база се може мало разликовати због ограничених ресурса који су били доступни.

Common Voice Corpus 18.0 база садржи 2183 аудио снимака за тренинг, валидациони скуп садржи 1645 и тест садржи 1385 аудио снимака.

База Јужне вести садржи 40 сати аудио снимака за тренинг, док валидациони и тест скуп садрже по 5 сати аудио снимака. Најдужи аудио снимак има 30 секунди, а најкраћи 2.3 секунде. За тренинг модела са основном архитектуром коришћена је цела база, док су у случају тренинга вишејезичког модела из базе уклоњени предуги аудио снимци који су доводили до недостатка меморије. Након тог филтрирања базе престало је 25 сати у тренинг сету, 3 сата у валидационом и 3 сата у тест сету, а најдужи аудио снимак има 18.7 секунди.

У бази *ParlaSpeech-RS 1.0* подела на тренинг, валидациони и тест скуп није унапред доступна. Најдужи аудио снимак има 214.5 секунди, а најкраћи 0.1 секунду. Уклоњени су сви аудио снимци дужи од 20 секунди и краћи од 5 секунди. Такође, постојао је још један значајан проблем са базом. Сви бројеви који се изговарају су у транскрипту наведени као цифре уместо као речи. То представља проблем јер модел зна да тумачи само токенизоване транскрипте, а као токени се користе само слова из српске азбуке и

неколико специјалних токена. Одлучено је да се сви аудио снимци у којима се изговара било који број избаце из базе. Пошто ови снимци чине око 15% целокупне базе, таква одлука није значајно умањила обим доступних података. Након свих филтрирања, тренинг скуп садржи 316.2 сати аудио снимака, валидациони 56.6 сати и тест скуп садржи 22.4 сати аудио снимака.. Ова база је коришћена само за тренинг модела са основном архитектуром. Разлог је велика количина података због које је тренинг основног модела трајао око 37 сати, док би тренинг вишејезичког модела трајао 2 до 3 пута дуже. Због ограниченог времена доступног за израду овог рада, одлучено је да се вишејезички модел не тренира на бази *ParlaSpeech-RS*.

Како би се базе података адекватно припремиле за тренинг модела било је потребно извршити предобраду аудио снимака и транскрипта. Сваки аудио снимак је ресемплован на брзину која је коришћена за тренинг оригиналног модела (16000 Hz). Затим је сваки аудио снимак стандардизован на нулту средњу вредност и јединичну варијансу. Из транскрипта су отклоњени сви знаци интерпункције као и велика слова. Затим је транскрипт токенизован тако да сваком слову из азбуке одговара по један број. Поред токена за 30 слова азбуке и за размак између речи, користе се и специјални токени.

4.2. Токенизација

Токенизација је један од кључних корака у процесу креирања модела за препознавање говора. Током овог процеса, свако слово српске азбуке мапира се на одређени број, а користи се и токен за размак између речи. Поред ових 31 токена, користе се и специјални токени <*BLANK*> и <*UNK*>. *Blank* токен игра важну улогу у рачунању *CTC* губитка, и пожељно је да се мапира на 0. *UNK* токен се користи за означавање непознатих симбола који се могу јавити у транскрипту. Да би се омогућила обука модела кроз *batch* величине веће од 1, потребно је изједначити дужину свих аудио снимака и транскрипата унутар једног *batch*-а. <*BLANK*> токен може да се користи и као токен за *CTC* функцију, али и за обуку модела кроз *batch*-еве. У процесу дообуке мапирање токена је урађено на следећи начин: <*BLANK*> токен на 0, <*UNK*> токен на 1, слова азбуке на бројеве од 2 до 31 и размак између речи на 32.

Када се за обуку користе базе Јужне Вести и *ParlaSpeech-RS, batch* величина је увек постављена на 1 због дугих аудио снимака који трају до 20 и 30 секунди. Такви снимци онемогућавају коришћење већег *batch*-а због ограничених меморијских ресурса. У случају обуке на *Common Voice Corpus* бази користи се *batch* једнак 4.

4.3. Метрике

У процесу процене перформанси модела за препознавање говора користе су две основне метрике *Word Error Rate* (*WER*) и *Character Error Rate* (*CER*). *WER* је најчешће коришћена метрика у области препознавања говора. *WER* представља однос броја погрешних речи у препознатом тексту у односу на укупан број речи у оригиналном транскрипту. *CER* је метрика која мери проценат грешака на нивоу карактера. Ниже вредности метрика указује на боље перформансе модела.

4.4. Тренинг акустичког модела

У овом раду су испитане перформансе 2 модела. Први има основну архитектуру и претрениран је на 960 сати нелабелираних аудио података. Други коришћени модел има проширену архитектуру и обучаван је на вишејезичкој бази са 56000 сати нелабелираних аудио података. Први модел је треniран на све 3 описане базе, док је други модел треniран на базама Јужне вести и *Common Voice Corpus 18.0*.

Тренинг је рађен на следећи начин. На претренирани модел је додат један на случај иницијализовани линеарни слој, а затим је он треniран минимизацијом *CTC* губитка. Користи се *Adam* оптимизатор. После тога су искључане све претрениране тежине осим кодера обележја и додатно трениране са малом брзином учења, такође минимизацијом *CTC* губитка. Оба модела су тренирана одвојено на свакој од база. Када је тренинг на једној бази завршен, односно, када је постигнут минимални валидациони губитак, модел је сачуван и тренинг је настављен на следећој бази.

4.5. Резултати на тест сету

У овом поглављу је дат упоредни приказ резултата три модела са основном архитектуром, модел треniран само на *ParlaSpeech-RS* бази, затим тај модел дотрениран на бази Јужне вести и на крају модел дотрениран на бази *Common Voice Corpus 18.0*. Такође су приказани резултати коришћењем *greedy* декодовања и декодовања са 5-грам језичким моделом. Коришћење *greedy* декодовања заправо представља резултате које постиже сам акустички *Wav2Vec* модел, без икаквог знања о контексту које даје примена језичког модела. Као тест сет података коришћена је унија свих података из тест сетова три коришћене базе за обуку. Поред тога, посебно су приказати резултати на тест сету базе *Common Voice Corpus 18.0* јер је то једина база које има добро поравнате све аудио снимке и транскрипте, чиме обезбеђује најрелевантнији тест модела. Коришћене су метрике *CER* и *WER*. Примена језичког модела има првенствено утицај на *WER* метрику. Сви резултати су приказани у табели 1. *Common Voice Corpus 18.0* је у табелама наведена као *CV18.*

У случају коришћења уније сва три тест скупа постоји небалансираност података јер тест скуп из базе *ParlaSpeech-RS* има значајно више података у односу на друге 2 базе. Због тога су резултати сва три модела на таквом тест сету слични, односно, чини се да додатна обука модела на базама Јужне вести и *Common Voice Corpus 18.0* није унапредила резултате. Међутим, уколико се погледају резултати које модели постижу на тест сету базе *Common Voice Corpus 18.0* (која је и најквалитетнија), видимо да додатна обука модела на бази Јужне вести (после обуке на *ParlaSpeech-RS*) знатно унапређује резултате. Очекивано, после тренинга на *Common Voice Corpus*

18.0 резултати су још бољи. Такође, примена језичког модела значајно унапређује вредности *WER* метрике, поготово после теста на *Common Voice Corpus* бази.

Табела 1. Приказ резултата добијених обуком модела са основном архитектуром.

Модел трениран на базама	Тест сет	*greedy*		језички	
		CER	WER	CER	WER
ParlaSpeech-RS	Унија	0.386	0.587	0.379	0.556
	CV18	0.214	0.508	0.205	0.411
ParlaSpeech-RS + Јужне вести	Унија	0.381	0.593	0.372	0.556
	CV18	0.147	0.385	0.125	0.278
ParlaSpeech + Јужне вести + CommonVoice	Унија	0.379	0.591	0.368	0.553
	CV18	0.085	0.264	0.055	0.143

За вишејезички модел су приказани резултати по истом принципу као за модел са основном архитектуром. Разлика је у томе што за обуку овог модела није коришћена *ParlaSpeech-RS*, већ обука полази од базе Јужне вести. Упоредни приказ резултата дат је у табели 2.

Табела 2. Приказ свих резултата добијених обуком вишејезичког модела.

Модел трениран на базама	Тест сет	*greedy*		језички	
		CER	WER	CER	WER
Јужне вести	Унија	0.427	0.725	0.413	0.615
	CV18	0.110	0.390	0.084	0.210
Јужне вести + CommonVoice	Унија	0.415	0.710	0.392	0.600
	CV18	0.068	0.263	0.034	0.103

Резултати се могу интерпретирати слично као за модел са основном архитектуром. Резултати који овај модел постиже на *Common Voice Corpus 18.0* бази су бољи од резултата основног модела. То су уједно и најбољи постигнути резултати у овом истраживању. *CER* од 0.034 значи да модел погрешно транскрибује тек 3.4% свих карактера, док *WER* од 0.103 значи да модел погрешно предвиди око 10% речи. Ипак у случају теста на унији три базе, вишејезички модел даје лошије резултате у поређењу са основним. То је и очекивано јер у тој унији доминирају узорци из *ParlaSpeech-RS* базе.

5. ЗАКЉУЧАК

Wav2Vec 2.0 модел показује велики потенцијал за препознавање говора на језицима са ограниченим ресурсима као што је српски. Резултати су показали да вишејезички модел постиже боље резултате на *Common Voice Corpus 18.0* бази у поређењу са основним моделом. Уз примену језичког модела, стопа грешке у препознавању карактера (*CER*) је 0.034, док је стопа грешке у препознавању речи (*WER*) 0.103. Будућа истраживања треба усмерити на побољшање квалитета података, прецизније поравнање транскрипата и аудио снимака, као и на примену напреднијих језичких модела ради даљег унапређења перформанси система.

6. ЛИТЕРАТУРА

[1] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," Oct. 22, 2020, *arXiv*: arXiv:2006.11477. Accessed: Sep. 24, 2024. [Online]. Available: http://arxiv.org/abs/2006.11477

[2] "Common Voice." Accessed: Sep. 24, 2024. [Online]. Available: https://commonvoice.mozilla.org/en

[3] "ASR training dataset for Serbian JuzneVesti-SR v1.0." Accessed: Sep. 24, 2024. [Online]. Available: https://www.clarin.si/repository/xmlui/handle/11356/1679

[4] "Parliamentary spoken corpus of Serbian ParlaSpeech-RS 1.0." Accessed: Sep. 24, 2024. [Online]. Available: https://www.clarin.si/repository/xmlui/handle/11356/1834

[5] A. Vaswani *et al.*, "Attention Is All You Need," 2017, *arXiv*. doi: 10.48550/ARXIV.1706.03762.

[6] A. van den Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," Jan. 22, 2019, *arXiv*: arXiv:1807.03748. Accessed: Sep. 24, 2024. [Online]. Available: http://arxiv.org/abs/1807.03748

[7] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*, Pittsburgh, Pennsylvania: ACM Press, 2006, pp. 369–376. doi: 10.1145/1143844.1143891.

Кратка биографија:

Тамара Бановац рођена је у Сремској Митровици 1999. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Обрада сигнала одбранила је 2024. године.
Контакт: tamarabanovac3@gmail.com

Владо Делић рођен је 1964. год. Докторирао је на Факултету техничких наука 1997. год., а од 2013. је у звању редовни професор. Област интересовања су говорне технологије.

UDK: 4.9
DOI: https://doi.org/10.24867/30BE40Banovac

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/30BE42Tesanovic

PROGRAMSKI MODEL PODESIVIH KOMPONENTI ZA FPS IGRE SA STRATEGIJSKIM MEHANIZMIMA

A PROGRAMMING MODEL OF CONFIGURABLE COMPONENTS FOR FPS GAMES WITH STRATEGY ELEMENTS

Nedeljko Tešanović, dipl. inž. elektr. i računar, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIČKO I RAČUNARSKO INŽENJERSTVO

Kratak sadržaj – *Trenutno ne postoje adekvatni modeli za brz razvoj prototipa modernih video igara određenih žanrova. Ovaj rad predlaže programski model podesivih komponenti čija implementacija predstavlja takav alat. Razvijeni alat je testiran kroz proces razvoja dva protipa igara. Rezultat je potvrda validnosti rješenja za datu problematiku. Rezultat daje osnovu za dalja istraživanja.*

Ključne reči: *Programski model, razvoj video igara, FPS, RTS, prototip*

Abstract – *There is a lack of adequate models for fast video game prototyping. This paper proposes a programming model of configurable components whose implementation represents such a tool. The tool is tested in the form of two game prototypes. The result confirms the validity of the proposed solution and also serves as a basis for future research.*

Keywords: *Programming model, game development, FPS, RTS, prototype*

1. UVOD

Video igre su trenutno jedan od najpopularnijih oblika zabave sa ogromnim tržištem i industrijom iza njih [0]. Potreba tržišta za novim igrama je neprestana i povećava se sa vremenom zbog rasta broja igrača, uz složenost tih igara. Zbog toga je za industriju video igara od ogromnog značaja da se proces razvoja novih igara ubrza i pojednostavi koliko god je to moguće.

2. OPIS PROBLEMA I POTREBE ZA RJEŠENJEM

Potreban je programski model koji jasno opisuje skup komponenti čija implementacija olakšava razvoj video igara proizvoljnog žanra. Implementacija ovog modela treba da omogući brz razvoj igara odabranog žanra sa što većoj fleksibilnosti u količini i složenosti igara.

Implementacija ovog modela ne mora da sadrži apsolutno sve njegove komponente, nego samo one komponente koje projektant želi da ima u svojim igrama. Pored toga, ovakav model ne treba da ograničava dizajn igre samo na mehanizme prisutne u ovom modelu, nego da omogući dodatna proširenja po potrebama vizije projektanta. Isto ako, ne treba da bude vezan za određen pogon, nego da je opšte primjenljiv – kako za više pogona, tako i za „domaći“ pogon.

Trenutno, jedan od najpopularnijih žanrova video igara jesu FPS igre (eng. *First Person Shooter*) [0]. Ovaj žanr je danas najsloženiji nego što je ikada bio i njegova složenost se povećava iz dana u dan. Nažalost, šabloni za pravljenje tih igara su ostali relativno jednostavni u odnosu na potrebnu složenost modernih igara. Za razvoj FPS igara u pogonima [1][2][3] je često dostupan samo paket za kontrolu karaktera iz prvog lica, odnosno za kretanje po svijetu konstantnom brzinom, skakanje i gledanje naokolo, ali ne i za ostale prethodno navedene mehanizme.

Zbog toga je za potrebe ovog rada odabran upravo FPS žanr. Tačnije, odabran je spoj FPS i RTS (eng. *Real-Time Strategy*) žanrova. Razlog tome jeste prisustvo određenih strategijskih mehanizama koji se mogu često pronaći u RTS igrama, a čija integracija u FPS igru bi omogućila dodatnu složenost sastava te igre i otvorila nove mogućnosti za dizajn.

U nastavku ovog poglavlja će biti navedeni skupovi mehanizama koji su često sadržani u FPS i RTS igrama i čija unija će predstavljati skup potrebnih mehanizama koje programski model treba da sadrži da bi se smatrao validnim rješenjem prethodno pomenute problematike. Skupovi mehanizama koji će biti navedeni nisu u potpunosti sadržani u svakoj igri, nego su sadržani neki od njihovih podskupova.

2.1. Mehanizmi česti u FPS igrama

Ovi mehanizmi se mogu pronaći u većini FPS igara, od relativno jednostavnih igara, do složenih vojnih simulacija [4][5][6]. Naravno, postoje i drugi mehanizmi koji se mogu naći u nekim FPS igrama, ali pokrivanje svih različitih mehanizama svih tih igara i formiranje modela koji ih sve uključuje je praktično nemoguće. Neki mehanizmi često sadržani u FPS igrama su lokomocija, kamera, interakcija, avatar igrača, oružje (blisko i daljinsko), neigrivi karakteri, logika igre, i korisnički interfejs.

2.2. Mehanizmi česti u RTS igrama

Igre RTS žanra su jedne od najstarijih video igara i obično se bave tematikom ogromnih bitaka i manevrisanja jedinica zarad nadmudrivanja protivnika i uništenja njihove baze komande [7][8]. Sa obzirom na to da FPS igre u prosjeku imaju mnogo manji opseg, tj. akcenat se daje na relativno manjim okršajima, od interesa je razmotriti strategijske mehanizme koji se mogu pronaći u ovim igrama radi potencijalnog poboljšanja i proširenja FPS igara. Neki od tih mehanizama koji već nisu nabrojani u

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

FPS igrama su upravljane jedinicama, baza, resursi, i regrutovanje jedinica. Kamera, logika igre, korisnički interfejsi neigrivi karakteri su takođe prisutni i u većini RTS igara, ali se razlikuju od njihovih predstava u FPS žanru, te je potrebno imati i to na umu prilikom osmišljavanja rješenja.

3. MODEL APSTRAKTNIH MEHANIZAMA

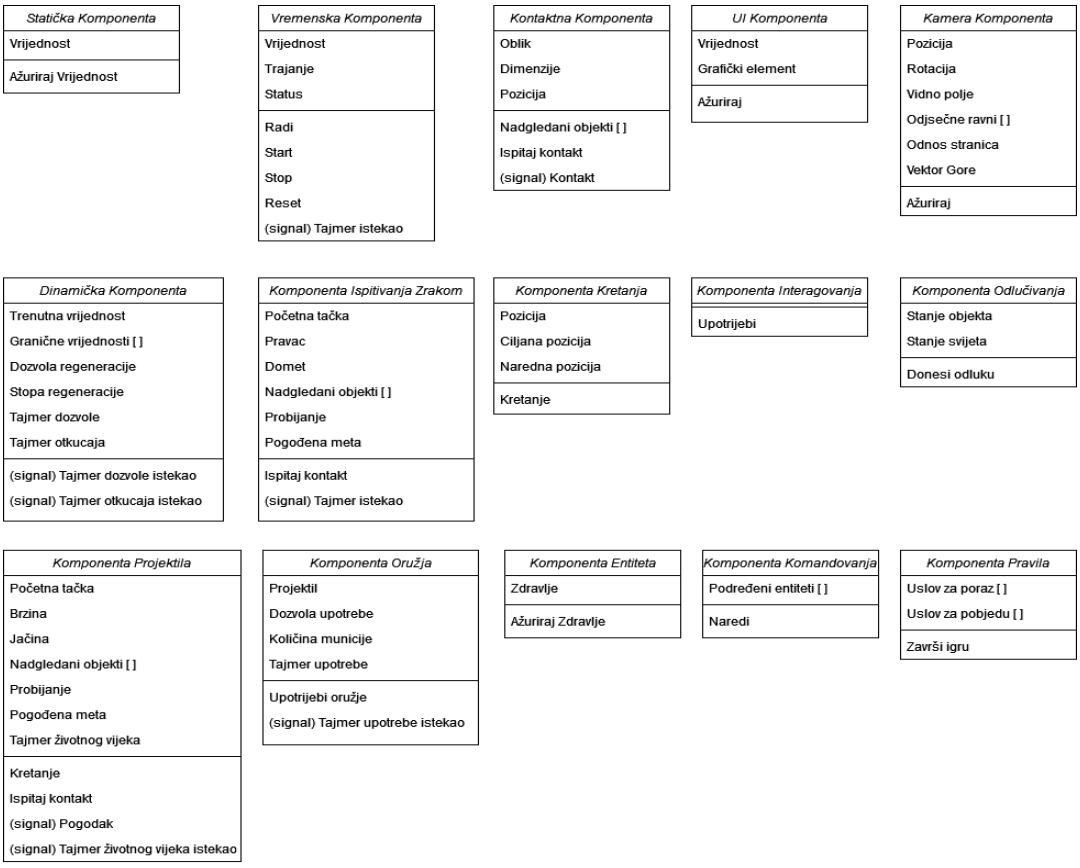
Imajući u vidu mehanizme nabrojane i pojašnjene u poglavlju 2.1, kao i specifikaciju problema u poglavlju 2, predlaže se sledeće rješenje objašnjeno u ovom poglavlju: **programski model podesivih komponenti za FPS igre sa strategijskim mehanizmima.**

Komponente ovog rješenja (prikazane na slici 1) su samo smjernice za funkcionalnosti koje bi trebale biti implementirane, a sam način implementacije je ostavljen projektantu na slobodan izbor. Akcenat je stavljen na upotrebi šema naslijeđivanja i kompozicije za ostvarenje komponenti čija implementacija može da obezbjedi ogromnu fleksibilnost u smislu varijanti stvari u igrama koje bi se pravile tim alatom. Komponente rješenja su one koje svojim kombinacijama ili uključivanjem omogućavaju implementiranje mehanizama koji se često nalaze u FPS i RTS igrama, odnosno onih opisanih u poglavlju 2. Bitno je naglasiti da se ovo rješenje odnosi na apstraktni nivo programskog dijela razvoja igre ili alata – implementiranje ovakvih komponenti podrazumijeva već postojeći pogon igre sa osnovnim funkcijama (programska petlja, renderer, animacije, 3D modeli, zvukovi, itd), bilo komercijalni ili domaći, za koji se ove komponente razvijaju.

Rješenje je modularno po prirodi, tako da projektant može da odabere koje komponente će da implementira u svom proizvodu i koje će da budu sadržane u nekoj formi u igrama razvijenim upotrebom tog proizvoda. Puna ili djelimična implementacija bi trebala da obezbjedi mogućnost lakog i ubrzanog razvoja video igara sa elementima FPS i RTS žanrova, ali i igara striktno FPS, odnosno RTS žanra. Drugim riječima, projektantu je na raspolaganju prošireni skup komponenti, ali ukoliko projektant želi da pravi samo FPS igre u retro stilu, dovoljno je da implementira podskup komponenti ovih rješenja koje on smatra potrebnim za ostvarenje tog cilja. Naravno, moguće je i dopuniti ovaj model sopstvenim komponentama - što je broj implementiranih komponenti veći, to je moguće i lakše pravljenje većeg broja i složenijih igara sa mnogo manje potrebnog truda i rada u fazama dizajna i implementacije prilikom razvoja igre.

4. PRIMJER IMPLEMENTACIJE MODELA

Kao što je već pomenuto, za ovaj rad je odabran spoj FPS i RTS žanrova i rješenje je orjentisano ka funkcionalnostima ta dva žanra, a radi jednostavnosni primjera, ograničeno je na jednog igrača. Fokus ove implementacije je na određenoj miješavini ta dva žanra gdje je najveći fokus na FPS dijelu, a strategijski mehanizmi igraju sekundarnu ulogu i služe da pomognu mehanizmima iz FPS žanra. Kretanje, perspektiva, akcija i interaktivne tehnike su preuzete iz FPS žanra, a ponašanje NPC-a (eng. *Non-Playable Character*) je uslovljeno mješavinom ponašanja čestim u FPS igrama, ali i raznim strategijskim mehanizmima - pogotovo koordinacija većeg broja NPC-a. Taj hibrid FPS i RTS žanrova će se u nastavku ovog rada skraćeno zvati FPSS (eng. *First Person Strategy Shooter*).



Slika 1: Apstraktne komponente modela

Implementacija modela je sastavljena od podskupa komponenti modela opisanog u poglavlju 3, odnosno njihovih implementacija specifičnih za odabrani pogon igre. Pogon koji je odabran za implementaciju modela je Godot [3], specifično trenutna verzija – Godot 4.

4.1. Godot 4

Godot je pogon za razvijanje igara opšte namjene. Glavni razlog za izbor Godota, kao i njegova glavna prednost u odnosu na druge pogone jeste njegova modularna arhitektura koja cijelu igru organizuje na bazi čvorova, stabla, scena i signalnih funkcija sa filozofijom maksimizacije upotrebe šema nasleđivanja i kompozicije, odnosno minimizacije broja komponenti. Sve to se poklapa sa načinom realizacijom ovog programskog modela, i trebalo bi da obezbjedi olakšanu implementaciju programskog modela. Ovi čvorovi, scene, stablo i signali su četiri glavna koncepta Godota.

Čvorovi su najsitniji, ugrađeni elementi Godota koji se redaju u hijerarhijska **stabla**. Svaki čvor ima određen skup funkcionalnosti i parametera za podešavanje tih funkcionalnosti. Pored toga, u slučaju da je potrebna dodatna funkcionalnost nekog čvora, moguće je zakačiti skriptu za njega, i pozivati njene funkcije, odnosno podešavati njene promjenljive. Skripte su pisane u GDScript jeziku, domaćem programskom jeziku Godot pogona. **Scene** se sastoje od čvorova i drugih scena i mogu se instancirati. **Signali** su Godotova implementacija *Observer* šeme [9].

4.2. Implementacija modela u obliku paketa za razvoj FPS i RTS igara - „FPSS paket“

Ovaj paket je namijenjen prvobitno za razvoj FPS igara sa strategijskim mehanizmima. Kao takav, ima potrebne komponente modela koje su krojene specifično za tu namjeru, od kojih su mnoge već potpuno ili djelimično sadržane u predefinisanim čvorovima Godot pogona. U nastavku ovog potpoglavlja će ukratko biti predstavljene implementirane komponente, bez detaljne razrade njihove implementacije, sa obzirom na to da postoji više načina za realizaciju svake komponente.

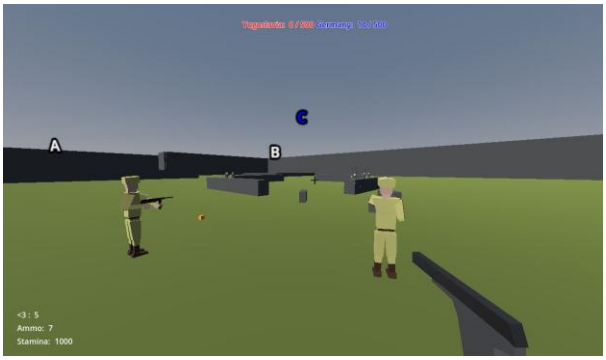
Statička komponenta je ograničena vrijednost sa funkcijom za ažuriranje i prvobitno se koristi kao brojač municije. **Vremenska komponenta** je već sadržana u predefinisanom *Timer* čvoru. **Dinamička komponenta** je kompozicija statičke i vremenske komponente i automatski mijenja svoju vrijednost tokom vremena. **Kontakt komponenta** je kombinacija Godotovih *Area3D* i *CollisionShape3D* čvorova i koristi se za ispitivanje dodira objekata. **UI komponenta** je predefinisani *Control* čvor sa propratnom skriptom. **Komponenta ispitivanja zrakom** je Godotov čvor *RayCast3D*. **Kamera komponenta** je takođe već prisutna kao *Camera3D* čvor. **Komponenta interagovanja** je implementirana kroz kod skripti zakačenih za druge čvorove i sastoji se jedne funkcije koja definiše posljedice upotrebe datog objekta. **Komponenta kretanja** sadrži *NavigationAgent3D* čvor i kod za kretanje po stazama stvorenim upotrebom *NavigationRegion3D* čvora. **Komponenta odlučivanja** je implementirana u formi funkcija koje diktiraju ponašanje entitetskih čvorova. **Komponenta projektila** se sastoji od vremenske komponente, *RigidBody3D* čvora i kontakt komponente i omogućava definisanje ponašanja projektila od prave putanje do krivih putanja balističkog proračuna. **Komponenta oružja** sadrži komponente interagovanja i projektila i definiše ponašanje oružja u igrama. **Komponenta entiteta** ima dinamičku komponentu zdravlja, kontakt komponentu i komponentu kretanja. **Komponenta komandovanja** podređenim entitetima zadaje instrukcije kroz svoje brojne funkcije u zavisnosti od "ličnosti" koja predstavlja stil komandovanja. **Komponenta pravila** je implementirana kroz kod i upravlja logikom za dodijeljivanje poena i završetak igre.

5. PRIMJER IGARA RAZVIJENIH IMPLEMENTACIJOM MODELA

U ovom poglavlju je pokazana primjena razvijenog FPSS starter paketa iz poglavlja 4 za razvoj dvije različite igre. Te dvije igre se razlikuju po tematici, kao i funkcionalnostima, što prikazuje fleksibilnost i valjanost paketa, i samim tim valjanost programskog modela na kome je on zasnovan. Ovaj rad neće ulaziti u sve detalje dizajna ovih igara, nego samo one komponente za koje je bio iskorišten FPSS starter paket.

5.1. FPS igra sa strategijskim mehanizmima

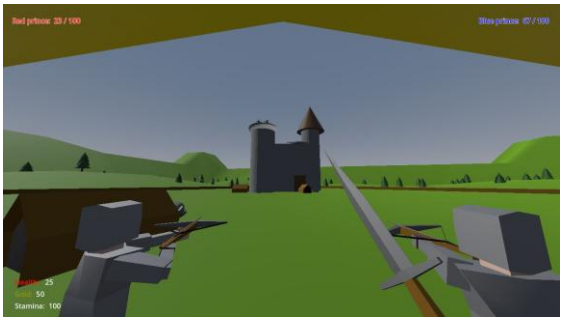
Kraljevina Jugoslavija je okupirana od strane sila osovine, a igrač se stavlja u ulogu partizanskog komandira i nadmudruje neprijatelje manevrisanjem svojih jedinica direktno sa bojnog polja. Sam igrač je hendikepiran u upotrebi oružja, te nije dovoljno jak da ide u direktan okršaj protiv neprijateljskih vojnika kao u klasičnoj FPS igri, nego mora da se uzda u svoje komandirske vještine, strategijsko razmišljanje i prijateljske jedinice.



Slika 2: Slika iz FPS igre sa strategijskim mehanizmima

5.2. RTS igra u prvom licu

Kraljevstvo je ostavljeno u krizi sa praznim prijestolom i bez imenovanog nasljednika. Postoje mnogi pretendenti, ali samo jedan od njih može postati kralj. Igrač se stavlja u ulogu jednog od princeva koji vrijebaju prijesto. Radi bolje legitimacije i morala, princevi moraju da vode svoje armije u bukvalnom smislu, izdavajući im naređenja direktno sa bojnog polja. Igrači imaju borbenu opremu u vidu oklopa, a mogu da koriste i svoje mačeve, samostrijele, katapulte i druga oružja, kao i da naređuju svojim jedinicama i radnicima.



Slika 3: Slika iz RTS igre u prvom licu

6. ZAKLJUČAK

Ovaj rad je predstavio programski model podesivih komponenti za FPS igre sa strategijskim mehanizmima, čija je arhitektura predložena u poglavlju 3. Ovaj programski model se pokazao kao sasvim validan alat za dopunjavanje funkcionalnosti već postojećih pogona za razvoj video igara, odnosno dopunjavanja izvornog koda domaćih pogona igara.

Modularne podesive komponente ovog alata, odnosno njihova pravilna implementacija je omogućila stvaranje paketa za brzi razvoj prototipa igara, a čija funkcionalnost je prikazana u poglavlju 4.2, koji sadrži osnovne elemente za stvaranje igara ne samo FPS žanra, nego i RTS žanra, kao i podžanra koji kombinuje njihove elemente. Iako je početni paket demonstrativno razvijen za pogon Godot, ne bi trebalo da postoji razlog zašto opšti programski model ne bi bio primjenljiv i na druge pogone, kao što su Unity, Unreal i slično.

Upotrebljivost implementacije programskog modela, odnosno početnog paketa FPSS je demonstrirana u poglavlju 5 na dvije igre drastično različite tematike – Drugog svjetskog rata i srednjeg vijeka. Ova varijacija u razvijenim igrama je pokazala valjanost i fleksibilnost paketa koji je podskup funkcija programskog modela. Mogućnosti razvoja dodatnih igara je praktično neograničena, a implementacijom dodatnih komponenti programskog modela bi se samo povećala.

Implementacija ovakvog modela bi trebala biti od koristi ne samo već uspostavljenim studijima za razvoj video igara, nego i nezavisnim projektantima koji žele da brzo razvijaju prototipe, odnosno razrađuju koncepte igara.

Ovaj programski model, iako skoncentrisan na FPS i RTS žanrove, je veoma opširan u svojim funkcionalnostima. Međutim, sa porastom složenosti današnjih video igara kao oblika zabave, javlja se veća i veća potreba za složenijim mehanizmima, kao što je objašneno u uvodnom poglavlju 1. Ovo stvara priliku za proširenje ovog programskog modela tako da obuhvata druge žanrove, npr. RPG, itd.

Čak i sama implementacija početnog paketa u Godot pogonu ima prostora za dalji rad i proširenje funkcijama, odnosno mehanizmima FPS i RTS igara, prvobitno onima koji se tiču režima igre sa više igrača, ali i drugima – mehanizmima za virtuelnu realnost, naprednijim algoritmima za pronalaženje staza i donošenje odluka. Interesantno bi bilo posmatrati i kako bi generativne neuronske mreže osmišljale i koristile ovakve modele, kao i kako ovakvi modeli utiču na brzinu i kvalitet razvoja igara u takmičarskom okruženju.

7. LITERATURA

[0] https://medium.com/spring-2024-information-expositions/video-games-popularity-and-longevity-over-time-72a748146f24 (pristupljeno u septembru 2024)

[1] https://unity.com/ (pristupljeno u septembru 2024)

[2] https://www.unrealengine.com/en-US (pristupljeno u septembru 2024)

[3] https://godotengine.org/ (pristupljeno u septembru 2024)

[4] https://www.gameopedia.com/evolution-of-first-person-shooter-fps-games/ (pristupljeno u septembru 2024)

[5] https://ultimatepopculture.fandom.com/wiki/First-person_shooter (pristupljeno u septembru 2024)

[6] https://reedkolbe.medium.com/an-analysis-of-first-person-shooters-what-makes-a-successful-fps-and-where-the-market-is-headed-e7b333ef5d05 (pristupljeno u septembru 2024)

[7] https://brokenkingdom.medium.com/what-are-real-time-strategy-games-6ddffd806ed1 (pristupljeno u septembru 2024)

[8] https://invogames.com/blog/what-is-real-time-strategy-games/ (pristupljeno u septembru 2024)

[9] https://docs.godotengine.org/en/stable/getting_started/introduction/key_concepts_overview.html (pristupljeno u septembru 2024)

Kratka biografija:

Nedeljko Tešanović je rođen 1998. godine u Bijeljini. Diplomski ispit na temu „Generisanje realističnih portreta upotrebom dubokih konvolucijskih generativno-protivničkih mreža“ je odbranio 2021. godine sa ocjenom 10,00.
kontakt: nedeljkotesanovic@uns.ac.rs

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/30BE43Radojevic

DINAMIKA NEURAVNOTEŽENIH KRATKIH SPOJEVA NA PRIKLJUČCIMA SINHRONOG GENERATORA

THE DYNAMICS OF UNBALANCED SHORT CIRCUITS AT THE TERMINALS OF A SYNCHRONOUS GENERATOR

Mila Radojević, Dejan Jerkan, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Predmet rada je analiza prelaznih pojava koji se odvijaju u namotajima sinhrone mašine prilikom neuravnoteženog kratkog spoja na statorskim priključcima. Izvršeno je analitičko uopštenje parametara koji karakterišu tropoloni (simetrični) kratak spoj na priključcima generatora (vremenske konstante i prelazne reaktanse), tako da se ih je moguće upotrebiti za kvantifikaciju neuravnoteženih kratkih spojeva. Na primeru dvopolnog kratkog spoja izvršen je proračun subtranzijentnih, tranzijentnih i ustaljenih reaktansi, te prirpadajućih vremenskih konstanti po uzdužnoj i poprečnoj osi.*

Ključne reči: *Sinhroni generator, kratki spojevi, tranzijenti*

Abstract *The subject of the paper is the analysis of transient phenomena that take place in the windings of a synchronous machine during an unbalanced short circuit at the stator terminals. It was performed the analytical generalization of the parameters characterizing the three-pole (symmetric) short circuit at the generator connections (time constants and transient reactances), so that they can be used for the quantification of unbalanced short circuits. On the example of a two-pole short circuit, the calculation of subtransient, transient and steady-state reactances, as well as the corresponding time constants along the longitudinal and transverse axis, was executed.*

Keywords: *Synchronous generator, short circuits, transients*

1. UVOD

Sinhrona mašina na kojoj se bazira predočeno proučavanje je trofazna, ima rotor sa isturenim polovima, jednosmerni naponski izvor pobuđivanja i dodatni prigušni rotorski namotaj. Kako je tema rada ispitivanje dinamičkih procesa u mašini, neophodno je izvršiti transformaciju matematičkog modela mašine opisanog u originalnom domenu u *dq* domen, čime se tronamotajni model mašine zamenjuje dvonamotajnim.

U osnovi, ova transformacija se vrši kako bi se eliminisala zavisnost matrice induktivnosti induktivnosti od vremena, odnosno položaja rotora koji zbog isturene konstrukcije dovodi do razlike u širini vazdušnog zazora prilikom obrtanja. Uz zanemarenje nelinearnosti feromagnetskog materijala od kojeg je izrađena mašina, te usvajanja konstantne brzine obrtanja tokom trajanja prelaznih procesa, dobija se linearna forma matematičkog modela, čime su stvoreni uslovi za primenu Laplasove transformacije: $\mathcal{L}\,[f(t)] = F(p) = p\int_0^\infty f(t)e^{-pt}dt.$

Ovim je model pretočen u algebarsku formu i verna je zamena polaznom nelinarnom diferencijalnom modelu u vremenskom domenu.

Početna forma sistema sastoji se od deset izraza, pri čemu se pet njih tiče jednačina naponske ravnoteže namotaja, a drugih pet jednačina njihovih fluksnih obuhvata, prikazanih na sledeći način:

$$u_d = R_s i_d + p\psi_d - \omega\psi_q \quad (1.1)$$
$$u_q = R_s i_q + p\psi_q + \omega\psi_d \quad (1.2)$$
$$u_f = R_f i_f + p\psi_f \quad (1.3)$$
$$0 = R_k i_D + p\psi_D \quad (1.4)$$
$$0 = R_k i_Q + p\psi_Q \quad (1.5)$$
$$\psi_d = L_d i_d + M_d i_f + M_d i_D \quad (1.6)$$
$$\psi_q = L_q i_q + M_q i_Q \quad (1.7)$$
$$\psi_f = M_d i_d + L_f i_f + M_d i_D \quad (1.8)$$
$$\psi_D = M_d i_d + M_d i_f + L_D i_D \quad (1.9)$$
$$\psi_Q = M_q i_q + L_Q i_Q \quad (1.10)$$

Simultano rešavanje sistema jednačina (1.1)-(1.10) odvija se iz dve etape. Tokom prve se vrši eliminacija jednačina i veličina koje opisuju pobudni namotaj u oznaci indeksa *f*: u_f – napon, R_f – aktivna otpornost, i_f – struja, ,ψ_f – fluksni obuhvat, L_f – sopstvena idnuktivnost. Druga etapa podrazumeva eliminaciju veličina i jednačina vezanih za prigušni rotorski namotaj: R_k – aktivna otpornost, i_D – struja po podužnoj osi, ψ_D – fluksni obuhvat po podužnoj osi, i_Q – struja po po poprečnoj osi, ψ_Q – fluksni obuhvat po poprečnoj osi, L_D – sopstvena induktivnost prigušnog namotaja po podužnoj osi, L_Q – sopstvena induktivnost prigušnog namotaja po poprečnoj osi. Osim navedenih, korišćene su i sledeće fizičke veličine za opisivanje statorskog namotaja: u_d – napon po podužnoj osi, u_q – napon po poprečnoj osi, R_s – rezistivnost namotaja, i_d – struja po podužnoj osi, i_q – struja po poprečnoj osi, ψ_d – fluksni obuhvat po podužnoj osi, ψ_q – fluksni obuhvat po poprečnoj osi, L_d – sopstvena induktivnost po podužnoj osi, L_q – sopstvena induktivnost po poprečnoj osi. Oznakama M_d i M_q definisane su međuinduktivnosti namotaja po *d* osi, odnosno po *q* osi.

Rezultat ovog postupka uprošćavanja prerasta u sistem jednačina koje opisuju samo statorske namotaje po *d* i *q* osi, ali u kojima je kroz izraze operatorskih induktivnosti dobijenih tokom postupka uprošćavanja sistema, ipak sadržan uticaj svih prisutnih namotaja. Ovakav sistem se naziva sistemom Parkovih jednačina i glasi:

$$u_d = R_s i_d + p\psi_d - \omega\psi_q \quad (1.11)$$
$$\psi_d = L_d(p) i_d + G(p) u_f \quad (1.13)$$

NAPOMENA:
Ovaj rad proistekao je iz master rada, čiji mentor je bio dr Dejan Jerkan, vanr. prof.

$u_q = R_s i_q + p\psi_q + \omega\psi_d$ (1.12) $\psi_q = L_q(p) i_q$ (1.14)

Pored već poznatih veličina uvedene su i tri nove:
$L_d(p)$ – podužna operatorska induktivnost, $L_q(p)$ – poprečna operatorska induktivnost i $G(p)$ – prenosna funkcija napona pobude.

2. OPERATORSKE INDUKTIVNOSTI

Podužna i poprečna induktivnost, date u zavisnosti od Laplasovog operatora p, produkt su postupka uprošćavanja sistema jednačina (1.1)-(1.10). Induktivnosti se tada izražavaju relacijama u kojima firgurišu omski otpor
(R_k, R_f) i rasipni (Λ_D, Λ_Q, Λ_f) parametri prigušnog i pobudnog namotaja, podužna i poprečna rasipna induktivnost statorskog namotaja, Λ_d i Λ_q, međuinduktivnosti M_d i M_q:

$$pL_d(p) = p\Lambda_d + pM_d \| (R_k + p\Lambda_D) \| (R_f + p\Lambda_f) \quad (2.1)$$

$$pL_q(p) = p\Lambda_q + pM_q \parallel (R_k + p\Lambda_Q) \quad (2.2)$$

Prilikom pojave naglih poremećaja u mašini, poput kratkih spojeva, dinamika odziva mašine je upravo donminatno određena ovim operatorskim induktivnostima.
Vremenski tok struja kratkog spoja karakterišu tri uočljiva stadijuma, koji se na osnovu oblika anvelope struje kvara mogu jasno raščlaniti. To su subtranzijentni, tranzijentni i ustaljeni period. Ovakva podela na intervale uzrokovana je konstrukcionim karakteristikama mašine sa isturenim polovima rotora i prigušnim namotajem. Naime, na samom početku kvara, ukupnoj impedansi mašine aktivno doprinose svi namotaji u mašini. Međutim, zbog izraženije vrednosti aktivne otpornosti prigušnog namotaja u poređenju sa otpornostima ostalih namotaja mašine, doprinos ovog namotaja u impedansi kratkog spoja najpre iščezava. Značajan uticaj prigušnog namotaja najbrže jenjava i time označava kraj subtranzijentnog, a početak tranzijentnog perioda. Aktivna otpornost pobudnog namotaja je i do 50 puta manja od prigušne, što uz ujedno i veliku sopstvenu induktivnost određuje da tranzijentni period vremenkog toka struje kvara traje značajno duže od subtranzijentnog. Sopstvena magnetna tromost pomenutih namotaja može se iskazati putem sledećih vremenskih konstanti:

$T_f = \frac{L_f}{R_f} = \frac{\Lambda_f + M_d}{R_f}$ – podužna vremenska konstanta pobudnog namotaja rotora

$T_D = \frac{L_D}{R_k} = \frac{\Lambda_D + M_d}{R_k}$ – podužna vremenska konstanta prigušnog namotaja rotora

Kao što je rečeno, strujama kvara statorskih namotaja doprinose svi namotaji mašine usled magnetskog sprezanja. Efekat ovog sprezanja je uvažen putem operatorskih induktivnosti $L_d(p)$ i $L_q(p)$. Stoga je vremenski tok struja kvara određen dinamikom svih međusobno spregnutih namotaja, tako da oni združeno određuju induktivnosti i vremenske konstante pomoću kojih se ta promena karakteriše. Osnov za dobijanje tih induktivnosti i vremenskih konstanti pronalazi se u okvirima teoreme o graničnim uslovima. Teoremom su obuhvaćena dva slučaja kojima se povezuju veličine u vremenskom i kompleksnom Laplasovom domenu:

- veza početnih vrednosti: $\lim_{t\to\infty} F(t) = \lim_{p\to 0} g(p)$,
- veza krajnjih vrednosti: $\lim_{t\to 0} F(t) = \lim_{p\to\infty} g(p)$,

gde je $F(t)$ – originalna funkcija u vremenskom domenu, a
$g(p)$ – originalna funkcija u kompleksnom Laplasovom domenu.
Operatorske induktivnosti se mogu primenom teoreme o graničnim vrednostima tumačiti kao konstantne veličine tokom trajanja nekog od tri pobrojana segmenta (sub,tr,ust) Prema tome, subtranzijentnom tj. početnom periodu prelaznog procesa odgovara primena prve navedene granične teoreme čime se dobija tzv. subtranzijentna induktivnost L_d'':

$$L_d^n = L_d(t \to 0) = L_d(p \to \infty)$$

$$= \Lambda_d + \left(M_d \left\| \left(\frac{R_k}{p} + \Lambda_D\right) \right\| \left(\frac{R_f}{p} + \Lambda_f\right) \right) = \Lambda_d + \left(M_d \| \Lambda_D \|_f \right). \quad (2.3)$$

Naredni je tranzijentni period koji za polaznu osnovu uzima izraz sličan prethodnom, ali bez učešća parametara vezanih za prigušni namotaj (Λ_f, R_f). Ovakav pristup je teorijski opravdan odnosnom vrednosti vremenske konstante prigušnog i pobudnog namotaja za koje važi $T_f \gg T_D$. Izraz za određivanje tranzijentne induktivnosti L_d', uz pobrujana ograničenja glasi:

$$L_d' = L_d(t \to 0) = L_d(p \to \infty) = \Lambda_d + (M_d \parallel \Lambda_f). \quad (2.4)$$

Po vremenskom toku poslednja, ustaljena induktivnost L_d, dobija se korišćenjem drugog graničnog uslova na sledeći način:

$$L_d = L_d(t \to \infty) = L_d(p \to 0) = \Lambda_d + \frac{1}{p}(pM_d \| R_k \| R_f)$$
$$= \Lambda_d + M_d. \quad (2.5)$$

Analognim postupkom definišu se i poprečne induktivnosti subtranzijentnog i ustaljenog perioda, L_q'' i L_q :

$$L_q'' = L_q(t \to 0) = L_q(p \to \infty) = \Lambda_q + \left(M_q \parallel \left(\frac{R_k}{p} + \Lambda_Q\right) \right)$$
$$= \Lambda_q + (M_q \parallel \Lambda_Q) \quad (2.6)$$

$$L_q = L_q(t \to \infty) = L_q(p \to 0) = \Lambda_q + M_q. \quad (2.7)$$

Poprečna operatorska induktivnost nije pod uticajem promene u tranzijentnom periodu zbog odsustva pobudnog namotaja po podužnoj osi, te stoga trivijalno sledi $L_q'' = L_q'$.
Vremenske kontante statorskih namotaja se mogu izvesti putem razlaganja na proste činioce izraza za operatorske induktivnosti. Svođenjem veličina u svakoj od jednačina ponaosob (2.6), (2.7) na zajednički imenitelj, kreira se novi oblik operatorskih induktivnosti:

$$L_d(p) = \Lambda_d + \frac{p^2 M_d \Lambda_d \Lambda_f + p(\Lambda_f M_d R_k + R_f M_d \Lambda_D) + M_d R_k R_f}{p^2(M_d \Lambda_D + \Lambda_f M_d + \Lambda_D \Lambda_f) + p(M_d R_k + R_f M_d + \Lambda_f R_k + R_f \Lambda_D) + R_k R_f} \quad (2.8)$$

$$L_q(p) = \frac{p\left(\Lambda_q M_q + \Lambda_Q(\Lambda_q + M_q)\right) + R_k(\Lambda_q + M_q)}{p(M_q + \Lambda_Q) + R_k}. \quad (2.9)$$

Oba izraza vidno sadrže polinome u svojim brojiocima i imeniocima, te se mogu razložiti u proizvode binoma, kako bi se ostvario faktorizovani zapis operatorskih induktivnosti:

$$L_d(p) = L_d \frac{(1+pT'_d)(1+pT''_d)}{(1+pT'_{d0})(1+pT'_{d0})} \quad (2.10)$$

$$L_q(p) = L_q \frac{1+pT''_q}{1+pT''_{q0}} \quad (2.11)$$

Ukoliko bi se dodatno definisale prigodne veličine za dalji matematički postupak, zapravo bi se izvršilo uvođenje koeficijenata rasipanja:

$\sigma_{fD} = 1 - \frac{M_d^2}{L_f L_D}$ – koeficijent rasipanja između pobudnog i podužnog prigušnog rotorskog namotaja

$\sigma_{df} = 1 - \frac{M_d^2}{L_d L_f}$ - koeficijent rasipanja između pobudnog i podužnog statorskog namotaja

$\sigma_{dD} = 1 - \frac{M_d^2}{L_d L_D}$ - koeficijent rasipanja između podužnog statorskog namotaja i podužnog prigušnog namotaja

Upotrebom navedenih koeficijenata rasipanja, jednačina () u sređenom obliku se može izraziti kao:

$$L_d(p) = \frac{L_d + pL_d(\sigma_{df}T_f + \sigma_{dD}T_D) + p^2 L''_d \sigma_{fD} T_f T_D}{1 + p(T_f + T_D) + p^2 \sigma_{fD} T_f T_D} \quad (2.12)$$

Upoređivanjem prethodne jednačine sa (2.10), mogu se uspostaviti jednakosti između članova koji se nalaze uz isti stepen polinoma te dve jednačine. Oslanjajući se na primenu objašnjenog uslova $T_f \gg T_D$, dobijaju se izrazi za četiri vremenske konstante:

$$T'_{d0} = T_f + T_D \approx T_f = \frac{\Lambda_f + M_d}{R_f} \quad (2.13)$$

$$T''_{d0} = \frac{\sigma_{fD} T_D}{1 + \underline{T_D}} \approx \sigma_{fD} T_D = \frac{\Lambda_D + (\Lambda_f \parallel M_d)}{R_k} \quad (2.14)$$

$$T'_d = \sigma_{df} T_f + \sigma_{dD} T_D \approx \sigma_{df} T_f = \frac{\Lambda_f + (\Lambda_d \parallel M_d)}{R_f} \quad (2.15)$$

$$T''_d = \frac{\Lambda_D + (\Lambda_d \| \Lambda_f \| M_d)}{R_k} \quad (2.16)$$

Apostrof $'$ se odnosi na konstante vezane za tranzijetni period prelaznog procesa, dok je dvosotruki apostrof " oznaka za subtranzijentni period. Dodatni indeks "0" naglašava da je reč o konstantama koje ne uključuju statorske rasipne induktivnost, što asocinara na režim praznog hoda, dok su veličine bez indeksa vezane za režim kratkog spoja. Odsustvo pobudnog namotaja po poprečnoj osi značajno uprošćava određivanje korespodentnih poprečnih vremenskih konstanti: subtranzijentne poprečne vremenske konstante u režimu praznog hoda $T''_{q0} = \frac{\Lambda_Q + M_q}{R_k}$ (2.17), dok je ista, ali za režim kratkog spoja: $T''_q = \frac{\Lambda_Q + (M_q \| \Lambda_q)}{R_k}$ (2.18).

3. VREMENSKE KONSTANTE NEURAVNOTEŽENIH KRATKIH SPOJEVA

Vremenske konstante izvedene u prethodnom poglavlju odgovaraju slučaju tropolnog kratkog spoja, koji je uravnotežen. Iako složen sa aspekta dinamike unutra sinhronog generatora, pojava tropolnog kratakpg spoja nenarušava uravnoteženost sistema, pa samim tim ni simetriju. Sa druge strane pri pojavi neuravnoteženih kvarova na priključcima generatora, poput jednopolnog ili dvopolnog kratkog spoja, dolazi do simultane pojave nesimetrije. Svi nesimetrični režimi u sinhronom generatoru izazvani nastankom neuravnoteženih kratkih spojeva su praćeni pojavom simetričnog režima inverznog redosleda. Dosadašnje induktivnosti su zapravo prećutno opisivale dinamiku mašine isključivo u direktnom redosledu simetrije, što povlači da je za opisivanje dinamike mašine u inverznom redosledu simetrije potrebno definisati nove. Induktivnost inverznog redosleda se tradicionalno obeležava kao L_2. Ona opisuje uspostavljanje struja u statorskim namotajima usled interakcije mašine sa obrtnim poljem inverznog smera obrtanja koje se prilikom nesimetričnog kvara uspostavlja u mašini. Ipak, situacija se dodatno usložnjava pri isturenom, nesimetričnom, obliku rotora koja uzrokuje pojavu dodatnih viših harmonika u električnim veličinama mašine (fluksevi, struje, indukovani naponi). Na osnovu iznetih promena u odnosu na prilike pri tropolnom kratkom spoju nameće se zaključak da i vremenske konstantne moraju biti različite od do sada izvedenih.

Ukoliko se, primera radi, analiza ograniči na dvopolni kratak spoj između faza b i c bez zemlje, određivanje vremenskih konstanti kvara započinje se od izraza za struju kvara dobijenih uz zanemarenje statorskog aktivnog otpora R_s, koja glasi:

$$i_b = -i_c = \frac{\sqrt{3}E}{L_d(p) + \sqrt{L_d(p) L_q(p)}} \times [\sin\theta - b\sin 3\theta + b^2 \sin 5\theta - b^3 \sin 7\theta + \cdots] - \frac{\sqrt{3}\,\mathrm{E}\sin\theta_0}{\sqrt{L_d(p) L_q(p)}} \times \left[\frac{1}{2} - b\cos 2\theta + b^2 \cos 4\theta - b^3 \cos 6\theta + \cdots\right] \quad (3.1)$$

U početnim trenucima kvara ovaj izraz se može aproksimirati kao:

$$i_b = -i_c = \underbrace{\frac{\sqrt{3}E}{L''_d + \sqrt{L''_d L''_q}}[\sin\theta - b\sin 3\theta + b^2 \sin 5\theta - b^3 \sin 7\theta + \cdots]}_{①} - \underbrace{\frac{\sqrt{3}\,\mathrm{E}\sin\theta_0}{\sqrt{L''_d L''_q}}\left[\frac{1}{2} - b\cos 2\theta + b^2 \cos 4\theta - b^3 \cos 6\theta + \cdots\right]}_{②} \quad (3.2)$$

Od posebnog interesa za dalji postupak određivanja vremenskih konstanti je imenilac iz člana obeleženog sa ①. Po uzoru na proračun ove vrste kvara u EES-u, daleko od generatora, uočava se sličan oblik izraza ispred zagrade. Po tom tumačenju je opravdano član $\sqrt{L'_d L''_q}$ progalsiti inverznom induktivnošću L_2.

$$\frac{1}{L''_d + L_2} = \frac{\frac{1}{L_d(p) + L_i}}{L_d \cdot \frac{1 + p(T'_d + T''_d) + p^2 T'_d T''_d}{1 + p(T'_{d0} + T''_{d0}) + p^2 T'_{d0} T''_{d0}} + L_2} = \frac{1 + p(T'_{do} + T''_{do}) + p^2 T'_{do} T''_{do}}{L_d + pL_d(T'_d + T''_d) + p^2 L_d T'_d T''_d + L_i + pL_2(T'_{do} + T''_{do}) + p^2 L_2 T'_{do} T''_{do}} \quad (3.3)$$

Impedansa, ili admitansa u ovom slučaju, se može tretirati kao prenosna funkcija napona i struje, pa tako važi da su polovi prenosne funkcije povezani sa vremenskim konstantama relacijom: $p = -\frac{1}{T}$. Izraz (3.3) se stoga treba faktorisati, pre svega imenilac u kojem su polovi sadržani. Zbog toga se zbog jednostavnijeg praćenja izdvaja imenilac i podvrgava daljem proračunu.

$$L_d + pL_d(T'_d + T''_d) + p^2 L_d T'_d T''_d + L_2 + pL_2(T'_{do} + T''_{do}) + p^2 L_2 T'_{do} T''_{do} =$$

$$= p^2(L_d T'_d T''_d + L_2 T'_{d0} T''_{d0}) + p(L_d(T'_d + T''_d) + L_2(T'_{d0} + T''_{d0})) + L_d + L_2$$

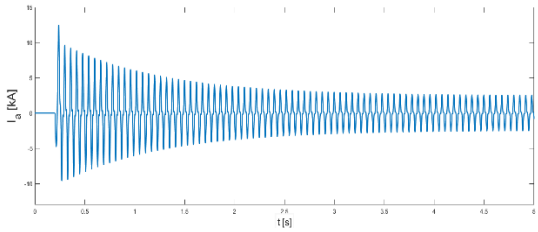
$$= p^2\left(L_d \cdot T''_{do} T'_{d0} \frac{L''_d}{L_d} + L_2 T'_{do} T''_{do}\right) + p\left(L_d\left(T'_{do}\frac{L'_d}{L_d} + \frac{T''_{d0} T'_{d0}}{T'_{d0}\frac{L''_d}{L_d}} \cdot \frac{L''_d}{L_d}\right) + L_2(T'_{do} + T''_{d0})\right) + L_d + L_2$$

$$= p^2 T'_{do} T''_{d0}(L''_d + L_2) + p\left(L_d\left(T'_{do}\frac{L'_d}{L_d} + T''_{d0}\frac{L''_d}{L'_d}\right) + L_2(T'_{do} + T''_{d0})\right) + L_d + L_2$$

$$= T'_{d0}T''_{d0}(L''_d + L_2)\left[p^2 + p\frac{L_d\left(T'_d\frac{L'_d}{L_d} + T'_{d0}\frac{L''_d}{L'_d}\right) + L_2(T'_{d0}+T''_{d0})}{T'_{d0}T''_{d0}(L''_d+L_2)} + \frac{L_d+L_i}{L''_d+L_i}\cdot\frac{1}{T'_{d0}T''_{d0}}\right]$$

$$= T'_{d0}T''_{d0}(L''_d + L_2)\left[p^2 + p\frac{\underbrace{T'_{d0}L'_d}_{①} + \underbrace{T'_{d0}T''_{d0}\frac{L''_d}{L'_d}}_{②} + \underbrace{L_2T'_{d0}}_{③} + \underbrace{L_2T''_{d0}}_{④}}{T'_{d0}T''_{d0}(L''_d+L_2)} + \frac{L_d+L_2}{L''_d+L_2}\cdot\frac{L'_d+L_2}{L'_d+L_2}\cdot\frac{1}{T'_{d0}T''_{d0}}\right] \quad (3.4)$$

Kako su drugi i četvrti sabirak brojioca uz član polinoma prvog stepena umnošci podužne subtranzijentne vremenske konstante praznog hoda, koja je po svoj prirodi mala brojčana vrednost, oni se u narednim koracima opravdano zanemaruju iz izraza. Pored toga, slobodni član polinoma u uglastoj zagradi se proširuje razlomkom $\frac{L'_d+L_2}{L'_d+L_2}$ kako bi se stvorili uslovi za definisanje novih vremenskih konstanti. Naime, elementi $\frac{L'_d+L_2}{L''_d+L_2}\cdot\frac{1}{T''_{d0}}$ u slobodnom članu kreiraju novu vremensku konstantu koja se naziva podužna subtranzijentna vremenska konstanta dvopolnog kratkog spoja $T'_{d(l-l)}$ (3.5). Primenjeni indeks *(l-l)* potiče od engleskog izraza za dvopolni kratak spoj, eng. *line-to-line*, pa će se sve vremenske konstante vezane za opisivanje prirode ovog kvara indeksirati na isti način. Preostali elementi koji figurišu u slobodnom članu polinoma se koriste u svrhu definisanja podužne tranzijentne vremenske konstante dvopolnog kratkog spoja koja glasi: $T''_{d(l-l)} = \frac{L_d+L_2}{L'_d+L_2}\cdot\frac{1}{T'_{d0}}$ (3.6). Izvdene vremenske konstante dokaz su da priroda prigušenog prelaznog procesa važi i pri neuravnoteženom, dvopolnom kratkom spoju bez zemlje, što je interpretirano na primeru struje i_a sinhronog generatora (slika 1). Prikazan odziv dobijen je na osnovu *FEM* modela sinhronog generatora, dok su rezultati proračuna nad takvim modelom prebačeni u okruženje *Matlab&Simulink* kako bi se ostvario predočen prikaz sa slike.



Slika 1. *Struja faze a pri dvopolnom kratkom spoju bez zemlje na priključcima sinhronog generatora*

4. ZAKLJUČAK

U radu je prikazan postupak određivaja vremsnkih konstanti i induktivnosti neophodnih za opisivanje dinamike prelaznog procesa kakav je uravnoteženi, tropolni kratak spoj na priključcima sinhronog generatora. Potom su na osnovu izvedenih parametara karakterističnih za simetričan kvar, analogno određeni i parametri karakteristični za uslove neuravnoteženog dvopolnog kratkog spoja bez zemlje, takođe na prikjučcima sinhronog generatora. Za potrebe svih navedenih izvođenja matematički model mašine je iz originalnog prebačen u *dq* sistem, a dodatno i u kompleksni Laplasovom domen.

5. LITERATURA

[1] Concordia, C., 1951. *Synchronous machines*, Theory and Performance. New York: John Wiley & Sons, INC.

[2] Adkins, B. и Harley, RG., 1975., *The general theory of alternating current machines: Application to Practical Problems*, Chapman and Hall.

[3] *Transient theory of synchronous generators under unbalanced conditions*, Y.K. Ching and B. Adkins, Volume 101, Issue 7, August 1954, p. 166 – 182.

[4] *Circuit Analysis of A-C Power Systems, Volume I: Symmetrical and Related Components*, Edith Clarke, General Electric Series, Wiley, 1943

[5] *Помоћни материјал – Предмет: Моделовање електричних машина*, Факултет техничких наука у Новом саду. 2023.

Kratka biografija:

Mila Radojević rođena je u Beogradu 2000. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine odbranila je 2024.god.

Dejan Jerkan je vanr. prof. na Fakultetu tehničkih nauka u Novom Sadu, na Katedri za Energetsku elektroniku i pretvarače. Oblast interesovanja su mu modelovanje i dijagnostika električnih mašina, kao i metoda konačnih elemenata.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/30BE44Bozic

ПРИЛАГОЂЕЊЕ ИНТЕРНЕТ ПРЕГЛЕДАЧА НА DIRECTFB ГРАФИЧКУ БИБЛИОТЕКУ

AN ADAPTATION OF INTERNET BROWSER TO DIRECTFB GRAPHICS LIBRARY

Милица Божић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Задатак овог рада јесте прилагођење интернет прегледача на произвољну графичку библиотеку, конкретно DirectFB. Овај задатак је сачињен од две целине, које је потребно испунити. У првој целини неопходно је развити тестну апликацију за приказивање графичког садржаја користећи DirectFB библиотеку. У другој целини потребно је прилагодити интернет прегледач, то јесте реализовати Ozone интеграцију, да користи DirectFB библиотеку за приказ садржаја на екран.*

Кључне речи: *Chromium, Ozone платформа, DirectFB графичка библиотека*

Abstract – *The task of this work is to adapt an internet browser to an arbitrary graphics library, specifically DirectFB. This task consists of two parts that need to be fulfilled. In the first part, it is necessary to develop a test application for displaying graphical content using the DirectFB library. In the second part, it is necessary to adapt the internet browser, i.e., to implement Ozone integration, to use the DirectFB library for displaying content on the screen.*

Keywords: *Chromium, Ozone platform, DirectFB graphics library*

1. УВОД

Брзи напредак интернет технологија и растућа потражња за високоперформансним интернет апликацијама истакли су важност ефикасних и прилагодљивих интернет прегледача. *Chromium*, пројекат интернет прегледача отвореног кода, постао је популаран због своје архитектуре и могућности прилагођавања. Разноврсност хардвера и оперативних система представља изазов у осигуравању оптималних перформанси на свим платформама.

Циљ овог рада јесте прилагођење интернет прегледача на *DirectFB* графичку библиотеку. Прво је потребно развити тестну апликацију за приказивање графичког садржаја користећи *DirectFB*. Затим је потребно прилагодити *Chromium* за *Ozone* интеграцију како би користио *DirectFB* за приказ садржаја. Ово омогућава ефикасан рад *Chromium*-а на уређајима са ограниченим графичким ресурсима.

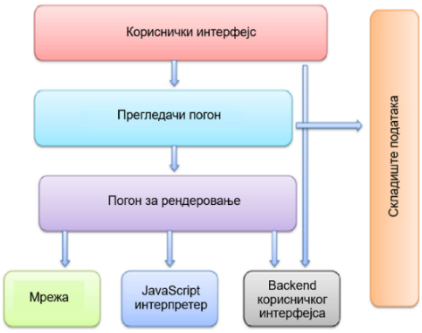
Мотивација за пројекат произилази из потребе за оптимизацијом перформанси интернет прегледача на *уграђеним* системима са мањом графичком моћи. DirectFB нуди ефикасно руковање графиком, што га чини идеалним за такве уређаје. Прилагођавање *Chromium*-а на *DirectFB* може значајно побољшати перформансе рендеровања и корисничко искуство, омогућавајући боље уграђене интернет апликације.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1. Интернет прегледачи

Интернет прегледачи су основне софтверске апликације које корисницима омогућавају приступ, преузимање и интеракцију са садржајем на *World Wide Web*-у. Они служе као интерфејс између корисника и огромних ресурса доступних на мрежи.

Главна функција интернет прегледача јесте приказивање интернет ресурса, захтевајући га од сервера, у интернет прозору. Ресурс је најчешће *HTML* документ. Лоциран је на месту означеном *URL* адресом.



Слика 1. Основне компоненте прегледача

Основне компоненте прегледача су (Слика 1):

- Кориснички интерфејс,
- Прегледачки погон,
- Погон за рендеровање,
- Мрежа,
- *Backend* корисничког интерфејса,
- *JavaScript* интерпретер,
- Складиште података.

Најпознатији интернет прегледачи укључују *Google Chrome*, *Mozilla Firefox*, *Microsoft Edge*, *Apple Safari* и *Opera*. Сваки од ових прегледача заузео је значајан удео на тржишту захваљујући својим јединственим карактеристикама, перформансама и корисничким интерфејсима. *Google Chrome* се посебно истиче као један од најпопуларнијих интернет прегледача на свету. Изграђен је на *Chromium* пројекту отвореног

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био проф. др Илија Башичевић.

кода, који служи као основа за неколико других прегледача.

2.2. *Chromium*

Chromium је пројекат интернет прегледача отвореног кода, који служи као основа за популарне прегледаче као што су *Google Chrome*, *Microsoft Edge* и *Brave*. Пројекат води *Google* уз доприносе бројних појединаца и организација. *Chromium* користи механизам *Blink*, настао од *WebKit*-а 2013. године.

Кључна карактеристика је његова природа отвореног кода, омогућавајући програмерима да прегледају, модификују и доприносе пројекту, што промовише транспарентност и заједнички развој. *Chromium* је изграђен на вишепроцесној архитектури која побољшава стабилност и безбедност изоловањем сваке картице у посебан процес.

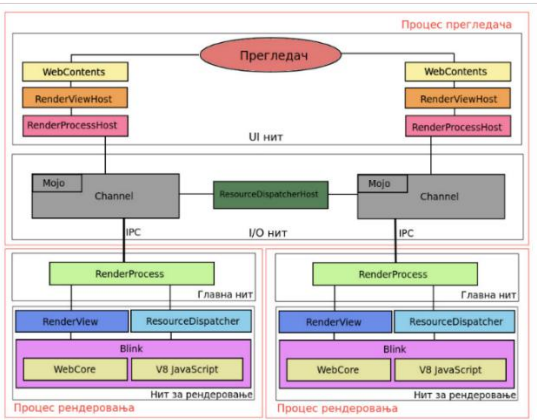
Подржава различите интернет стандарде и технологије као што су *HTML*, *CSS*, *JavaScript* и *WebGL*, и укључује брзи *JavaScript* погон *V8*. Активна заједница програмера доприноси пројекту кроз извештаје о грешкама, закрпе и нове функције, а редовна ажурирања побољшавају безбедност и перформансе.

Google Chrome додаје додатне функције као што су синхронизација са *Google* налозима, уграђени *PDF* прегледач, подршка за власничке медијске кодексе и интеграција са *Google* услугама.

2.2.1. Архитектура *Chromium*-а

Chromium користи више процеса да заштити целокупну апликацију од грешака и кварова у погону рендеровања или другим компонентама. Такође ограничава приступ из сваког процеса за рендеровање другим процесима и остатку система. На неки начин, ово доноси претраживању веба предности које су оперативним системима донели заштита меморије и контрола приступа [1].

Архитектура *Chromium*-а је дизајнирана као вишепроцесни модел који се састоји од процеса прегледача и више процеса рендеровања (Слика 2).



Слика 2. Приказ мултипроцесне архитектуре *Chromium*-а

Процес прегледача:

- *UI* нит: Одговорна за приказивање интернет страница.
- *I/O* нит: Рукује међупроцесном комуникацијом (*IPC*) и мрежним комуникацијама.

Главни објекти *UI* нити:

- *RenderProcessHost*: Повезује се са једним процесом рендеровања и одржава однос један-према-један. Шаље поруке специфичне за приказ на *RenderViewHost*.
- *RenderViewHost*: Прима поруке специфичне за приказ од *RenderProcessHost*-а и управља навигационим командама и догађајима уноса.
- *WebContents*: Представља таб који приказује интернет страницу.
- *Browser*: Представља прозор прегледача и може садржати више *WebContents* инстанци (*tab*-ова).

Главни објекти *I/O* нити:

- *Channel*: Олакшава *IPC* комуникацију између процеса прегледача и процеса рендеровања.
- *ResourceDispatcherHost*: Управља свим мрежним захтевима и комуникацијама.

Процес рендеровања:

- Главна нит: Управља међунитном комуникацијом.
- Нит за рендеровање: Рукује конструкцијом интернет страница.

Главни објекти главне нити:

- *RenderProcess*: Олакшава комуникацију између нити за рендеровање и *I/O* нити процеса прегледача.

Главни објекти нити за рендеровање:

- *ResourceDispatcher*: Прослеђује захтеве за преузимање садржаја процесу прегледача путем *IPC*-а.
- *Blink*: Погон за рендеровање, одговоран за конструкцију *DOM*-а и распоред страница, укључује *WebCore* и *V8 JavaScript* интерпретер.

Вишепроцесна архитектура *Chromium*-а побољшава перформансе и сигурност изолацијом процеса, осигуравајући стабилно и брзо искуство прегледања.

3. КОНЦЕПТ РЕШЕЊА

У циљу логичког поједностављења имплементације и лакшег праћења током израде, рад је подељен на три целине, односно на три фазе израде. Свака фаза представља логички заокружену целину.

У првом сегменту потребно је подесити окружење и *Chromium* пројекат *Linux* платформи.

У другој фази неопходно је упознавање са постојећом документацијом и њена анализа везано за *Chromium Web Browser* и *DirectFB* графичку библиотеку.

У трећој фази потребно је постојеће *DirectFB* примере наместити и прилагодити *Linux* платформи тако да се могу покренути приказујући садржај. С обзиром да *Linux Ubuntu* има проблем са покретањем *DirectFB* апликација, резултат треба да буде *DirectFB* апликација која шаље податке на диск као слику.

У другом делу треће фазе потребно је креирати *BUILD.gn* скелет за нови порт платформе и имплементирати неопходне класе *DirectFB* графичке библиотеке тако да *Chromium* ради користећи графичку библиотеку *DirectFB*.

3.1. Компоненте система

Рад се састоји из две целине. Прва је *DirectFB* апликација која шаље податке на диск као слику. Друга целина је *Chromium* пројекат који ради користећи графичку библиотеку *DirectFB.*

3.2. Платформа *Ozone*

Платформа *Ozone* је део *Chromium* екосистема, који служи да апстрахује код специфичан за различите платформе и прозорске системе. Ова апстракција омогућава *Chromium*-у да ради у различитим окружењима без потребе за поновним писањем кода за сваку платформу.

Ozone је слој апстракције платформе испод *Aura* прозорског система који се користи за унос ниског нивоа и графику. Када се заврши, апстракција ће подржати основне системе у распону од уграђених *SoC* циљева до нових *X11*-алтернативних прозорских система на Linux-у као што су *Wayland* или *Mir* да би се приказао *Aura Chromium* пружањем имплементације интерфејса платформе [2].

Главне карактеристике *Ozone* платформе су:

- Апстракција платформе,
- Вишеструки бекендови,
- Интеграција графичког стека,
- Руковање уносом,
- Управљање прозорима,
- Приступачност,
- Изазови интеграције,
- Сарадња у заједници,
- Безбедност и стабилност.

Ozone платформа је кључни део *Chromium* архитектуре, који омогућава конзистентне перформансе и понашање на различитим оперативним системима, прилагођавајући се специфичним аспектима сваке платформе и промовишући модуларност, прилагодљивост и сарадњу у пројекту отвореног кода.

3.2. Графичка библиотека *DirectFB*

DirectFB (*Direct Frame Buffer*) је софтверска библиотека са малим меморијским отиском која обезбеђује убрзање графике, руковање улазним уређајима и слој апстракције, као и интегрисани систем прозора са подршком за прозирне прозоре и више слојева приказа на врху *Linux* бафера оквира без потребе модификације језгра. *DirectFB* је бесплатан софтвер отвореног кода који подлеже условима *GNU* мање опште јавне лиценце (*LGPL*) [3].

DirectFB је танка библиотека која обезбеђује потпуни слој апстракције хардвера са софтверским резервама за сваку графичку операцију коју основни хардвер не подржава, *DirectFB* додаје графичку снагу уграђеним системима и поставља нови стандард за графику под *Linuxom*.

DirectFB извршава само следеће задатке преко */dev/fb* [4]:

- Подешавање видео режима (резолуција, дубина боје и времена),
- Меморијско мапирање бафера оквира и *I/O* портова и
- Промена области приказа (нпр. За двоструки бафер).

3.4. *BUILD.gn*

GN систем за грађење је мета-систем за грађење који је развио *Google* првенствено за *Chromium* пројекат. Дизајниран је да генерише датотеке за грађење за *Ninja*, који је брз систем за грађење ниског нивоа. *GN* има за циљ да ефикасно рукује сложеношћу великих пројеката, пружајући баланс између флексибилности, брзине и лакоће коришћења.

Кључне функционалности *BUILD.gn* укључују:

- Дефинисање циљева грађења – специфицира различите типове циљева грађења, као што су извршне датотеке, дељене библиотеке, статичке библиотеке и скупови извора.
- Управљање зависностима – управља зависностима између различитих компоненти пројеката, осигуравајући да свака компонента буде изграђена у исправном редоследу.
- Конфигурационе поставке – укључује разне конфигурационе поставке, као што су заставице компајлера, директоријум за укључивање и платформско-специфичне поставке, које контролишу процес компилације.

4. ПРОГРАМСКО РЕШЕЊЕ

4.1. Тестна апликација за приказивање графичког садржаја користећи *DirectFB* библиотеку

Први задатак је био развој апликације за приказивање графичког садржаја користећи *DirectFB* библиотеку. Због проблема са покретањем *DirectFB* апликација на *Ubuntu*-у, решење је било чување излазних података на диску у облику слике.

На *Github*-у се могу наћи примери једноставних *DirectFB* апликација за демонстрацију различитих функција. За овај задатак је коришћен пример *df_window*, који показује како креирати, управљати и комуницирати са више прозора користећи *DirectFB.*

Као и у другим примерима, коришћена је макро дефиниција за проверу повратне вредности *DirectFB* функција, која избацује поруку о грешци ако резултат није *DFB_OK*.

За чување излазних података на диску коришћена је функција *Dump()*, која штампа информације о површини као што су величина, формат и други детаљи, помажући у инспекцији и дијагнози током развоја.

4.2. Прилагођење интернет прегледача да користи *DirectFB* библиотеку за приказ садржаја на екран

Ова целина описује имплементацију прилагођавања *Chromium* интернет прегледача за коришћење *DirectFB*-а кроз *Ozone* слој апстракције платформе. Након подешеног развојног окружења први корак је био креирање скелета *BUILD.gn* за нови платформски порт, а срж софтверског решења укључивала је имплементацију компоненти специфичних за *DirectFB*:

- *DirectFBConnector*,
- *DirectFBPlatformWindow*,
- *DirectFBPlatformScreen*,

- *DirectFBSurface,*
- *DirectFBSurfaceFactory,*
- *GLOzoneEGLDFB,*
- *OzonePlatformDirectFB.*

DirectFBConnector класа служи као синглтон одговоран за иницијализацију и деиницијализацију *DirectFB* оквира. Управља главним *DirectFB* интерфејсом, слојем приказа, креирањем прозора и компонентама за руковање догађајима. Ова класа прати синглтон образац како би се осигурало да постоји једна инстанца одговорна за одржавање *DirectFB* стања током животног циклуса апликације. Она обухвата заплетености иницијализације *DirectFB*-а, креирања прозора и управљања ресурсима, пружајући чист и организован интерфејс за остатак апликације за интеракцију са *DirectFB* оквиром.

Класа *OzonePlatformDirectFB* служи као специјализована имплементација *Ozone* платформе, прилагођена конкретно за *DirectFB* окружење. Интегрише различите компоненте неопходне за управљање прозорима, површинама, методама уноса и приказима, чиме омогућава беспрекорну графичку и улазни интерфејс за апликације.

Све те класе заједно чине окосницу графичког и подсистема корисничког интерфејса дизајнираног за интеграцију са *DirectFB* платформом. Свака класа има специфичну улогу, доприносећи укупној функционалности и модуларности система. Употреба шаблона дизајна, као што су фабрички образац и синглтон образац, побољшава организацију кода и могућност одржавања. Интеграција графичких библиотека као што су *Skia* и *OpenGL* показује свестраност и проширивост система за руковање различитим графичким захтевима.

5. РЕЗУЛТАТИ

5.1. Тестна апликација за приказивање графичког садржаја користећи *DirectFB* библиотеку

С обзиром да *Ubuntu* има проблем са покретањем *DirectFB* апликација, решење је била *DirectFB* апликација која излазне податке чува на диску у виду слике. За резултат смо требали добити слику са *DirectFB* логом, што се и види на слици (Слика 3).

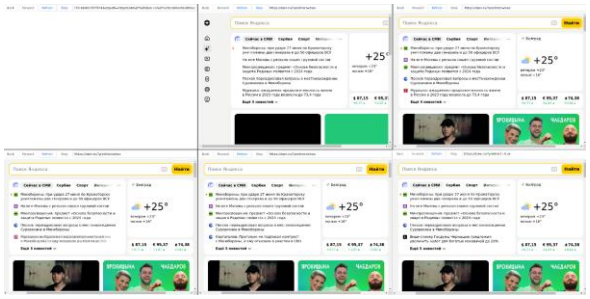
Слика 3. Приказ резултата при покретању *DirectFB* апликације

5.2. Прилагођење интернет прегледача да користи *DirectFB* библиотеку за приказ садржаја на екран

У сврху верификације резултата, покренули смо различите странице и видели како се страница парсира, рендерује и приказује заправо.

Кључну улогу има функција *SaveAsImage()* у класи *DirectFBConnector*. Она служи као алтернативни механизам за проверу графичког излаза приликом покретања *Chromium*-а на *Linux*-у са *DirectFB*-ом. Ова функција хвата тренутно стање *DirectFB* прозора и чува га као слику на диску.

За пример је узето учитавање странице која константно мења свој изглед због реклама које се смењују на страници. Као, на пример, код *Yandex.ru* странце. Страница се врло брзо учитавала али при томе је константо мењала изглед, те је сачувано чак 170 слика пре него је прекинуто. Учитавање је било успешно као што се може видети на сликама (Слика 4).



Слика 4. Приказ учитавања странице *Yandex.ru*

6. ЗАКЉУЧАК

Циљ рада је био да реши изазов прилагођавања *Chromium* интернет прегледач за коришћење *DirectFB*, лагане графичке библиотеке, кроз *Ozone* апстрактни слој платформе. Интеграција *DirectFB* у *Chromium* имала је за циљ да побољша перформансе прегледача на уграђеним системима са ограниченим могућностима графичке обраде. Систематским приступом овом задатку, демонстрирали смо изводљивост и предности таквих адаптација, доприносећи вредним увидима и практичним решењима у области развоја интернет прегледача.

Овај рад доприноси текућем развоју и оптимизацији интернет прегледача за разноврсне хардверске платформе. Успешна интеграција *DirectFB*-а у *Chromium* истиче флексибилност архитектуре *Chromium*-а, показујући његову прилагодљивост различитим графичким библиотекама. Увиди стечени из овог пројекта могу бити основа за будући рад на оптимизацији интернет прегледача за уграђене системе и друге специјализоване употребе.

7. ЛИТЕРАТУРА

[1] Multi-process Architecture; URL: https://www.chromium.org/developers/design-documents/multi-process-architecture/
[2] Ozone Overview; URL: https://chromium.googlesource.com/chromium/src/+/HEAD/docs/ozone_overview.md
[3] Wikipedia: DirectFB; URL: https://en.wikipedia.org/wiki/DirectFB
[4] DirectFB; URL: https://elinux.org/DirectFB

Кратка биографија:

Милица Божић рођена је у Добоју 1998. год. Мастер рад на Факултету техничких наука из области Рачунарство и аутоматика одбранила је 2024. год.
контакт: milicab98@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/30BE45Popovic

NEUROMORFNO RAČUNARSTVO – IMPULSNE NEURONSKE MREŽE

NEUROMORPHIC COMPUTING – SPIKE NEURAL NETWORKS

Anđela Popović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Neuromorfno računarstvo je oblast u razvoju koja nalazi sve veću primenu. U ovom radu biće ukratko prikazani principi funkcionisanja neuromorfnog računarstva, kao i rezultati implementacije jedne Spiking (impulsne) neuronske mreže u kojoj su sinapse obučene da svoje težine prilagode odgovarajućim ulazima.*

Ključne reči: *Neuromorfno računarstvo, Impulsne neuronske mreže, LIF model, STDP pravilo*

Abstract – *Neuromorphic computing is an evolving field with increasing practical applications. This paper will briefly present the principles of neuromorphic computing, as well as the results of implementation a Spiking Neural Network, where the synapses are trained to adjust their weights according to the corresponding inputs.*

Keywords: *Neuromorphic computing, Spiking Neural Networks, LIF model, STDP rule*

1. UVOD

Neuromorfno računarstvo predstavlja interdisciplinarno polje koje je inspirisano principima funkcionisanja mozga [1]. U nedostatku prevoda na srpski jezik termin engl. *Spiking Neural Networks (SNN)* prevešćemo kao Impulsne neuronske mreže, čime se ističe priroda signala odnosno *spike*-a.

2. NEUROMORFNO RAČUNARSTVO

Neuromorfno računarstvo zasniva se na podražavanju biološkog nervnog sistema (engl. *neuro-* odnosi se na nervni sistem, i engl. *morphic-* odnosi se na oblik ili strukturu). Kod tradicionalnih veštačkih neuronskih mreža (engl. Artificial Neural Networks - ANN) svaki neuron vrši nekakvo računanje, rezultat propagira u naredni sloj, ponovo svaki neuron iz tog sloja vrši računanje, propagira dalje, i tako dok se ne dođe do rešenja. Ne moraju svi neuroni u mozgu da se pobude i obrade novu informaciju (stimulus), ovo bi trajalo predugo. Dovoljno je da samo određeni neuroni pošalju impulse, pa u zavisnosti od toga koji neuroni i kada, mozak donosi zaključak odnosno rešenje.

2.1. Neuroni, sinapse i akcioni potenijal

Neuron je osnovna procesna jedinica neuronske mreže. Neuroni informacije primaju, obrađuju i prenose putem električnih signala, u biologiji poznatih kao akcioni potencijali. Ovi signali se prenose putem veza između neurona koje se zovu sinapse [1]. Ključni koncept u Impulsnim neuronskim mrežama (SNN) je plastičnost sinapsi, odnosno sposobnost sinaptičkih veza da se menjaju tokom vremena na osnovu iskustva.

Akcioni potencijal je kratkotrajna promena membranskog potencijala koja se dešava kada se dostigne određeni prag. Generisanje akcionog potencijala je rezultat depolarizacije membrane neurona do tačke praga. Do stvaranja akcionog potencijala („okidanje neurona") dolazi kada potencijal dosegne prag „od dole". Okidanje neurona je drugim rečima ispaljivanje impulsa odnosno *spike*-a odakle i naziv.

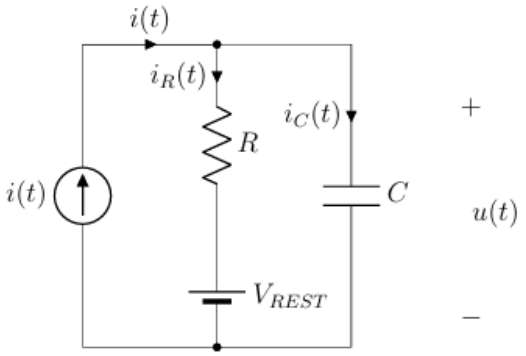
Nakon okidanja dolazi period refrakcije. Tokom ovog perioda je teže ili nemoguće generisati novi impuls.

2.2. Modeli neurona

Najpoznatiji modeli neurona su Hodgkin–Huxley model (HH) [2], Leaky Integrate-and-Fire model (LIF) [1] i Ižikevič model (IZK) [3].

Hodgkin–Huxley model je jedan od prvih detaljnih matematičkih modela koji opisuju elektrofiziološko ponašanje neurona. Razvijen je 1952. godine od strane Alana Hodžkinsa i Endrua Hakslija, za šta su dobili Nobelovu nagradu. Matematičke jednačine koje opisuju dinamiku membranskog potencijala u HH modelu su kompleksne i uključuju nelinearne diferencijalne jednačine koje opisuju brzinu promene membranskog potencijala u zavisnosti od vremena i membranskog potencijala. Zbog toga, nemoguće je napraviti veliku SNN sa ovakvim modelom neurona.

Leaky Integrate-and-Fire model je jednostavan ali efikasan model koji se modeluje pomoću kondenzatora i otpornika, predstavljen na slici 1 i narednim jednačinama.



Slika 1. *LIF model*

$$i(t) = i_R(t) + i_C(t) \tag{1}$$

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

$$\mathrm{u(t)} = \mathrm{R} \cdot \mathrm{i_R(t)} + \mathrm{V_{REST}}$$
$$\Longrightarrow \mathrm{i_R(t)} = \frac{\mathrm{u(t)} - \mathrm{V_{REST}}}{\mathrm{R}} \quad (2)$$

$$i_C(t) = C\frac{du(t)}{dt} \quad (3)$$

$$C\frac{du(t)}{dt} + \frac{u(t) - V_{REST}}{R} = i(t) \quad (4)$$

$$\frac{du(t)}{dt} + \frac{u(t)}{RC} = \frac{1}{C}i(t) + \frac{V_{REST}}{RC} \quad (5)$$

Vidimo da je diferencijalna jednačina (5) u obliku:

$$\frac{du(t)}{dt} + p(t)u(t) = q(t)$$

gde je $\mathrm{p(t)} = 1/\mathrm{RC}$ i $q(t) = \frac{1}{C}i(t) + \frac{V_{REST}}{RC}$. Možemo rešiti ovu jednačinu koristeći faktor $\mu(t)$, tako da je:

$$\mu(t)p(t) = \frac{d\mu(t)}{dt} \Longrightarrow \frac{d\mu(t)}{\mu} = p(t)dt \quad (6)$$

Integrirajući obe strane jednačine (5) dobija se:

$$\int \frac{d\mu(t)}{\mu} = \int p(t)dt \Longrightarrow \ln \mu\,(t) = \int p(t)dt \quad (7)$$

Prema tome

$$\mu(\mathrm{t}) = \mathrm{e}^{\int \mathrm{p(t)dt}} = \mathrm{e}^{\int 1/\mathrm{RCdt}} = \mathrm{e}^{\mathrm{t/RC}} \quad (8)$$

Dalje, množenjem obe strane diferencijalne jednačine (5) sa $\mu(t)$:

$$\mu(t)\frac{du(t)}{dt} + \mu(t)p(t)u(t) = \mu(t)q(t) \quad (9)$$

Uvrštavanjem jednačine (6) dobija se:

$$\mu(t)\frac{du(t)}{dt} + \frac{d\mu(t)}{dt}u(t) = \mu(t)q(t) \quad (10)$$

Primenom Lajbnicovog pravila, $(u \cdot v)' = uv' + u'v$ sledi

$$\frac{d}{dt}[\mu(t)u(t)] = \mu(t)q(t) \Longrightarrow \frac{d}{dt}\left(e^{t/RC}u(t)\right) = e^{t/RC}\left[\frac{1}{C}i(t) + \frac{V_{REST}}{RC}\right] \quad (11)$$

Da bi se rešila ova jednačina po $u(t)$, potrebno je integrirati obe strane jednačine:

$$e^{t/RC}u(t) = \int e^{t/RC}\left[\frac{1}{C}i(t) + \frac{V_{REST}}{RC}dt\right]$$
$$= \frac{1}{C}\int e^{t/RC}i(t)dt + \frac{V_{REST}}{RC}\int e^{t/RC}dt$$
$$= \frac{1}{C}\int e^{t/RC}i(t)dt + V_{REST} \cdot e^{t/RC} \quad (12)$$

U jednostavnom LIF neuronskom modelu, pretpostavljamo $i(t) = \delta(t)$. Delta funkcija je jednaka nuli svuda osim za $t = 0$. Jedna od posledica toga je da ako se pomnoži bilo kojom funkcijom, nije bitno koje su vrednosti te funkcije sem za $t = 0$. To znači da:

$$\int_{-\infty}^{+\infty} f(t)\delta(t)dt = \int_{-\infty}^{+\infty} f(0)\delta(t)dt$$
$$= f(0)\int_{-\infty}^{+\infty}\delta(t)dt$$
$$= f(0) \quad (13)$$

Sada se može rešiti integral za jednačinu (13):

$$e^{t/RC}\mathrm{u}(t) = \frac{1}{C}\int e^{t/RC}i(t)dt + V_{REST} \cdot e^{t/RC}$$
$$= \frac{1}{C}\int e^{0/RC}\delta(t)dt + V_{REST} \cdot e^{t/RC}$$
$$= \frac{1}{C}e^{0/RC} + V_{REST} \cdot e^{t/RC}$$
$$= \frac{1}{C} + V_{REST} \cdot e^{t/RC} \quad (14)$$

Stoga

$$\mathrm{u(t)} = \frac{1}{\mathrm{C}}\mathrm{e}^{-\mathrm{t/RC}} + \mathrm{V_{REST}} \quad (15)$$

U projektu koji je opisan u ovom radu implementiran je LIF model, zbog svoje zadovoljavajuće biološke podobnosti i računarske efikasnosti.

Ižikevič model neurona postavio je Eugen Ižikevič 2003. godine u kom je neuron opisao dvodimenzionalnim setom diferencijalnih jednačina (16):

$$\frac{dv(t)}{dt} = 0.04v^2(t) + 5v(t) - u(t) + i(t) + 140$$
$$\frac{du(t)}{dt} = a\big(bv(t) - u(t)\big) \quad (16)$$

Resetovanje potencijala membrane se modeluje:

Ako $v \geq 30mV$, onda $= \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases}$ (17)

Ovde, v i u su bezdimenzione promenljive, a, b, c, d su bezdimenzioni parametri, t je vreme. Promenljiva v predstavlja membranski potencijal neurona, promenljiva u predstavlja promenljivu oporavka membrane (u čije detaljnije objašnjavanje se nećemo upuštati u ovom radu).

3. IMPULSNE NEURONSKE MREŽE

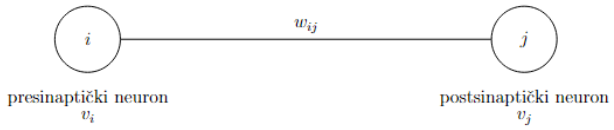
Ključne karakteristike su [1]:

- Bitna karakteristika SNN je modelovanje vremenskih karakteristika informacija putem sekvenci akcionih potencijala. Umesto obrade statičkih ulaznih podataka, SNN obrađuju temporalne sekvence informacija koje se prenose putem impulsa.
- Učenje u SNN često se zasniva na biološki inspirisanim principima učenja, kao što su Spike-Timing-Dependent Plasticity (skraćeno STDP), ili različitim varijantama ovog pravila. Ova pravila omogućavaju mreži da prilagodi svoje sinapse odnosno težine na osnovu tačnog vremenskog odnosa između akcionih potencijala neurona, odnosno njihove uzročnosti.
- Razlike u odnosu na ANN: ANN koriste kontinuirane vrednosti (obično realne brojeve) za predstavljanje aktivnosti neurona i propagaciju signala kroz mrežu. SNN koriste diskretne impulse kako bi predstavile aktivnost neurona. Ovi impulsi se generišu u određenim vremenskim trenucima i prenose se kroz mrežu.

3.1. STDP pravilo

STDP pravilo učenja je biološki inspirisano pravilo učenja koje opisuje kako jačina sinapsi između neurona slabi ili jača u skladu sa vremenskim odnosom između impulsa tih neurona. Donald Heb ga je 1949. godine

definisao sledećom rečenicom: ,,*Neurons that fire together, wire together*“ (neuroni koji okidaju zajedno, međusobno se povezuju) [4]. Na slici 2 prikazana je šema dva neurona i sinapse između njih.



Slika 2. *Presinaptički i postsinaptički neuron, sinapsa*

Ako neuron i koji je povezan sa neuronom j izaziva okidanje neurona j onda njihovu vezu (sinapsu) w treba ojačati. Drugim rečima, ako presinaptički neuron generiše impuls pre postsinaptičkog neurona, sinapsu treba ojačati. Ako presinaptički neuron generiše impuls posle postsinaptičkog neurona, sinapsu treba oslabiti. Jačina sinapse se menja zavisno od tačnog vremenskog intervala između presinaptičkog i postsinaptičkog impulsa, $\Delta t = t_{post} - t_{pre}$. Ta promena Δw modeluje se eksponencijalnom funkcijom (jednačine 18 i 19), gde su $A_+ > 0$ i $A_- < 0$ ponderišući faktori.

$$\Delta w_+ = A_+(w) * e^{\frac{-|\Delta t|}{t}};\ \Delta t > 0 \quad (18)$$

$$\Delta w_- = A_-(w) * e^{\frac{-|\Delta t|}{t}};\ \Delta t < 0 \quad (19)$$

Konačno ažuriranje jačine sinapsi računa se na sledeći način:

$$w_{ij} = \begin{cases} w_{ij} + \eta \Delta w_{ij}(w_{max} - w_{ij}), & \Delta t > 0 \\ w_{ij} + \eta \Delta w_{ij}(w_{ij} - w_{min}), & \Delta t < 0 \end{cases} \quad (20)$$

Gde je η koeficijent učenja koji se bira iz opsega $w_{min} < w_{ij} < w_{max}$ [4].

4. IMPLEMENTACIJA

Projekat koji je opisan u ovom radu implementira SNN sa LIF modelom, STDP pravilom, i „pobednik uzima sve“ (engl. *winner take all*) strategijom. Ova strategija znači da kada jedan neuron „pobedi“ za određeni ulaz, odnosno ima najprilagođenije težine sinapsi, ostali neuroni će biti prigušeni (smanjiće im se potencijal membrane).

Algoritam radi tako što za svaki neuron izračunava potencijal pomoću LIF modela. Zatim proverava da li je došlo do impulsa, ako jeste, primenjuje se pravilo „pobednik uzima sve“. Dalje se nad svim neuronima primenjuje STDP pravilo. Ova tri pravila izvršavaju se za sve vremenske odabirke. Nakon prolaska kroz sve vremenske odabirke prelazi se na sledeći ulaz (sliku) i nakon što se prođe kroz sve slike, sledi naredna epoha u kojoj se sve ponavlja. Jedini uslov konvergencije algoritma je ispunjenje zadatog broja epoha.

Ovo praktično znači da jedan neuron može da ispali impuls više puta (ili nijednom) tokom trajanja vremenskog intervala. Ako je neuron na kraju pobednik, to znači da se matrica sinapsi vezana za taj neuron najbolje ,,naučila“ na ulaz za koji je neuron pobedio. Pobednički neuron određuje se za svaki ulaz nakon celog vremenskog intervala. Intuitivno ovo je kao da smo uzeli jednu sliku, pustili sve neurone da je ,,vide“ npr. 200ms, a onda im pokazivali redom slike po 200ms, i tako npr. 20 puta (epoha). Na kraju svih epoha, za svaku sliku imamo finalni pobednički neuron, čije sinapse su naučene upravo na tu sliku. U projektu opisanom u ovom radu sinapse su učene da mapiraju odgovarajuće piksele, tako da se na kraju učenja mogu rekonstruisati ulazi.

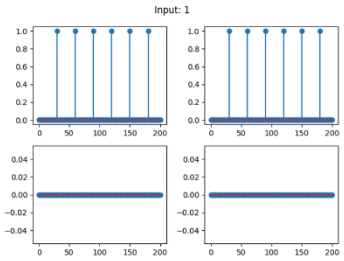
4.1. Predobrada ulaznih slika

Pre nego što počne učenje, potrebno je predobraditi (engl. *preprocessing*) ulazne podatke, u ovom slučaju slike. Prvo se intenziteti svakog piksela slike normalizuju na određeni opseg. Zatim sledi enkodiranje, odnosno potrebno je doći do povorke ulaznih impulsa.

Na osnovu membranskog potencijala svakog piksela, interpolacijom između minimalne i maksimalne frekvencije, dobija se frekvencija impulsa za taj piksel. Niži potencijali imaće niže frekvencije okidanja impulsa i obrnuto. Na osnovu frekvencije izračunava se interval između impulsa, po obrascu *1/frekvencija*. Sada se formira niz diskretnih događaja (vrednosti jedan) gde su impulsi razmaknuti za širinu izračunatog intervala. Ovo je ustvari niz Dirakovih impulsa.

Primera radi, za sliku dimenzija 2x2 gde je gornja polovina bela a donja polovina crna (desni deo slike 4), povorka impulsa (za svaki piksel) data je na slici 3. Crna boja ima vrednost 0, stoga nema nijednog impulsa, bela boja izazvaće najveću gustinu impulsa, a ostale nijanse sive bile bi između.

Svrha ove predobrade je da se od ulaznih podataka dobiju nizovi ulaznih impulsa, u odnosu na koje će se računati vreme Δt za STDP pravilo.



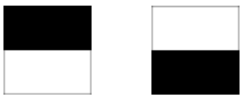
Slika 3. *Povorke impulsa po pikselu*

5. REZULTATI

Mreža je testirana na tri različita skupa podataka. Broj neurona treba da bude veći od broja izlaznih klasa, jer će neki neuroni naučiti šum. Mreža je učena u 20 epoha.

5.1. Početni skup

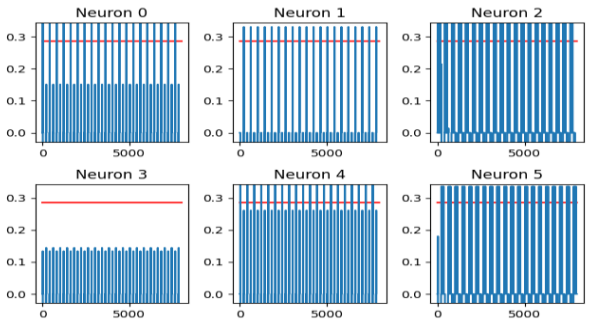
Prvi skup je jednostavan, *proof of concept* skup, koji se sastoji od samo dve ulazne slike (slika 4).



Slika 4. *Prvi ulazni skup*

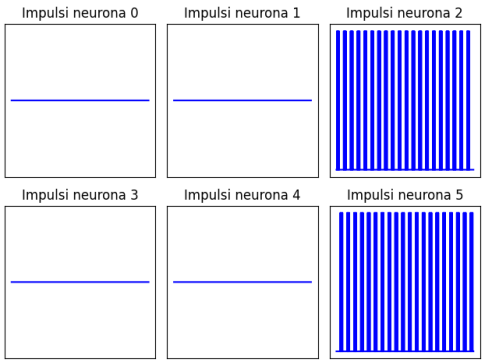
Za ovako mali skup praktično je prikazati potencijale neurona. Na slici 5 prikazani su potencijali 6 neurona gde je na y osi potencijal, a na x osi prikazano je vreme. Crvenom horizontalnom linijom označen je prag. Vidi se da za razliku od ostalih, potencijal neurona 3 nikada ne pređe prag a neuroni 2 i 5 najviše puta prelaze prag, vremenski naizmenično. Po tome možemo zaključiti da su upravo ti neuroni pobednici, svaki za odgovarajuću sliku (naravno algoritam na kraju ispiše pobednike pa nema

potrebe za analiziranjem grafika). Ovo znači da su se oni ,,naučili" za odgovarajuće slike, i samo kada je ta slika ,,prikazana" neuronu, on ispaljuje impulse i prelazi prag.



Slika 5. *Potencijali neurona*

Prikaz ispaljivanja impulsa po neuronu dat je na slici 6. Ovde vidimo da samo pobednički neuroni 2 i 5 ispaljuju impulse. Neuroni 0, 1, 3 i 4, iako nekada pređu prag, ne ispaljuju impulse jer bivaju ,,ugušeni" od strane pobedničkih neurona („pobednik uzima sve").



Slika 6. *Ispaljeni impusi po neuronima*

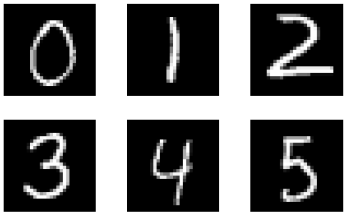
Ako pogledamo vizuelizovan prikaz svih sinapsi po svim neuronima (slika 7) vidimo da sinapse neurona 2 i 5 odgovaraju ulaznim slikama

Slika 7. *Finalne vrednosti sinapsi (x osa) po neuronima (y osa)*

5.2. Rezultati na MNIST skupu

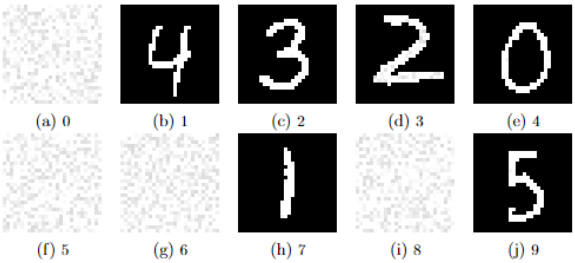
Drugi ulazni skup koji se sastoji od 6 različitih slika dimenzija 2x2 mreža takođe uspešno rekonstruiše.

Treći ulazni skup je MNIST skup od 6 slika (slika 8), cifara 0-5. Radi konciznosti neće ponovo biti dati grafici potencijala, ispaljenih impulsa i sinapsi po neuronima.



Slika 8. *Ulazni skup*

Rezultati rekonstrukcije sinapsi (slika 9) gde je ispod svake slike označen redni broj neurona su:



Slika 9. *Rekonstrukcija MNIST skupa*

Kao što vidimo sinapse pobedničkih neurona su uspešno naučene, dok su ostale naučile šum.

Mreža je uspešno testirana na tri različita ulazna skupa, uz identične parametre, što potvrđuje njenu sposobnost učenja i robustnost. Važno je prilagoditi broj neurona kojih mora biti barem 20% više od ulaznih klasa.

6. ZAKLJUČAK

Ovo istraživanje pokazuje da je SNN mreža koja implementira LIF model, STDP pravilo, odgovarajuće predobrade (i ostale detalje implementacije) sposobna da uspešno prilagodi sinapse (,,nauči") ulaznim slikama, što je jasno prikazano rekonstrukcijom ulaza na osnovu sinaptičkih težina.

7. LITERATURA

[1] W. Gerstner, W.M. Kistler, "Spiking neuron models: Single neurons, populations, plasticity", New York, US: Cambridge University Press, 2002. isbn: 978-0-521-81384-6. doi: 10.1017/CBO9780511815706.

[2] A.L. Hodgkin, A.F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," J. Physiol., Vol. 117, no. 4, pp. 500–544, Aug. 1952. [Online]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/

[3] E. Izhikevich, "Simple model of spiking neurons", *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003. doi: 10.1109/TNN.2003.820440.

[4] N. Caporale, Y. Dan, "Spike timing-dependent plasticity: a Hebbian learning rule," eng, Annual Review of Neuroscience, Vol. 31, pp. 25–46, 2008. doi: 10.1146/annurev.neuro.31.060407.125639.

Kratka biografija:

Anđela Popović rođena je u Šapcu 2000. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva - Računarstvo i automatika odbranila je 2023.god.
kontakt: andjela9popovic@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: (Upisuje redakcija, ostaviti ovaj red)
ДОИ: (Upisuje redakcija, ostaviti ovaj red)

КОМПАРАТИВНА АНАЛИЗА *MAVEN SUREFIRE* И *ALLURE* РАДНИХ ОКВИРА ЗА ГЕНЕРИСАЊЕ ТЕСТ ИЗВЕШТАЈА

COMPARATIVE ANALYSIS OF MAVEN SUREFIRE AND ALLURE TEST REPORT FRAMEWORKS

Ана Гавриловић, *Факултет техничких наука, Нови Сад*

Област –ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Овај рад представља компаративну анализу радних оквира за генерисање извештаја о тестовима Maven Surefire и Allure, користећи као студију случаја веб апликацију за биоскоп и GitHub Actions pipeline-ове. Рад евалуира метрике као што су време генерисања извештаја, употреба ресурса и величина извештаја, као и кориснички интерфејс, опције прилагођавања и лакоћа интеграције у оквиру CI/CD pipeline-а, истичући предности и мане сваког оквира. Резултати су показали да се Maven Surefire истиче брзином и ефикасношћу, генеришући мање извештаје са мањом потрошњом ресурса, док Allure пружа детаљније, прилагодљиве извештаје, али уз већу цену у погледу перформанси и коришћења ресурса.*

Кључне речи: *тестирање софтвера, тест извештаји, Allure Reports, Maven Surefire Report*

Abstract – *This thesis presents a comparative analysis of the Maven Surefire and Allure test report frameworks, using the cinema web application and GitHub Actions pipelines as a case study. The paper evaluates metrics such as report generation time, resource usage, and report size, as well as user interface, customization options, and ease of integration within a CI/CD pipeline, highlighting the strengths and weaknesses of each framework. The results showed that Maven Surefire excels in speed and efficiency, generating smaller reports with less resource consumption, while Allure provides more detailed, customizable reports, but at a higher cost in terms of performance and resource usage.*

Keywords: *software testing, test reports, Allure Reports, Maven Surefire Report*

1. УВОД

У области развоја софтвера, тестирање и извештавање играју важну улогу у одржавању квалитета кода, а са све већом сложеношћу софтверских система, потреба за ефикасним алатима за извештавање је значајно порасла. Избор одговарајућег оквира за извештавање је изазован због разноврсности доступних опција и фактора као што су перформансе, лакоћа интеграције, јасноћа извештаја, скалабилност и употребљивост. Овај рад има за циљ да кроз компаративну анализу *Maven Surefire* и *Allure* радних оквира процени њихове предности и мане у контексту поменутих фактора, пружајући увид у то који је алат боље прилагођен специфичним потребама тестирања.

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

2. СПЕЦИФИКАЦИЈА СИСТЕМА

Ефикасност упоредних анализа у великој мери се ослања на добро дефинисан и структуриран систем. Ово поглавље пружа детаљан преглед система који се користи као студија случаја за ово истраживање.

2.1. Веб апликација за биоскоп

Радни оквири за које је вршена компаративна анализа генеришу тест извештаје за тестове *full-stack* веб апликације намењене за употребу биоскопа. Примарна сврха апликације је да олакша онлајн резервацију карата и управљање, за раднике и посетиоце биоскопа. Неке од основних функционалности апликације су преглед филмова, преглед пројекција, резервације пројекција, преглед биоскопских сала, кориснички налози, админ панел и друге.

Frontend је развијен коришћењем *React*-а, *backend* је имплементиран у програмском језику Java и радном оквиру *Spring Boot*, док за базу података апликација користи *PostgreSQL*.

2.2. Тестирање софтвера

За биоскопску веб апликацију, усвојен је вишеструки приступ тестирању, који укључује више врста тестова који покривају различите аспекте апликације. Коришћена је комбинација *unit* тестова, тестова интеграције и E2E (*end-to-end*) тестова. *Unit* тестови представљају основну јединицу тестирања јер се фокусирају на појединачне компоненте у изолованом окружењу. Они осигуравају да свака јединица кода ради како је очекивано. За писање *unit* тестова коришћени су *JUnit* и *Mockito*. Интеграциони тестови служе за тестирање интеракција између различитих компоненти апликације, на пример *backend*-а и базее података. E2E тестови обезбеђују процену рада целе апликације, симулирајући стварне корисничке сценарије како би се потврдило да систем ради како је предвиђено од почетка до краја. Ови тестови су написани помоћу *Selenium* алата.

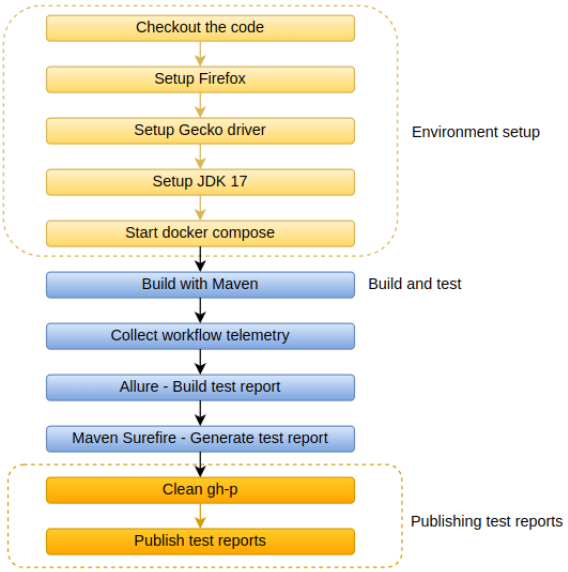
Како би се додатно повећала покривеност тестовима за најразличитије вредности улазних параметара, већина тестова је параметризована. Параметризовани тестови представљају напредну технику тестирања, где се иста тестна логика покреће са различитим улазним параметрима. На овај начин, један тестни случај може покрити широк спектар сценарија и граничних случајева. У контексту упоредне анализе алата за генерисање тест извештаја, параметризовани тестови су били посебно значајни јер се помоћу њих може симулирати веома велики број тест случајева, какав се очекује код већине реалних апликација.

Укупан број тест случајева је 41143, од чега су 31000 *unit* тестови, 10019 интеграциони и 123 E2E тестови.

2.3. Дизајн CI pipeline-а

Токови континуиране интеграције (енгл. *Continuous Integration pipelines* - CI *pipelines*) играју кључну улогу у савременим праксама развоја софтвера, јер аутоматизују процес интегрисања кода написаног од стране више програмера у један заједнички репозиторијум, тако што извршавају све акције које су неопходне пре саме интеграције.

За ову апликацију, написан је CI *pipeline* који је дизајниран тако да аутоматизује процесе *build*-а, тестирања и генерисања извештаја тестирања. У два одвојена корака *pipeline*-а, генеришу се два тест извештаја, један од стране *Maven Surefire* алата, а други од стране *Allure* радног оквира. Пре ових корака, врши се неколико других корака који служе за подешавање окружења за даљи ток *pipeline*-а. Такође је подешена и акција која мери перформансе и потрошњу ресурса корака који генеришу тест извештаје. На слици 1 приказани су кораци CI *pipeline*-а биоскопске апликације.



Слика 1. *Ток и кораци CI pipeline-а апликације*

2.4. Maven Surefire Report

У оквиру алата за аутоматизацију *Apache Maven*-а, постоји више додатка (енгл. *pulgin*) који доприносе процесу тестирања. Најзначајнији додатак је *Maven Surefire Plugin*, дизајниран посебно за покретање тестова. Овај додатак генерише извештаје о тестирању у формату обичног текста или XML формату. Међутим, у овим форматима, тест извештаји нису читљиви обичном кориснику, те их је пожељно конвертовати у HTML формат.

За те сврхе користи се *Maven Surefire Report Plugin [1]*, чија је сврха да парсира тест артефакте из XML формата у HTML фајл, који се може отворити и прегледати у веб претраживачу. Ти извештаји пружају детаљне информације о извршењу теста, укључујући број покренутих, успешних, неуспешних и прескочених тестова, дистрибуцију тестова по пакетима, поруке о грешкама и друге.

За интеграцију оба *pugin*-а потребно их је додати у конфигурацију *Maven* пројекта, затим покренути тестове, и након тога помоћу команде *surefire-report:report-only* изгенерисати извештај.

2.5. Allure Reports

Allure [2] оквир за извештаје је алат дизајниран за генерисање детаљних и визуелно привлачних извештаја о тестирању. Има богат скуп функција и прилагодљив је различитим потребама тестирања. *Allure* подржава широк спектар тест оквира и језика. Овај радни оквир користи тест артефакте добијене након извршавања тестова од стране коришћеног радног оквира за те сврхе и, на основу њих, генерише тест извештаје. Добијени тест извештај представља статичку веб страницу са приказом резултата извршавања тестова, њиховом броју, подељености по категоријама, пакетима, и многобројним графиконима.

Allure се састоји од адаптера за радни оквир и *allure* програма за командну линију. Тест извештаји се генеришу кроз следећа три корака:

1. Инсталирање *Allure* адаптера и програма за командну линију, и додавање подршке за поменути адаптер у оквиру конфигурације пројекта
2. Покретање тестова на исти начин на који би се покретали у технологијама у којима је пројекат имплементиран
3. Генерисање HTML извештаја користећи једну од команди из програма командне линије (*allure generate* или *allure serve*) [3]

Allure има подршку за интеграцију са различитим алатима, као што су *Azure DevOps, GitHub Actions, Jenkins, JetBrains IDEs, Visual Studio Code* и другим. У том случају потребно је додати одговарајућу конфигурацију у односу на алат у који се *Allure* интегрише.

3. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

Ово поглавље има за циљ да пружи увид у практичну имплементацију и техничке детаље делова система који су неопходни како би се извршила упоредна анализа радних оквира за генерисање тест извештаја.

3.1. Конфигурација CI pipeline-а

CI *pipeline* за апликацију биоскопа је имплементиран помоћу *GitHub Actions* алата. *GitHub Actions [4]* представља CI/CD платформу која је интегрисана у оквиру *GitHub* екосистема. *GitHub Actions* омогућава креирање прилагодљивих токова посла (енгл. *workflows*) писањем YAML конфигурационих фајлова.

У главном (енгл. *root*) директоријуму *GitHub* репозиторијума, креиран је нови директоријум *.github/workflows* и у њему документ *ci.yml*. Почетни део документа, у ком се специфицирају назив *pipeline*-а и догађаји га окидају, као и кораци који су задужени за подешавање окружења и за извршавање *build*-а дати су на листнгу 1.

```yaml
name:  CI pipeline for Cinema Application

on:
  pull_request:
    branches: [ "master" ]
  push:
    branches: [ "*" ]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout the code
        uses: actions/checkout@v4.1.1

      - name: Setup Firefox
        uses: browser-actions/setup-firefox@latest

      - name: Setup gecko driver
        uses: browser-actions/setup-geckodriver@latest

      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: maven

      - name: Start docker compose
        run: docker-compose up -d

      - name: Build with Maven
        run: mvn -B package -DGECKODRIVER_PATH=$(which geckodriver) \
                    -DFIREFOX_PATH= --file backend/pom.xml
```

Листнг 1. *CI pipeline - почетна конфигурација*

3.2. Конфигурација Github акције за мерење перформанси

За прикупљање телеметријских података у оквиру CI *pipeline*-а, коришћена је *catchpoint/ workflow-telemetry-action* [5] *GitHub* акција, верзије 1.8.7. Ова акција је дизајнирана тако да хвата метрике перформанси током извршавања CI *pipeline*-а. Интеграцијом ове акције можемо пратити различите аспекте *pipeline*-а, као што су коришћење процесора, потрошња меморије, диска, активност мреже, као и времена извршавања различитих корака. На листингу 2 приказана је конфигурација ове акције у оквиру *pipeline*-а.

```yaml
- name: Collect Workflow Telemetry
  uses: catchpoint/workflow-telemetry-action@v1.8.7
  with:
    metric_frequency: 1s
```

Листинг 2. *Конфигурација акције за мерење перформанси корака pipeline-а*

3.3. Конфигурација и генерисање Maven Surefire извештаја

За *build* и *packaging Spring Boot* апликације користи се *spring-boot-maven-plugin*, док *maven-surefire-report-plugin* генерише извештаје о тестирању. На листингу 3 представљен је начин на који се ови *plugin*-ови додају у пројекат, у оквиру *pom.xml* фајла, а на листингу 4 дата је конфигурација корака *pipeline*-а у оквиру којег се генеришу *Maven Surefire* извештаји.

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

<!-- Maven surefire test reports plugin -->
<reporting>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-report-plugin</artifactId>
            <version>3.2.2</version>
        </plugin>
    </plugins>
    <outputDirectory>github-pages/surefire</outputDirectory>
</reporting>
```

Листинг 3. *Plugin-ови за Maven Surefire у оквиру pom.xml*

```yaml
- name: Maven Surefire - Generate test report
  run: |
    cd ./backend
    mvn surefire-report:report-only -DreportOutputDirectory=github-pages/surefire/
```

Листинг 4. *Конфигурација корака за генерисање Maven Surefire тест извештаја*

3.4. Конфигурација и генерисање Allure извештаја

За интеграцију *Allure* радног оквира у пројекат, прво је потребно укључити *allure-junit5 dependency* у пројекат (листинг 5). Овај *dependency* осигурава да *Allure* може да обради постојеће резултате тестова и генерише извештаје на основу њих.

```xml
<dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-junit5</artifactId>
    <scope>test</scope>
    <version>2.13.0</version>
</dependency>
```

Листинг 5. *Dependency за Allure Reports у pom.xml*

Након додавања зависности, следећи корак је конфигурација CI *pipeline*-а да покрене *Allure* акцију за генерисање извештаја и то је приказано на листингу 6.

```yaml
- name: Allure - Load test report history
  uses: actions/checkout@v4.1.1
  if: always()
  continue-on-error: true
  with:
    ref: gh-p
    path: gh-p

- name: Allure - Build test report
  uses: simple-elf/allure-report-action@v1.9
  if: always()
  with:
    allure_history: ./backend/github-pages
    allure_results: ./backend/target/allure-results
    subfolder: allure-history
```

Листинг 6. *Конфигурација корака за генерисање Allure тест извештаја*

4. РЕЗУЛТАТИ

У овом поглављу су дате вредности метрика као и квалитативних особина радних оквира за генерисање извештаја о тестовима *Maven Surefire* и *Allure*. CI *pipeline* је извршен више пута и иако метрике благо варирају, сваки пут су биле приближних вредности и трендова, те су за анализу узете метрике које су добијене као резултат једног од извршавања. Табела 1 пружа резултате и поређење специфичних аспеката радних оквира, укључујући време генерисања, коришћење ресурса (CPU, RAM, коришћење диска и мреже), величину извештаја, детаљност извештаја, лакоћу интегрисања, лакоћу коришћења и тумачења резултата.

	Maven Surefire	*Allure*
Време генерисања	5s	15s
Величина извештаја	11MB	242MB
Заузеће процесора - системски процеси	81%	89%
Заузеће процесора - кориснички процеси	79%	72%
Потрошња RAM-а	2000MB	3000MB
Network I/O Read (MB/s)	0	19MB/s
Network I/O Write (MB/s)	0	3MB/s
Disk I/O Read (MB/s)	1	0
Disk I/O Write (MB/s)	300MB/s	110MB/s
Кориснички интерфејс	једноставан, функционалан за основне проблеме, без напредних функционалности	атрактиван, модеран, са интерактивним елементима, интерфејс модерних веб апликација
Прилагодљивост	лимитирана прилагодљивост	веома прилагодљив
Лакоћа интеграције	једноставна интеграција као део *Maven build* процеса	захтева инсталацију алата или додатну конфигурацију у зависности од интеграције
Документација	добро документовано	добро документовано

Табела 1. *Резултати метрика и квалитативних особина оба радна оквира за извештавање о тестирању*

5. ЗАКЉУЧАК

Компаративна анализа радних оквира за генерисање извештаја о тестовима *Maven Surefire* и *Allure* истакла је различите предности и мане сваког од алата. *Maven Surefire* је показао брже време генерисања извештаја и мање величине извештаја, што га чини ефикаснијим у смислу брзине и складиштења. Тачније, *Maven Surefire* је генерисао извештаје за 5 секунди са величином од 11 MB, док је *Allure*-у требало 15 секунди и произвео је извештаје од 242 MB. Што се тиче коришћења системских ресурса, *Maven Surefire* је показао нижу употребу RAM-а, од 2000 MB, у поређењу са *Allure*-ових 3000 MB. Међутим, *Allure* је показао веће брзине читања са мреже (19 MB/s) и умерене брзине слања преко мреже (3 MB/s). Поред тога, перформансе брзине писања и читања са диска су варирале, при чему је *Maven Surefire* имао веће брзине писања (300 MB/s), али ниже брзине читања (1 MB/s) од *Allure*-а. Оба оквира су била приближна у коришћењу процесора.

Затим, *Allure* се истакао у дизајну корисничког интерфејса, нудећи визуелно привлачне, интерактивне и прилагодљиве извештаје, који побољшавају могућност корисника да анализира резултате тестова. *Maven Surefire*, иако је био једноставнији, пружио је јасне и детаљне резултате теста. Лакоћа интеграције била је још једна тачка поређења, при чему је *Maven Surefire* био једноставнији за интеграцију у постојеће пројекте засноване на *Maven*-у, док је *Allure* захтевао додатну конфигурацију, али је нудио веће прилагођавање и детаљније карактеристике извештавања. Документација за оба оквира је била добра, иако је *Allure*-ова била опсежнија због ширег скупа функција.

На основу анализе, могу се дати препоруке за коришћење *Maven Surefire* алата када су примарни захтеви брзо генерисање извештаја, ниска потрошња ресурса и једноставна интеграција. *Allure* би било препоручљиво користити у сценаријима где су детаљни и прилагодљиви извештаји од суштинског значаја, а постоји и потреба за интерактивним корисничким интерфејсом како би се олакшала боља анализа резултата теста.

4. ЛИТЕРАТУРА

[1] A. Ramirez, “Maven Surefire Report Plugin – Introduction.” Accessed: Aug. 18, 2024. [Online]. Available: https://maven.apache.org/surefire/maven-surefire-report-plugin/

[2] “Allure Report Docs — Introduction.” Accessed: Aug. 18, 2024. [Online]. Available: https://allurereport.org/docs

[3] “How it works.” Accessed: Aug. 18, 2024. [Online]. Available: https://allurereport.org/docs/how-it-works/

[4] “GitHub Actions documentation,” GitHub Docs. Accessed: Aug. 18, 2024. [Online]. Available: https://docs.github.com/en/actions

[5] “GitHub - catchpoint/workflow-telemetry-action” GitHub. Accessed: Aug. 18, 2024. [Online]. Available: https://github.com/catchpoint/workflow-telemetry-action

Кратка биографија:

Ана Гавриловић рођена је у Шапцу 23. септембра 1999. године. Факултет техничких наука у Новом Саду, студијски програм Рачунарство и аутоматика уписала је 2018. године. Након завршених основних студија, 2022. године, уписала је мастер академске студије из исте области. контакт: ana.gavrilovic247@gmail.com

UDK: 4.41

DOI: https://doi.org/10.24867/30BE46Gavrilovic

Зборник радова Факултета техничких наука, Нови Сад

UDK: 4.9
DOI: https://doi.org/10.24867/30BE47Stojic

МОБИЛНА АПЛИКАЦИЈА ЗА ПРАЋЕЊЕ ЛОКАЦИЈЕ И АУТОМАТИЗАЦИЈУ ИСПОРУКЕ ПОШИЉАКА КОРИШЋЕЊЕМ QR КОДОВА

MOBILE APPLICATION FOR LOCATION MONITORING AND AUTOMATION OF SHIPMENT DELIVERY USING QR CODES

Кристина Стојић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – У раду је представљено проширење постојећег информационог система који је намијењен запосленима у пошти, кроз развој нове мобилне апликације и интеграцију са постојећом веб апликацијом. Главни циљ рада је аутоматизовање праћења испоруке пошиљака и праћење локације поштара. Нова апликација омогућава поштарима да скенирају QR кодове пошиљака како би потврдили њихов статус након испоруке. Поред тога, апликација омогућава праћење локације у реалном времену.

Кључне речи: *пошта, QR код, праћење локације, испорука пошиљака*

Abstract – *The paper presents the extension of the existing information system intended for post office employees, through the development of a new mobile application and integration with the existing web application. The main goal of the work is to automate the tracking of shipment delivery and tracking the location of the postman. The new app allows postmen to scan QR codes on packages to confirm their status after delivery. In addition, the application allows real-time location tracking.*

Keywords: *post office, QR code, location tracking, shipment delivery*

1. УВОД

Пошта је предузеће чија је основна дјелатност пријем, пренос и достава пошиљака. Такође, обавља и додатне услуге попут платног промета, продаје разних артикала (мобилни телефони, SIM картице…), као и пружање услуга интернет провајдера и кабловског оператера [1].

Овај рад је настао као наставак на већ постојећи рад који обухвата информациони систем намијењен запосленима у пошти, са циљем да се унаприједе и поједноставе информациони системи исте. Постојећи систем побољшава прегледност и доступност података, олакшава и убрзава процес уноса уплата и пошиљака, те побољшава евиденцију тренутних статуса истих.

Главна мотивација за наставак овог пројекта лежи у потреби за даљим унапрјеђењем ефикасности и продуктивности поштанске службе. Иако је почетни систем донио значајна побољшања, идентификоване су области које захтијевају додатну аутоматизацију, посебно у сегментима праћења испоруке пошиљака и локације запослених. Аутоматизација ових процеса не само да повећава транспарентност и контролу, већ доприноси бржем и прецизнијем обављању задатака, чиме се побољшава укупни квалитет услуге.

Циљ овог рада је да унаприједи и прошири функционалности постојећег информационог система поште кроз развој мобилне апликације која ће аутоматизовати праћење испоруке пошиљака и омогућити праћење локације поштара. Нова апликација ће омогућити поштарима да скенирају QR кодове пошиљака како би евидентирали оне које је потребно доставити и како би потврдили њихов статус након испоруке. Поред тога, апликација ће омогућити праћење локације поштара у реалном времену, која ће периодично бити ажурирана и слата на постојећу веб апликацију, и самим тим доступна управницима на преглед, чиме би могли да прате ефикасност и кретање својих запослених.

2. СПЕЦИФИКАЦИЈА СИСТЕМА

У овом поглављу биће описана спецификација система, његове компоненте, њихова међусобна комуникација, начин аутентификације корисника и интеграција са екстерним сервисима.

У систему се разликују два типа корисника: управник поште и поштар. Управник има приступ веб апликацији, док поштар може да користи само мобилну.

Након пријаве на веб апликацију, управник има опцију да, из листе свих поштара који раде у његовој пословници, изабере оног чију локацију жели да прати. На мапи се појављује означена тренутна локација поштара, која се ажурира у складу са његовим кретањем. Пријавом на мобилну апликацију почиње праћење локације поштара, коју је могуће прегледати на веб апликацији.

На почетној страници мобилне апликације се налази табела са прегледом информација о пошиљкама које је поштар задужио за испоруку, као и њихов тренутни статус. Поред тога, поштар има могућност да задужи

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

нове пошиљке за испоруку. Након избора те опције, укључује се камера телефона којом је потребно скенирати QR код пошиљке, након чега се она додјељује поштару и приказује у табели са осталим пошиљкама. Приликом испоруке пошиљке, потребно је изабрати ту опцију и поново скенирати QR пошиљке. Након овог скенирања, статус пошиљке се ажурира и она се у табели означава као испоручена.

На слици 1. је визуелно представљена архитектура система. Главне дијелове система чине једна серверска апликација са којом комуницирају и размјењују податке двије клијентске апликације (мобилна и веб), као и *JavaScript* апликација која представља *WebSocket* сервер на који се са мобилне апликације шаљу поруке са информацијама о локацији, које се касније преузимају и приказују на веб апликацији. За складиштење података користи се *PostgreSQL* база података, док је интеграција са *Google Maps* API-јем урађена за приказ локација на мапи.

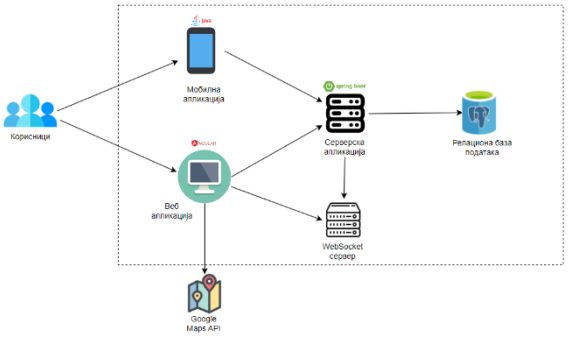
Серверску страну чини монолитна апликација у којој је имплементирана сва пословна логика потребна за реализацију овог система. Код апликације је написан у *Java* програмском језику, уз коришћење *Spring Boot* радног оквира.

Веб апликација служи као графички интерфејс преко ког корисници приступају функционалностима система. Написана је у *TypeScript*-у, при чему је коришћен *Angular* радни оквир, уз коришћење *html* 5, *css* и *Bootstrap*-а за стилизацију. За приказ локације на мапи је урађена интеграција са *Google Maps* API-јем.

Пословна логика мобилне апликације је написана у *Javi*, док се за дефинисање корисничког интерфејса користи XML (*Extensible Markup Language*).

Што се тиче базе података, коришћен је *PostgreSQL* објектно-релациони систем за управљање базама података.

Комуникација између клијентске и серверске стране се врши преко REST-а (*Representational state transfer*). Приликом слања захтјева са клијентске стране ка серверској провјеравају се ауторизација и аутентификација. Сви захтјеви, осим оних за пријављивања корисника, садрже JWT (*JSON Web Token*) токен који се поставља у HTTP (*Hypertext Transfer Protocol*) заглавље.



Слика 1. *Приказ архитектуре система*

3. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

У овом поглављу ће бити описани детаљи имплементације за следеће функционалности:

- скенирање QR кода
- одређивање локације на мобилној апликацији
- слање локације са мобилне на веб апликацију
- интеграција са *Google Maps API*-јем и приказ локације на веб апликацији

3.1. Скенирање QR кода

Функционалност скенирања QR кода са шифром пошиљке омогућава додавање пошиљака за испоруку и ажурирање информација о њима.

Да би била могућа обрада QR кодова извршена је интеграција са ZXing (*Zebra Crossing*) библиотеком [2].

Када се покрене функција за скенирање QR кода, прво је потребно провјерити да ли апликација има дозволу за коришћење камере. Ако дозвола није већ одобрена, апликација тражи исту. У случају да је дозвола одобрена, покреће се камера и започиње скенирање. Ако корисник одбије дозволу, добија обавјештење да је иста неопходна за скенирање QR кода.

Након успјешног скенирања QR кода, обрађује се резултат скенирања који представља шифру пошиљке која се упоређује са евидентираним пошиљкама из система, и ако тамо постоји, додаје у листу пошиљака за испоруку.

3.2. Одређивање локације на мобилној апликацији

Након успјешне пријаве на апликацију почиње праћење локације мобилног уређаја.

Да би била могуће одређивање локације извршена је интеграција са *Google Play Services Location* API-јем [3], додавањем његове зависности у *build.gradle* датотеку. С обзиром да се детектовање локације врши у позадини апликације, креиран је посебан *LocationService* који се покреће након пријаве и који ради без обзира који дио корисничког интерфејса је активан.

Након што је сервис креиран и стартован, позива се метода која се користи за иницијализацију сервиса и постављање почетних вриједности и објеката који ће бити коришћени током рада сервиса.

Сваки пут када се сервис покрене, врши се конектовање на *WebSocket* сервер и затим, периодично, на сваке 2 секунде, позива метода за одређивање локације мобилног уређаја.

Најприје се провјерава да ли апликација има дозволу за детектовање локације. Ако дозвола није већ одобрена, апликација тражи исту. У случају да је дозвола одобрена, иницијализује се *FusedLocationProviderClient* и позивањем методе *getLastLocation()* добија локација уређаја из које се извлаче географска ширина и дужина.

3.3. Слање локације на *WebSocket* сервер

Након што је локација одређена, чува се у бази и шаље на *WebSocket* сервер, одакле ће бити преузета приказана на мапи. За потребе рада са *WebSocket*-има, сервер је написан као посебна апликација помоћу *JavaScript*-а која се извршава у *Node.js* окружењу.

Приликом слања поруке на *WebSocket* сервер са информацијама о локацији као и времену кад је забиљежена, најприје се креира JSON објекат у који се уписују географска ширина, дужина и вријеме, а затим се исти шаље на сервер ако постоји конекција са њим.

3.4. Интеграција са Google Maps API-јем и приказ локације на веб апликацији

Први корак при интеграцији је инсталација пакета за *Google Maps* у Angular апликацији.

Прије него што се апликација повеже са *Google Maps* API-јем, потребно је направити API кључ који обезбјеђује приступ API-ју. Да би се добио кључ, прво је потребно креирати Google Cloud налог [4].

Након што се изврши пријава на налог, потребно је креирати нови пројекат и на њему омогућити коришћење *Google Maps* API-ја. Приликом креирања пројекат потребно је унијети његов назив, и по потреби, изабрати организацију.

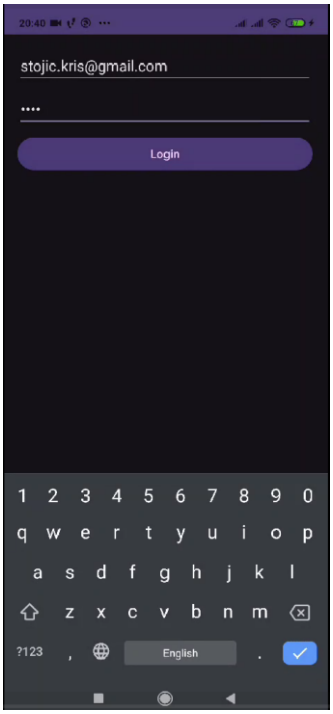
С обзиром да Google наплаћује коришћење својих API-ја, након креирања пројекта потребно је уписати информације са картице за плаћање. Корисницима је доступно 200 долара месечног бесплатног кредита за коришћење *Google Maps* API-ја, а све преко тога се додатно наплаћује.

Послије креирања пројекта, потребно је у библиотеци API-ја пронаћи *Maps JavaScript* API [5] и активирати га да буде омогућен за коришћење на креираном пројекту, а затим креирати API кључ који се може користити за приступ API-ју.

Након што је добијен валидан API кључ, потребно је искористити га у веб апликацији и тако омогућити приказивање локације на мапи. То се остварује тако што се у датотеку *src/app/app.module.ts* укључи и конфигурише *AgmCoreModule*, при чему се уноси претходно изгенерисани API кључ. Након тога је потребно додати одговарајући *html* и *css* код који креира мапу са означеном географском ширином и дужином. Поред тога, потребно је извршити повезивање на *WebSocket* сервер и омогућити са њега преузимање порука са вриједностима географске ширине и дужине које ће бити приказане на мапи.

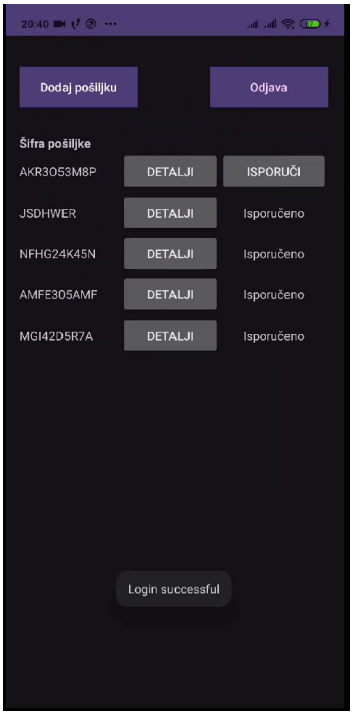
5. ПРИКАЗ ИМПЛЕМЕНТИРАНОГ РЈЕШЕЊА

Приликом покретања мобилне апликације, прва страница која се појави је страница за пријављивање корисника у систем. На слици 2. је приказана форма за пријављивање коју корисник треба да попуни како би могао да користи систем.



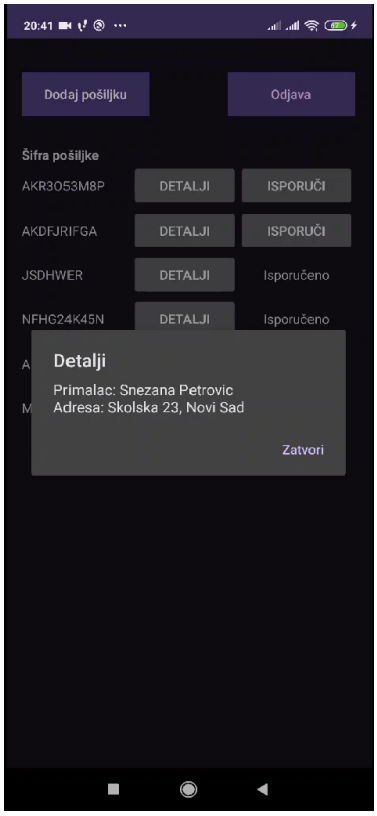
Слика 2. Пријава корисника у систем

Када се корисник успјешно пријави, приказује му се почетна страница која је приказана на слици 3. и која садржи табелу са шифрама пошиљака које је потребно испоручити и њиховим статусима. Кликом на дугме „DODAJ POŠILJKU“, покреће се камера телефона којом је потребно скенирати QR код пошиљке, након чега ће иста бити додата у табелу. Ако корисник жели да означи да је пошиљка испоручена, потребно је да кликне да дугме „ISPORUČI“, након чега се поново покреће камера којом је потребно да скенира QR код пошиљке и тиме потврди њену испоруку.



Слика 3. Почетна страница

На слици 4. приказане су информације о примаоцу пошиљке и адреса на коју је потребно доставити исту, које се отварају кликом на дугме „DETALJI“ на почетној страници.



Слика 4. Детаљи пошиљке

5. ЗАКЉУЧАК

У раду је представљено проширење постојећег информационог систем који је намијењен запосленима у пошти кроз развој нове мобилне апликације и интеграцију са постојећом веб апликацијом. Главни циљ рада је аутоматизовање праћења испоруке пошиљака и праћење локације поштара. Нова апликација омогућава поштарима да скенирају QR кодове пошиљака како би евидентирали оне које је потребно доставити и како би потврдили њихов статус након испоруке. Поред тога, апликација омогућава праћење локације у реалном времену. Детаљно су описани спецификација и архитектура система, технологије које су коришћене у изради, као и детаљи имплементације најбитнијих функционалности.

Иако овај рад представља проширење једног, већ постојећег, може бити додатно унапријеђен и проширен. Неко од проширења система може бити чување и анализирање рута кретања, као и израда статистике и извјештаја на основу пређених километара и времена потрошеног за испоруку, како би се омогућило праћење ефикасности поштара и оптимизација рута. Поред тога, проширење система може бити увођење функционалности која би омогућила приказ свих пошиљака на мапи, након чега би систем могао да генерише оптималну руту, која би узимала у обзир најкраће растојање између локација, и коју би поштар требао да слиједи при испоруци. Додатно, поштар би могао да има могућност да прегледа цијели план испоруке на почетку дана, укључујући предвиђену руту и очекивано вријеме доласка на сваку локацију.

6. ЛИТЕРАТУРА

[1] https://sr.wikipedia.org/sr-ec/%D0%9F%D0%BE%D1%88%D1%82%D0%B0

[2] https://github.com/zxing/zxing

[3] https://developers.google.com/android/reference/com/google/android/gms/location/LocationServices

[4] https://console.cloud.google.com/

[5] https://developers.google.com/maps/documentation/javascript/overview

Кратка биографија:

Кристина Стојић рођена је у Зворнику 1999. год. Завршила је основне академске студије 2022. на Факултету техничких наука. Уписала је мастер студије исте године, студијски програм Електронско пословање.

контакт: stojic.kris@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.41
DOI: https://doi.org/10.24867/30BE48Miskovic

RAZVOJ FUNKCIONALNOSTI UPRAVLJANJA SASTANCIMA ODBORA NA CAPCADE APLIKACIJI

DEVELOPMENT OF BOARD MEETING MANAGEMENT FUNCTIONALITY ON THE CAPCADE APPLICATION

Anđela Mišković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIČKO I RAČUNARSKO INŽENJERSTVO

Kratak sadržaj – *Ovaj rad obuhvata ključne aspekte specifikacije i razvoja softvera, sa fokusom na metodologije proširenja postojećeg softvera sa ciljem zadovoljavanja klijentskih zahteva. Rad obuhvata analizu tržišta, konkurenata, predloge rešenja, procenu uticala i napora predloženih ideja, dizajn korisničkog interfejsa i podelu zadataka, kao i način promocije i praćenja korisnika nove funkcionalnosti upravljanja sastancima odbora u okviru Capcade aplikacije.*

Ključne reči: *Metodologije rada softvera, Capcade, upravljanje sastancima odbora, prioritizacija funckcionalnosti, istraživanje konkurenata*

Abstract – *This paper covers key aspects of software specification and development, with a focus on methodologies for extending existing software to meet client requirements. The work includes analysis of the market, competitors, proposed solutions, assessment of the impact and efforts of the proposed ideas, user interface design and division of tasks, as well as the way to promote and monitor users of the new board meeting management functionality within the Capcade application.*

Keywords: *Software development methodologies, Capcade, board meeting management, functionality prioritization, competitor research*

1. UVOD

Capcade [1] je softver koji nudi centralizaciju alata za orkestraciju poslovnih procesa u *fintech* kompanijama. Ovaj rad istražuje metodologije razvoja softvera u *business-to-business* okruženjima, kao i načine prioritizacije funkcionalnosti. Cilj rada je da predloži specifikaciju i metode proširenja *Capcade* aplikacije funkcionalnosti za upravljanje sastancima odbora na zahtev klijenta.

2. METODOLOGIJA SPECIFIKACIJE I RAZVOJA SOFTVERA U POSLOVNIM OKRUŽENJIMA

Metodologije razvoja softvera u poslovnim okruženjima, posebno u *business-to-business* sektorima, fokusiraju se na rešavanje konkretnih problema kroz jasno definisane faze koje obuhvataju analizu tržišta, prioritizaciju funkcionalnosti i kreiranje zadataka, sve sa ciljem postizanja usklađenih ciljeva i povećanja vrednosti proizvoda za korisnike.

Razumevanje problema klijenta je ključna početna faza koja postavlja temelje za dalji razvoj softverskog rešenja kroz detekciju bolnih tačaka i prostora za unapređenje poslovnih procesa.

U fazi analize tržišta, procenjuju se konkurentna rešenja i razvija se poređenje funkcionalnosti, čime se omogućava jasnija slika o pozicioniranju i specifičnim potrebama korisnika. Uporedna tabela funkcionalnosti [2] igra ključnu ulogu u ovoj fazi, pružajući uvid u to kako postojeća rešenja zadovoljavaju potrebe korisnika i gde postoje mogućnosti za inovaciju [3] [4].

Odabir i prioritizacija funkcionalnosti uključuje istraživanje i definisanje ključnih funkcionalnosti na osnovu korisničkih zahteva i poslovnih potreba [7][8]. Proces prioritizacije se vrši uzimajući u obzir uticaj i napor potreban za implementaciju svake funkcionalnosti, što omogućava efikasno planiranje resursa i vremena [9].

Kada su funkcionalnosti definisane, sledeća faza je kreiranje zadataka kroz definisanje bekloga i principa za održavanje i kreiranje korisničkih priča. Beklog funkcioniše kao lista prioriteta koja se kontinuirano ažurira i optimizuje, a kvalitetne korisničke priče osiguravaju da razvojni tim ima jasan uvid u zahteve korisnika. U ovoj fazi koriste se metodologije kao što su *MoSCoW* i *BUC [9][10]* metode za prioritizaciju zadataka i određivanje veličine priča, čime se omogućava jasnija raspodela resursa i vremena.

U fazi praćenja korisnika i postavljanja ciljeva, ključni rezultati se definišu kroz metode kao što su *OKR [11]*, a platforme poput *Mixpanel*-a omogućavaju precizno praćenje uspeha i definisanje narednih koraka u razvoju proizvoda. Razumevanje potreba korisnika i postavljanje ciljeva na osnovu povratnih informacija od korisnika igra ključnu ulogu u daljem unapređenju proizvoda.

Na kraju, faza marketinga i prodaje obuhvata strategije za povećanje interakcije korisnika sa novom funkcionalosti, kao i poboljštanja položaja softverskog rešenja na tržištu.

Iterativan pristup, fokusiran na kontinualnu evaluaciju i prilagođavanje funkcionalnosti potrebama korisnika, osigurava da se razvijaju rešenja koja će ispuniti specifične zahteve poslovnog okruženja i doneti maksimalnu vrednost korisnicima.

3. CAPCADE

Capcade je platforma za poslovnu saradnju dizajnirana za *B2B* okruženja sa visokim zahtevima za usklađenošću, poput *fintech* usluga. Omogućava sigurne, višestrano

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

povezane radne tokove između organizacionih jedinica i eksternih partnera. Ključne funkcionalnosti uključuju upravljanje dokumentima u realnom vremenu, zadacima, internu komunikaciju i alate za pregovore, čime se povećava efikasnost poslovnih procesa. Platforma smanjuje oslanjanje na *e-mail*, poboljšava sigurnost i transparentnost, i posebno je korisna za složene projekte sa više strana. *Capcade* integriše saradnju i upravljanje podacima radi bolje produktivnosti i usklađenosti.

Capcade ima mikroservisnu arhitekturu, a napisan je u *Spring Boot* i *Angular* radnim okvirima. U kompaniji se praktikuju agilne metodologije razvoja softvera. Struktura kompanije obuhvata *product* tim, *UX/UI* tim, *devops* tim, i nekoliko radnih timova koje sačinjavaju *full-stack* i *front-end* inženjeri, kao i testeri softvera.

4. ANALIZA KLIJENTSKIH ZAHTEVA

Klijent se suočava sa problemom neefikasnog upravljanja sastancima odbora, što uključuje složene procese poput ručnog zapisivanja minuta, odobravanja odluka i evidentiranja glasanja. Ove tradicionalne metode nisu usklađene sa savremenim potrebama za digitalizacijom i optimizacijom radnih procesa. Cilj je kreirati rešenje koje bi omogućilo efikasnije vođenje sastanaka, automatizaciju ključnih koraka i bolje praćenje odluka, čime bi se poboljšala ukupna produktivnost i transparentnost unutar organizacije.
Analizom tržišta, došlo se do zaključka da postoji sve veća potreba za naprednim digitalnim rešenjima za upravljanje sastancima. B2B kompanije sve više koriste digitalne alate kao što su *Zoom* i *Microsoft Teams*, a sa porastom broja sastanaka mesečno, efikasnost i automatizacija postaju ključни. Očekivani rast tržišta alata za upravljanje sastancima od 13% godišnje do 2026. godine jasno pokazuje da postoji značajan potencijal za razvoj novih funkcionalnosti koje bi optimizovale poslovne procece.
Kako bi sastanak bio produktivan i efikasan, potrebno je da postoji cilj, da su svi gosti sastanka dobro pripremljeni, da sastanak ima jasno definisan tok (agenda). Samo oni koji imaju šta da doprinesu treba da budu prisutni na sastanku. Učesnici treba da napuste sastanak sa jasnom slikom o donešenim odlukama i narednim koracima koji će doprineti cilju.
Rezultat analize poznatih rešenja poput *Google Calendar*-a [13], *Calendly*-a [16], *Microsoft Teams*-a [15], *Zoom*-a [17] i *Apple Calendar*-a [14] izdvojen je u tabelu konkurenata [5] na slici 1.

Opis funkcionalnosti iz tabele:

- Podsetnici za sastanke: Obaveštavanje korisnika o predstojećim sastancima radi bolje organizacije.
- Ponovljajući sastanci: Kreiranje sastanaka koji se ponavljaju u intervalima (dnevno, nedeljno, mesečno).
- Prilagođeni šabloni sastanaka: Kreiranje i korišćenje šablona za sastanke prema specifičnim potrebama.
- Zakazivanje slobodnih termina: Pronalaženje i rezervacija slobodne termine za sastanke, često putem deljenja rasporeda.
- Podrška za više kalendara: Upravljanje i praćenje više kalendara iz jednog interfejsa.
- Skladištenje snimka sastanaka: Čuvanje snimaka sastanaka za kasniji pregled.
- AI-generisani sažetak sastanka: Automatsko generisanje sažetka sastanka pomoću veštačke inteligencije.
- Alati za timsku saradnju: Alatke za deljenje dokumenata i komunikaciju unutar platforme.

Funkcionalnost	Google Calendar	Calendly	Microsoft Teams	Zoom	Apple Calendar
Podsetnici za sastanke	☑	☑	☑	☑	☑
Ponavljajući sastanci	☑	☑	☑	✗	☑
Prilagođeni šabloni sastanka	✗	✗	☑	✗	✗
Zakazivanje slobodnih termina	✗	☑	☑	✗	✗
Podrška za više kalendara	☑	☑	☑	✗	☑
Skladištenje snimaka sastanka	✗	✗	☑	☑	✗
Sažetak sastanka generisan pomoću veštačke inteligencije	✗	✗	☑	✗	✗
Alati za timsku saradnju	☑	✗	☑	☑	✗

Slika 1 – Poređenje konkurenata

Uvođenje funkcionalnosti za upravljanje sastancima u *Capcade* donosi brojne benefite za korisnike. Omogućava centralizovano upravljanje svim informacijama o sastancima i transakcijama na jednoj platformi, čime se značajno smanjuje vreme potrebno za organizaciju i pripremu. Povećava efikasnost i saradnju među timovima, omogućavajući lakše deljenje informacija i praćenje napretka projekata. Uz bolje upravljanje pristupom i već postojeće korisničke strukture unutar platforme, povećava se sigurnost podataka, što je od ključne važnosti za *B2B* klijente. Ova funkcionalnost predstavlja logičnu ekstenziju *Capcade*-a, čineći ga kompletnijim alatom za upravljanje poslovnim procesima.

5. PREDLOG NAČINA RAZVOJA

Nakon odluke o razvoju nove funkcionalnosti, potrebno je definisati minimalno održiv proizvod (*MVP*) koji donosi vrednost. Trenutno *Capcade* kalendar služi samo za zahteve za fajlove i podsetnike. Unapređenje bi omogućilo korisnicima da planiraju, zakazuju i upravljaju svim poslovnim sastancima i događajima na jednom mestu. Različiti prikazi - nedeljni, mesečni i godišnji - omogućiće prilagođavanje pregleda prema potrebama korisnika. Kalendar će nuditi detaljan prikaz svakog sastanka, sa mogućnostima filtriranja prema timu, projektu ili tipu sastanka, što će poboljšati organizaciju vremena i obaveza.
MVP verzija će uključivati osnovne atribute za sastanke: ime, opis, lokaciju, boju, vreme i trajanje, te goste.

Korisnici će moći da kreiraju, pregledaju, odlažu i brišu sastanke, uz obaveštenja za sve promene i mogućnost praćenja statusa odgovora gostiju. Takođe, korisnici će moći da označe bitne događaje i primaju prilagođena obaveštenja.

U procesu razvoja proizvoda, ideje prolaze kroz faze procene i selekcije. Svakoj ideji se dodeljuje status kao što su „predložena ideja“, „potrebno više informacija“ ili „odložene ideje“. Ideje se ocenjuju na osnovu njihovog uticaja na proizvod i korisnike, kao i truda potrebnog za implementaciju. Ključni faktor je poređenje „uticaja na proizvod“ i „truda za implementaciju“, pri čemu ideje s visokim uticajem i manjim trudom dobijaju prioritet. Procenjuje se važnost funkcionalnosti za trenutne klijente i strateški značaj. Na kraju, ideje prolaze kroz glasanje prodajnih i produkt timova kako bi se odlučilo da li će se ideja odmah realizovati ili odložiti za kasnije. Ovaj proces omogućava transparentnost i saradnju unutar kompanije. U prikazanoj tabeli su prikazani uticaj i trud za ideje navedene u radi, kao i kratko objašnjenje svake od njih, pri čemu se uticaj meri na osnovu koristi za klijente, a trud procenjuje kao da jedan razvojni tim radi na funkcionalnosti.

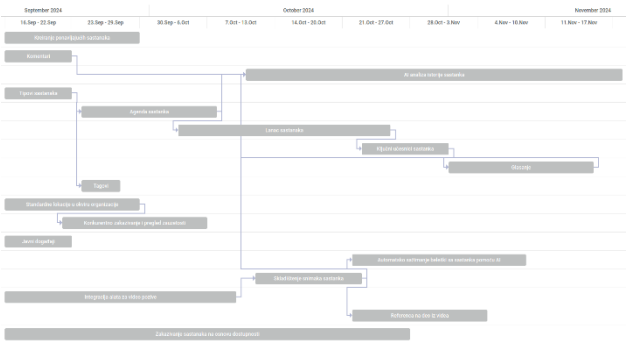
Kratak opis ideja iz tabele:

- Kreiranje ponavljajućih sastanaka – mogućnost zakazivanja više sastanaka odjednom na svakih nekoliko dana, nedelja, meseci ili određenim danima u nedelji;
- Agenda sastanka – definisanje tačaka sastanka sa okvirnim vremenom trajanja i referenciranim dokumentima;
- Standardne lokacije u okviru organizacije – konferencijske sale vezane za organizaciju;
- Konkurentno zakazivanje i pregled zauzetosti – zakazivanje sastanaka i događaja na osnovu dostupnosti lokacije;
- Tipovi sastanaka – šabloni za kreiranje sastanaka;
- Javni događaji – događaji u okviru organizacije kojima svi zaposleni treba da imaju pristup;
- Tagovi – specifične oznake za sastanke definisane na nivou organizacije;
- Zakazivanje sastanaka na osnovu dostupnosti – mogućnost nalaženja termina sastanka koji odgovara najvećem broju pozvanih učesnika;
- Skladištenje snimaka sastanaka – referenca sastanka na snimak u *dataroom*-u;
- Automatski sažetak beleški sa sastanka – mogućnost sažimanja sastanka pomoću veštačke inteligencije;
- Ključni učesnici sastanka – mogućnost označavanja bitnih učesnika bez kojih sastanak ne treba da se održi;
- Integracija sa alatima za video pozive – održavanje sastanaka preko *Capcade* aplikacije;
- Referenca na deo iz videa – mogućnost označavanja dela u videu gde je donešena odluka;
- Glasanje – mogućnost donošenja odluka pre/posle/u toku sastanka;
- Lanac sastanaka – ulančavanje sastanaka kako bi se lakše pratile istorije donošenja odluka i zavisnoti;
- Komentari – ostavljanje komentara na sastancima;
- Unapređenje sastanaka analizom istorije – praćenje komentara, glasanja, diskusija, dolaznosti, poštovanja agende i analiza informacija sa predlozima za unapređenje implementirana pomoću veštačke inteligencije.

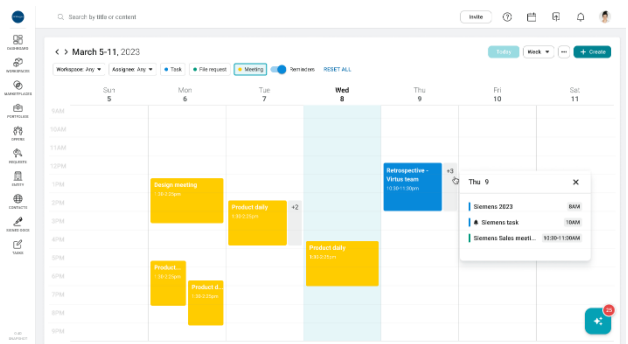
Funkcionalnost	Uticaj	Napor
Kreiranje ponavljajućih sastanaka	●●●●●	2 nedelje
Agenda sastanka	●●●●	2 nedelje
Standardne lokacije u okviru organizacije	●●	3 dana
Konkurentno zakazivanje i pregled zauzetosti	●●●	1 nedelja
Tipovi sastanaka	●●●	2 nedelje
Javni događaji	●	1 nedelja
Tagovi	●●	3 dana
Zakazivanje sastanaka na osnovu dostupnosti	●●●●●	2 meseca
Skladištenje snimaka sastanka	●●●●	3 dana
Automatski sažetak beleški sa sastanka	●●	3 dana
Ključni učesnici sastanka	●●●●	3 dana
Integracija sa alatima za video pozive	●●●●●	1 mesec
Referenca na deo iz videa	●●	1 mesec
Glasanje	●●●●●	2 nedelje
Lanac sastanaka	●●	2 nedelje
Komentari	●●●●	1 nedelja
Unapređenje sastanaka analizom istorije	●●●●●	2 meseca

Slika 2 – Procena uticaja i napora

Ono što je ključno kod procene napora i uticaja jeste identifikacija međusobne zavisnosti ideja. Mnoge od ideja mogu blokirati jedna drugu i to nikako ne treba zanemariti pri određenju prioriteta.



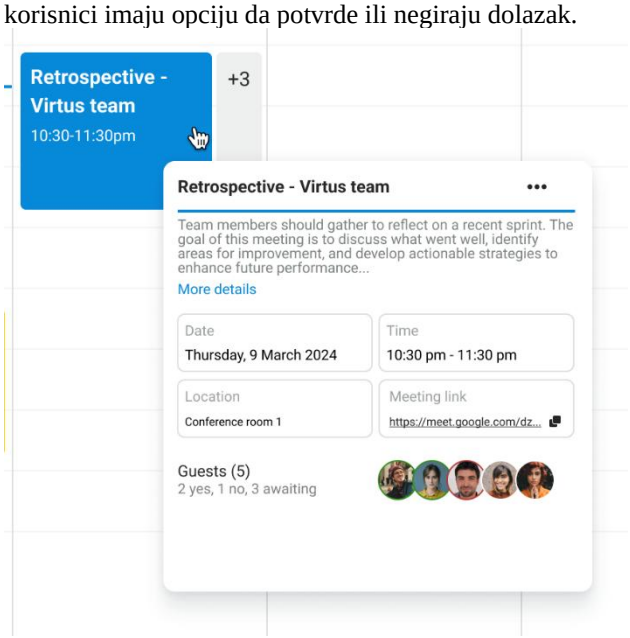
Slika 3 – Međusobna zavisnost ideja



Slika 4 – Nedeljni prikaz kalendara

Kreiranje MVP verzije funkcionalnosti podrazumeva kreiranje korisničkih priča. Najpre je potrebno proširiti kalendar *Capcade* aplikacije nedeljnim prikazom kao na

slici. Zatim, omogućavanje kreiranja novog događaja kroz dijalog od 3 koraka (osnovne informacije o sastanku, odabir vremena i lokacije, pozivanje gostiju). Nakon kreiranja, događaj se mora prikazati na kalendaru.
Klikom na kreirani događaj, korisnik treba da vidi osnovne informacije o događaju. Takođe, ukoliko je on kreirao događaj, može ga izmeniti ili otkazati. Pozvani korisnici imaju opciju da potvrde ili negiraju dolazak.



Slika 5 – Detaljni prikaz događaja na kalendaru

Tokom i nakon implementacije funkcionalnosti, potrebno je kontinuirano sakupljati povratne informacije od korisnika. Ono što je ključno jeste promocija nove funkcionalnosti na različite načine. Već postojeći korisnici treba da budu svesni novih opcija na aplikaciji kako bi mogli da ih isprobaju i potencijalno počnu da ih koriste. Za potencijalne klijente naše aplikacije, nova funkcionalnost može biti dobar *selling point* i zbog toga treba da se reklamira na mrežama i istakne na sajtu kompanije.
Kroz definisanje relevantnih metrika, omogućavamo lako praćenje uspeha nove funkcionalnosti.

6. ZAKLJUČAK

Ovaj rad se bavi proširenjem *Capcade* softvera sa funkcionalnosti za upravljanje i analizu online sastanaka. Analizirana su tržišna rešenja poput *Google Calendar*-a, *Microsoft Teams*-a i *Calendly*-a, i procenjena je kompatibilnost nove funkcionalnosti sa postojećim rešenjem. Dat je predlog implementacije prve verzije rešenja, počevši od osnovnih funkcija poput zakazivanja sastanaka i obaveštavanja korisnika, uz ideje i planove za napredne mogućnosti poput glasanja, ključnih učesnika sastanaka, kao i analize istorije sastanaka pomoću veštačke inteligencije. Naglašena je važnost konstantne saradnje sa prodajnim timovima i klijentima, i praćenje korisničkog *feedback*-a za dalji razvoj. Budući radovi iz ove oblasti mogu uključivati konkretnu implementaciju navedene funkcionalnosti, kao i dalje istraživanje novih metodologija razvoja softvera.

7. LITERATURA

[1] Capcade: https://www.capcade.com/
[2] Phaal, R., Farrukh, C. J. P., & Probert, D. R. (2004). "Technology Roadmapping — A Planning Framework for Evolution and Revolution." Technological Forecasting and Social Change, 71(1), 5-26.
[3] Cooper, R. G. (2001). Winning at New Products: Accelerating the Process from Idea to Launch.
[4] McGrath, M. E. (2001). Product Strategy for High Technology Companies. McGraw-Hill.
[5] Ries, E. (2011). The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business.
[6] Gartner, Inc. (2019). Market Guide for Competitive Intelligence Platforms.
[7] Ulwick, A. W. (2005). What Customers Want: Using Outcome-Driven Innovation to Create Breakthrough Products and Services. McGraw-Hill.
[8] Porter, M. E. (1980). Competitive Strategy: Techniques for Analyzing Industries and Competitors.
[9] Patton, J. (2014). User Story Mapping: Discover the Whole Story, Build the Right Product.
[10] Cohn, M. (2005). Agile Estimating and Planning.
[11] Doerr, J. (2018). Measure What Matters: How Google, Bono, and the Gates Foundation Rock the World with OKRs.
[12] Microsoft Outlook: https://support.microsoft.com/en-us/office/schedule-meetings-in-outlook-07d8c542-dcf4-4a9b-9439-6b7e07052e07
[13] Google Calendar: https://support.google.com/calendar/answer/2465776
[14] Apple Calendar: https://support.apple.com/en-us/HT201239
[15] Microsoft Teams: https://support.microsoft.com/en-us/teams
[16] Calendly: https://calendly.com/
[17] Zoom API: https://marketplace.zoom.us/docs/api-reference/introduction

Kratka biografija:

Anđela Mišković rođena je je u Valjevu 2000. godine. Diplomirala je na Fakultetu tehničkih nauka u Novom Sadu, na smeru Softversko inženjerstvo i informacione tehnologije 2023. godine sa prosečnom ocenom 9.59. Master rad na Fakultetu tehničkih nauka u Novom Sadu iz oblasti Elektrotehničko i računarsko inženjerstvo – Razvoj funkcionalnosti upravljanja sastancima odbora u *Capcade* aplikaciji, odbranila je 2024.godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 681.324.06
DOI: https://doi.org/10.24867/30BE50Bratic

MODELOVANJE MEMRISTORA KORIŠĆENJEM TEORIJE HISTEREZISA

MODELING OF MEMRISTORS USING HYSTERESIS THEORY

Stojanka Bratić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je predstavljena veza između postojanja klasične histerezisne petlje u fluks-naelektrisanje ravni, i uštinute histerezisne petlje u naponsko-strujnoj ravni memristora.*

Ključne reči: *memristor, histerezis, Takačev model*

Abstract – *This paper presents a relationship between the existence of classical hysterezis loop in the flux-charge plane, and pinched hysteresis loop in the voltage-current plane of memristor.*

Keywords: *memristor, hysteresis, Takacs model*

1. UVOD

U ovom radu je prikazana povezanost između postojanja klasične histerezisne petlje (eng. *Hysteresis Loop*, kratko HL) u fluks-naelektrisanje ravni memristora, i uštinute histerezisne petlje (eng. *Pinched Hysteresis Loop*, kratko PHL) u naponsko-strujnoj ravni memristora. PHL u *u-i* ravni predstavlja jedan od „otisaka prstiju“ (eng. *fingerprints*) memristora, kada je njegova pobuda neparna periodična funkcija čija je srednja vrednost jednaka nuli.

Histerezis predstavlja složenu nelinearnost (eng. *compound nonlinearity*) koja se pojavljuje u mnogim inženjerskim domenima, ili bistabilnost u radu različitih uređaja od inženjerskog interesa. U literaturi se mogu sresti HL-ovi različitih oblika: „klasičan“ (dve pomerene sigmoidalne funkcije koje se seku), kružni, elipsoidni, kvadratni, pravougaoni, u obliku romba i romboida.

U ovom radu je razmatran klasičan histerezis, a za njegovo modelovanje je korišćen Takačev model histerezisa. Kao specijalni slučajevi ovog histerezisa se za odgovarajući izbor parametara mogu dobiti i histerezisi u obliku kvadrata, pravougaonika, romba i romboida.

Teorija histerezisa je dobro utemeljena i razvijena oblast. Sa druge strane, uštinuta histerezisna petlja, mora pre svega biti histerezisna petlja, kako to njeno ime sugeriše. Začuđujuće, veze između HL-a i PHL-a nema u dostupnoj literaturi, prema našim najboljim saznanjima. Glavni rezultat rada je uspostavljanje veze između analitičkih izraza za HL modelovan Takačevim modelom, i odgovarajućeg PHL-a. Rezultat je matematički izveden u kontekstu modelovanja memristora.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Staniša Dautović, vanr. prof.

2. MEMRISTOR – POTREBNE DEFINICIJE I POJMOVI

Memristori su definisani konstitutivnom relacijom $F(q,\varphi)$ u naelektrisanje-fluks ravni. Konstitutivna relacija idealnog memristora je definisana u $\varphi - q$ ravni nekom nelinearnom funkcijom,

$$F(\varphi, q) = 0, \tag{1}$$

gde su

$$\varphi(t) = \varphi_0 + \int_{t_0}^{t} v(t)\,dt, \qquad q(t) = q_0 + \int_{t_0}^{t} i(t)\,dt. \tag{2}$$

U prethodnom izrazu, φ_0 i q_0 su početne vrednosti fluksa i naelektrisanja, respektivno, u početnom trenutku $t = t_0$.
Za memristor se kaže da je kontrolisan naelektrisanjem ako je njegova konstitutivna relacija iskazana u obliku $\varphi=f(q)$ [1]. Dualno, za memristor se kaže da je kontrolisan fluksom ako je njegova konstitutivna relacija iskazana u obliku $q=f(\varphi)$.

Relacija koja povezuje napon i struju memristora glasi

$$u=Mi, \tag{3}$$

gde M označava memristansu memristora, $M=d\varphi/dq>0$. Često se umesto simbola M u radovima sreće simbol R (otpornost zavisna od promenjive stanja), budući da je fizička jedinica za memristansu takođe om [Ω]. Ekvivalentno, veza između struje i napona može se iskazati kao

$$i=Wu, \tag{4}$$

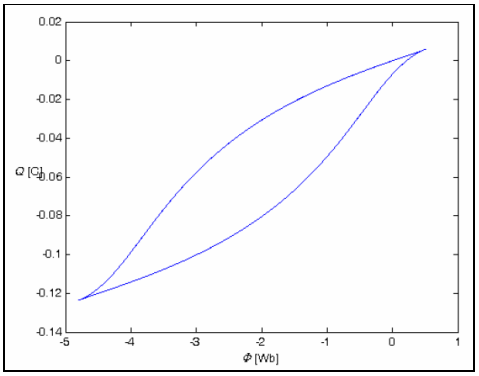
gde W označava memduktansu memristora, $W=dq/d\varphi=1/M>0$. Često se umesto simbola W u radovima sreće simbol G (provodnost zavisna od promenjive stanja), čija je fizička jedinica Simens, u oznaci [S]. Iz relacija (3) i (4) je očigledno da napon i struja na memristoru moraju biti istovremeno jednaki nuli.
Iz praktičnih razloga potrebno je da model memristora bude kontrolisan ili strujom ili naponom. Za memristor priključen na strujnu pobudu kaže se da je kontrolisan strujom, dok se za memristor priključen na naponsku pobudu kaže da je kontrolisan naponom. U savremenoj literaturi, memristori se klasifikuju u četiri grupe: idealni, idealni generički, generički i prošireni [1].

3. PRIMERI HISTEREZISA U FLUKS-NAELEKTRISANJE RAVNI MEMRISTORA

Nelinearnost (1) ne mora biti jednoznačna funkcija, već može biti i složena nelinearnost, tj. histerezis. Histerezisi kod memristora su u dostupnoj literaturi uočeni u različitim ravnima od interesa: *φ-q*, *R-i*, *R-u*, *G-i*, *G-u*. Ovde će kao motivišući primeri biti navedeni samo neki

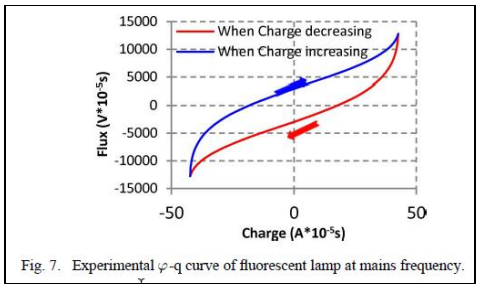
među njima, tj. oni gde se klasičan histerezis javlja u φ-q ravni.

Primer 1 iz [2] i [3]: Termistor. Tangencijalna PHL u i-u ravni, samopresecajuća „leptir“ PHL u G-u ravni, klasični HL u q-φ ravni, koji je prikazan na Slici 1.



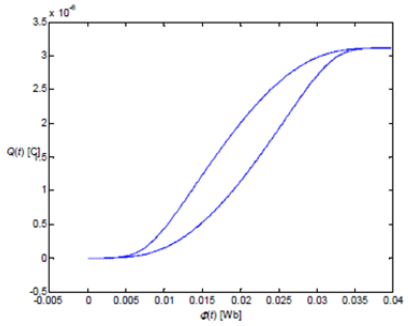
Slika 1. HL u q-φ ravni termistora.

Primer 2 iz [4]: Gas discharge lamp, inverzna HL u φ-q ravni, prikazana na Slici 2.



Slika 2. Inverzna HL u φ-q ravni (preuzeto iz [4]).

Primer 3 iz [5]: Generalizovani memristor, PHL u obliku “vesla” u i-u ravni. Odgovarajući klasičan HL u q-φ ravni nije prikazan u originalnom članku [5], ali je u ovom radu softverski modelovan i prikazan na Slici 3.



Slika 3. HL u φ-q ravni memristivnog diodnog mosta kaskadno povezanog sa RLC filtrom.

4. TAKAČEV MODEL HISTEREZISA

Takačev model [6] je baziran na $T(x)$ funkciji, koja je definisana kao linearna kombinacija tangens hiperbolične funkcije i linearne funkcije:

$$T(x) = B_0 \tanh C_0 x + A_0 x. \tag{5}$$

Da bi se opisale grane simetrične HL, hiperbolični deo $T(x)$ funkcije je transliran horizontalno (ili desno ili levo za a_0) i vertikalno (ili gore ili dole za b_1). Uzlazna grana $f_+^{A_0}$ za rastuće vrednosti x je opisana sa

$$f_+^{A_0}(x) = B_0 \tanh\big(C_0(x-a_0)\big) + A_0 x + b_1, \tag{6}$$

dok je silazna grana $f_-^{A_0}$ za opadajuće vrednosti x opisana sa

$$f_-^{A_0}(x) = B_0 \tanh\big(C_0(x+a_0)\big) + A_0 x - b_1. \tag{7}$$

U kontekstu magnetskih materijala, parametar A_0 u izrazima (6) i (7) se odnosi na reverzibilnu magnetizaciju, i može biti iskorišćen da “iskosi” HL. U razmatranjima koja slede, koeficijent A_0 je postavljen na nulu. Za $A_0 = 0$, izrazi (6) i (7) prelaze u

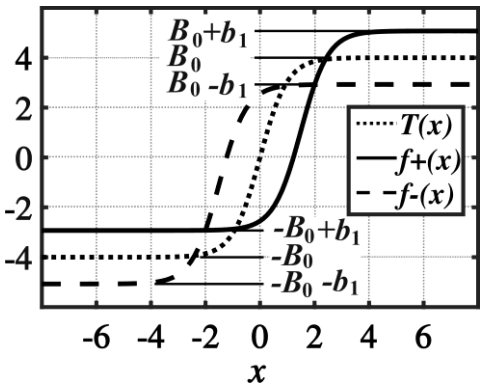
$$f_+(x) = B_0 \tanh\big(C_0(x-a_0)\big) + b_1, \tag{8}$$

$$f_-(x) = B_0 \tanh\big(C_0(x+a_0)\big) - b_1. \tag{9}$$

Kod zatvorenih HL, grane imaju dve zajedničke tačke na vrhovima (“špicevima”) histerezisa. Za periodičnu pobudu $x = x(t)$, sa nultom srednjom vrednošću i $-X_m \le x \le X_m$, vrhovi se pojavljuju za $x = \pm X_m$. Simetrična HL je zatvorena kada je $f_+(X_m) = f_-(X_m)$. Ova relacija se može rešiti po b_1:

$$b_1 = \frac{B_0}{2}\Big[\tanh\big(C_0(X_m + a_0)\big) - \tanh\big(C_0(X_m - a_0)\big)\Big]. \tag{10}$$

Primer klasične zatvorene HL je prikazan na Slici 4.



Slika 4. Primeri funkcija $T(x)$, $f_+(x)$ i $f_-(x)$ za vrednost parametara $A_0 = 0$, $B_0 = 4$, $C_0 = 1$, $a_0 = 1.5$, $X_m = 2$.

5. VEZA IZMEĐU HL-a I PHL-a

U ovom delu će biti uspostavljena veza između postojanja HL-a u φ-q ravni memristora, i pojave PHL-a u u-i ravni istog memristora. Izvođenje će biti dato na primeru strujom kontrolisanog memristora. Kako kod strujom kontrolisanog memristora pobuda mora biti nezavisni vremenski promenjiv strujni generator, čiji je talasni oblik periodična funkcija koja je neparna i sa srednjom vrednošću jednakoj nuli, izvođenje je sprovedeno za sinusni talasni oblik generatora, $i_g(t) = I_m \sin(\omega t)$. U tom slučaju, na osnovu (2) naelektrisanje je jednako

$$q(t) = q_0 - \frac{I_m}{\omega}\big(1 - \cos \omega t\big), \tag{11}$$

gde je q_0 početni uslov. U izvođenjima će nam biti potrebni prvi i drugi izvod naelektrisanja, koji su tada

$$i(t)=\frac{dq(t)}{dt}=I_m\sin\omega t\ , \tag{12}$$

$$\frac{d^2q(t)}{dt^2}=I_m\cdot\omega\cdot\cos\omega t\ . \tag{13}$$

Histerezisnu petlju, $\varphi(q(t))$, možemo opisati pomoću Takačevog modela histerezisa i Hevisajdove funkcije $h(\cdot)$,

$$\varphi(q(t))=f_+(q(t))h\left(\frac{dq(t)}{dt}\right)+f_-(q(t))h\left(-\frac{dq(t)}{dt}\right). \tag{14}$$

U slučaju kada je HL u φ-q ravni, i kada je q dato sa (11), parametar b_1 (10) se može izračunati iz preseka dve grane histerezisa, u dve tačke (dva "špica histerezisa") u kojima je $f_+(q(t))=f_-(q(t))$. U desnoj gornjoj presečnoj tački za b_1 dobijamo

$$b_1=\frac{B_0}{2}\left(\tanh\left(C_0\left(q_0+a_0\right)\right)-\tanh\left(C_0\left(q_0-a_0\right)\right)\right), \tag{15}$$

a u donjoj levoj presečnoj tački

$$b_1=\frac{B_0}{2}\left(\tanh\left(C_0\left(q_0-\frac{2I_m}{\omega}+a_0\right)\right)-\tanh\left(C_0\left(q_0-\frac{2I_m}{\omega}-a_0\right)\right)\right), \tag{16}$$

odakle sledi da početni uslov mora biti $q_0=I_m/\omega$ da bi histerezisna petlja bila simetrična i zatvorena.

Izvod po vremenu jednačine (14) daje

$$\begin{aligned}u(t)=\frac{d\varphi(q(t))}{dt}=&\frac{d}{dq}f_+(q(t))\frac{d}{dt}q(t)\cdot h\left(\frac{d}{dt}q(t)\right)+\\&+f_+(q(t))\cdot\delta\left(\frac{d}{dt}q(t)\right)\frac{d^2}{dt^2}q(t)+\\&+\frac{d}{dq}f_-(q(t))\frac{d}{dt}q(t)\cdot h\left(-\frac{d}{dt}q(t)\right)-\\&-f_-(q(t))\cdot\delta\left(-\frac{d}{dt}q(t)\right)\frac{d^2}{dt^2}q(t)\ ,\end{aligned} \tag{17}$$

gde su $h(\cdot)$ i $\delta(\cdot)$ oznake za Hevisajdovu i Dirakovu funkciju, respektivno. Iz prethodnog izraza, izdvojimo sabirke koji sadrže Dirakove impulse,

$$\begin{aligned}D=&f_+(q(t))\cdot\delta\left(\frac{d}{dt}q(t)\right)\frac{d^2}{dt^2}q(t)-\\&f_-(q(t))\cdot\delta\left(-\frac{d}{dt}q(t)\right)\frac{d^2}{dt^2}q(t)\ .\end{aligned} \tag{18}$$

U onome što sledi, pokazaćemo da je izraz (18) jednak nuli. U izrazu (18) se javljaju Dirakovi impulsi složene funkcije, koji se izračunavaju na sledeći način

$$\delta(f(t))=\sum_{t_k}\frac{1}{\left|\frac{d}{dt}f(t)\right|_{t=t_k}}\cdot\delta(t-t_k), \tag{19}$$

gde su t_k nule funkcije $f(t)$, tj. $f(t_k)=0$.

Korišćenjem osobine parnosti Dirakovog impulsa, $\delta(x)=\delta(-x)$, i osobine (19), za $f(t)=q(t)=I_m\sin(\omega t)$ sledi

$$\begin{aligned}\delta\left(\frac{d}{dt}q(t)\right)=\delta\left(-\frac{d}{dt}q(t)\right)=\delta\left(I_m\sin\omega t\right)=\\\sum_{t_k}\frac{1}{\left|\frac{d}{dt}I_m\sin\omega t\right|_{t=t_k}}\cdot\delta\left(t-t_k\right).\end{aligned} \tag{20}$$

U prethodnom izrazu (20), trenuci t_k predstavljaju nule funkcije $I_m\sin\omega t$, kojih ima beskonačno mnogo,

$$I_m\sin\omega t=0\rightarrow t_k=0,\pm\frac{\pi}{\omega},\pm\frac{2\pi}{\omega},\ldots.=\frac{k\pi}{\omega};\ k\in\mathbb{Z}\ . \tag{21}$$

Uvrštavajući (21) u (20), (20) dobija oblik

$$\delta\left(\frac{dq(t)}{dt}\right)=\sum_{t_k}\frac{1}{\left|I_m\omega\cos\omega t\right|_{t=t_k}}\cdot\delta\left(t-t_k\right)=\frac{1}{I_m\omega}\sum_{k=-\infty}^{\infty}\delta\left(t-\frac{k\pi}{\omega}\right). \tag{22}$$

Koristeći (22) i (13), izraz (18) se može predstaviti

$$\begin{aligned}D=&\delta\left(\frac{dq(t)}{dt}\right)\frac{d^2q(t)}{dt^2}\left(f_+(q(t))-f_-(q(t))\right)=\\&\frac{I_m\omega\cos\omega t}{I_m\omega}\left(\sum_{k=-\infty}^{\infty}\delta\left(t-\frac{k\pi}{\omega}\right)\right)\left(f_+(q(t))-f_-(q(t))\right).\end{aligned} \tag{23}$$

U izvođenju nam je potrebna osobina odabiranja Dirakovog impulsa,

$$f(t)\cdot\delta(t-T)=f(T)\cdot\delta(t-T) \tag{24}$$

u tačkama $t_i=T$=const. Označimo razliku pozitivne i negativne grane histerezisa u (23) sa $f(q(t))$,

$$f(q(t))=f_+(q(t))-f_-(q(t)). \tag{25}$$

Koristeći (24) i (25), jednačina (23) postaje

$$D=\left(\sum_{k=-\infty}^{\infty}\delta\left(t-\frac{k\pi}{\omega}\right)\right)\left(\cos\omega t\cdot f(q(t))\right)\Big|_{t=t_k}\ . \tag{26}$$

Razlika pozitivne i negativne grane histerezisa (25) u tačkama odabiranja t_k jednaka je

$$\begin{aligned}f(q(t_k))=&\left(f_+(q(t_k))-f_-(q(t_k))\right)=\\&B_0\tanh\left(C_0\left(q(t_k)-a_0\right)\right)-\\&B_0\tanh\left(C_0\left(q(t_k)+a_0\right)\right)+2b_1\end{aligned}\ . \tag{27}$$

Rastavimo izraz (27) u dva slučaja:

<u>Slučaj 1</u>, kada je k parno tj. $k=0,\pm\frac{2\pi}{\omega},\pm\frac{4\pi}{\omega},\ldots,$ $k=2n,\ n\in\mathbb{Z}$. U ovim tačkama vrednost $\cos\omega t_{2n}$ je jednaka jedinici, odnosno, $q(t_{2n})=q_0-\frac{I_m}{\omega}(1-1)=q_0$ sledi da je izraz (27)

$$\begin{aligned}f(q(t_{2n}))=&B_0\tanh\left(C_0\left(q_0-a_0\right)\right)-B_0\tanh\left(C_0\left(q_0+a_0\right)\right)+\\&B_0\left(\tanh\left(C_0\left(q_0+a_0\right)\right)-\tanh\left(C_0\left(q_0-a_0\right)\right)\right)=0.\end{aligned} \tag{28}$$

<u>Slučaj 2</u>, kada je k neparno tj. $k=\pm\frac{\pi}{\omega},\pm\frac{3\pi}{\omega},\ldots,$ $k=2n+1,\ n\in\mathbb{Z}$. U ovim tačkama vrednost $\cos\omega t_{2n+1}$ je

minus jedan, a naelektrisanje je $q(t_{2n+1}) = q_0 - 2\frac{I_m}{\omega} = -q_0$.

Korišćenjem osobine neparnosti tanh(·) funkcije, tanh(–x)= –tanh(x), izraz (27) i u ovom slučaju biva jednak nuli,

$$\begin{aligned} f(q(t_{2n+1})) = {} & B_0 \tanh\left(C_0\left(-q_0 - a_0\right)\right) - \\ & B_0 \tanh\left(C_0\left(-q_0 + a_0\right)\right) + \\ & B_0 \tanh\left(C_0\left(q_0 + a_0\right)\right) - \\ & B_0 \tanh\left(C_0\left(q_0 - a_0\right)\right) = 0 \end{aligned} \quad (29)$$

Sumirajući dva slučaja (28) i (29), izraz (27) postaje jednak nuli, a time i izraz (18) postaje takođe jednak nuli. Budući da je napon na memristoru jednak $u(t)=d\varphi(t)/dt$, a struja $i(t)=dq(t)/dt$, izraz (17) se može napisati u obliku

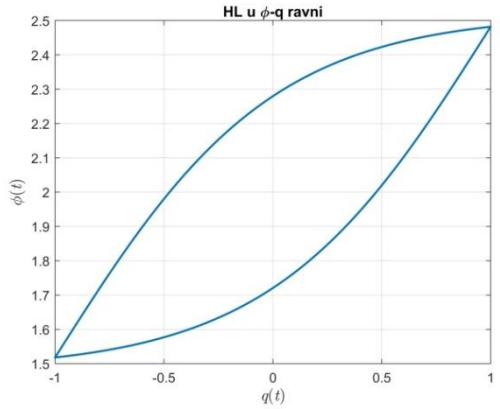
$$u(t) = \left(\frac{d\, f_+\left(q(t)\right)}{dq} h\left(i(t)\right) + \frac{d\, f_-\left(q(t)\right)}{dq} h\left(-i(t)\right)\right) i(t), \quad (30)$$

odnosno

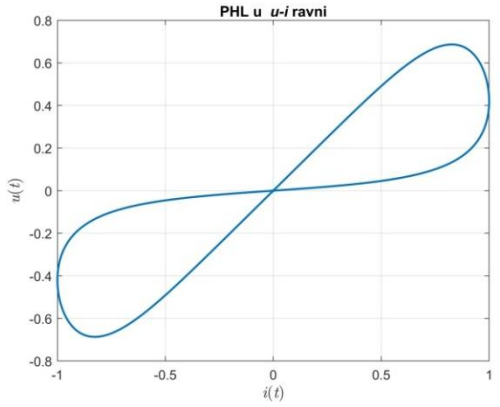
$$u(t) = M(q,i) \cdot i(t), \quad (31)$$

gde je memristansa M(q,i) memristora koji ima HL u φ-q ravni jednaka prvom činiocu u zagradi izraza (30). Na osnovu izvršenih analiza možemo zaključiti da izvod po vremenu HL-a u φ-q ravni memristora, rezultuje pojavom PHL-a u u-i ravni memristora.

Radi ilustracije dobijenih rezultata, polazna parametarska jednačina (14) za HL u φ-q ravni je prikazana na Slici 5, a rezultujuća parametarska jednačina (30) za PHL u u-i ravni memristora je prikazana na Slici 6.



Slika 5. HL memristora u φ-q ravni. Vrednosti parametara I_m=1, ω=1, q_0=I_m/ω, B_0=1, C_0=1, a_0=1, $q(t)$=q_0- I_m/ω(1-cosωt).



Slika 6. PHL u u-i ravni memristora. Vrednosti parametara I_m=1, ω=1, q_0=I_m/ω, B_0=1, C_0=1, a_0=1, $q(t)$=q_0- I_m/ω(1-cosωt).

Kao prateći rezultat sprovedene analize, kao nov rezultat ovog rada, iz parametarske jednačine za HL

$$\begin{aligned} f_{HL}\left(x(t)\right) = {} & f_{HL+}\left(x(t)\right) \cdot h\left(\frac{d}{dt}x(t)\right) + \\ & f_{HL-}\left(x(t)\right) h\left(-\frac{d}{dt}x(t)\right) \end{aligned}, \quad (32)$$

izvedena je parametarska jednačina za PHL

$$\begin{aligned} f_{PHL}\left(x(t)\right) = {} & \frac{d}{dt} f_{HL+}\left(x(t)\right) \cdot h\left(\frac{d}{dt}x(t)\right) + \\ & \frac{d}{dt} f_{HL-}\left(x(t)\right) h\left(-\frac{d}{dt}x(t)\right) \end{aligned} \quad (33)$$

kao glavni rezultat i doprinos ovog rada.

6. ZAKLJUČAK

U ovom radu je izvedena parametarska jednačina za uštinutu histerezisnu petlju, i uspostavljena njena veza sa klasičnom histerezisnom petljom modelovanom korišćenjem Takačevog modela. Rezultat je dobijen u kontekstu matematičkog modelovanja memristora koji ima složenu histerezisnu nelinearnost u fluks-naelektrisanje ravni.

7. LITERATURA

DOI: https://doi.org/10.24867/30BE50Bratic

[2] S. P. Adhikari, M. P. Sah, H. Kim and L. O. Chua, "Three Fingerprints of Memristor," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 60, no. 11, pp. 3008-3021, Nov. 2013, doi: 10.1109/TCSI.2013.2256171.

[3] L. Chua, "Memristor, Hodgkin–Huxley, and Edge of Chaos", Nanotechnology, Vol. 24, No 11, Sept. 2013, DOI 10.1088/0957-4484/24/38/383001

[4] D. Lin, S. Y. R. Hui and L. O. Chua, "Gas Discharge Lamps Are Volatile Memristors," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 61, no. 7, pp. 2066-2073, July 2014, doi: 10.1109/TCSI.2014.2304659.

DOI: https://doi.org/10.24867/30BE50Bratic

[6] Takacs, J. *Mathematics of Hysteretic Phenomena*. Weinheim (Germany): Wiley-VCH, 2006. ISBN: 9783527404018

Kratka biografija:

Stojanka Bratić rođena je u Trebinju 1998. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti elektrotehnike i računarstva - embeded sistemi i algoritmi odbranila je 2022. godine.

kontakt: braticstojanka@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

UDK: 621.38/.39
DOI: https://doi.org/10.24867/30BE51Pavlovic

СИСТЕМ ЗА ЕВИДЕНЦИЈУ ПРОЛАСКА ЉУДИ СА LoRaWAN Комуникацијом

LoRaWAN-BASED PEOPLE TRACKING SYSTEM

Косана Павловић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Kratak sadržaj – *Рад се бави хардверским и софтверским аспектом реализације система за евиденцију просласка људи кроз врата у једном или другом правцу. Биће објашњена хардверска структура, алгоритам програма уграђеног у систем, начин преношења информација до одговарајуће платформе, као и backend и frontend апликативног сервера до ког информације пристижу.*

Кључне речи: *Brojanje ljudi, LoRaWAN, IoT*

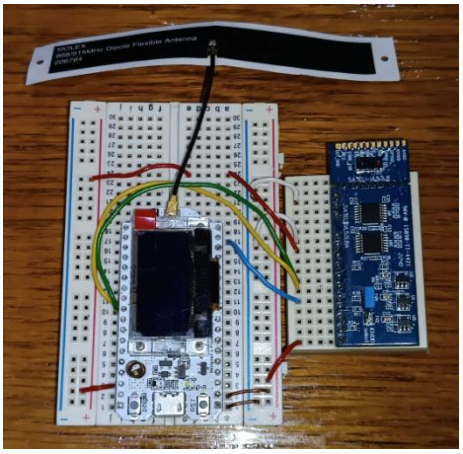
Abstract – *The paper addresses both the hardware and software aspects of a people tracking system. It will explain the program embedded in the system, the method of transferring information to the appropriate platform, as well as the backend and frontend components of the application server where the information is received.*

Keywords: *People counting, LoRaWAN, IoT*

1. УВОД

Савремени изазови у управљању просторијама захтевају паметна решења за праћење броја људи. Увођење IoT уређаја специјализованих за те сврхе постаје све неопходније како би се оптимизовало управљање у циљу побољшања утиска посетилаца и уштеде енергије. Тачна статистика о броју људи отвара могућност аутоматизације система попут грејања, климатизације или осветљења. За систем за евиденцију проласка људи са слике 1. Коришћена је WiFi LoRa 32 (V2) [1] IoT развојна плоча базирана на ESP32 чипу. Садржи LoRa модул због чега је погодно користити LoRaWAN технологију за слање података са уређаја на удаљени сервер. Плоча је повезана са *Time-of-Flight* (ToF) сензором VL53L8A [2] који мери удаљеност до објекта. Подаци са уређаја пристижу на *ThingPark Wireless* платформу компаније Actility и прослеђују даље на *third-party* сервер где је имплементиран *backend* и *frontend* сервис. Статистички приказ броја људи у одређеној просторији је табеларно и графички представљен на веб страници која се налази на URL адреси https://ues.edu.rs/pavlovic.e192/results.php. Обезбеђен је лак приступ подацима без компликованог приступа серверу/бази података и инсталацијом додатног софтвера.

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



Slika 1. *Систем за евиденцију проласка људи*

2. АЛГОРИТАМ ПРОГРАМА СИСТЕМА

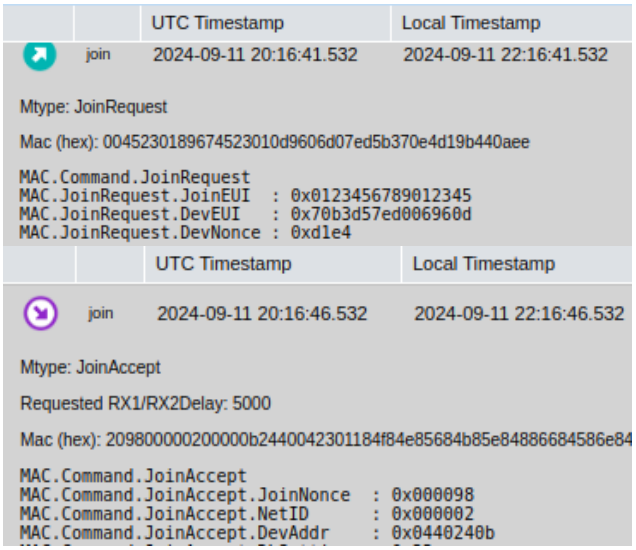
На почетку програма се сензор ресетује, иницијализује и конфигурише да мери удаљеност до објекта у 16 или 64 различитих области. Као резултат мерења добија се матрица удаљености од 16 или 64 елемената. Започиње се мерење удаљености и чека се да резултати мерења буду спремни. Тај тренутак се препознаје услед имплементирања *polling* методе или прекидне рутине. Резултати се обрађују тако што се матрица удаљености раздвоји на предњу и задњу видну зону сензора. За сваку зону се обрачуна просечна удаљеност и пореди са унапред дефинисаним прагом. Све док је просечна удаљеност у једној од зона мања од прага, особа је детектована и може се утврдити њен смер кретања. Ако је просечна удаљеност у обе зоне мања од прага, тада је детектовано присуство особе на средини видне зоне. Правац њеног кретања се може одредити тек кад особа пређе у једну од зона и он се одређује на основу распореда преласка видних зона сензора.

Алгоритам детектује пролазак особе само ако она прође кроз обе видне зоне. Ако особа одлучи да се врати одакле је дошла, то неће бити регистровано као прелазак. Детекцијом проласка особе се број особа повећава или смањује у зависности од правца кретања. Промењен број особа је потребно проследити на удаљени сервер помоћу LoRaWAN технологије [3]. Због карактеристике ове технологије да наредну поруку не може послати док не прими потврду претходне, нови број особа мора бити смештен у бафер. На тај начин је осигурано да свака промена броја људи ће бити послата. Из тог бафера ће

се ишчитавати подаци који ће се паковати у LoRaWAN поруке и слати до апликационог сервера.

3. LORAWAN ТЕХНОЛОГИЈА

Када се уређај укључи, покрене се OTAA активација уређаја ваздухом како би се он регистровао на LoRaWAN мрежу. Активација се реализује слањем захтева за придруживање мрежи са параметрима JoinEUI и DevEUI (слика 2). ThingPark Wireless платформа шаље назад одговор са додељеном динамичком адресом уређаја DevAddr и JoinNonce параметром на основу ког уређај креира безбедносне кључеве сесије. Уређај може слати поруке тек након обављене активације.



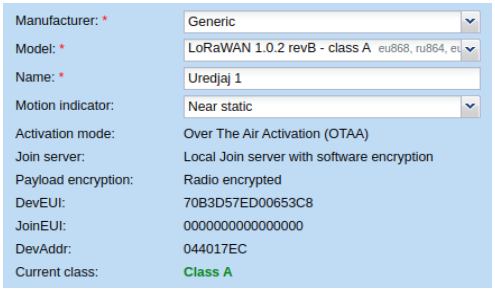
Слика 2. *Информације о захтеву за придруживање и прихватању придруживања*

Ако у баферу нема података спремних за слање, уређај иде у режим спавања како би уштедео енергију све док не стигне податак у бафер. Ако бафер није празан, чита се податак који је најраније смештен у њега и пакује у LoRaWAN поруку. Она се шаље радио таласима фреквенције из европског опсега (EU868) до свих суседних гејтвеја у домету. Нкон што уређај прими потврду о послатој поруци, шаље наредни податак из бафера ако он постоји. Гејтвеји прослеђују поруку *ThingPark Wireless* платформи која игра улогу мрежног сервера у архитектури LoRaWAN мреже.

4. *THINGPARK WIRELESS* PLATFORMA

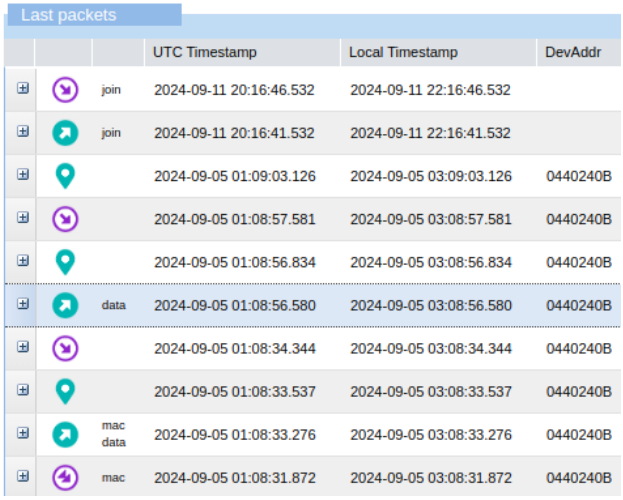
Почетна страна платформе садржи две опције: *Device Manager* и *Wireless Logger*. У *Device Manager*-у треба креирати уређај са OTAA активацијом и са вредностима параметара JoinEUI и DevEUI идентичним као у програму уређаја (слика 3).

За модел уређаја коришћена је опција LoRaWAN revB 1.0.2 - класа А. У том процесу се додељује динамичка адреса уређаја. Поред креирања уређаја, под опцијом *Application servers* на постојећем серверу је потребно пронаћи *Destination* поље у које се уноси адреса https://ues.edu.rs/pavlovic.e192/actility.php. Тиме се омогућује пренос порука пристиглих до платформе на *third-party* апликациони сервер на тој адреси.

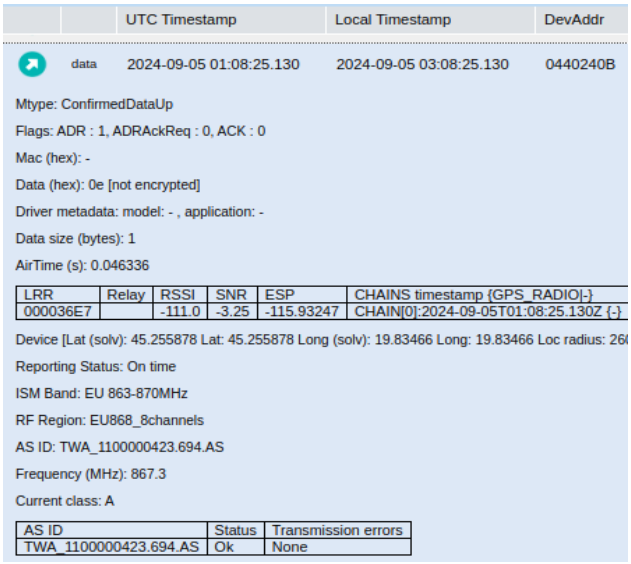


Слика 3. *Device Manager – креирање уређаја*

У *Wireless Logger*-у се налази преглед свих порука размењених између платформе и уређаја (слика 4). За сваку поруку коју платформа прими или пошаље се кликом на плус приказују детаљне информације попут типа поруке, локалног и глобалног времена, динамичке адресе уређаја, фактора ширења, корисног података, фреквенције радио сигнала којим је порука послата и слично (слика 5). Слика 2 приказује детаљније информације из *Wireless Logger*-а за поруку захтева за придруживање и поруку прихватања придруживања респективно.



Слика 4. *Приказ Wireless Logger-а*

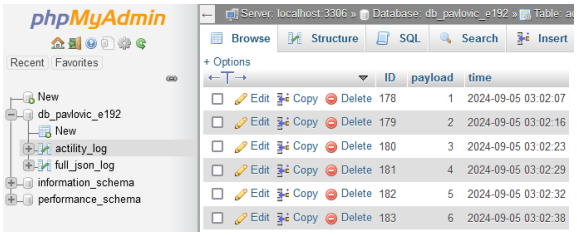


Слика 5. *Приказ информација о поруци пристиглој на платформу*

5. *BACKEND* И *FRONTEND* СЕРВИСИ

У позадини апликационог сервера је датотека https://ues.edu.rs/pavlovic.e192/actility.php у коју прослеђене поруке стижу путем HTTP POST методе. У овој php скрипти се из HTTP захтева издваја пристигла порука у JSON формату. Из ње је потребно извући користан податак и време слања поруке који се налазе кључем *'DevEUI_uplink'*. Под ознаком *'payload_hex'* је користан податак, а под ознаком *'Time'* је глобално време са временском зоном које је потребно конвертовати у локално време. Користан податак се налази у хексадецималном облику па га треба конвертовати у децималан облик.

На апликационом серверу се налази и база података *db_pavlovic_e192*. Припрема се и извршава SQL упит како би се издвојени подаци унели у табелу *actility_log* базе података. Табела има поље *payload* типа *integer* и поље *time* као низ од ограниченог броја *varchar* карактера (слика 6).

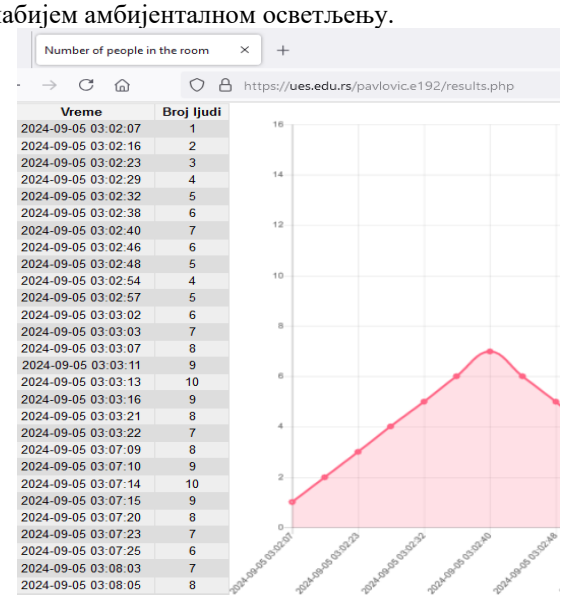


Slika 6. *Приказ података у табели actility_log*

На страни апликационог сервера постоји и датотека *https://ues.edu.rs/pavlovic.e192/results.php*. Ако се ова адреса убаци у веб претраживач, учитаће се веб страница са називом *Number of people in the room*. Представља табеларни и графички приказ евиденције проласка људи прочитане из базе података. У оквиру фајла *results.php* је <html> елемент у ком су табела и график. Редови табеле су попуњени редовима прочитаним из табеле *actility_log*. На хоризонталној оси графика су вредности колоне *time*, а на вертикалној оси су вредности колоне *payload* табеле *actility_log* базе. Део поменуте веб странице се налази на слици 7.

6. ЗАКЉУЧАК

Резултати тестирања система за евиденцију броја људи у просторији су показали да систем исправно функционише. Овакви ембедед системи могу бити корисни у великим канцеларијама, конференцијским салама, тржним центрима или јавним установама где се често беспотребно троши велика количина енергије, поготово ако нема људи у њима. Људска безбедност у случају непредвиђених ситуација може бити осигурана помоћу оваквих система. Ако дође до потребе за хитном евакуацијом, од критичне је важности знати број људи у просторији да би се убрзала реакција. Праћење да капацитет просторија не буде прекорачен и да се поштују безбедносне мере може бити олакшано. Ови системи су корисни и са маркетиншког аспекта јер менаџери могу прилагођавати понуде на основу статистике о броју посета радњама у одређеном периоду. Фиксан праг удаљености у програму ограничава примену система и потенцијално изазива погрешне резултате (нпр. немогућност детекције особа нижих од 150 центиметара). Једном програмиран систем се мора монтирати увек на исту висину како би резултати били валидни. Ако се систем постави тако да у његово видно поље улази статичан објекат до ког је удаљеност мања од прага, резултати ће бити погрешни. Од плавокосу особу ће се мање фотона одбити у односу на тамнокосу особу [4]. Пожељно је у програму уграђеном у систем имплементирати аутокалибрациону методу са великим бројем мерења [4]. Помоћу ње је потребно одредити праг који мора бити неколико центиметара мањи од најмањег измереног растојања. Такође, ова метода треба да садржи и проверу видног поља при постављању система тако да сензор покрива само област која је од значаја. Пожељно је почетно мерење извршити при слабијем амбијенталном осветљењу.



Slika 7. *Приказ дела веб странице на адреси https://ues.edu.rs/pavlovic.e192/results.php*

7. ЛИТЕРАТУРА

[1]https://heltec.org/project/wifi-lora-32v2/, Септембар 2024.

[2]https://www.st.com/en/imaging-and-photonics-solutions/vl53l8cx.html?icmp=tt38832_gl_lnkon_may2024#overview, Септембар 2024.

[3] https://lora-alliance.org/about-lorawan/, Септембар 2024.

[4] https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.st.com/resource/en/user_manual/um2600-counting-people-with-the-vl53l1x-longdistance-ranging-timeofflight-sensor-stmicroelectronics.pdf&ved=2ahUKEwjU69mb18OIAxW98rsIHWWEATsQFnoECBkQAQ&usg=AOvVaw1se7k-8fFTiVgxPi_CYDvm, Септембар 2024.

Кратка биографија:

Косана Павловић рођена је у Београду 2000. год. Дипломски рад на Факултету техничких наука из области Електротехнике и рачунарства – Примењена електроника одбранила је 2023.год.
контакт: kosanapavlovic66@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.738
DOI: https://doi.org/10.24867/30BE52Mijonic

ARHITEKTURA I PERFORMANSE EVM BLOK EKSPLORER-A

ARCHITECTURE AND PERFORMANCE OF EVM BLOCK EXPLORER

Nikola Mijonić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – *Ovaj rad pruža uvid u implementaciju EVM blok pretraživača sa fokusom na arhitekturu i performanse. Rad sadrži opis ključnih pojmova Ethereum platforme kako bi rad bio razumljiviji.*

Ključne reči: *blokčejn, pretraživač, arhitektura, performanse*

Abstract – *This paper provides an insight into the implementation of an EVM block explorer with a focus on architecture and performance. It includes a description of key concepts of the Ethereum platform to make the paper more comprehensible.*

Keywords: *blockchain, explorer, architecture, performance*

1. UVOD

Razumevanje tehnologije lanca blokova [1] kao i njene upotrebne vrednosti zahteva osnovno poznavanje istorije novca i monetarnih sistema. Nepoznavanje materije i finansijska nepismenost često dovode do netačnih zaključaka kao što su osporavanje finansijske vrednosti pojedinih kriptovaluta kao i same upotrebne vrednosti. Postojeći monetarni sistemi su veoma centralizovani što u svojoj osnovi nije dobro jer se javlja problem centralizacije moći i neadekvatnog upravljanja novcem u vidu nekontrolisanog štampanje bez ikakvog pokrića.

Osnovna ideja *Ethereum* [2] mreže je da pruži infrastrukturu za razvoj decentralizovanih aplikacija. Problemi vezani za centralizaciju monetarnih sistema prisutni su i u domenu centralizovanih aplikacija. Cenzure na društvenim mrežama, sprečavanje informisanje u vidu centralizovanih medija, olakšani hakerski napadi usled postojanja centralizovane tačke ispada samo su neki od problema koji bi trebalo da budu rešeni decentralizovanim aplikacijama.

Podaci koji su pohranjeni na *Ethereum* mrežu ne mogu se menjati što ujedno znači da ne postoji mogućnost malverzacije prethodno sačuvanim informacijama. Detaljan opis *Ethereum* mreže može se pronaći u okviru ovog rada. Postajanje podataka nema preveliku vrednost ukoliko ne postoji jednostavan način da se ti podaci pregledaju odnosno čitaju. Upravo to je razlog postojanja blokčejn pretrazivača.

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

Postojanje mogućnosti da se u svakom trenutko može videti svaka tanskacija koja se dogodila na *Ethereum* mreži koja je po svojoj prirodu javna je nešto što je za očekivati. Interesovanja za analizu velike količine podataka je sve veće zbog vrednosti koju te informacije donose. Na istom principu postoji potreba da podaci prisutni u *Ethereum* mreži budu pristupačni.

Kao što se može i pretpostaviti usled mogućnosti monetizacije pretraživača blokčejn mreže nije jednostavno steći uvid u arhitekturu i performanse jednog pretraživača. Motiv za ovaj rad je pružanje uvida u stavke o kojima treba voditi računa pri implementaciji pretraživača EVM mreže kao i potencijalnim optimizacijama kada su performanse u pitanju. U okviru ovog rada može se pronaći detaljan opis postupka implementacije blokčejn pretraživača koji je baziran na *Ethereum* mreži, odnosno podatke dobavlja sa pomenute mreže.

1.1. Primena lanca blokova

Transfer novca između država predstavlja nešto sa čime se mnogi susretnu u nekom trenutku života. Troškovi koji nastaju u jednom takvom procesu ne idu u korist onoga ko šalje ili prima novac. Kada na sve to dodamo vreme potrebno da se transakcija procesuira i procenat koji razni entiteti u tom procesu uzimaju može se zaključiti da tu postoji dosta prostora za promene. Prenos novca upotrebom blokčejn tehnologije može biti izuzetno jeftin i uklanja se problem postojanja graničnih podela na države.

Glasanje o donošenju važnih odluka vrlo često je podložno manipulacijama i nekorektnostima. Ovo je nešto što vrlo jednostavno može da se reši upotrebom tehnologije lanca blokova. Pojedinac može da se identifikuje svojom javno dostupnom adresom i ostavi glas, a da je i dalje pseudo anoniman.

Igrice su u velikom broju slučajeva sastavni deo jednog perioda života muške populacije. Neretko se dešava da se enormne količine vremena utroše na aktivnosti ovog tipa što kod većine u zrelijim dobima uzrokuje osećaj griže savesti usled izgubljenog vremena. Nakon prestanka igranja igrice u većini slučajeva ne postoji nikakva mogućnost ostvarivanja monetarnih dobara. Lanac blokova omogućava čuvanje raznoraznih kolekcija koje se nakon prestanka igranja pojedinih igrica mogu prodati i na taj način se može prihodovati.

2. OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA

2.1. Tehnologije

C# [3] - predstavlja objektno orijentisani programski jezik razvijen od stane Majkrosofta (eng. *Microsoft*). Veliku primenu pronašao u razvoju desktop i veb aplikacija.

ASP .Net Core [4] – predstavlja okvir (eng. Framework) javno dostupnog koda koji je namenjen za ravoj zadnje strane veb aplikacija.

Azurne funkcije (eng. *Azure Functions*) [5] - predstavlja proizvod razvijen od strane Majkrosofta (eng. Microsoft) i pruža mogućnost izvršavanja bez servera (eng. *Serverless*). Svoju primenu pronašle su u situacijama kada imamo kod koji će se izvršavati po potrebi.

React [6] – prestavlja biblioteku razvijenu od straje Fejsbuka (eng. Facebook) koja omogućava razvoj modernih veb aplikacija.

Tajpskript (eng. *TypeScript*) [7] – predstavlja jezik otvorenog koda napravljen kako bi rešio i olakšao programiranje u već postojećem javaskript (eng. *JavaScript*) jeziku.

HTML (eng. *HyperText Markup Language*) [8] – predstavlja opisni jezik namenjen za opis prednje strane veb aplikacije.

CSS (eng. *Cascade Style Sheet*) [9]- predstavlja jezik čijom upotrebom se definiše izgled prethodno kreiranih HTML tagova.

Elastični pretraživač (eng. *Elasticsearch*) [10] - predstavlja distribuirani, *RESTful* pretraživač i analitički motor zasnovan na *Apache Lucene*. Dizajniran je da bude brz, skalabilan i lako proširiv, omogućavajući pretragu, analizu i vizualizaciju ogromnih količina podataka u realnom vremenu.

2.2. Alati

Vižual studio kod (eng. *Visual Studio Code*) [11] - predstavlja kod editor koji je razvijen od strane Majkrosofta (eng. Microsoft).

Vižual studio 2022 (eng. *Visual Studio 2022*) [12] - predstavlja razvojno okruženje razvijeno od strane Majkrosofta (eng. *Microsoft*).

Doker desktop (eng. *Docker Desktop*) [13] – predstavlja aplikaciju koja se na *Windows* operativni sistem instalira u nekoliko jednostavnih koraka. Ovaj alat korišćen je za pokretanje elastičnog pretraživača.

Kibana [14] – predstavlja vizualizacijski alat otvorenog koda koji se koristi za pretragu, analizu i vizualizaciju podataka pohranjenih u *Elasticsearch.*

3. ETERIJUM

Ethereum je globalno decentralizovana računarska infrastruktura koja funkcioniše kao svetski kompjuter. Iz perspektive računarskih nauka *Ethereum* je deterministička mašina stanja sa globalno dostupnim jedinstvenim stanjem i virtuelnom mašinom koja primenjuje promene na to stanje. Praktično gledano, *Ethereum* koristi lanac blokova za sinhronizaciju i skladištenje promena stanja sistema, dok se kriptovaluta *Ether* koristi za merenje i ograničavanje troškova resursa izvršenja.

Sposobnost Eterijuma da izvršava programe u svojoj virtuelnoj mašini, poznatoj kao *Ethereum Virtual Machine* (*EVM*) [15], dok čita i piše podatke u memoriji, čini ga Turing-kompletnim [16] sistemom. To znači da *Ethereum* može izvršavati bilo koji algoritam koji je izvodljiv na Turingovoj mašini, pod uslovom da postoji dovoljno memorije.

Ethereum uvodi mehanizam zvan gas, koji meri i ograničava količinu resursa koje jedan pametni ugovor može potrošiti. Svaka instrukcija u *EVM*-u ima unapred određenu cenu u jedinicama gasa, a kada transakcija pokrene izvršenje pametnog ugovora, mora sadržati određeni iznos gasa koji postavlja gornju granicu za izvršenje.

Transakcija je pojedinačna instrukcija koja je kriptografski potpisana i konstruisana od strane aktera u mreži. Pošiljalac transakcije ne može biti pametan ugovor. Iako se pretpostavlja da će konačni spoljnji akter biti ljudske prirode, za konstrukciju i distribuciju transakcija koriste se softverski alati.

Ethereum novčanik je alat koji omogućava korisnicima da upravljaju svojim sredstvima i interaguju sa *Ethereum* lancem blokova. On čuva privatne ključeve koji su ključni za pristup *Ethereum* adresama i sredstvima, dok se javni ključ povezuje sa adresom na kojoj su sredstva pohranjena. Novčanik omogućava slanje i primanje *ETH*-a i tokena, kao i interakciju sa pametnim ugovorima i decentralizovanim aplikacijama.

Blok je osnovna jedinica podataka u *Ethereum* lancu blokova, koja sadrži informacije o transakcijama i promenama stanja u mreži. Svaki blok sadrži skup transakcija koje su izvršene u određenom vremenskom periodu, zajedno sa meta podacima kao što su vreme kreiranja bloka, referenca na prethodni blok (poznata kao heš prethodnog bloka), i druge podatke neophodne za validaciju i integraciju sa lancem blokova.

Konsenzus mehanizam *Proof of Stake* (*PoS*) [17] u *Ethereum* mreži predstavlja ključnu promenu u načinu na koji se novi blokovi dodaju u mrežu i održava sigurnost mreže. U *PoS* sistemu, umesto da rudari rešavaju složene matematičke probleme kao što je to bio slučaj u *Proof of Work* (*PoW*) [18] sistemu, validatori su odabrani na osnovu količine Eterijuma koju su založili kao garanciju.

4. PRETRAŽIVAČ LANCA BLOKOVA

Pretraživač lanca blokova je sofisticirani alat koji omogućava korisnicima da pretražuju, pregledaju i analiziraju sve podatke unutar mreže. Njegova svrha je pružiti transparentnost, uvid u aktivnosti i informacije koje se odvijaju na mreži, te omogućiti lakše praćenje i razumevanje funkcionalnosti i transakcija unutar mreže.

4.1. Funkcionalnosti

Vrlo često se javlja potreba pregleda poslednjih blokova odnosno transakcija na mreži. Inicijalna stranica implementiranog pretraživača sadrži vizuelni prikaz poslednjih blokova i transakcija na mreži. U okviru trake za navigaciju postoji mogućnost pretrage po sledećem kriterijumima: heš vrednosti ili broju kada je reč o blokovima i transakcijama kao i adresi novčanika. U zavisnosti od rezultata korisnik se preusmerava na

odgovarajući prikaz koji sadrži detalje rezultata pretrage. Manipulacija tržištem kpritovalute je prisutna u velikoj meri zbog toga što tržište i dalje nema dovoljno likvidnost što se koristi od strane pojedinaca koji raspolažu velikim sredstvima. Dešavanja u okviru njihovih novčanika često može ukazati na pravac u kojem će se tržište kretati. Upravo iz ovog razloga implementiran je poseban prikaz koji omogućava praćenje velikih transakcija u prethodnom periodu sa ciljem pružanja mogućnosti adekvatno reagovanja na potencijalne promene na tržištu

4.2. Arhitektura i performanse

Jedan od načina za dobavljanje podataka sa mreže bi bio pokretanje lokalno Eterijum čvora što iziskuje određene hardverske resurse kao i samo održavanje u vidu ažuriranja softvera i tako dalje. Kako bi se izbegli problemi koji se mogu javiti prilikom pokretanja jednog takvog čvora donešena je odluka da se iskoristi eksterni servis pod nazivom *Infura* [19] koji pruža usluge manipulacije podacima raznih mreža uključujući i Eterijum. Podaci dobavljeni sa eksternog servisa čuvaju se u elastičnom pretraživaču (eng. *Elasticsearch*).
Kako bi pretraživač imao mogućnost prikazivanja poslednjih blokova sa mreže neophodan je mehanizam sinhronizacije sa dešavanjima prisutnim na mreži, odnosno neophodno je dobavljati nove blokove i transakcije u okviru njih u lokalnu bazu podataka. Upravo ovaj problem rešen je upotrebom Azurne funkcije (eng *Azure function*) koja pruža mogućnost izvršavanja koda bez servera (eng. *Serverless*).
Kako bi lokalno sačuvani podaci postali dostupni prednjoj stranii aplikacije implementiran je interfejs za programiranje aplikacija (eng. *Application Programming Interface*). Ovaj projekat sadrži biznis logiku celokupne aplikacije.
Sa ciljem poboljšanja korisničkog iskustva kada je reč o performansama donešena je odluka da se za bazu podataka koristi elastični pretraživač. Ova tehnologija optimizovana je za veliki broj zahteva koji čitaju podatke. Kako bi se dodatno unapredile performanse donešena je odluka da se implementirana skladištenje podataka u memoriji. *Redis* je izuzetno brz sa mogućnošću obrade miliona zahteva u sekundi.

5. ZAKLJUČAK

Pretraživač lanca blokova predstavlja ključni alat za omogućavanje transparentnosti i analize podataka u okviru mreža. Ovaj alat omogućava korisnicima, istraživačima, regulatorima i razvojnim timovima da efikasno pregledaju sve transakcije, blokove, pametne ugovore i druge aktivnosti unutar mreže, pružajući uvid u decentralizovane sisteme i njihove procese. Tokom istraživanja u ovom radu, pokazano je da pretraživač lanca blokova nudi širok spektar funkcionalnosti, uključujući praćenje stanja adresa, verifikaciju transakcija i analizu blokova. Pored toga, pretraživač se koristi kao alat za analizu učinka mreže, praćenje aktivnosti u cilju prepoznavanja potencijalnih prevara. Buduća istraživanja i razvoj u oblasti pretraživača lanca blokova treba da se fokusiraju na poboljšanje performansi, uvođenje funkcija za zaštitu privatnosti, kao i na interoperabilnost između različitih mreža.

5.1. Predlozi za dalja usavršavanja

Pretraživač lanca blokova treba da bude intuitivan i jednostavan za korišćenje, kako za tehničke korisnike, tako i za širu publiku. Uvođenje vizuelnih prikaza podataka kao što su grafikoni transakcija, dinamika mreže, ili statistike o založenim sredstvima i validatorima može učiniti pretraživač pristupačnijim i razumljivijim. Personalizovani prikazi, filtriranje transakcija prema specifičnim kriterijumima i prikaz ključnih metrika u realnom vremenu dodatno bi poboljšao korisničko iskustvo.
Sa sve većim brojem decentralizovanih aplikacija na platformama poput Eterijuma, neophodno je da pretraživač pruži detaljniji uvid u rad i stanje pametnih ugovora. Ovo uključuje funkcionalnosti za pregled izvornog koda pametnih ugovora, praćenje interakcija sa ugovorima i analizu izvršenja. Mogućnost da se prati istorija ažuriranja ugovora, kao i uvida u potrošnju gasa i efikasnost ugovora, može biti korisna za programere i korisnike.
Kako broj mreža raste, potreba za interoperabilnošću između njih postaje sve važnija. Pretraživač bi trebalo da omogući praćenje više različitih mreža lanca blokova na jednoj platformi, što bi korisnicima omogućilo uvid u transakcije i blokove na različitim mrežama. To bi moglo uključivati i prikaz mostova između mreža, transakcija, kao i stanja u više lanaca.
Uvođenje opcije za korisnike da ocenjuju ili komentarišu određene transakcije ili pametne ugovore može pomoći u jačanju zajednice i omogućiti korisnicima da lakše identifikuju sumnjive ili pouzdane transakcije i ugovore.
Decentralizovane finansije (eng. *DeFi*) [20] su popularan segment ekosistema. Pretraživač može biti proširen kako bi omogućio uvid u interakcije sa *DeFi* protokolima, uključujući pozajmljivanje sredstava, zalaganje, prikupljanje kamate i slično.
Sa rastućom popularnošću tokena i nezamenljivih tokena (eng. *NFT*) [21], pretraživač može biti unapređen tako da omogući detaljno praćenje ERC-721 standarda. Korisnici bi mogli jednostavno da prate transfere tokena, stanje na adresama koje poseduju određene tokene, i pregledaju informacije o samim tokenima, poput njihove vrednosti, vlasništva i meta podataka za NFT tokene.

6. LITERATURA

[1] Blockchain, https://en.wikipedia.org/wiki/Blockchain
[2] Ethereum, https://ethereum.org/en/
[3] Mahesh Chand, Programming C# for Beginners, Garnet Valley PA. Sept 01, 2014, str. 4-7
[4] Kenneth Yamikani Fukizi, Jason De Oliveira and Michel Bruchet, Learn ASP .NET Core 3, Packt Publishing Ltd. Livery Place 35 Livery Street Birmingham B3 2PB, UK. ISBN 978-1-78961-013-0, str. 13-15

[5] Azurne funkcije, https://learn.microsoft.com/en-us/azure/azure-functions/functions-over
[6] React, https://react.dev/
[7] Tajpskript, https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html
[8] Jon Duckett, HTML & CSS design and build webistes, John Wiley & Sons, Inc. 10475 Crosspoint Boulevard Indianapolis, IN 46256, str. 20-24
[9] Jon Duckett, HTML & CSS design and build webistes, John Wiley & Sons, Inc. 10475 Crosspoint Boulevard Indianapolis, IN 46256, str, 227-233
[10] Elastični pretraživač, https://www.elastic.co/elasticsearch
[11] Vižual studio kod, https://code.visualstudio.com/
[12] Vižual studio 2022, https://visualstudio.microsoft.com/vs/
[13] Doker desktop, https://www.docker.com/products/docker-desktop
[14] Kibana, https://www.elastic.co/kibana
[15] EVM, https://ethereum.org/en/developers/docs/evm/.
[16] Turing kompletan, https://en.wikipedia.org/wiki/Turing_completeness
[17] PoS, https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/
[18] PoV, https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/
[19] Infura, https://www.infura.io/
[20] Decentralizovane finansije, https://www.coinbase.com/learn/crypto-basics/what-is-defi
[21] NFT, https://ethereum.org/en/nft/

7. BIOGRAFIJA

Nikola Mijonić je rođen 03.06.1998. godine u Somboru. Srednju elektrotehničku školu “Mihajlo Pupin” završio u Novom Sadu 2017. godine. Iste godine upisao se na studije primenjenog softverskog inženjerstva Fakulteta tehničkih nauka. Položio je sve ispite predviđene planom i programom. Master studije na programu Elektronsko poslovanje upisao 2021. godine i položio sve ispite.

Зборник радова Факултета техничких наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/30BE53Markovic

УНАПРЕЂЕЊЕ РАДА *CNC* МАШИНЕ КОРИШЋЕЊЕМ *GRBLHAL* БИБЛИОТЕКЕ

IMPROVING CNC MACHINE PERFORMANCE USING GRBLHAL LIBRARY

Никола Марковић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду изложено је унапређење CNC машине по питању брзине и перформанси. За управљање је коришћена Clicker 4 развојна плоча са STM32F407 контролером и одговарајућим драјверима. Такође, приказане су и предности grblHAL-а у односу на оригинални GRBL.*

Кључне речи: *grblHAL, управљачка електроника, драјвери за корачне моторе.*

Abstract – *This paper presents the improvement of a CNC machine in terms of speed and performance. A Clicker 4 development board with the STM32F407 controller and corresponding drivers was used for control. Additionally, the advantages of grblHAL over the original GRBL are also demonstrated.*

Keywords: *grblHAL, control electronics, step-mottor drivers.*

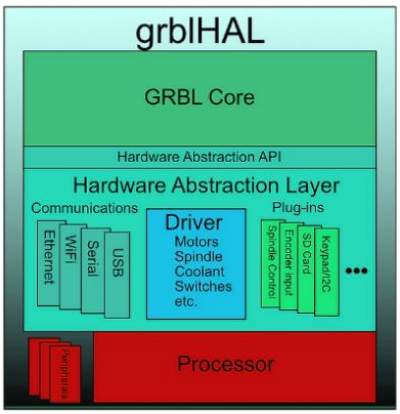
1. УВОД

GRBL је софтвер који омогућава управљање *CNC* машинама користећи *G-code*. Покреће се на 8-битним микроконтролерима, као што је *Arduino*, и служи за претварање *G-code* команди у покрете *CNC* машине. *GRBL* је познат по својој једноставности, поузданости и могућности рада на приступачним хардверским платформама. Поред ефикасног управљања *CNC* машинама, главна предност *GRBL*-а је то што је бесплатан за коришћење, што га чини доступним широкој заједници хобиста и професионалаца. Самим тим, омогућено је креирање јефтиних *CNC* система, што је одиграло велику улогу у популаризацији *CNC* машина међу хобистима и малим произвођачима.

Како је 8-битни *GRBL* достигао своје границе на *Arduinu*, почело се радити на преласку на неку другу хардверску платформу тј. на неке од све бројнијих, јефтиних 32-битних контролера, посебно на *ARM* базиране микроконтролере. Из тог разлога, задатак овог рада јесте реализација управљања *CNC* машином помоћу библиотеке *grblHAL* и контролера *STM32F407*.

2. *GRBL* БИБЛИОТЕКА

Због тога што је *GRBL* био оптимизован да ефикасно користи *Atmel* микроконтролер на *Arduinu*, код специфичан за машину био је помијешан са кодом независним од машине, што је отежавало прелазак на друге хардверске платформе. Такође, свака нова функција која је додата или грешка исправљена у једној верзији, морала би се прилагодити другим портовима. Није постојала главна верзија изворног кода из које би се могле изградити све различите 32-битне верзије. Прелазак на нови микроконтролер или чак на другачију варијанту, унутар исте породице, захтијевао је дубоко разумијевање функционисања *GRBL*-а [1]. Рјешење тог проблема било је да се *GRBL* подијели на два дијела: један који садржи сав процесорски зависан код – слој апстракције хардвера (*HAL*) и други који не садржи – *GRBL* језгро. Тако је настао *grblHAL*. *HAL* садржи код који иницијализује процесор, управља тајмерима, *PWM* хардвером, портовима, адресама улазно-излазних прикључака, комуникацијом итд. *GRBL* језгро комуницира само са *HAL*-ом. Пребацивање *grblHAL*-а на нови микроконтролер траје неколико недјеља, па чак и дана у неким случајевима, јер је *HAL* слој релативно мали. Подјела *grblHAL*-а на *GRBL* и *HAL* дио, као и сама организација *grblHAL*-а приказана је на слици 1.



Слика 1. *Организација grblHAL-а [1]*

Пребацивање *grblHAL*-а на нови микроконтролер је једноставно. Један драјвер фајл дефинише хардверску апстракцију за циљни процесор. Програмери могу почети са постојећим драјвером или *ARM* шаблоном драјвера. Пошто се већина кода, који треба да модификују, налази у драјвер фајлу, не морају да разумију *GRBL* језгро.

2.1. Предности *GrblHAL-а*

GrblHAL је 32-битна верзија *GRBL*-а (који ради на 8-битним *Arduinima*). Док је *Arduino* са *GRBL*-ом врло јефтина платформа за изградњу *CNC* машина, 8-битни

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

Табела 1. *Поређење Grbl-а и GrblHAL-а*

	Grbl	GrblHAL
Подржани микроконтролери	ATmega328	STM32, ESP32
Број оса	3 осе	До 6 оса
Перформансе	Спорији процесор	Бржи процесор
Комуникациони протоколи	USB	USB, WiFi, Ethernet
Лакоћа подешавања	Једноставно, стабилно	Напредније, али сложеније за подешавање
Погодност за почетнике	Веома погодно	Више за напредне кориснике
Компатибилност са софтвером	Одлична подршка за једноставне пројекте	Шира подршка за сложеније пројекте
Цијена хардвера	Јефтинији	Скупљи хардвер
Меморијски капацитет	32KB Flash, 2KB SRAM	STM32F4: 1MB Flash, 192KB Sram

процесор озбиљно ограничава његове могућности. Са *grblHAL*-ом који ради на 32-битним *ARM* процесорима, доступне су много боље перформансе и знатно више функција. За почетак, подржано је више од 3 осе, а могуће су знатно веће брзине корака. Поред тога, *grblHAL* подржава многе додатке који омогућавају додавање нових функција. Поређење *Grbl-а и GrblHAL-а* је приказано у табели 1.

Покретањем на 32-битним микроконтролерима, *grblHAL* доноси бројне предности:

• више перформансе - оригинални *Arduino* ради на 16MHz, док 32-битни *ARM* чипови могу радити знатно брже,

• више меморије - много већи простор за програме омогућава значајно више функција,

• ниски трошкови - *grblHAL* је бесплатан, а процесори на којима ради су веома јефтини,

• конзистентност - *GCode* је конзистентан на свим платформама које подржавају исте функције,

• брзо портовање на нове микроконтролере и варијанте подржаних чипова,

• *grblHAL* има архитектуру са додатним модулима (Plug-in),

• лака прилагођавања функција за специфичне апликације и машине.

3. ПРОЈЕКТОВАЊЕ *CNC* СИСТЕМА ЗАСНОВАНОГ НА *GrblHAL*-У

За пројектовање минималног *CNC* система неопходно је неколико компонената [2]:

• контролер,

• корачни мотори,

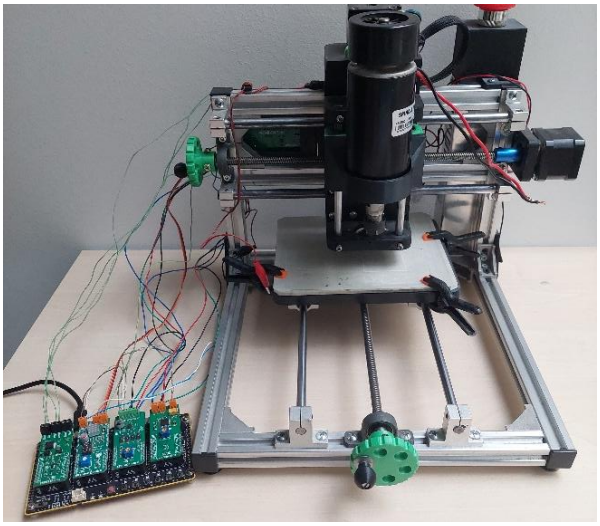
• обрадни мотор,

• драјвери за моторе,

• обрадни алат,

• напајање.

Одговарајућим повезивањем ових компонената и прилагођавањем *grblHAL* софтвера датом контролеру могуће је остварити функционисање *CNC* машине. На слици 2. приказан је изглед *CNC* машине са повезаним контролером и драјверима.



Слика 2. *Компоненте неопходне за повезивање минималног CNC система [2]*

4. КОНТРОЛЕР И ДРАЈВЕРИ

На самом почетку потребно је одлучити који контролер ће се користити у раду. Контролера који имају подршку за *grblHAL* је све више и више, а због брзине и перформанси одлука је пала на *STM32* фамилију микроконтролера. *STM32* контролери су јако популарни, па самим тим постоји много развојних плоча на којима су уграђени (најпопуларније су *Nucleo* плоче).

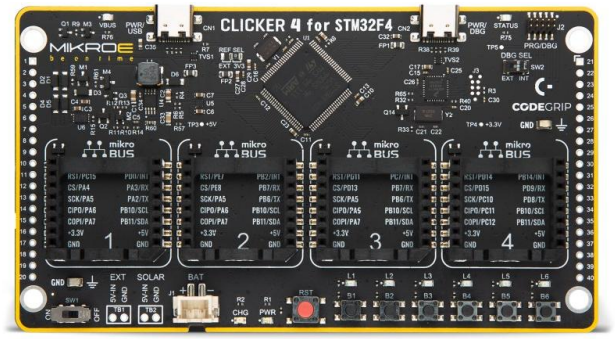
Међутим, за израду овог рада изабрана је развојна плоча *Clicker 4* за *STM32F4* која је произведена у српској компанији *Микроелектроника.* Разлог за бирање ове плоче је то што иста компанија производи и велики број драјвера потребних за покретање мотора. Сви драјвери су прављени по *mikroBus* стандарду, а

Clicker 4 посједује четири *mikroBus* конектора, па је повезивање драјвера са плочом изузетно једноставно.

4.1. *Clicker 4* за *STM32F4*

Clicker 4 за *STM32F4* представља компактну развојну плочу дизајнирану као комплетно рјешење које се може користити за брзо прављење сопствених уређаја са јединственим функционалностима. Опремљена *STM32F407VGT6* микроконтролером и четири *mikroBUS* конектора за повезивање *click* плочица, представља савршено рјешење за брзи развој различитих типова апликација.

Повезивање спољњих уређаја може се остварити преко два 1x20 *pin header*-а или преко четири унапријеђена *mikroBUS* конектора који омогућавају повезивање огромног броја *click* плочица. Да би се омогућило повезивање развојног система *Clicker4* и самих мотора који се налазе на *CNC* машини, користе се одговарајуће *click* плочице. То су уређаји који служе као драјвери за покретање мотора. На самој *CNC* машини налазе се четири мотора, па су због тога потребна четири драјвера тј. *click* плочице. Изглед *Clicker 4* развојне плоче приказан је на слици 3.



Слика 3. *Развојни систем Clicker 4 [3]*

4.2. Драјвери за моторе

Микроелектроника има велики избор драјвера за моторе. Тренутно имају више од 40 различитих драјвера за корачне моторе. Подијељени су по фамилијама, па постоје *Stepper*, *Multi Stepper*, *Silent Step*, *STSPIN* фамилија итд. Приликом израде рада, на располагању је било пет различитих драјвера за корачне моторе. То су *Stepper 4*, *Stepper 11*, *STSPIN820*, *Silent Step* и *Silent Step 3*. Сваки од ових драјвера је намијењен за покретање биполарних корачних мотора и сваки има своје предности и мане које су приказане у табели 2. Одлучено је да се користи по један драјвер из сваке фамилије и то су *Stepper 4 click*, *STSPIN820 click*, *Silent Step click*.. На слици 4. је приказан сваки од њих.

Слика 4. *Изглед коришћених драјвера [3]*

Табела 2. *Поређење драјвера за корачне моторе*

	Предности	Мане
Stepper 4 click	Максимална струја 2A (струјни скок до 4A)	Само 1/32 микрокорака
Stepper 11 click	Максимална струја до 3A	Само 1/32 микрокорака
STSPIN820 click	Висока прецизност због 1/256 микрокорака и мала дисипација снаге	Мала струја до 1,5A што ограничава коришћење већих мотора
Silent Step click	Висока прецизност због 1/256 микрокорака	Мала струја до 1,2A и комплексније подешавање због *SPI* модула
Silent Step 3 click	Висока прецизност због 1/256 микрокорака	Комплексније подешавање због *SPI* модула

4.3. Управљање обрадним мотором

За управљање обрадном главом машине је такође потребан неки драјвер. Микроелектроника и за ту примјену има велики избор драјвера, као што су нпр. *PROFET* или *Relay click* плочице. Основна разлика је што *PROFET* драјвери користе снажне *MOSFET* транзисторе за управљање, док *Relay* драјвери користе механичке релеје. Главна предност *PROFET* драјвера је њихова брзина, а предност *Relay* драјвера је рад на ниским фреквенцијама и примјенама гдје је потребна велика снага. За израду овог рада користи се *PROFET 2 click*, приказан на слици 5. На плочи се налази коло *BTS7080-2EP* које има способност управљања оптерећењима до 3A. Овај драјвер подржава спољашње напајање за *BTS7080-2EP*, које може бити повезано на улазни терминал означен као *VIN* и требало би да буде у опсегу од 4.1V до 28V.

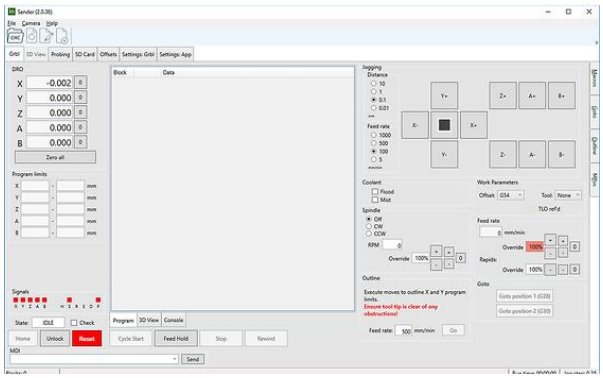
Слика 5. *Изглед драјвера за обрадни мотор [3]*

5. ПРОГРАМИ ЗА ПОКРЕТАЊЕ *CNC* МАШИНЕ

Како би се успјешно управљало *CNC* машином неопходно је користити неки од софтверских алата који имају могућност слања *G-code* команди. Један од таквих програма је *IoSender*. Прије коришћења неопходно је провјерити да ли је све повезано, јер програм функционише тако што шаље команде преко

серијске комуникације, па је неопходно да постоји *Virtual Com Port* преко кога контролер прима податке.

IoSender је софтвер који се углавном користи у комбинацији са *grblHAL* библиотеком и ради као *G-code* интерпретер [4]. Да би се покренуо задатак, потребно је учитати *G-code* и поставити осе у одговарајући положај. Такође, постоји и 3Д преглед путања алата, што помаже да се визуелно прати процес обраде. На слици 6. приказан је изглед *IoSender* програма.

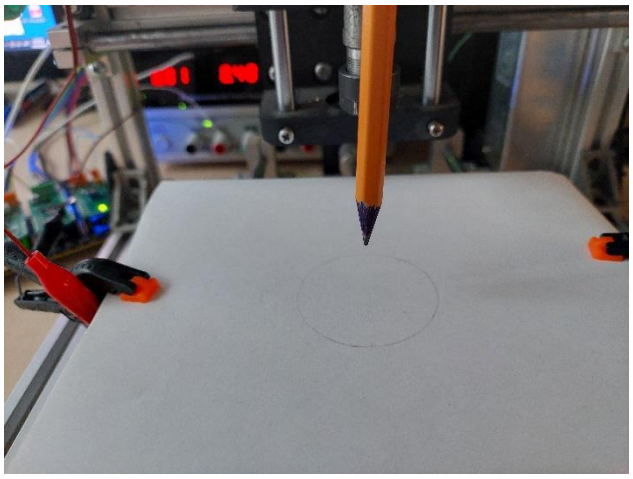


Слика 6. *Изглед IoSender програма [4]*

Слање *G-code* команди је могуће и помоћу неког од алата за серијску комуникацију. Један од таквих програма је *Docklight*. Као и код *IoSender*-а и овде је могуће слати команде.

6. ТЕСТИРАЊЕ

За тестирање функционалности *CNC* машине, као обрадни алат постављена је оловка, чији је задатак да исцрта објекат или слику, која се задаје кроз *G-code* команде. На примјеру који је приказан на слици 7, задатак је био да се исцрта кружница и може се видјети да је задатак успјешно извршен.



Слика 7. *Приказ исцртане кружнице*

7. ЗАКЉУЧАК

У овом раду приказана је минимална конфигурација са само неопходним компонентама. Минимална конфигурација може се унаприједити уградњом неких додатака као што су тастери за покретање и паузирање циклуса, гранични прекидачи који служе за принудно заустављање, додавање *Ethernet*-а (поузданији од *USB*-а), додавање *WiFi-a* итд. Такође је приказана способност *grblHAL*-а да брже и ефикасније обавља задатке које добија од контролера.

8. LITERATURA

[1] https://www.grbl.org/what-is-grblhal (приступљено у јуну 2024.)

[2] https://www.grbl.org/building-a-grblhal-machine (приступљено у јуну 2024.)

[3]https://www.mikroe.com/?srsltid=AfmBOorxqXSZtWdzGYirDU62e24SuYCXRxywwkqx4yT49v-GcGGvXRuy (приступљено у јулу 2024.)

[4] https://www.grbl.org/single-post/one-sender-to-rule-them-all (приступљено у августу 2024.)

Кратка биографија:

Никола Марковић рођен је у Теслићу 1999. god. Дипломски рад на Факултету техничких наука из области Електротехнике и рачунарства – Примењена електроника одбранио је 2022.god.
контакт: nmarkovic999@gmail.com

.

Зборник радова Факултета техничких наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/30BE54Tesovic

3D ШТАМПАНЕ МЈЕРНЕ ТРАКЕ

3D PRINTED STRAIN GAUGES

Драгана Тешовић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак саджај – *У раду је приказана фабрикација и карактеризација 3D штампаних мјерних трака. Посебан акценат стављен је на спрезање мјерне траке са инструментом. За мјерење промјене отпорности коришћен је дигитални мултиметар. Такође, приказан је и начин за одређивање мјерног фактора мјерне траке.*

Кључне речи: *Мјерне траке, мјерни фактор, 3D штампа.*

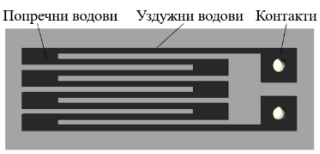
Abstract – *The paper presents the fabrication and characterization of 3D printed strain gauges. Special emphasis is placed on the integration of the strain gauge with the instrument. A digital multimeter was used to measure changes in resistance. The paper also outlines the method for determining the gauge factor of the strain gauge.*

Keywords: *Strain gauge, gauge factor, 3D printing.*

1. УВОД

Мјерења силе, притиска и напрезања представљају мјерења која су од великог значаја у многим гранама индустрије и научним дисциплинама, као што су медицина, роботика, грађевинарство, аутомобилска индустрија итд. Силе и напрезања мјере се веома често јер су важни параметри система који се користе у процесима управљања или чак могу бити индикатори непредвиђених оштећења и најава будућих озбиљнијих кварова, који указују на животни вијек система.

Мјерна трака (енг. *strain gauge*) чини отпорнички сензор који се користи за мјерење деформација. Постављају се на површину објекта који се испитује, а свака деформација објекта услед његовог оптерећења доводи до одговарајуће деформације мјерне траке, што све заједно омогућује мјерење промјене отпорности траке. На слици 1. приказан је изглед мјерне траке израђене технологијом *3D* штампе.



Слика 1. *Изглед 3D штампане мјерне траке*

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

Задатак овог рада јесте карактеризација и тестирање мјерне траке реализоване примјеном технологије *3D* штампе. Главна предност коришћења *3D* штампе јесте могућност израде у више слојева, чиме се директно утиче на отпорност. Поред тога, постоји могућност штампање више трака истовремено на једном супстрату.

1.1. Деформације чврстих тијела под дејством силе

Дејством силе, интензитета *F*, на чврсто тијело настаје напрезање. Механички напон дефинисан је количником силе (*F*) и површине на коју дјелује та сила (*S*) и по димензији једнак је притиску (Pa). Релативно издужење тијела, које је последица напрезања, назива се дилатација. Дилатација (деформација) може бити позитивна, када се тијело издужује или негативна, када тијело трпи сабијање. Величина дилатације зависи од силе која дјелује на тијело, али и од еластичних својстава материјала од кога је тијело направљено.

1.2. Типови мјерних трака

Постоје два типа мјерних трака – слободне и лијепљене мјерне траке. Слободне мјерне траке представљају разапету жицу између рамова који се могу помјерати један у односу на други, доводећи до напрезања слободне мјерне траке. Други тип мјерних трака јесу лијепљене мјерне траке. Нарезана односно нагрижена фолија, депоновани филм или метална жица причвршћује се на подлогу са којом се заједно лијепе на површину материјала који се испитује [1].

Од материјала који се користе за израду мјерних трака, захтијева се да имају велику специфичну отпорност, што мањи температурни коефицијент и велику осјетљивост. На основу врсте материјала који се користи за израду, мјерне траке могу бити полупроводничке и металне.

За израду полупроводничких мјерних трака најчешће се користи силицијум, *Si*, који се обликује у мале траке које се излажу истезању. При томе долази до повећања дужине траке и то је један од узрока промјене њене отпорности. Други узрок јесте заправо промјена специфичне отпорности силицијума. Овај ефекат је знатно важнији код полупроводничких мјерних трака од првог, јер је промјена отпорности због специфичне отпорности вишеструко већа од промјена отпорности услед промјене дужине траке.

Металне мјерне траке мјере помјерање на основу промјене електричне отпорности при промјени дужине отпорника. Трака се састоји од отпорне жице савијене

и затим постављене на савитљиву подлогу облика траке. Уколико се трака истеже, повећава се дужина жице, а тиме и отпорност. Отпорност допунски расте и услед промјене специфичне отпорности, али је први ефекат доминантнији.

1.4. Мјерење силе и напрезања

Мјерне траке лијепе се на тијела која трпе напрезања и, захваљујући специјалним смолама за обезбјеђивање квалитетног налијегања, траке су изложене готово идентичној дилатацији као и само тијело. Под дејством силе, мјерна трака се истеже или сабија, чиме се мијењају њена дужина и попречни пресјек.

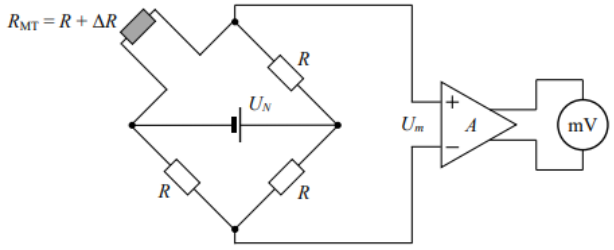
У општем случају, за произвољан отпорник било би веома тешко одредити у каквој сразмјери се мијењају дужина и попречни пресјек, када на тај отпорник дјелује сила. Међутим, савремени технолошки процеси омогућавају производњу мјерних трака код којих постоји стална (константна) сразмјерност између дилатације и промјене отпорности, која се назива осјетљивост мјерне траке или мјерни фактор, одређена изразом (1):

$$GF = \frac{\Delta R/R}{\Delta l/l} = \frac{\Delta R/R}{\varepsilon} \qquad (1)$$

Осјетљивост мјерне траке зависи од изабраног материјала, кроз Поасонов коефицијент и промјене електричне отпорности полупроводника или метала услед механичке деформације кроз пиезорезистивни коефицијент.

Типичан мјерни фактор за металне мјерне траке је око 2, док је за полупроводничке мјерне траке већи од 150.

Да би се могле мјерити мале промјене отпорности погодније је везати мјерну таку у мјерни мост, а потом мјерити напон мјерне дијагонале моста [2]. За веома мала напрезања и напон мјерене дијагонале је мали, па га је прије довођења на инструмент, некада потребно додатно појачати колом са операционим појачавачем, као на слици 2.



Слика 2. *Везивање једне мјерне траке и три стална отпорника у мјерни мост [2]*

Мјерење се обавља тако што се мост доводи у равнотежу када је трака ненапрегнута (када нема дилатације). Да би се то остварило, стални отпорници у гранама моста морају имати једнаку отпорност као и ненапрегнута мјерна трака. Приликом дјеловања силе и дилатације мјерне траке, отпорност мјерне траке ће се промијенити за неку вриједност ΔR и мост ће се раздесити.

2. ТЕХНОЛОГИЈА *3D* ШТАМПЕ

Технологија *3D* штампе, позната као адитивна производња, омогућава стварање физичких објеката, слој по слој, користећи дигиталне моделе. *3D* штампа може се замислити као штампа танких хоризонталних слојева. Ти слојеви представљају хоризонтални пресјек предмета. Производи који се *3D* штампају могу бити од различитих материјала, од пластике, преко керамике до метала.

2.1. *3D* штампачи

3D штампачи стварају тродимензионалне предмете од дигиталног фајла тако што их граде слој по слој, узастопно, све док предмет у потпуности није готов. Сваки слој је прецизан, танко сјечен, хоризонтални пресјек објекта који настаје. Изглед *3D* штампача приказан је на слици 3.

Слика 3. *Изглед 3D штампача*

3D штампач састоји се од механичке конструкције штампача са актуаторима, материјала за штампање, врућег краја, топлог кревета, млазнице, истискивача филамента, интерфејса са корисником (екран) и управљачке електронике [3].

Када постоји идеја како би прототип за *3D* штампање могао да изгледа, потребно је направити модел у неком од софтвера за *3D* моделовање. Неки од бесплатних комерцијално доступних алата су: *Tinkercad*, *ScatchUp*, *123D Design*, *FreeCAD*, *AutoCAD*, *OnShape*, *Blender*. *STL* датотека чува информације о *3D* моделима. Овај формат описује само геометрију површине тродимензионалног објекта без представљања боје, текстуре или других уобичајених атрибута *3D* модела. *STL* датотеке обично су генерисане програмом за аутоматско пројектовање (*CAD*), као крајњи производ процеса *3D* моделовања. „*.stl*“ је екстензија датотеке *STL* формата датотеке.

Софтвери за припрему *3D* штампе који долазе уз *3D* штампаче не могу да направе *3D* модел онога што се штампа већ само служе за [4]:

• позиционирање и орјентацију *3D* модела у радном простору штампача,

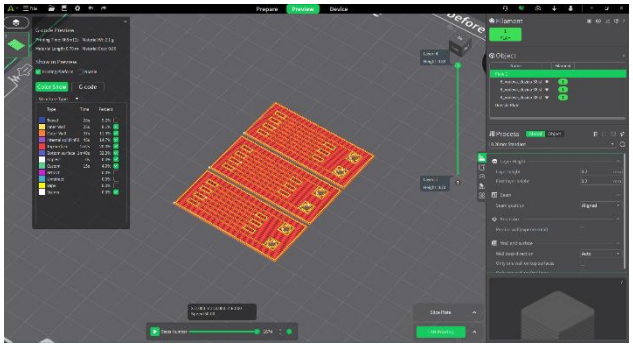
• генерисање *G*-кода за путању кретања главе/дизне штампача,

• умножавање комада за штампу више истовјетних комада у једном процесу,

• дефинисање параметара штампе – дебљина зидова, проценат испуне, геометрија испуне, облик потконструкције, итд.

GCODE датотека садржи наредбе у *G*-коду, који је језик који се користи да опише како *3D* штампач треба да штампа одређени *3D* модел. Она чува команде у обичном тексту, при чему сваки ред представља другачију наредбу, као што су брзина штампања којом треба да штампа одређени слој, температура на којој треба да се постави кревет и истискивач филамента и гдје би се дијелови штампања требали кретати.

Након припремљеног *3D* модела, слика 4, може се прећи на израду. За израду супстрата, на који се штампа *NinjaTek EEL* проводни филамент, користио се *PLA* филамент. Параметри штампе који су се подешавали при раду јесу температура кревета од 60°C, која остаје иста за оба филамета, температура *PLA* филамента од 220°C и температура проводног филамента од 230°C. Број водова, дужина, ширина и дебљина водова чине параметре мјерне траке који су се мијењали при фабрикацији.



Слика 4. *Модел slice-ован у софтверском алату*

3. РЕЗУЛТАТИ ТЕСТИРАЊА

Поглавље 3 посвећено је процесу практичног тестирања функционалности и анализи добијених резултата.

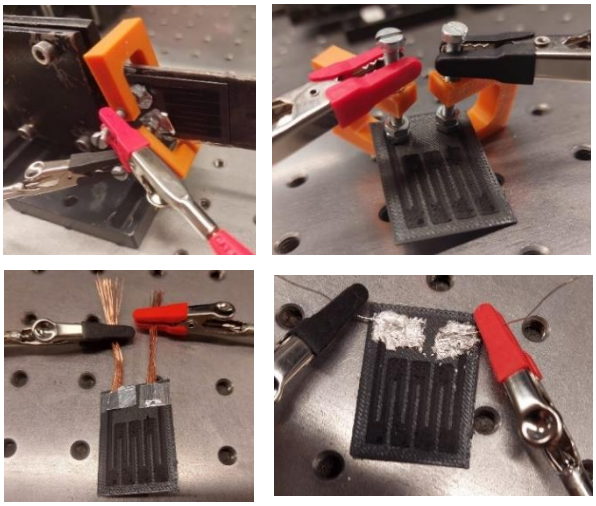
Табела 1. *Параметри мјерне траке и отпорности*

Параметри	Отпорност [kΩ]
Број водова: 6	150.1
Број водова: 8	158.5
Број водова: 10	191.8
Број водова: 12	205.8
Дужина вода: 34mm	176.5
Дужина вода: 38mm	239.5
Дужина вода: 42mm	250.1
Ширина вода: 1.2mm	115
Ширина вода: 1.6mm	90.1
Ширина вода: 2mm	52.7
Дебљина вода: 0.2mm	158.5
Дебљина вода: 0.4mm	61.3
Дебљина вода: 0.6mm	39.81
Дебљина вода: 0.8mm	29.98

Као што је већ поменуто, различити параметри мјерне траке су се мијењали па самим тим и њена отпорност, што је приказано у табели 1. Тестирања су вршена на различитим поставкама, односно на различитим начинима мјерења отпорности мјерне траке.

У зависности од начина спрезања мјерне траке са дигиталним мултиметром, како би се измјерила отпорност, долази до промјене укупне отпорности саме траке. У наставку ће бити приказана четири начина остваривања контаката између мјерне траке и инструмента.

На слици 5. приказана су четири начина мјерења отпорности мјерне траке: мјерна трака залијепљена на метални профил, мјерна трака директно везана за сонде дигиталног мултиметра, мјерна трака са лицнастом жицом и мјерна трака са термалном пастом.



Слика 5. *Поставке за мјерење отпорности*

Посматра се промјена отпорности у зависности од остварених контаката, у интервалу од 20 минута, са биљежењем отпорности на сваких 5 минута.

Резултати мјерења за сва четири начина приказани су у табели 2.

Табела 2. *Резултати мјерења*

	Први начин	Други начин	Трећи начин	Четврти начин
t [min]	*R* [kΩ]	*R* [kΩ]	*R* [kΩ]	*R* [kΩ]
0	111.88	74.12	91.12	552
5	109.72	73.25	84.84	536
10	107.39	73.16	84.47	527
15	106.22	73.11	83.79	535
20	105.81	73.11	83.64	529

График 1. приказује промјене отпорности у току времена за сва четири описана начина мјерења.

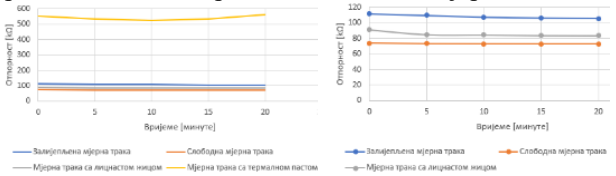


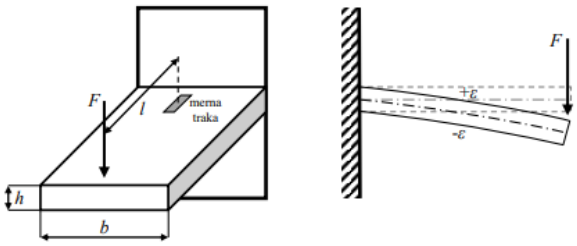
График 1. *Промјена отпорности током времена*

Може се закључити да мјерење промјене отпорности има најбоље резултате уколико се користи поставка мјерне траке са лицнастом жицом. Резултати добијени на тај начин приближно су константни током времена и приближни номиналној вриједности отпорности тестиране траке (90.1kΩ, ширина вода 1.6mm). Поставка мјерне траке са термалном пастом даје

најлошије резултате, гдје се очитава отпорност која је вишеструко већа од отпорности саме мјерне траке.

3.1. Одређивање мјерног фактора

Осјетљивост мјерне траке или мјерни фактор GF (енг. *gauge factor*) представља се као релативна промјена отпорности за дато напрезање. Дјеловањем силе на конзолу, на чијем другом крају се налази мјерна трака, долази до промјене отпорности траке, слика 8.



Слика 8. *Дејство силе на конзолу [2]*

На основу дјеловања силе, за траку чија је дужина водова 42mm и савијање у опсегу до 20mm, са кораком од 5mm, добијају се промјене отпорности приказане у табели 3.

Савијање [mm]	Отпорност [kΩ]
0	273.3
5	278.6
10	286.3
15	294.1
20	301.6

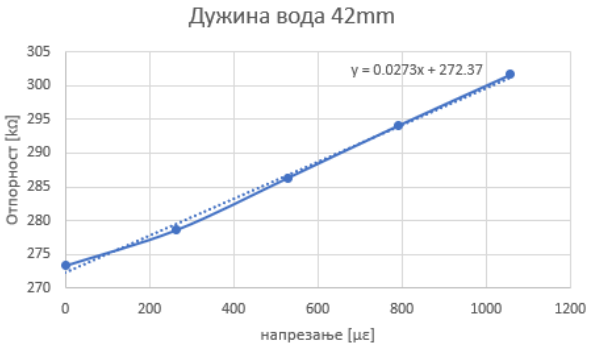
Табела 3. *Промјена отпорности за различита напрезања*

Како би се одредио мјерни фактор, неопходно је одредити напрезање траке. Напрезање се одређује помоћу израза (2):

$$\varepsilon = \frac{6(l-z)\delta h}{4l^3} \qquad (2)$$

гдје је l дужина конзоле, z растојање мјерне траке од фиксираног дијела, δ савијање, h дебљина конзоле. На основу одређеног напрезања, мјерни фактор се рачуна помоћу израза (3):

$$GF = \frac{\Delta R}{R} * \frac{1}{\varepsilon} \qquad (3)$$



Grafik 2. *Одређивање мјерног фактора*

На графику 2. приказана је зависност отпорности од напрезања. Мјерни фактор у овом случају има вриједност *GF*=100. Честице унутар проводног филамента, коришћеног за водове и контакте, мијењају специфичну отпорност која доминира у овом случају, због свог различитог распореда.

4. ЗАКЉУЧАК

Првобитни циљ који је постављен у овом раду била је фабрикација и карактеризација *3D* штампаних мјерних трака, али и осмишљање мјерне поставке за тестирање трака. Сами контакти доста утичу на отпорност траке, па је неопходно пажљиво бирање поставке за тестирање.

С обзиром на то да се приликом тестирања добијао мјерни фактор чија вриједност је већа од очекиване, може се закључити да пиезоефекат није занемарљив. На отпорност трака такође утиче и начин налијегања филамента приликом саме израде.

5. ЛИТЕРАТУРА

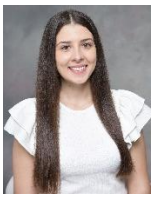
[1]https://www.optolab.ftn.uns.ac.rs/images/NASTAVA/SIA/files/predavanja/4-merenje-sile-pritiska-i-naprezanja.pdf (приступљено у јуну 2024.)

[2]http://www.kelm.ftn.uns.ac.rs/literatura/mut/10_Merenje_sile_i_naprezanja.pdf (приступљено у јуну 2024.)

[3] https://print24.com/rs/journal/osnove-stampanja/3d-stampanje (приступљено у јулу 2024.)

[4] https://solfins.com/en_US/3dstampa-3dskeniranje/softveri-za-3d-stampu-i-3d-skeniranje (приступљено у августу 2024.)

Кратка биографија:



Драгана Тешовић рођена је у Фочи 1999. год. Дипломски рад на Факултету техничких наука из области Електротехнике и рачунарства – Примењена електроника одбранила је 2022.год.
контакт: tesovicdragana195@gmail.com