УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

ЗБОРНИК РАДОВА
ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке – зборници
Година: XLI
Број: 8/2026

Нови Сад

Едиција: „Техничке науке – Зборници“
Година: XLI Свеска: 8
Издавач: Факултет техничких наука Нови Сад

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

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

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

ОБЛАСТИ (СТУДИЈСКИ ПРОГРАМИ)

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

- **Електротехничко и рачунарско инжењерство**

ПРЕДГОВОР

Поштовани читаоци,

Пред вама је осма 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 Факултета техничких наука, али и 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.

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.

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

ЕЛЕКТРОТЕХНИЧКО И РАЧУНАРСКО ИНЖЕЊЕРСТВО

Зборник радова Факултета техничких наука, Нови Сад

УДК: 621.31
ДОИ: https://doi.org/10.24867/35BE01Banjac

Испитивање модула пренапонске заштите код нисконапонских каблова

Testing of Surge Protection Modules for Low-Voltage Cables

Милан Бањац, Дејан Јеркан, *Факултет техничких наука, Нови Сад*

Студијски програм – ЕНЕРГЕТИКА, ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ

Кратак садржај – *У раду је анализирана функција модула пренапонске заштите у продужном каблу, као и њен утицај на заштиту крајњих потрошача прикључених на тај продужни кабл. Објашњено је шта су пренапони и њихове кључне карактеристике, као и функција компоненти које се налазе у модулу пренапонске заштите. Такође је објашњен и принцип рада самог модула.*

Кључне речи: *пренапон, модул пренапонске заштите, продужни кабл.*

Abstract – *This paper analyzes the function of the surge protection module in power strip, as well as its impact on protecting end users connected to that power strip. It explains what surges are and their key characteristics, along with the function of the components contained within the surge protection module. The operating principle of the module itself is also described.*

Keywords: (three to five): *surge, surge protection module, power strip*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Дејан Јеркан, ванредни професор.

1. УВОД

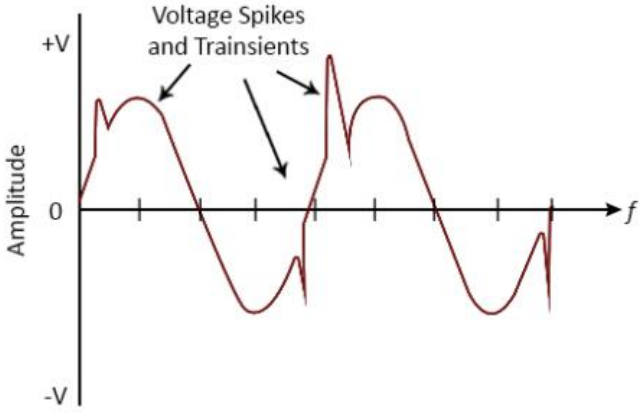
Савремено друштво је незамисливо без поуздане и стабилне испоруке електричне енергије. Са развојем технологије и растом броја електронских уређаја у домаћинствима и индустрији, значајно се повећала и осетљивост корисника на квалитет напајања електричном енергијом. Чак и краткотрајна одступања од номиналних вредности напона могу изазвати озбиљна оштећења, прекиде у раду или смањење животног века електронске опреме. Међу свим поремећајима у електроенергетским мрежама, посебно место заузимају пренапони - краткотрајни скокови напона који могу имати природно или вештачко порекло. Управо пренапони представљају један од главних узрока отказа електричне и електронске опреме, која није димензионисана тако да може да их издржи самостално без помоћи потпуних каскадних заштита од пренапона.

Иако се основна заштита од пренапона у електроенергетским инсталацијама обезбеђује заштитама од пренапона типа 1 и типа 2, њихова ефикасност често није довољна да у потпуности заштити осетљиве електронске уређаје. Због тога тип 3 заштите, имплементиран директно у продужне каблове, има кључну улогу у очувању опреме прикључене у утичнице наших домова и канцеларија. На овај начин обезбеђује се додатни ниво сигурности и повећава поузданост свакодневне употребе електричне енергије.

На тај начин тип 3 заштита у продужним кабловима омогућава да се наша опрема заштити од делимичног оштећења или потпуног квара, без потребе за додатном уградњом, ангажовањем мајстора или скупом опремом као приликом инсталације заштите код типа 1 и типа 2. Довољно је користити продужни кабл са интегрисаном пренапонском заштитом, који се једноставно прикључује и спречава материјалне и физичке последице на наше уређаје изазване пренапонима.

1.1. Шта су пренапони

Пренапон је појава краткотрајног, али значајног повећања напона у електричној мрежи изнад предвиђене називне вредности. За дистрибутивне мреже у Европи, називни напон је 230 V (фаза-нула), са дозвољеним варирањем од ±10 %. Сваки напон који значајно и краткотрајно премаши овај опсег (нпр. досегне стотине или чак хиљаде волти) сматра се пренапоном, што сликовито приказује Слика 1.



Слика 1. Приказ мрежног напона са пренапонским импулсима

Кључне карактеристике пренапона су:

1. Временско трајање: Врло кратко, од микросекунди до неколико милисекунди.
2. Амплитуда: Може бити до неколико десетина хиљада волти.
3. Енергетски садржај: Варира од занемарљивих количина енергије (код комутационих пренапона) до веома високих (код атмосферских пражњења).
4. Облик таласа: Најчешће се описује стандардизованим импулсима (нпр. 8/20 μs за струју, 1.2/50 μs за напон).

2. КОМПОНЕНТЕ МОДУЛА ПРЕНАПОНСКЕ ЗАШТИТЕ У ПРОДУЖНОМ КАБЛУ И ЊИХОВА УЛОГА

Да би пренапонска заштита успешно обавила своју функцију и заштитила прикључене уређаје, од кључног је значаја правилно пројектовање модула. Ово подразумева не само избор одговарајућих компоненти већ и њихову оптималну међусобну везу и распоред унутар самог кућишта. Правилно димензионисан модул, где свака компонента има своју специфичну улогу, омогућава ефикасно скретање и апсорпцију енергије пренапона, чиме се обезбеђује поуздана и дуготрајна заштита, истовремено омогућавајући једноставну употребу – довољно је да се продужни кабл укључи, без потребе за додатном инсталацијом или интервенцијом стручњака.

Модул пренапонске заштите унутар продужног кабла фирме ELCO пројектован је као Тип 3 или Тип D (према IEC 61643-11), намењен за заштиту крајњих потрошача у нисконапонској мрежи 230 V AC.

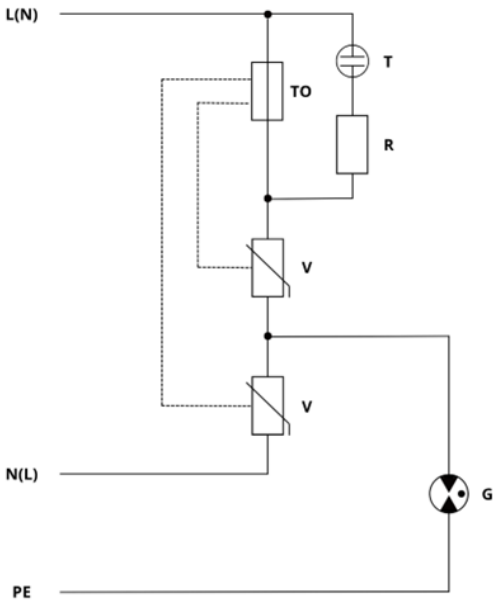
Компоненте које се налазе унутар модула пренапонске заштите (приказане на Слици 2) су:

1. TO (термички осигурач – A1-3A-F, 102 °C) – у серији са варистором. Он прекида струју ако се варистор прегреје (нпр. после више удара пренапона варистор почиње да проводи и да се загрева).
2. R (отпорник 100 kΩ) и T (тињалица – 65 V) – паралелно повезани. Тињалица је сигнална лампица – светли када је пренапонска заштита исправна и када кроз отпорник протиче мала струја. Отпорник ограничава струју кроз тињалицу.
3. G (гасни одводник пренапона – N81A350X) – везан између заједничке тачке варистора и PE. Он се пали на још вишем напону него варистори и обезбеђује додатну заштиту тако што га директно одводи у земљу.
4. V (варистори – S14K250) – два комада, везани између фазе и нуле и између фазе и PE. Они „се активирају“ тек кад напон пређе одређену границу (нпр. 275 V AC). Тада постају проводници и одводе вишак енергије.

Основна топологија комбинује брзо ограничење напона (V) и енергетски робустно премошћавање према заштитном потенцијалу (G), уз сигурносно искључење (термички осигурач) и визуелну индикацију стања (тињалица + отпорник).

Као што је раније наведено, модул пренапонске заштите располаже термичким осигурачем који у случају прегоревања заштитних елемената доводи до њиховог искључења, што се сигнализује паљењем сигналне тињалице.

Захваљујући хибридној конструкцији која комбинује варистор (V), гасни одводник (G) и термичку заштиту (T), њени параметри су побољшани у односу на стандардне Тип 3 уређаје. Ова комбинација омогућава бољу апсорпцију пренапонских импулса и ефикасно смањење пренапона, што у пракси пружа заштиту приближну Типу 2. Важно је нагласити да формална класификација остаје Тип 3, док се побољшане перформансе односе на стварну способност заштите прикључене електронике од пренапонских удара.



Слика 2. Шема повезивања модула пренапонске заштите у продужном каблу

3. ПРИНЦИП РАДА СКЛОПА

Компоненте уграђене у модул пренапонске заштите унутар продужног кабла фирме ELCO приказане су на Слици 3.1. Принцип рада овог модула састоји се из следећих стања:

3.1. Нормални услови (230 V AC):

- Варистори не проводе (само су „на чекању“).
- Гасни одводник не ради.
- Тињалица не светли.

3.2. Пренапонски удар:

- Ако напон порасте преко прага варистора, они постају проводници и одводе вишак енергије према нули и PE.
- Ако пренапон траје или је веома јак, укључује се и гасни одводник (велики удар), који директно одводи импулс у земљу.

3.3. Заштита од оштећења:

- Ако се варистор прегреје (после много интервенција или услед квара), термички осигурач прекида везу и спречава пожар.
- Тада се пали тињалица јер сада мали напон тече кроз њу и кориснику показује да пренапонска заштита више није активна тј. исправна.

4. ИСПИТНА МЕТОДА

Сва испитивања су извршена у складу са препорукама Међународне електротехничке комисије IEC 1312 и IEC 61643, односно IEC 60-1 и IEC 60-2, као и препорукама телекомуникационе уније ITU K20, ITU K21 и ITU K44.

Испитивање се ради на три узорка. Испитивање се састоји из следећих секвенци:

- 5 удара позитивног поларитета у прикључак L према прикључку N који је везан за PE
- 5 удара негативног поларитета у прикључак L према прикључку N који је везан за PE
- 5 удара позитивног поларитета у прикључак N према прикључку L који је везан за PE
- 5 удара негативног поларитета у прикључак N према прикључку L који је везан за PE
- 5 удара позитивног поларитета у прикључке L и N према прикључку PE, снимање на L
- 5 удара позитивног поларитета у прикључке L и N према прикључку PE, снимање на N
- 5 удара негативног поларитета у прикључке L и N према прикључку PE, снимање на L
- 5 удара негативног поларитета у прикључке L и N према прикључку PE, снимање на N

Због симетричности прикључака, потпуно је свеједно који ће прикључак бити декларисан као линијски (L), а који као неутрални (N).

Укупан број удара које трпи сваки узорак је 40. Са овим је тестирање извршено како за случај када имамо пренапон између фаза (L–N), тако и за случај када је пренапон према земљи (L–PE, N–PE, L+N–PE) за оба поларитета.

Овде ће бити приказана само табеле на којима су приказане максималне вредности параметара при мерењу на сваком од три узорка.

Табела 1. Максималне вредности параметара при мерењу 1. узорка

Umin(kV)	Umax(kV)	Sigma(kV)	Usred(kV)
0.996	1.140	0.032	1.074

Табела 2. Максималне вредности параметара при мерењу 2. узорка

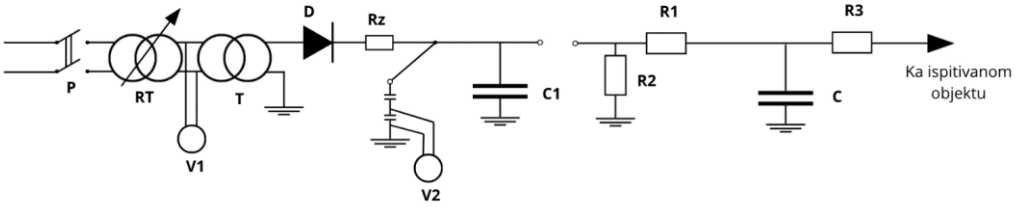
Umin(kV)	Umax(kV)	Sigma(kV)	Usred(kV)
0.955	1.118	0.042	1.041

Табела 3. Максималне вредности параметара при мерењу 3. узорка

Umin(kV)	Umax(kV)	Sigma(kV)	Usred(kV)
0.623	1.140	0.057	1.069

5. ШЕМА ВЕЗА И МЕРНА ОПРЕМА ЗА ДОБИЈАЊЕ УДАРНОГ НАПОНА СТАНДАРДНОГ ОБЛИКА 10/700 И РЕГИСТРАЦИЈУ ТАЛАСНИХ ОБЛИКА

Испитивање ударним напоном вршено је помоћу ударног напонског генератора сопствене производње за добијање атмосферских ударних таласа облика 10/700 μs. Принципијелна шема уређаја дата је на Слици 3:



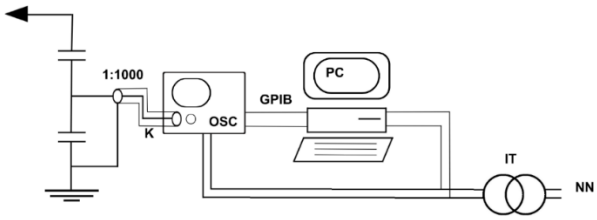
Слика 3. Заменска шема за испитивање ударним напоном облика таласа 10/700 μc

Ознаке на Слици 3 имају следеће значење:

- P - главни прекидач на командној табли уређаја
- V1 - показни волтметар који мери примарни напон ВН трансформатора
- D - високонпонска диода
- R2 - заштитни отпорник за ограничавање пуњења кондензатора
- C1 - батерија кондензатора називног напона 10kV, капацитета 20 μF
- S - високонпонска капацитивна сонда 1000:1 за прикључак мерног инструмента до 20kV Tektronix P 6015A
- V2 - волтметар за мерење једносмерног напона пуњења кондензатора C1
- R1 - отпорник промењиве отпорности за подешавање времена чела
- R2 - отпорник промењиве отпорности за подешавање времена зачеља
- C2 - кондензатор за подешавање чела и зачеља таласа
- R3 - излазни отпорник од 25 Ω, који се прикључује између ударног генератора и испитиваног објекта

Снимање напона је урађено на самом прикључку објекта који се испитује. Снимање се врши помоћу

високонпонске сонде за 20 kV, помоћу дигиталног осцилоскопа и рачунара.
Шема склопа за мерење дата је на Слици 4:



Слика 4. Шема Система за регистрацију ударних напона

6. ЗАКЉУЧАК

Резултати спроведених испитивања показују да је испитивани модул за пренапонску заштиту успешно ограничио стандардизовани атмосферски импулсни напон (4 kV, 10/700 μs) у оба испитивана мода:

- у случају пренапона између фаза (L–N) на максималну вредност од 1180 V,
- у случају пренапона према земљи (L–PE) на максималну вредност од 787 V.

Ови резултати потврђују високу ефикасност и поузданост заштитног модула из следећих разлога:

1. Прекорачене декларисане перформансе:

Измерене вредности од 1180 V (пренапон између фаза) и 787 V (пренапон према земљи) значајно су ниже од декларисаних граничних вредности од 1300 V и 1000 V. То показује да је модул пројектован са великом сигурносном маргином и резервом у енергетском капацитету, што указује на висок квалитет коришћених компоненти (варистора и гасних одводника) и њихову правилну координацију.

2. Ефикасност у заштити осетљиве електронике:

Ограничење пренапона између фаза на 1180 V обезбеђује да улазни елементи напајања са импулсним претварањима (SMPS), који су у складу са стандардом IEC 61000-4-5 пројектовани да издрже краткотрајне импулсне напоне до око 1500 V, остану у безбедном режиму рада без појаве оштећења или деградације. Иако је ова вредност изнад њиховог номиналног радног опсега (600–1000 V), кратко трајање импулса и ограничена енергија обезбеђују да напон на улазу уређаја остане у границама отпорности компоненти. Истовремено, ограничење пренапона према земљи на 787 V значајно смањује ризик од појаве високог потенцијала између фазе и кућишта уређаја, чиме се спречавају пробоји изолације, оштећења комуникационих интерфејса и евентуалне електричне сметње или удари..

3. Отпорност на екстремне услове:

Испитивања су спроведена импулсом 10/700 μs, који садржи већу енергију од стандардног импулса 8/20 μs, чиме су заштитне компоненте изложене значајно већем оптерећењу. Упркос томе, нису забележена никаква оштећења ни деградација функције модула, што потврђује његову робусност и дуготрајну стабилност у реалним условима рада.

7. ЛИТЕРАТУРА

[1] IEC 61643: Low-voltage surge protective devices – Part 11: Surge protective devices connected to low-voltage power systems – Requirements and test methods.

[2] IEC 61643: Low-voltage surge protective devices – Part 12: Surge protective devices connected to low-voltage power distribution systems – Selection and application principles.

[3] IEC 61643: Low-voltage surge protective devices – Part 1: Surge protective devices connected to low-voltage power distribution systems – Requirements and tests.

[4] IEC 61643: Components for low-voltage surge protection - Part 331: Specification for metal oxide varistors (MOV).

[5] ITU-T Recommendation K.20: Resistibility of telecommunication equipment installed in a telecommunication centre to overvoltages and overcurrents.

[6] ITU-T Recommendation K.21: Resistibility of telecommunication equipment installed in customer premises to overvoltages and overcurrents.

[7] ITU-T Recommendation K.44: Resistibility tests for telecommunication equipment exposed to overvoltages and overcurrents - Basic Recommendation.

[8] IEC 60060: High-voltage test techniques, Part 1: General definitions and test requiremenets 11/1989.

[9] IEC 60060: High-voltage test techniques, Part 2: High voltage test techniques – Part 2: Measuring systems, 11/1994

Кратка биографија:

Милан Бањац рођен је 1994. године у Руми. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Енергетска електроника и електричне машине одбранио је 2025. год.

Дејан Јеркан је ванредни професор на Факултету техничких наука у Новом Саду, на катедри за Енергетску електронику и претвараче. Област интересовања су му моделовање и дијагностика електричних машина као и метода коначних елемената.

Контакт: banjacm94@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.3
ДОИ: https://doi.org/10.24867/35BE02Ilic

Једно решење претраге релационих база података уз помоћ Graph RAG система

One Approach to Relational Database Search Using a Graph RAG System

Игор Илић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Релационе базе података су и даље ослонац пословних информационих система, али рад са њима захтева познавање SQL-а. Иако су велики језички модели значајно приближили могућност постављања упита на природном језику, претварање тих питања у поуздане и прецизне одговоре над структурираним подацима остаје изазов. Чисто текстуални RAG (енгл. Retrieval Augmented Generation) често занемарује шеме, кључеве и ограничења, док је директно превођење природно језичког упита у SQL уз помоћ LLM-ова крхко и тешко провељиво. Овај рад представља GraphRAG (енгл. Graph Retrieval Augmented Generation) решење проблема, којим се омогућава природним језиком приступ релационим подацима.*

Кључне речи: *Релациона база података, граф база података, векторска база података, велики језични модели, вештачка интелигенција*

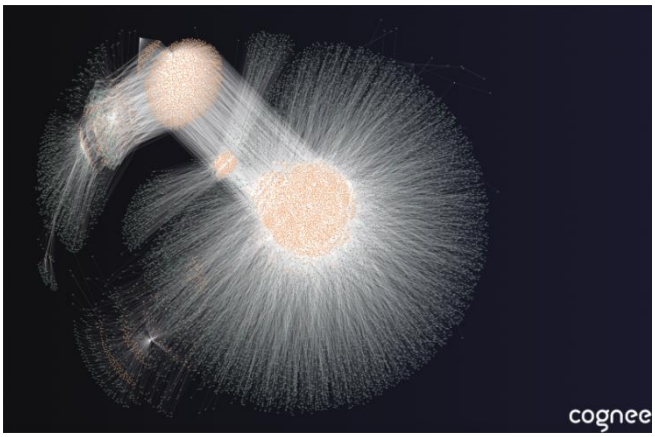
Abstract – *Relational databases remain the backbone of business information systems, but working with them requires knowledge of SQL. While Large Language Models have significantly brought closer the possibility of querying using natural language, turning those questions into reliable and precise answers from relational data remains a challenge. Pure textual RAG (Retrieval Augmented Generation) often ignores schemas, keys, and constraints, while the direct translation of a natural language query into SQL using LLMs is fragile and difficult to verify. This paper presents the GraphRAG (Graph Retrieval Augmented Generation) solution to the problem, which enables natural language access to relational data.*

Keywords: *Relational Databases, Graph Databases, Vector Databases, Large Language Models, Artificial Intelligence*

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

1. УВОД

Релационе базе података су и даље ослонац пословних информационих система, али рад са њима захтева познавање SQL-а, шема и логике спајања табела. Иако су велики језички модели (енгл. *Large Language Models, LLM*) значајно приближили могућност постављања упита на природном језику, претварање тих питања у поуздане одговоре над структурираним подацима остаје изазов. Чисто текстуални RAG (енгл. *Retrieval Augmented Generation*) занемарује шеме, кључеве и ограничења, док је директно превођење природно језичког упита у SQL уз помоћ LLM-ова крхко. Мигрирањем релационих података у GraphRAG (енгл. *Graph Retrieval Augmented Generation*) систем, омогућиће се природно језичким приступ релационим подацима. GraphRAG систем спаја граф претрагу са векторском семантичком претрагом како би се премостио јаз између природног језика и релационе структуре. Предложени дизајн избегава крхкост директне генерације SQL синтаксе тако што утемељује LLM у структуираним доказима извученим из изворне SQL базе. На слици 1. можемо да видимо резултат миграције релационе базе у граф базу за GraphRAG системе.



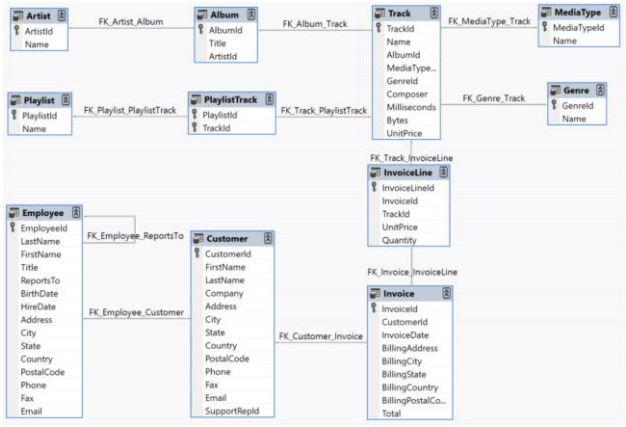
Слика 1. Граф визуализација мигриране релационе базе података у GraphRAG систем

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1. Релационе базе података

Релационе базе података заснивају се на математичким темељима теорије скупова и предикатне логике првог

реда. Концепт је формализовао Едгар Ф. Код 1970. године [1]. Користе се за управљање организованим подацима у различитим секторима, као што су финансије (банкарство, управљање рачунима), електронска трговина (наруџбине, купци, производи), здравство (медицински картони пацијената), логистика (праћење пошиљки) и управљање залихама (праћење нивоа залиха). На слици 2. можемо видети пример шеме података музичке радње у једној релационој бази, иста ова база података ће бити коришћена за верификацију резултата касније у раду.



Слика 2. Пример шеме релационе базе података

2.2. Граф базе података

Граф базе података моделују податке као чворове (ентитете) и везе (релације) са својствима, у оквиру *property grap*h парадигме. *Cypher* је декларативни језик за подударање патерна (енгл. *pattern matching*) и модификацију таквих графова. Предности укључују експлицитно моделирање веза, интуитивне упите над путањама и добру читљивост. Граф базе су природна подлога за GraphRAG системе јер експлицитно моделују везе, омогућавају ефикасне *multi-hop* претраге и нуде алгоритме који подижу квалитет и објашњивост контекста.

2.3. Векторске базе података

Дигитална трансформација и експоненцијални раст неструктурираних података — попут текста, слика, аудио и видео записа — поставили су нове изазове пред традиционалне системе за управљање подацима. Конвенционалне релационе базе података показале су се недовољним за руковање сложеним семантичким претрагама које захтевају савремене апликације вештачке интелигенције. Векторска база података је специјализована врста базе података дизајнирана за индексирање и управљање високодимензионалним векторским уграђивањима (енгл. *embeddings*) [2]. За разлику од традиционалних база података које се ослањају на прецизно подударање вредности (нпр. SQL упити *WHERE*), векторске базе података омогућавају претрагу сличности (енгл. *Similarity search*). Оне проналазе податке који су семантички слични упиту, а не нужно они који садрже идентичне кључне речи. Срце сваке векторске базе су векторска уграђивања. То су нумерички прикази (вектори) који претварају сложене, неструктуриране податке у низове бројева у високодимензионалном простору. Ови бројеви хватају семантичко значење и односе између података.

2.4. Велики језички модели

Вештачка интелигенција постала је катализатор револуционарних промена у начину на који машине разумеју и генеришу људски језик. Велики језички модели (енгл. *Large Language Models, LLM*) представљају врхунац деценија истраживања у области обраде природног језика (енгл. *Natural Language Processing*, *NLP*) и дубоког учења. Велики језички модели су врста неуронских мрежа темељених на архитектури трансформера [3], обучених на масовним количинама текстуалних података, који могу разумети, генерисати и манипулисати људским језиком на софистициран начин.

2.5. *Retrieval Augmented Generation*

Retrieval-Augmented Generation је техника за побољшање тачности и поузданости генеративних AI модела помоћу информација добијених из специфичних и релевантних извора података. Ова спољашња база знања може укључивати интерне документе компаније, истраживачке радове или било који други корпус структурираних или неструктурираних података. Основна идеја је да се генеративне способности LLM-а прошире механизмом за проналажење информација, омогућавајући моделу да даје одговоре који су не само кохерентни, већ и утемељени на проверљивим, ажурним подацима. Ово се постиже дељењем докумената у сегменте, векторски уграђивање тих сегмената па потом у времену претраге издвајањем најрелевантнијих сегмената из свих докумената.

2.6. Graph Retrieval Augmented Generation

Из недостатака адекватног повезивања информација из различитих сегмената докумената у RAG системима нови приступ настаје под именом GraphRAG [4]. GraphRAG је свеобухватни назив за различите системе који интегришу граф базе података за свој систем меморије за велике језичне моделе. Овај рад је заснован на *Cognee* GraphRAG пројекту отвореног кода. GraphRAG решава изазове адекватног повезивања информација тако што користи графове знања као основу за претрагу информација. Графови знања су структурирани као мреже ентитета (чворови) и њихових међусобних релација (везе), што GraphRAG-у омогућава семантичко претраживање које прати логичке везе, уместо да се ослања искључиво на статичку векторску сличност.

3. КОНЦЕПТ РЕШЕЊА

GraphRAG системи нису направљени са нативном подршком за релационе базе, намењени су фундаментално за обраду и процесуирање текстуалних података попут докумената. Ти документи су анализирани и процесуирани уз помоћ LLM-ова за релативне концепте и ентитете. Те информације се чувају уз сегменте и потом користе као контекст за

будуће упите од корисника (енгл. *User Query*). У случају релационих база, анализа и процесуирање података уз помоћ LLM-ова нису ни могући ни потребни јер је свака информација нужна и већ сведена на сам минимум корисности. Направљен је систем директног мапирања података из релационе базе у податке који GraphRAG системи могу и знају да користе. Мигрирани подаци су сачувани у релационој бази у граф бази података, па су потом ови подаци векторски уграђени (енгл. *Embedding*) и прикладно сачувани у векторску базу података. Овакав систем је направљен унутар *Cognee* GraphRAG система као проточна структура (енгл. *custom pipeline*) која служи само за мигрирање релационих база у *Cognee* GraphRAG систем [5].

3.1. Мапирање ентитета

TableType, *TableRow* и *ColumnValue* чине троделну, онтолошки утемељену репрезентацију релационих података у граф-парадигми. Они наслеђују метамодел *DataPoint* који је дефинисан у *Cognee* систему, метамодел *DataPoint* омогућава директно мапирање елемената у графу са елементима у векторској бази и омогућава дефинисање тачно шта од информација треба бити прослеђено LLM-у током упита корисника. Сваки наследник метамодела *DataPoint*-а треба да дефинише адекватне вредности у пољима:

1. *metadata* – У овом пољу се спецификује на основу које вредности треба радити векторско уграђивање, очекиван улаз за вредност је конкретно поље из модела (на пример, *description* поље)
2. *description* – У овом пољу се спецификује које текстуалне информације треба приложити LLM-у приликом давања одговора на упит корисника, када је конкретни чвор део релевантних резултата претраге.

Наслеђивањем *DataPoint*-а на овај начин нам је омогућено коришћење *Cognee* метода за креирање GraphRAG репрезентације, односно прављењем сопствене онтолошке репрезентације података и коришћењем *Cognee* система претраге података. Када је мапирање завршено додани су новокреирани чворови и везе у саму граф базу података. *Cognee* систем има своју апстракцију за комуникацију са граф и векторским базама података, има јасно дефинисане интерфејсе који су исти за све различите моделе који су наследили *DataPoint* и за све подржане типове база података.

Систем врши пресликавања ентитета описана у Табели 1. Систем додатно спаја ентитете у везе у графу на основу начина дефинисаних у Табели 2.

Табела 1.Мапирање релационих модела на GraphRAG моделе

Релациони модел	**GraphRAG модел**
Табела	*TableType* чвор
Ред у табели	*TableRow* чвор
Примарни кључ	Јединствени идентификатор *TableRow* чвора
Вредност колумне у реду у табели	*ColumnValue* чвор

Табела 2.Мапирање веза у GraphRAG систему на основу релационих односа

Ентитет извор	**Ентитет циљ**	**Име веза**	**Опис везе**
TableRow	*TableType*	*is_part_of*	Припадност реда његовој основној табели.
ColumnValue	*TableRow*	Име колоне	Вредност у колони конкретних редова у табели.
TableRow	*TableRow*	Име колоне страног кључа	Везу између редова кроз страни кључ.

3.2. Векторско уграђивање

Након мапирања података из релационе базе у чворове и везе у граф базу ти подаци су векторски уграђени и сачувани у векторску базу података. Ово је једноставно урађено у *Cognee* систему коришћењем већ постојећих метода:

- *index_data_points* – ради мапирање чворова у векторску базу података.
- *index_graph_edges* – ради мапирање веза у векторску базу података.

Позивањем ових функција добијене су четири табеле у којима се чувају векторски уграђене вредности. Конкретно:

- *ColumnValue_properties*
- *TableRow_properties*
- *TableType_name*
- *EdgeType_relationship_name*

3.3. Претрага по упиту корисника

Cognee методе претраге аутоматски раде са мигрираном релационом базом. Сама претрага је тренутно написана тако да ради пројекцију граф и векторске базе у RAM меморију, одатле се прави листа свих триплета (два чвора и веза између њих) и сваки триплет се бодује за релевантност на основу следеће формуле:

$$R = d_{č1} + d_{č2} + d_v$$

где је: $d_{č1}$- векторска косинусна удаљеност чвора 1 од упита корисника, $d_{č2}$ – векторска косинусна удаљеност чвора 2 од упита корисника, а d_v- векторска косинусна удаљеност везе од упита корисника.

Векторска косинусна удаљеност је мера сличности између два вектора у вишедимензионалном простору. Она мери косинус угла између та два вектора. Од листе свих триплета узима се *top_k* број триплета са најмањим резултатом укупне удаљености од корисниковог упита. Листа релевантних триплета се претвара у текст. Ово је постигнутно тако што *DataPoint* моделу у себи имају поље *description* које се директно претвара у опис чвора за LLM, у овом пољу је специфицирано тачно шта представља који податак

и шта се у њему садржи. Ова вредност је динамички генерисања током процеса мигрирања. На пример током мигрирања *ColumnValue* чвора опис је подешен да буде:
"column from relational database table={table name}. Column name={key} and value={value}. This column has the following ID: {column_node_id}"
Из ових информација на корисниково питање LLM успева успешно да разазна вишеслојно порекло информација и њихово значење и релације и да да адекватан одговор базиран само на приложеном контексту.

4. РЕЗУЛТАТИ

У наставку је описан резултат миграције релационе базе података једне фиктивне музичке радње [6]. Шема ове релационе базе се може видети на слици 2. На слици 1 се види визуализација граф базе настале овом миграцијом. Граф база настала од ове миграције садржи 19437 чворова и 57818 веза. Миграција је трајала 27 минута и 5 секунди. Величина меморије датотеке *Kuzu* граф базе настале миграцијом је 63MB. Величина датотеке *LanceDB* векторске базе настале од ове миграције је 246,6MB. Величина иницијалне *SQLite* релационе базе за мигрирање је 1.1MB. После мигрирања података могуће је радити природно језичке упите везане за податке који се налазе у релационој бази података**.** Питања попут: "Које су фактуре од муштерије *Leonie Köhler*", "Које су све песме део плејлисте *Heavy Metal Classic*", "Које су све муштерије из Прага?" су све примери питања која се успешно и конзистентно одговарају. Приликом верификације резултата над мигрираном релационом базом музичке продавнице постављено је питање да се пронађу фактуре везане за сваку муштерију појединачно и резултати су верификовани поређењем резултата SQL претраге за исте информације, а за сваку муштерију су успешно пронађене његове фактуре без халуцинација и изузетака. Докле год тачан одговор на питање не захтева више резултата него што је доступно контексту дефинисаним *top_k* бројем релевантних триплета. Са друге стране одговори на комплекснија питања попут "Који су све радници исте националности као муштерија Leonie Köhler" *Cognee* претрага не може да одговори јер питање о националности је потребно експлицитно да се постави. Није могуће квалитетне имплицитне претраге правити због лимитација LLM контекста, односно јер неће довољно релевантних информација бити нађено за такво имплицитно питање.

5. ЗАКЉУЧАК

На примеру јавно доступне базе музичке радње демонстриран је комплетан процес: од дефинисања онтологије и процеса миграције, преко изградње графа знања и векторског индекса, до практичне употребе система за претрагу и одговор на питања. Исправност имплементације верификована је комбинацијом аутоматизованих тестова и ручног увида.

Практичне импликације су двоструке: миграцијом релационе базе у GraphRAG систем омогућава се интуитивнији приступ подацима из релационих извора путем природног језика, али уз реалне трошкове у погледу простора и времена миграције и претраге. Тренутно се ти трошкови чини превише великим за конзистентно и лако коришћење на комерцијалном нивоу и нивоу свакодневног коришћења, али имплементација може бити интересантна у истраживачком пољу са аспекта омогућавања природно језичке анализе података који се касније морају верификовати да се потврди тачност резултата.

6. ЛИТЕРАТУРА

[1] Codd, E. F. (1970) "A Relational Model of Data for Large Shared Data Banks." Communications of the ACM 13 (6): pp. 377-387. https://doi.org/10.1145/362384.362685.

[2] Grohe, M. (2020). "word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data", in *proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, pp. 1-16.

[3] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. (2017) "Attention is all you need", Advances in neural information processing systems 30.

[4] Jonathan Larson, Steven Truitt (2024). "GraphRAG: Unlocking LLM discovery on narrative private data", https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/ (приступано 10.11.2025.)

[5] Cognee Open Source project. https://github.com/topoteretes/cognee/tree/main (приступано 10.11.2025.)

[6] Chinook (n.d.) Music store relational database. https://github.com/lerocha/chinook-database/tree/master (приступано 10.11.2025.)

Кратка биографија:

Игор Илић, рођен у Новом Саду, 04.11.1996. године.

Основне студије завршио 2019. године, смер Рачунарство и аутоматика, на Факултету техничких наука, Нови Сад.

Контакт: igorilic03@yahoo.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 621.391
ДОИ: https://doi.org/10.24867/35BE03Martinov

Анализа расподеле примљене снаге VLC система у затвореном простору

Analysis of Received Power Distribution in Indoor VLC Systems

Драгана Мартинов, *Факултет техничких наука, Нови Сад*

Студијски програм – ЕНЕРГЕТИКА, ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ

Кратак садржај – *У оквиру рада, разматрано је испитивање расподеле примљене снаге пријемника при видљивој светлосној комуникацији* (енг. *Visible Light Communication - VLC*), *у просторији при промени параметара попут максималног угла видног поља оптичког пријемника, полуугла предајника, висине просторије и броја LED лампи.*

Кључне речи (три до пет): *VLC, LED, FOV*

Abstract – *The paper examined the distribution of the received receiver power within Visible Light Communication* (*VLC*), *in a room when changing parameters such as the maximum field of view of the optical receiver, half – amgle of the transmitter, the height of the room, and the number of LED lamps.*

Keywords: (three to five): *VLC, LED, FOV*

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

1. УВОД

У оквиру дистрибуираног осветљења у затвореним просторима, оптичка бежична комуникација представља иновативан правац развоја технологија које омогућавају ефикасно повезивање уређаја, посебно у контексту интернета ствари (IoT).

Једна од кључних предности ове технологије је могућност коришћења постојеће инфраструктуре осветљења, односно диода које емитују светлост (LED), као канала за комуникацију. Тиме се значајно смањују трошкови имплементације и олакшава примена у паметним кућама, канцеларијама, па чак и индустријским постројењима. Ова технологија додатно доприноси смањењу загушења радио-фреквентног (RF) спектра, што је од посебног значаја с обзиром на све већи број повезаних IoT уређаја [1].

Видљива светлосна комуникација (VLC), као подскуп оптичке бежичне технологије (OWC), користи диоде које емитују светлост у опсегу видљиве светлости. Захваљујући напретку у развоју LED велике снаге, ова технологија нуди енергетски ефикасну, еколошки прихватљиву алтернативу традиционалним RF технологијама. Тиме се омогућава развој OWC система који функционишу унутар већ постојећих осветљајних мрежа.

Имплементација паметног осветљења заснованог на технологији комуникације путем видљиве светлости не само да обезбеђује ефикасну комуникацију и осветљење, већ омогућава и прецизну контролу потрошње енергије. Осветљеност у просторији зависи од низа фактора, укључујући удаљеност од извора светлости, угао емисије LED светала и конфигурацију мреже. Оптимизација ових параметара омогућава да се осветљеност усклади са положајем корисника у простору. Како се позиције корисника динамички мењају, систем мора да прилагоди расподелу светла у реалном времену, чиме се додатно повећава енергетска ефикасност [2].

У овом раду приказана је симулација испитивања расподеле примљене снаге на страни пријемника у просторији у зависности од промене параметара попут максималног угла видног поља оптичког пријемника (FoV), полуугла предајника, висине просторије и броја LED лампи.

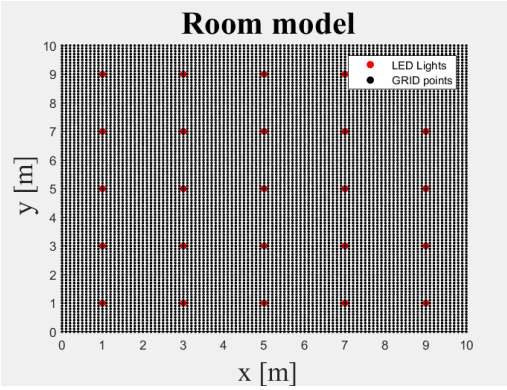
2. ИСПИТИВАЊЕ ПРИЈЕМА СНАГЕ КОД КОРИСНИКА У ПРОСТОРИЈИ

Решења заснована на LED осветљењу доносе нове перспективе систему локализације у затвореном простору. Унапред инсталирани LED уређаји за осветљење могу обезбедити инфраструктуру за систем позиционирања у затвореном простору. Такође, информације о позицији могу се преносити модулацијом емитоване светлости. Скоро свака техника сигнала/сензора може се трансформисати у систем који предвиђа информације о положају комуникационог уређаја. Ови системи су засновани на мерењу неких параметара окружења или комуникационог канала [3].

Како VLC укључује обоје и комуникацију и осветљење, значајно је истражити расподелу примљене снаге LED предајника. У наставку је дат приказ модела просторије као и резултата симулације при промени параметара попут броја LED лампи, висине просторије и максималног угла видног поља оптичког пријемника. Симулација је вршена на основу извора светлости и пропагационих параметара, базирајући се на изразу који одређује пријемну снагу

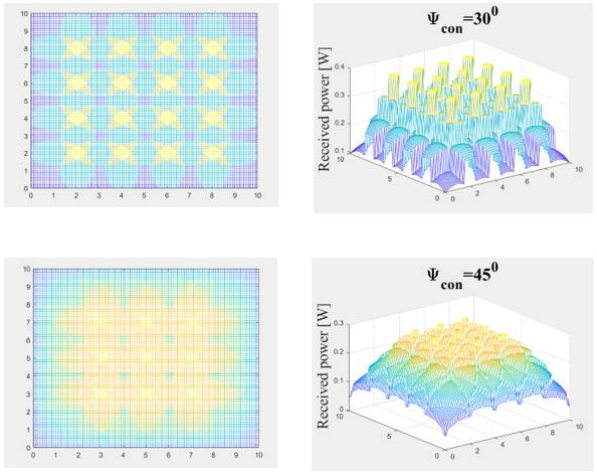
$$P_r = P_t \cdot \frac{(m_l+1)}{2\pi d^2} \cos^{m_l}(\emptyset) \cdot T_s(\psi) \cdot g(\psi) \cdot \cos(\psi), \quad 0 \le \psi \le \psi_{con} \qquad (1)$$

где је P_t предајна снага, $\emptyset$ угао осветљења у односу на осу нормалну на површину предајника, m_l је ред Ламбертовог зрачења, ψ угао упада у односу на осу нормалну на површину пријемника, $T_s(\psi)$ је пропусност филтра, $g(\psi)$ и ψ_{con} су добитак концентратора и FoV, респективно, а d је удаљеност између LED предајника и површине детектора.



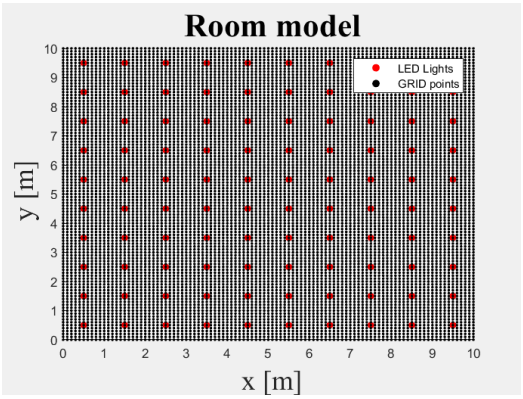
Слика 1. Модел просторије (10x10x3, број LED лампи=5).

На слици 1. приказан је модел просторије једнаке дужине и ширине од 10 m, док висина износи 3 m. Црвене тачке представљају позиције LED лампи, а црне тачке потенцијалне позиције корисника. Број LED лампи је 5.
На слици 2. дат је приказ резултата симулације за максимални угао видног поља оптичког пријемника од 30° и 45°.



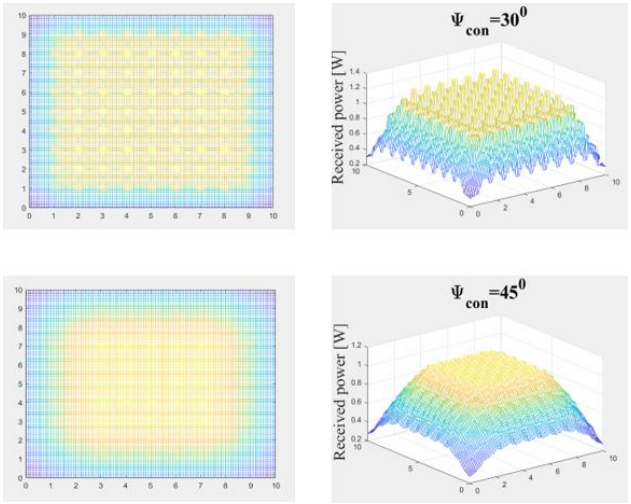
Слика 2. Понашање примљене снаге при промени максималног угла видног поља оптичког пријемника за модел просторије (10x10x3, број LED лампи=5).

Приликом повећања максималног угла видног поља оптичког пријемника, уочено је да примљена снага опада, али да је већа покривеност. Више тачака упада у FoV и вредност примљене снаге у тим тачкама више није нула.
На слици 3. приказан је модел просторије једнаке дужине и ширине од 10 m, док висина износи 3m. Црвене тачке представљају позиције LED лампи, а црне тачке потенцијалне позиције корисника. Број LED лампи је 10.



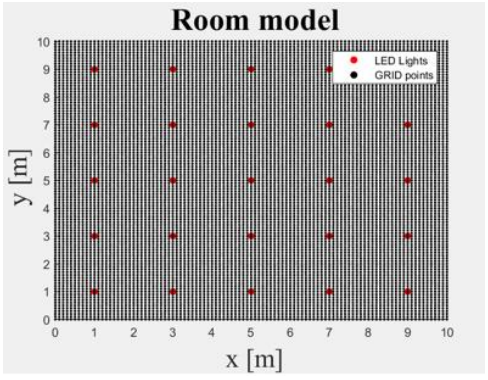
Слика 3. Модел просторије (10x10x3, број LED лампи=10).

На слици 4. дат је приказ резултата симулације за максимални угао видног поља оптичког пријемника од 30° и 45°.



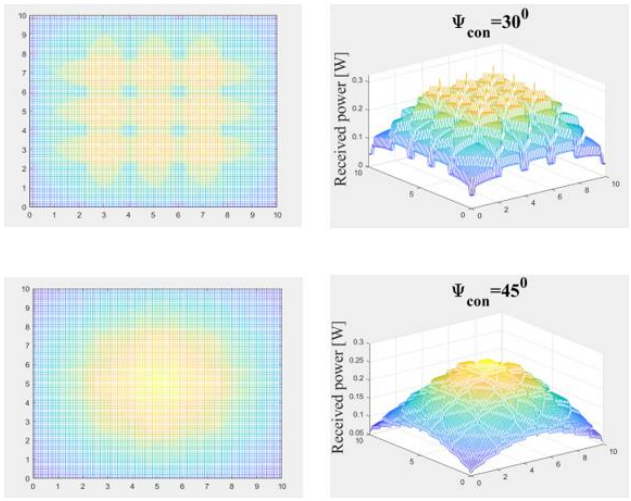
Слика 4. Понашање примљене снаге при промени максималног угла видног поља оптичког пријемника за модел просторије (10x10x3, број LED лампи=10).

Уколико би се извршило поређење између добијених резултата са слике 2. и 4. приметно је да је повећана вредност примљене снаге, а узрок тога је повећање броја LED лампи. Као и на слици 2. и овде се уочава да са повећањем FoV интензитет примљене снаге по тачки опада, али да је захваћено више тачака.



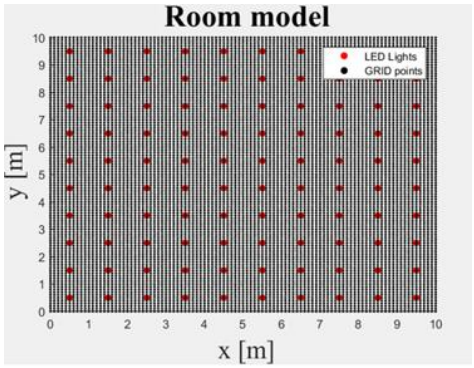
Слика 5. Модел просторије (10x10x5, број LED лампи=5).

На слици 5. приказан је модел просторије једнаке дужине и ширине од 10 m, док висина износи 5 m. Црвене тачке представљају позиције LED лампи, а црне тачке потенцијалне позиције корисника. Број LED лампи је 5.
На слици 6. дат је приказ резултата симулације за максимални угао видног поља оптичког пријемника од 30° и 45°.



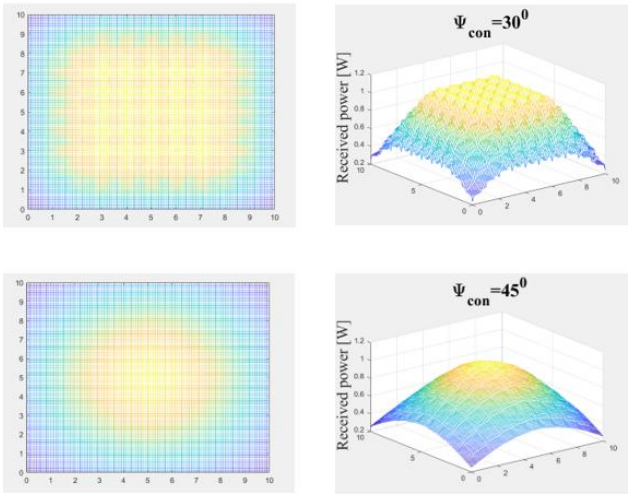
Слика 6. Понашање примљене снаге при промени максималног угла видног поља оптичког пријемника за модел просторије (10x10x5, број LED лампи=5).

На слици 6. уочава се да је повећање висине плафона просторије узроковало да примљена снага има нижу вредност. Промена FoV је са повећањем утицала на смањење интензитета примљене снаге, али и на већу покривеност тачака.
На слици 7. приказан је модел просторије једнаке дужине и ширине од 10 m, док висина износи 5 m. Црвене тачке представљају позиције *LED* лампи, а црне тачке потенцијалне позиције корисника. Број LED лампи је 10.



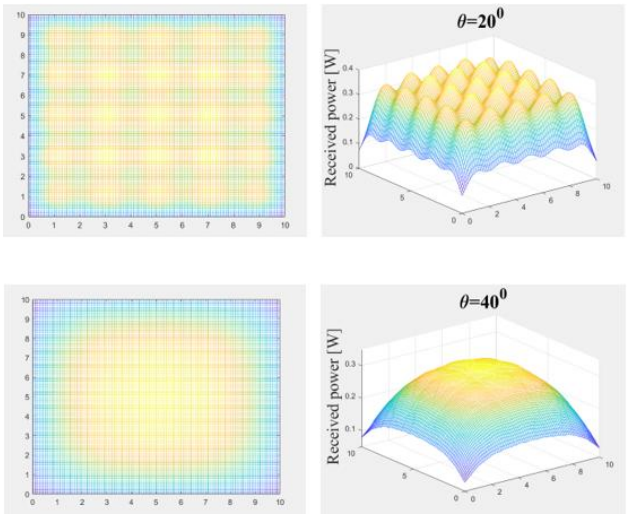
Слика 7. Модел просторије (10x10x5, број LED лампи=10).

На слици 8. дат је приказ резултата симулације за максимални угао видног поља оптичког пријемника од 30° и 45°.

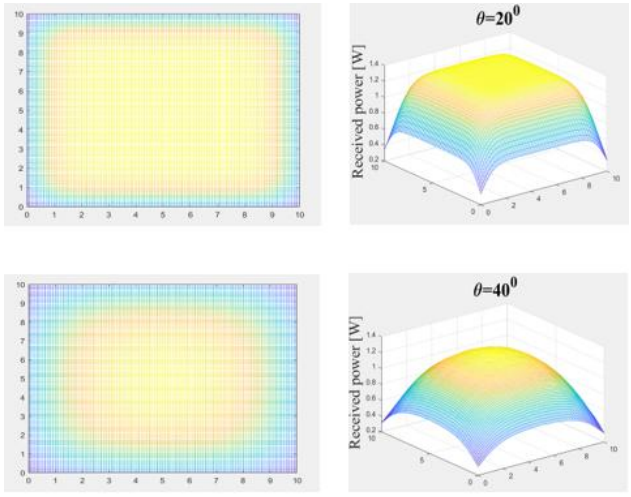


Слика 8. Понашање примљене снаге при промени максималног угла видног поља оптичког пријемника за модел просторије (10x10x5, број LED лампи=10).

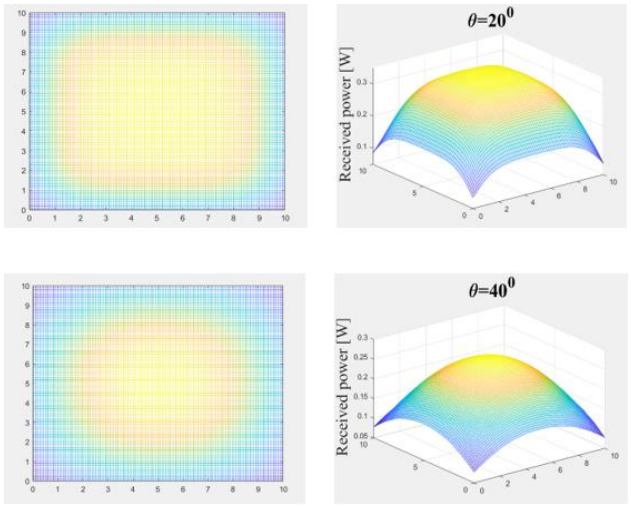
У овом случају запажа се да је повећање броја *LED* лампи у просторији узроковало веће вредности примљене снаге, али да је утицај повећања висине просторије и даље присутан, уколико би се поредили добијени резултати приказани на слици 6. и 8.
Горе наведени резултати симулације испитују понашање примљене снаге у зависности од мењања FoV.
У наредним случајевима приказан је резултат симулације када је FoV=60°, при промени параметара попут висине, броја LED лампи и полуугла предајника. За исте моделе просторије, такође се симулирало за фиксан FoV=60°, за полуугао предајника 20° и 40°. Сходно томе, у наставку је дат преглед овог случаја, на сликама 9, 10, 11 и 12, респективно. Приметно је да мање мање вредности полуугла предајника, узрокују израженије вредности примљене снаге на датим позицијама LED лампи, због ужег светлосног снопа.



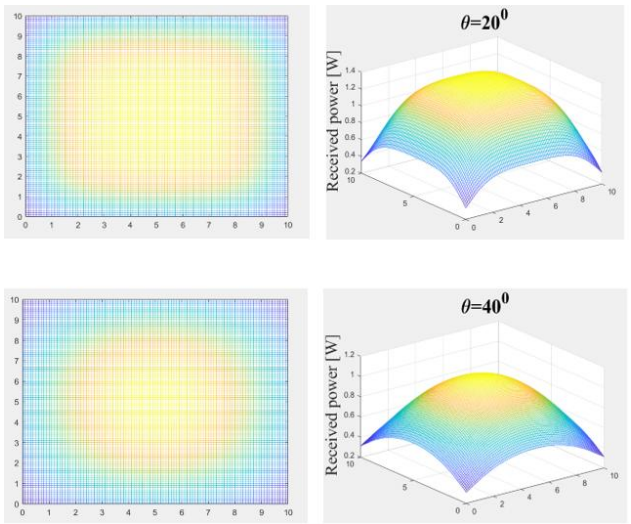
Слика 9. Понашање примљене снаге при промени полуугла предајника за модел просторије (10x10x3, број LED лампи=5).



Слика 10. Понашање примљене снаге при промени полуугла предајника за модел просторије (10x10x3, број LED лампи=10).



Слика 11. Понашање примљене снаге при промени полуугла предајника за модел просторије (10x10x5, број LED лампи=5).



Слика 12. Понашање примљене снаге при промени полуугла предајника за модел просторије (10x10x5, број LED лампи=10).

3. ЗАКЉУЧАК

Анализом резултата, уочено је да повећање висине плафона смањује укупну примљену снагу на страни пријемника. Покушај компензације за већу висину огледа се у повећању броја LED лампи. Уколико би се повећавао број LED лампи, растао би ниво интерференције односно било би више преклапања и шума. У зависности да ли је циљ да се постигне већа покривеност или већа примљена снага у одређеним тачкама, мењају се и вредности максималног угла видног поља оптичког пријемника. Такође, осим испитивања случајева где је мењан FoV, одрађена је и анализа уколико је FoV фиксан, а промењене су вредности полуугла предајника.

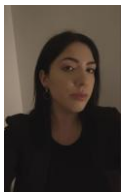
4. ЛИТЕРАТУРА

[1] Hamza, Abdelbaset; Tripp, Tyler, Optical Wireless Communication for the Internet of Things: Advances, Challenges, and Opportunities. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.12659789.v2, 2020.

[2] Abhishek Gupta, Dr. Xavier Fernando, Exploring Secure Visible Light Communication in Next-generation (6G) Internet-of-Things, International Wireless Communications and Mobile Computing (IWCMC), 2021.

[3] Optical Wireless Communications-System and Channel Modelling with MATLAB, Z. Ghassemlooy, W. Popoola, S. Rajbhandari, 2019.

Кратка биографија:



Драгана Мартинов рођена је у Новом Саду 1998. године. Дипломски рад на Факултету техничких наука из области електротехнике и рачунарства одбранила је 2022. године.

Контакт: draga.martinov@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.85
ДОИ: https://doi.org/10.24867/35BE05Mirkovic

Развој подршке за парсирање за SeamlessMDD развојни оквир уз ослонац на Web API

Development of parsing support for SeamlessMDD framework based on Web API

Драган Мирковић, *Факултет техничких наука, Нови Сад*

Студијски програм - СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *Овај рад представља проширење SeamlessMDD оквира за развој подршком за парсирање путем удаљених Web API сервиса. Циљ је био избећи јаку зависност од библиотека за парсирање, уколико би се оне укључиле у језгро алата. Предложени приступ дефинише апстрактни интерфејс IApiParser и двије имплементације, тако да клијент изражава намјеру у доменским појмовима, а конкретно извршење обавља удаљени сервис за парсирање. Рјешење смањује зависност, јасно раздваја одговорности и омогућава лаке замјене технологија, што је показано кроз двије имплементације, AdvancedHTMLParser и lxml, које се интегришу без измјена у клијентском коду. На тај начин SeamlessMDD добија чистију архитектуру и подршку за проширивост новим парсерима.*

Кључне речи (три до пет): *развој базиран на моделима (MDD), SeamlessMDD, парсирање, Web API*

Abstract – *This paper presents an extension to the SeamlessMDD framework that introduces support for parsing via remote Web API services. Our goal was to minimize dependence on parsing libraries that would otherwise be included in the tool's core. The proposed approach defines an abstract interface (IApiParser) along with two implementations, allowing the client to express intent in domain terms while a remote parsing service carries out the execution. This solution reduces coupling, clearly separates responsibilities, and facilitates easy technology swaps, as demonstrated through two implementations, with AdvancedHTMLParser and lxml, that integrate without requiring changes to the client code. In this way, SeamlessMDD achieves a cleaner architecture and supports extensibility with new parsers.*

Keywords: (three to five): *Model-Driven Development (MDD), SeamlessMDD, parsing, Web API*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је била др Гордана Милосављевић, ред. проф.

1. УВОД

Развој вођен моделима (Model Driven Development – MDD) има за циљ да тежиште рада пребаци са имплементационих детаља на доменске концепте, тако што модел постаје извориште из којег се аутоматски изводе код, конфигурације и други артефакти. Тај приступ подразумијева јасно дефинисане трансформације улазних на излазне елементе, како би се промјене у моделу имплементирале као минимални, контролисани захвати у излазном коду, без губитка ручно писаног кода и уз предвидљиво понашање током еволуције система **Error! Reference source not found.**. SeamlessMDD је алат за MDD развој који проблем интеграције генерисаног и ручно писаног кода адресира инкременталним, селективним трансформацијама модела на програмски код и очувањем ручно писаних измјена. Његов циљ је неометан рад програмера кроз подршку за директно спајање ручно писаног и генерисаног кода у оквиру исте датотеке и мијењање од стране алата само оних дијелова кода захваћених промјеном модела **Error! Reference source not found.**5,6]. То се постиже API базираним генераторима кода који раде над синтаксним стаблима, коришћењем парсера за језике на којима ће се вршити развој циљног решења.

Циљ овог рада је да се у SeamlessMDD алат уведе подршка за парсирање за различите програмске језике, уз избјегавање јаке зависности између језгра алата и библиотека/сервиса за парсирање, што би отежало замјену технологије и тестирање. Из тог разлога уводимо механизам за парсирање кода употребом удаљених API-ја, тако да алат говори једним доменским „језиком намјере“, а парсирање и операције изводе одвојени сервиси **Error! Reference source not found.**. Конкретно, у овом раду представљамо проширење које уводи апстрактни интерфејс за API парсирање и измијешта манипулацију стаблом у засебне удаљене сервисе, чиме се поједностављује архитектура и омогућава лакша употреба различитих парсерских библиотека **Error! Reference source not found.**.

Структура рада је сљедећа. У другој секцији представљамо SeamlessMDD развојни оквир и његов начин рада. У секцији 3 представљамо дизајн нашег рјешења и дискутујемо пројектантске одлуке. Секција

4 даје детаље имплементације а секција 5 показује како се додаје нови парсер без промјена у клијентском коду. Секција 6 је закључак рада, гдје резимирамо постигнуто и дефинишемо правце будућег развоја.

2. SEAMLESSMDD ALAT

SeamlessMDD у средиште ставља проблем који се у пракси MDD-а упорно враћа: како синхронизовати генерисани код са ручним измјенама, тако да крајње рјешење ради и да се те измјене очувају кроз будуће циклусе генерисања [6]. Умјесто трансформација базираних на шаблонима које приликом промене модела поновно генеришу све артефакте, SeamlessMDD инсистира на инкременталним, селективним трансформацијама. Мијења се само релевантни дио циљног кода на основу промењених елемената модела, чиме се смањује вријеме извршавања и ризик од конфликата [6]. Ова идеја „source-preserving incrementality“ поткријепљена је и у литератури о инкременталном model-to-text превођењу, с јасном препоруком да обим пропагиране промјене буде пропорционалан само утицају промјене, а не величини улазног модела [6].

Да би такво понашање било могуће, архитектура се ослања на два основна појма: „делту“ и интерну репрезентацију артефакта (Internal Artifact Representation, IAR). Делта је минималан и прецизан опис промјене у моделу (шта је додато, уклоњено или измијењено), док је IAR структуриран приказ постојећег излазног артефакта (нпр. HTML/код) у облику стабла са типовима, идентификаторима и контекстом. На основу делти се граде циљане операције над IAR-ом (додај, замијени, обриши конкретан чвор или атрибут), па се тек затим IAR серијализује у текст излазне датотеке. Тако се не преписује цијела датотека него се локално захвата само њен дио захваћен променом елемента модела. Када артефакт не постоји, генератор га ствара из шаблона. Ако постоји, парсира се постојећи садржај и претвара у синтаксно стабло са контекстом (IAR) над којим API-базирани генератор додаје, мијења или брише чворове [6]. Овим се циљна датотека третира као структура, не као обичан текст, па су и интервенције прецизније.

На основу овог приступа, SeamlessMDD у средиште поставља процес који је усмјерен на мале, провјерљиве кораке. Прво се детектују промјене на моделу путем поређења тренутне и претходне верзије, за генератору релевантне дијелове. Резултати тих промјена су делте (додато/обрисано/мијењано) на нивоу елемента модела. Затим слиједи валидација која користи API базиране генераторе да провјере синтаксну исправност намјераваних захвата над IAR-ом и да семантички испитају усклађеност са моделом. Тек послије тога долазимо до корака извршавања трансформације. Трансформација обухвата функције: убаци, модификуј и обриши. Извршавање се врши над IAR-ом, а корисник може да бира између два режима рада: „Preview“ (увид у планиране измене у коду без серијализације) и „Generate“ (трајна серијализација промјена), при чему се ослања на VCS - систем за контролу верзија који омогућава евиденцију измјена, поређење стања и повратак на претходне ревизије [6].

Ова подјела на преглед и упис значајно смањује проблеме при усвајању MDD-а, јер се ефекти могу испробати и кориговати прије него што постану трајни. Поред овога, за SeamlessMDD је кључно да се омогући ко-еволуција модела и кода, односно њихов паралелан и усклађен развој: промјене у моделу доводе до минималних, циљаних измјена у коду, док се ручно унесене измјене у коду очувају и узимају као приоритет при наредним циклусима. Ово се постиже мапирањем елемената модела на кодне фрагменте, провјерама интегритета фрагмената и интерактивним рјешавањем конфликата на нивоу елемената модела, а не линија текста [6]. Ово има за циљ да подстакне употребу овог алата и у системима гдје се рад врши „по дијеловима“ и гдје је само дио система моделован (остало се наставља развијати ручно), што омогућава постепено усвајање MDD-а. Веома је важно и да рјешавање конфликата не буде на нивоу линије како VCS-ови функционишу (поређењем редова обичног текста који занемарује контекст), већ да алат прикаже конфликт у контексту модела и понуди смислене опције за његово разрешавање. Такође, обим трансформација треба да се смањи колико је могуће, да би могућност настајања конфликта била мања.

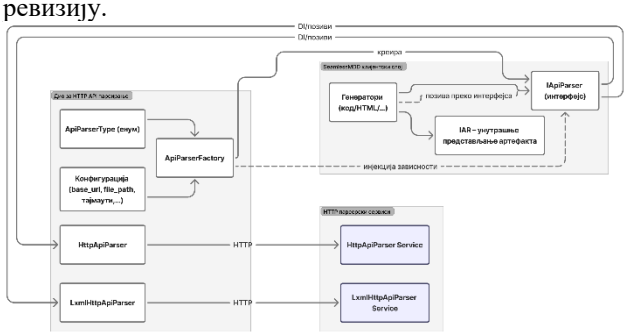
Разлог зашто озбиљни напори у SeamlessMDD-у иду баш у овом правцу јесте дуго познат „round-trip“ проблем [12]. MDD природно уводи више нивоа и више међусобно повезаних артефаката; кад промјена крене „одоздо“ (из кода), веома је тешко поуздано подићи апстракцију нагоре и пренијети значење у модел, а поновно генерисање има тенденцију да често изгуби ручно писан код. Отуда нагласак на IAR-у, на селективности, на приоритету ручних измјена и могућности увида у измјене, у Preview режиму.

3. ДИЗАЈН РЈЕШЕЊА

Анализирајући SeamlessMDD оквир, може се уочити да парсирање представља уско грло самог алата и да је чување њега у језгру алата потенцијалан проблем. Код за парсирање као дио језгра алата доводи до јаке зависности између генератора и конкретне библиотеке за парсирање. Свака промјена библиотеке, верзије или начина адресирања чворова у стаблу прелива се на клијентски код и на логичке токове генератора. Тестирање је скупо и подложно променама јер зависи од исте те библиотеке: тешко је изоловати неуспјехе, симулирати статусне кодове или провјерити понашање у граничним случајевима. Подршка више парсерских технологија значи одржавање паралелних грана кода или разгранавања по условима, што увећава технички дуг. Операције над стаблом (читања, уметања, замјене, брисања) постају уско везане за специфични DOM/AST модел, што отежава увођење другог алата са другачијим семантичким нијансама. Перформансе и стабилност су тешко мјерљиве јер се све дешава унутар истог процеса, без јасних мјеста за кеширање, временска ограничења или поновне покушаје. Оперативно, тимови не могу лако да изаберу најбољи парсер за свој домен, а да то не утиче на остатак система. Прелазак на удаљени сервис касније би свакако био нужан ради скалабилности, али тренутни спој парсирања и генератора то отежава. На крају,

додавање трећег или четвртог парсера тражи понављање шаблона и ризикује неконзистентност у обради грешака, формату одговора и уговора који клијент очекује. Све ово кочи еволуцију, повећава трошак одржавања и смањује повјерење у минималне, селективне трансформације које SeamlessMDD заговара.

Рјешење проблема је дизајнирано тако да централни дио алата не зависи од конкретне технологије парсирања, већ од стабилног и јасно дефинисаног интерфејса за API парсирање. Умјесто да клијентска апликација у себи садржи парсере за све језике и формате, парсирање и манипулација стаблима се изводе у посебном сервису изложеном преко програмског интерфејса IAPIParser (слика 1). Клијент и сервис комуницирају јасним уговором: „нађи елемент по идентификатору“, „провјери да ли чвор постоји у овом контексту“, „убаци подструктуру на дату путању“, „замјени елемент датим HTML-ом“. Клијент не мора знати ни којом је библиотеком стабло изграђено, ни како се интерно представља чвор. Тако се добија неколико директних користи. Прво и основно, могућност пружања подршке парсирања за много различитих структура података, те лакше мијењање употребљених библиотека за парсирање. Друго, скалабилност: сервиси се могу хоризонтално умножавати, распоређивати на локалну машину или у мрежу, а клијент се не мијења. Треће, бољи квалитет парсирања. Пошто сервис ради над стаблима, лакше је обезбиједити да су измјене минималне, да чувају ручне измене и да остављају чист траг за тестирање и ревизију.



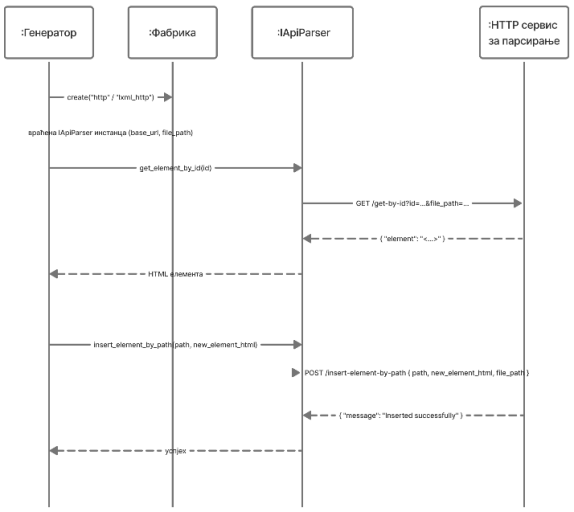
Слика 1. Комуникација компоненти за парсирање

Да би се остварила максимална добит, било потребно употријебити још додатних помоћних образаца, као што су уметање зависности (енг. *Dependency Injection*) и фабрика (енг. *Factory*). Клијент зависи само од апстрактног интерфејса парсера; конкретна имплементација се умеће приликом покретања или конфигурације. Фабрика, са друге стране, се брине и чува логику избора имплементације. То се остварује на основу типа, адресе сервиса или окружења, враћа се парсер који говори истим интерфејсом. Ова комбинација креира ниску зависност између клијента и имплементације и пружа „*hot-swap*“ могућност – лако је замијенити технологију парсирања без интервенција у клијентском коду или у остатаку алата. Уз то, тестови се пишу према структури интерфејса: једном дефинисан скуп понашања може се провјеравати над симулираним сервисима, локалним серверима или правим инстанцама у интеграционом окружењу.

Ово даје нову димензију SeamlessMDD оквиру, јер је много лакше пронаћи постојећи сервис парсирања и само се прилагодити његовом API-у, него њега интегрисати у наш код.

4. ИМПЛЕМЕНТАЦИЈА РЈЕШЕЊА

Имплементација прати дизајн кроз двије конкретне класе IApiParser интерфејса и два помоћна сервиса. Прва имплементација HttpApiParser комуницира са сервисом заснованим на AdvancedHTMLParser-у [9], а друга имплементација, LxmlHttpApiParser са сервисом на lxml-у [10]. Методе интерфејса мапирају се на HTTP руте сервиса за читање (нпр. GET /get-by-id, /get-by-name, /check-exists, /get-elements-by-path) и мутације (нпр. POST /insert-element-by-path, PUT /update-by-path, DELETE /delete-by-path, као и операције по ID-у). Одговори се нормализују у уједначене JSON облике (нпр. { "element": "<…>" } за читања, { "exists": true|false } за провјере, { "message": "…" } за успјешне мутације), тако да клијент не зависи од специфичности библиотека за парсирање. Обрада статуса је централизована: 200 означава успјех, 404 се тумачи као „не постоји“, 400 сигнализује лош улаз, 500 серверску грешку, 501 „није подржано“ (на примјер, AdvancedHTMLParser сервис не нуди све операције које пружа lxml варијанта). Конфигурација имплементација кроз фабрику прима параметре за подешавање комуникације, чиме се извршење лако прилагођава локалним или удаљеним окружењима. На овај начин, SeamlessMDD задржава исти начин рада (Preview/Generate над IAR-ом), док конкретно извршење обавља изабрани сервис са стандардизованим интерфејсом и понашањем (слика 2).



Слика 2. Добављање и упис елемената стабла

5. ДОДАВАЊЕ НОВОГ ПАРСЕРА

Архитектура је осмишљена тако да увођење новог парсерског сервиса не захтијева измјене у клијентском коду већ само локализоване допуне у слоју имплементација. Полазно, проширује се енум тип којим се бира имплементација и додаје се нова класа која реализује интерфејс IApiParser, прилагођена API-ју конкретног сервиса (путање, параметри, повратни формати). Конфигурација нове имплементације

прихвата адресу сервиса, путању до обрађиване датотеке и подешавања HTTP клијента, уз могућност додавања заглавља за аутентификацију/ауторизацију када је потребно. Фабрика се допуњава једном граном која нову вриједност енум типа мапира на нову класу, чиме избор парсера остаје централизован и транспарентан. При имплементацији метода интерфејса, сваку операцију треба досљедно пресликати на руте сервиса, те нормализовати одговоре у уједначене JSON облике које клијент већ очекује. Нови парсер може радити локално или удаљено, за SeamlessMDD је то прозирно, јер се клијент и даље обраћа само IApiParser-у и наставља свој Preview/Generate ток над IAR-ом. Практично, постојећи интеграциони сценарији се могу поново искористити над новом имплементацијом, што омогућава брзу провјеру усклађености са интерфејсом. На овај начин, увођење трећег или четвртог парсера своди се на предвидљив шаблон корака, без додира у генераторима и остатку система. Резултат је линија проширивости која омогућава да се технологија парсирања бира према доменским критеријумима, а да архитектура остане чиста и стабилна.

6. ЗАКЉУЧАК

У овом раду смо адресирали уочено уско грло у SeamlessMDD-у: чврсту спрегу између језгра алата и конкретних парсерских библиотека. Увели смо стабилну, замјењиву апстракцију за парсирање (IApiParser) и централизовали избор имплементације кроз Factory шаблон, тако да клијентски дио изражава намјеру у доменским појмовима, док конкретно извршење обавља сервис. На тај начин задржана је филозофија SeamlessMDD-а о минималним, селективним захватима над IAR-ом и приоритету ручних измјена, а истовремено је смањена зависност од технологија и поједностављено увођење различитих парсера. Двије изведене имплементације, на основу AdvancedHTMLParser-а и lxml-а, показују да исти сценарији могу да се изведу без измјена у клијентском коду, што директно потврђује циљеве о смањењу зависности ка библиотекама за парсирање, повећању лакоће тестирања и природној проширивости [6,7].

Ограничења тренутне верзије односе се прије свега на број подржаних сервиса и на оперативне аспекте мрежне комуникације. Нису уведени механизми временских ограничења и поновних покушаја, нити подршка за аутентификацију/ауторизација за заштићене сервисе. Такође, нису спроведена систематска мјерења перформансии. Будући рад усмјерићемо на додавање нових парсера (и за друге језике/формате), оснаживање комуникационог слоја, подршку безбједносним захтјевима, као и увођење метрика и трајног логовања за оперативну употребу у већим тимовима. Планирано је и проширење уговора за богатије семантике ажурирања елемената и експлицитну идемпотентност, као и формалнија валидација конзистентности између различитих имплементација. На тај начин, предложена архитектура остаје стабилна основа за даљу еволуцију SeamlessMDD-а ка ширем, индустријски одрживом еко-систему парсерских сервиса.

7. ЛИТЕРАТУРА

[1] B. Selic, “The Pragmatics of Model-Driven Development,” IEEE Software, 20(5), 2003.

[2] D. C. Schmidt, “Guest Editor’s Introduction: Model-Driven Engineering,” Computer, 39(2), 2006.

[3] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice, 2nd ed., Morgan & Claypool, 2017.

[4] B. Hailpern, P. Tarr, “Model-driven development: The Good, the Bad, and the Ugly,” IBM Systems Journal, 45(3), 2006.

[5] B. Dragaš et al., “SeamlessMDD: Framework for Seamless Integration,” технички извјештај, GitHub: github.com/BojanaZ/SeamlessMDD, приступљено: окт. 2025.

[6] B. Dragaš, N. Todorović, T. Rajačić, G. Milosavljević, Ž. Vuković. "A Novel Approach to Integration of Manual Changes in Generated Code: SeamlessMDD." *Journal of Web Engineering* 24, no. 4 (2025): 499-528.

[7] N. Todorović, B. Dragaš, G. Milosavljević, “Supporting Integrative Code Generation with Traceability Links and Code Fragment Integrity Checks,” in Disruptive Information Technologies for a Smart Society, Springer, Cham, 2024.

[8] W3C, “XML Path Language (XPath) 3.1 (Recommendation),” w3.org/TR/xpath-31/, приступљено: окт. 2025.

[9] lxml, “Parsing XML and HTML with lxml,” lxml.de/parsing.html, приступљено: окт. 2025.

[10] AdvancedHTMLParser, “Project page (PyPI),” pypi.org/project/AdvancedHTMLParser/, приступљено: окт. 2025.

[11] B. J. Ogunyomi, L. Rose, D. Kolovos, “Incremental execution of model-to-text transformations using property access traces,” Software & Systems Modeling, 18:1–17, 2019.

[12] B. Hailpern, P. Tarr, „Model-driven development: The Good, the Bad, and the Ugly“, IBM Systems Journal, 45(3), 2006.

Кратка биографија:

Драган Мирковић основну и Гимназију „Васо Пелагић“ у Брчком завршио је као вуковац. Од 2019. студира на ФТН-у у Новом Саду (Софтверско инжењерство и информационе технологије). Основне студије окончао је 2023., а мастер студије уписује исте те године, смијер Софтверско инжењерство и информационе технологије, усмјерење Софтверско инжењерство.
Контакт: dmirkovic.se@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.4
ДОИ: https://doi.org/10.24867/35BE06Radman

Алгоритам за праћење, одређивање оријентације и детекцију храњења свиња

Algorithm for Pig Tracking, Orientation Estimation, and Feeding Detection

Иван Радман, Дарко Чапко, *Факултет техничких наука, Нови Сад*

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У раду је представљен алгоритам за праћење свиња на снимку, који се у комбинацији са алгоритмом за одређивање оријентације свиња користи за надзор и увид у храњење свиња у шталама.*
Кључне речи (три до пет): *компјутерска визија, праћење, процена оријентације, детекција храњења, анализа понашања*

Abstract – *The paper presents an algorithm for tracking pigs in videos, which in combination with an algorithm for determining their orientation, is used for monitoring and analyzing pig feeding behavior in barns.*

Keywords: (three to five): *computer vision, tracking, orientation estimation, eating detection, behavior analysis*

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

1. УВОД

Захваљујући развоју технологије и дигитализације, савремено сточарство се налази у фази интензивне модернизације. Традиционални начини надзора животиња се све више замењују аутоматизованим системима који омогућавају прецизно мерење, праћење и контролу параметара у реалном времену. Примена сензора, камера и интелигентних система доприноси већој ефикасности производње, побољшању животног стандарда животиња и смањењу потребе за сталним физичким надзором.

Вештачка интелигенција, као данас најбрже растућа област, има све већу улогу у процесу модернизације. Употребом метода машинског и дубоког учења, као и компјутерске визије, омогућено је аутоматско препознавање образаца понашања животиња, детекција болести у раним фазама и праћење активности као што су храњење, одмор или насилно понашање [1]. На тај начин, фармерима и истраживачима се обезбеђује детаљнији увид у понашање животиња и могућност бржег реаговања.

Упркос могућностима, аутоматизација праћења животиња у домаћем сточарству још увек није довољно развијена. Због тога је, у сарадњи Института БиоСенс и компаније Карнекс, формиран алгоритам за праћење свиња и одређивање њихове оријентације, са циљем анализе понашања и активности. У раду је фокус стављен на храњење, али овакав систем може послужити као основа за друге типове активности, поспешујући даљи развој решења и истраживања у области аутоматизоване анализе понашања животиња.

2. ПОДАЦИ И МОДЕЛ

2.1. Подаци

Две камере су биле постављене унутар штале, свака усмерена на засебну комору са по једном хранилицом. Свиње највећи део дана проводе у коморама, што омогућава континуално праћење и анализу њиховог понашања. Камере су бележиле видео снимке током целог дана, чиме је обезбеђен велики број података погодних за анализу.

Из видео снимака су издвојени појединачни фрејмови прилагођавањем фреквенције чувања у зависности од активности свиња. Већа фреквенција коришћена је у динамичним сценама, а мања током периода мировања. Такође, водило се рачуна да се подаци прикупе у различитим условима осветљења и са обе камере ради избегавања пристрасности.

Обележавање података извршено је над 165 слика, при чему је свака садржала по неколико десетина свиња. Ради проширења скупа података, коришћен је и јавно доступан скуп са већ означеним сликама свиња које су биле релативно сличне коришћеним, што је додатно побољшало перформансе модела.

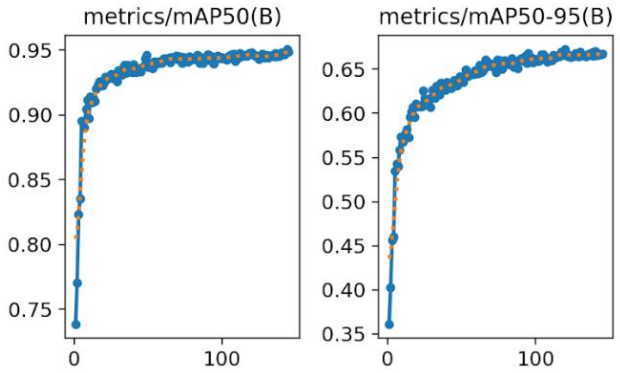
Због ограниченог броја слика, примењена је аугментација ради проширења скупа података. Над сваком сликом су вршене операције као што су ротација, промена контраста и замагљивање, чиме је иницијални скуп троструко увећан.

2.2. Модел

Као основни модел за детекцију коришћена је конволутивна неуронска мрежа *YOLOv8* [2], позната по високој брзини и тачности. Модел је посебно погодан јер је претходно трениран над скупом података *OpenImagesV7* [3] који садржи ознаке за свиње.

Обука је сроведена применом технике *Transfer Learning*, где је првих 15 слојева модела замрзнуто ради задржавања универзалних особина као што су препознавање ивица и контура, док су виши слојеви прилагођени новим подацима. Обука је вршена са максималним бројем епоха од 200, а као алгоритам

оптимизације је коришћен пондерисани Адам. Праг за рано заустављање је био постављене на 20 епоха и најбољи резултати су добијени за почетни корак учења 0.015. Вредности метрика у току обуке могу се видети на слици 1.



Слика 1. Метрике у току обуке

Модел је обучен у окружењу Google Colab, што се показало као довољно за потребе истраживања. Јасно је да за продукцијске верзије овакав модел не би био нити погодан нити довољан, али то свакако није фокус рада.

3. АЛГОРИТАМ ЗА ПРАЋЕЊЕ

Након иницијалног тестирања постојећег алгоритма који је интегрисан у библиотеку, показало се да није давао задовољавајуће резултате у условима густе и динамичне сцене унутар штале. Честе појаве преклапања свиња, њихово нестајање из кадра и хаотично кретање доводили су до великог броја погрешних идентификација, на снимцима од једног минута и треидесетак свиња на снимцима, било идентификовано и по више хиљада свиња.
Покушаји исправке и филтрирања резултата показали су се недовољним, па је донета одлука да се развије потпуно нов, прилагођени алгоритам за праћење, прилагођен специфичностима сцене и понашању свиња.

3.1. Прилагођени алгоритам за праћење

Нови алгоритам је осмишљен тако да омогући стабилно праћење и идентификацију свиња кроз динамичке сцене и услове велике густине и делимичне заклоњености. Структура алгоритма се ослања на основне принципе сваког алготитма за праћење, али су кључни механизми филтрирања, мапирања и компензације губитака објеката специфично прилагођени датом проблему.

3.2. Основа алгоритма

Основу овог алгоритма представњају скупови објеката који се прате унутар њега и то конкретно наредна три дисјункта скупа објеката:

- Потенцијални објекти - ново детектоване свиње које се тек појављују у кадру и морају бити видљиве пет фрејмова за редом како би се потврдило да не представљају шум или лажну детекцију.
- Активни (праћени) објекти - свиње које су успешно потврђене и чије се позиције континуирано ажурирају у сваком фрејму. Уз њих се бележи и број фрејмова током којих су присутне.
- Изгубљени објекти - свиње које су нестале из кадра. Чува се последња позната позиција, како би се омогућило поновно препознавање након појављивања.

Сваки објекат да би доспео у активне или изгубљене објекте мора проћи кроз потенцијалне и сачекати своју потврду.

3.3. Филтрирање

Да би се спречила обрада предикција које нису од интереса, дефинисана је област од интереса (*Region of Interest*) у којој се врши анализа. Објекти ван области, као и они са мером поузданости мањом од 0.4, одбацују се. Поред тога, примењен је праг на вредност преклапања области предикција (*Intersection over Union - IoU*) од 0.5 како би се избегло понављање детекција за исту свињу у истом кадру.

3.4. Мапирање објеката

Кључни део алгоритма је механизам мапирања нових детекција са постојећим објектима. За сваку нову детекцију се у односу на све објекте из три наведеа скупа рачунају:

- Дистанца између центроида
- *IoU* вредност

Да би мере биле упоредиве, дистанца се нормализује на опсег [0,1] формулом:

$$dist_{norm} = max\ (0{,}1 - \left(\frac{dist}{100}\right)) \qquad (1)$$

Чиме се дистанца од 0 пиксела, што представа потпуно поклапање, пресликава у 1, а све преко 100 пиксела у 0. Мере су обједињене у један коначан скор блискости који се добија као пондерисана комбинација мерених вредности:

$$score = \alpha * IOU + \ (1 - \alpha) * dist_{norm} \qquad (2)$$

где је емпиријски утврђена добра вредност за α 0.4. На основу највишег скора, врши се упаривање објеката између фрејмова. Уколико ниједна детекција нема скор који је већи од 0.55 (такође емпријска вредност, међутим може се изнети интерпретација да је просек границе за удаљеност ~60 пиксела и за IoU 0.4, где побољшање једне комензује другу до неке границе), генерише се нови идентификатор.
Овај приступ је донео значајно побољшање, где се број свиња препознатих на снимцима који трају око 2 минуте смањио са неколико хиљада, на по 28, 29 (од стварних 21) у зависности од снимка, што представља велики напредак у односу на иницијални алгоритам.

3.5. Механизам загревања

Додатно је уведен и механизам загревања, који служи за стабилизацију броја праћених свиња. Први део снимака (~10%) користи се као фаза иницијализације, током које се прикупљају све потенцијалне детекције. Након тога, задржавају се само оне свиње које су биле присутне у већем делу загревања (~85%). Тачне вредности процената нису стриктне и зависе од сцене

видео снимка (може се упоредити са расподелом података на скупове за обуку, валидацију и тест), само је битно да се обезбеди да је период загревања довољно репрезентативан и да је праг за присутност у загревању довољно висок, али да допушта неки ниво нестајања свиња.
На овај начин је постигнута елиминација детекција које постоје веома кратко,а које су у већини случајева лажне. Након загревања, алгоритам наставља рад само са провереним објектима, по претходно описаном принципу. Резултати са загревањем су додатно смањиви број детекција, где је на снимцима од 2 минуте била додата једна, односно 2 свиње.

4. ОДРЕЂИВАЊЕ ОРИЈЕНТАЦИЈЕ

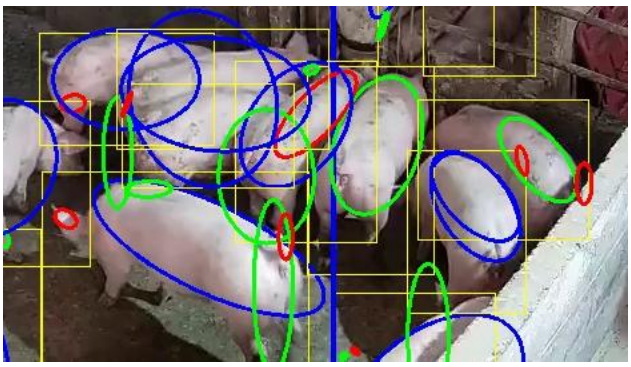
Алгоритам за праћење свиња представља основу за анализу положаја и кретања, али сам по себи није довољан за дубљу анализу понашања. Иако обезбеђује информације о позицији и путањи сваке свиње, он не даје податке о њеној оријентацији, што је кључно за процену разних активности, међу којима је и храњење. Уколико се посматра конкретно храњење, на основу положаја објекта, могуће је проценити да ли се свиња налази у близини хранилице, али не и да ли је окренута ка њој.
Имајући у виду наведене ставке, реализован је поступак одређивања оријентације свиња.

4.1. Поступак одређивања оријентације

Постоји више различитих модета за одређивање оријентације објеката на слици [4], а у овом раду су примењене само методе обраде слике. Процес је одрађен превенствено са циљем да се издвоји свиња од позадине унутар дектекција, а затим да се на основу њеног облика одреди угао оријентације тела.
Да би се добила информације о оријентацији свиње, свиње унутар предикција су апроксимиране елипсом. Елипса је коришћена јер тело свиње интуитивно може да се представи елипсом и зато што се након познавања параметара елипсе унутар предикција врло лако може утврдити где је елипса тј. свиња усмерена. Поступак израчунавања оријентације свиње обухвата следеће кораке:

1. Извајање дела слике где се налази деткција
2. Конверзија исечене слике у сиву слику
3. Еквивализација хистограма
4. Бинаризација слике, где се за праг бинаризације користи средња вредности хистограма
5. Издвајање контура на добијеној бинарној слици
6. Фитовање елипси над пронађеним контурама
7. Избор одговарајуће елипсе

Након издвајања, еквивализација хистограма се користи да би се поправио контраст унутар слике и лакше диференцирала позадина од свиње. Даље, бинаризацијом се постуже издвајање објеката са слике од позадине. Коришћење средње вредности хистограма за праг јесте вид адаптације прага према слици, коришћен из разлога што различите свиње на како једној, тако и различитим сликама захтевају различите вредности и није могуће одредити универзални праг.
Надаље, контуре се издвајају из бинаризоване слике применом уграђене функције *findContours* из библиотеке *OpenCV* [5], након чега се над сваком контуром са пет тачака врши фитовање елипсе методом *fitEllipse*. Коришћеном методом се формирају елипсе на основу затворених контура унутар слике и резултат се може видети на слици 2.

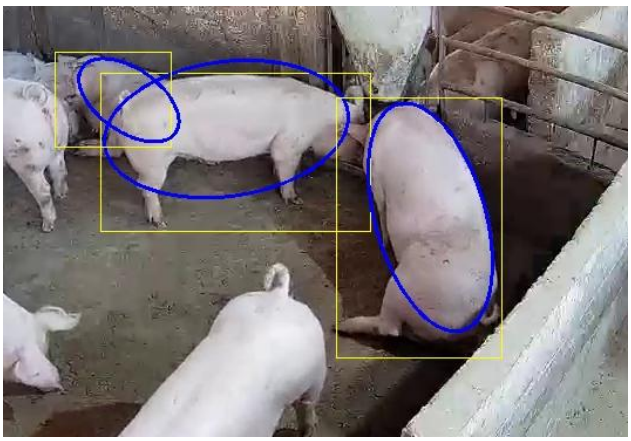


Слика 2. Фитовање елипси

Јасно је да се овакви резултати не могу користити, тако да је се за сваку добијену елипсу рачунају површина и позиција центроида и врши се поступак филтрирања елипси, тако што се задрже само елипсе које задовољавају следеће услове:

- Барем 25% елипсе се налази унтар правоуганика (филтрирање очигледно превеликих елипси)
- Центроид предикције се налази унутар елписе (филтрирање елпипси које су далеко од центроида)

Након филтрирања, највећа елипса од преосталих се проглашава за одабрану и користи се као представник оријентације свиње.
Овај поступак се примењује за сваку детекцију у сваком фрејму и користи за даље аналите. Резултат примене алгортима над конретним фрејмом и свињама се може видети на следећој слици 3.



Слика 3. Резултат одређивања оријентације

5. ИНТЕГРГАЦИЈА И УПОТРЕБА ОРИЈЕНТАЦИЈЕ У АЛГОРИТМУ ЗА ПРАЋЕЊЕ

Алгоритам за одређивање оријентације је на крају било потребно интегрисати унутар алгортима за праћење и искористити га за детекцију храњења. Срећом, то је

било врло једноставно, где је била потребно направити једноставу модификацију и проширење.
Унутар аллгортма за праћење, за сваку свињу која се налазила у релтивној близини хранилице, што је опет обележеном новом области од интереса, рађен је поступак одређивања елипсе ради оријентације.
На основу добијене елипсе одређује се угао између главне осе елипсе и праве која спаја центроид предикције и хранилицу (која се једноставно обележи тачком). Овим поступком се добија угао који представља усмереност свиње према хранилици, при чему мањи угао указује на већу усмереност свиње ка хранилици.
Поред угла, рачуна се и дисатнца свиње од хранилице, јер је она такође релевантан фактор у процени да ли свиња заиста једе.
На крају, обе вредности се комбинују у јединствени скор храњења који се израчунава преко нормализованих вредности угла и удаљености, где се нормализован угао дефинише као:

$$angle_{norm} = angle/90 \quad (4)$$

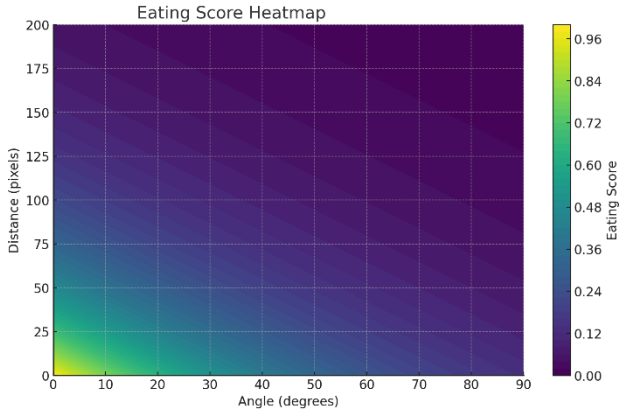
док се нормализована дистанца дефинише:

$$dist_{norm} = dist/100 \quad (5)$$

Коначно, финални скор се добија преко следеће формуле:

$$score = exp\left(-\alpha * angle_{norm} - \beta * dist_{norm}\right) \quad (6)$$

За вредности пондера, који служе да се контролише осетљивост финалног резултата на угао и удањеност, коришћене су редом вредности 0.4 и 1.5. Вредности финалног скора се могу видети на следећем графику:



Слика 4. Графички приказ скора

Зато, на крају је коришћен праг од 0.25, где уколико скор прелази дату вредност, сматра се да свиња једе и унутар алгоритма се инкренемтира број фрејмова у којима дата свиња једе. Такође, на излазном видео снимку се она обележава другачијом бојом, ради боље видљивости.
Овиме је постигнуто да се поред праћења свиња на снимку, добије увид у информацију да ли свиња једе. Иако генерално добро ради, само одређивање оријентације је осетљиво на нестајање и нагле промене осветљености свиње, што уме да се искаже у лажним детекцијама да свиња не једе. Наравно, овај проблем се делимично и решава повећањем тачности коришћеног модела, као и даљим унапређивањем коришћених метода.

6. ЗАКЉУЧАК

Формирани алгоритам показује способност да се веома добро функционише у тешким условима са којима је суочен и са те стране се може сматрати врло успешним. Препознавање, праћење и повезивање свиња на оваквим снимцима је врло захтеван задатак за човека и чињеница да алгоритам може да се носи са њима оправдава његов циљ, што јесте аутоматизовано праћење храњења свиња.
Поред тога, алгоритам није прављен са идејом да се максимизују перформансе на конкретном скупу видео снимака, нити да користи што бољи модел као основу, већ да се истраже и развију методе који ће омогућити праћење у комплексним и густим срединама. У раду су су дискутоване и предложене разне методе са циљем компензације различитих проблема, које се могу користити као такве, али и као основа за даљи развој метода и прилагођења.
Иако је у раду је дискутовано и тестирано само посматрање храњења свиња, које представља једну од најзначајнијих активности које фармери посматрају, логика и приступ који су озложени у раду се могу користи и проширити и на друге типове активности, што ће додатно допринети аутоматизованом надзору и анализи понашања животиња, олакштати и убрзати посао фармерима и коначно допринети повећању квалитета живота свиња.

7. ЛИТЕРАТУРА

[1] L. S. Saoud A. Sultan, M. Elmezain, M. Heshmat, L. Seneviratne, I. Hussain, “Beyond observation: Deep learning for animal behavior and ecological conservation“, Khalifa University, United Arab Emirates, 2024

[2] YOLOv8 документација https://docs.ultralytics.com/models/yolov8/

[3] OpenImagesV7 Скуп података https://storage.googleapis.com/openimages/web/index.html

[4] G Savathrakis, A. Argyros *“An Automated Method for the Creation of Oriented Bounding Boxes in Remote Sensing Ship Detection Datasets“*, In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 830-839. 2024.

[5] OpenCV документација https://docs.opencv.org/4.x/index.html

Кратка биографија:

Иван Радман рођен је у Новом Саду 2001. год. Мастер рад на Факултету техничких наука из области Рачунарство и аутоматика – Рачунарски управљачки системи одбранио је 2025.год.
Контакт: ivannradman@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.3
ДОИ: https://doi.org/10.24867/35BE07Lakic

Филтрирање у индустријским системима: компаративна студија и имплементација у софтверском осцилоскопу

Filtering in industrial systems: comparative study and implementation in a software oscilloscope

Анђела Лакић, *Факултет техничких наука, Нови Сад*

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Овај рад представља компаративну студију техника филтрирања за индустријску аутоматизацију и њихову имплементацију у софтверском осцилоскопу KeStudio Scope компаније KEBA. Анализиране су три групе филтера: IIR филтери (Баттерворт, Чебишев I и II), адаптивни филтери (LMS, NLMS, RLS) и нелинеарни филтери (Калманов, EKF, UKF, SIR честични филтер). Резултати показују да Чебишев II постиже најбоље перформансе за стационарне сигнале, RLS за адаптивне примене, док SIR честични филтер даје најтачније резултате за нелинеарне системе. На основу анализе, имплементирано је хибридно решење које комбинује NLMS и честични филтер.*
Кључне речи: *iir филтри, адаптивни филтри, нелинеарни филтри, софтверски осцилоскоп*

Abstract – *This paper presents a comparative study of filtering techniques for industrial automation and their implementation in the KeStudio Scope software oscilloscope by KEBA. Three filter groups are analyzed: IIR filters (Butterworth, Chebyshev I and II), adaptive filters (LMS, NLMS, RLS), and nonlinear filters (Kalman, EKF, UKF, SIR particle filter). Results demonstrate that Chebyshev II achieves the best performance for stationary signals, RLS for adaptive applications, while SIR particle filter provides the most accurate results for nonlinear systems. Based on the analysis, a hybrid solution combining NLMS and particle filter was implemented.*

Keywords: *iir filters, adaptive filters, nonlinear filters, software oscilloscope*

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

1. УВОД

У индустријској аутоматизацији, инжењери се ослањају на прецизно мјерење сигнала за управљање и оптимизацију сложених система. Дигитални осцилоскопи су важан алат за визуализацију временских серија података са сензора и контролера. Иако показују одличне способности визуализације, обично немају уграђене функције обраде сигнала.

Потреба за аутоматским филтрирањем сигнала јавила се у дизајну софтверског осцилоскопа компаније *KEBA Industrial Automation*. У сарадњи са инжењерима индустријске производње дошло се до закључка да би комбинација филтрирања у реалном времену и на већ снимљеним сигналима омогућила већу поузданост алата и уштеду времена.

Овај рад има за циљ процјену и имплементацију различитих техника филтрирања: једноставних филтера са бесконачним импулсним одзивом (Батерворт и Чебишев), адаптивних филтера који динамички прилагођавају параметре, и нелинеарних филтера отпорних на одступања. На основу компаративне анализе, најбоље методе се интегришу у софтверски осцилоскоп *KeStudio Scope*.

2. *KESTUDIO SCOPE* СОФТВЕРСКИ ОСЦИЛОСКОП

KeStudio Scope је софтверски осцилоскоп развијен од стране компаније *KEBA*, посвећен симулацији и дијагностици робота и машина. Програм преузима сигнале са PLC контролера добијене од робота путем сензора, како би се анализирали и визуализовали ради испитивања динамике система и оптимизације рада машина.

Додавање филтара чини анализу, детекцију аномалија и дијагностику специфичнијим, те обезбеђује прецизније и поузданије податке.

3. ПРЕГЛЕД КОРИШТЕНИХ ФИЛТАРА

У раду су анализиране 3 групе филтара: дигитални филтри бесконачног импулсног одзива, адаптивни филтри и нелинеарни филтри. У наставку је дат преглед филтара са теоријским основама за сваки кориштени филтер из групе.

3.1. Дигитални филтри бесконачног импулсног одзива

Дигитални филтри са бесконачним импулсним одзивом (*IIR*) могу ефикасно формирати фреквенцијски одзив са мањим редом филтера. Њихова рекурзивна структура омогућава постизање снажних ефеката када су ресурси ограничени [1].

Батерворт филтер има максимално равну амплитудску карактеристику у пропусном опсегу без таласања.

Модуо преносне функције нископропусног Батерворт филтера реда *n,* са граничном фреквенцијом ω_c је:

$$|H(j\omega)| = \frac{1}{\sqrt{1 + (\frac{\omega}{\omega_c})^{2n}}} \quad (1)$$

Чебишев тип I има таласање у пропусном опсегу *ε,* али оштрији прелаз [2]:

$$|H(j\omega)|^2 = \frac{1}{\varepsilon^2 T_n^2(\frac{\omega}{\omega_c})} \quad (2)$$

где је $T_n(x)$ Чебишев полином првог типа.
Чебишев тип II нема таласања у пропусном опсегу, али се таласање појављује у зауставном делу.

3.2. Адаптивни филтри

Адаптивни филтери прилагођавају своје параметре током времена на основу улазног сигнала. Основна структура заснива се на минимизацији средње квадратне грешке [3]:

$$J(n) = E\{e^2(n)\} \quad (3)$$

Ако је *w(n)* вектор тежина, *x(n)* улазни вектор, а μ корак учења, LMS алгоритам ажурира тежине по формули:

$$w(n+1) = w(n) + \mu x(n) e(n) \quad (4)$$

NLMS нормализује корак учења:

$$w(n+1) = w(n) + \frac{\mu_0}{x^T(n)x(n) + \delta} x(n)e(n) \quad (5)$$

где је μ_0 номинална вриједност корака учења, а δ мала позитивна константа.
RLS алгоритам минимизује експоненцијално пондерисану средњу квадратну грешку и има бржу конвергенцију уз већу рачунску сложеност [4].

3.3. Нелинеарни филтри

Нелинеарни филтери су направљени за системе гдје односи између промјенљивих нису линеарни. Циљ је израчунати условну расподјелу $p(x_k|y_{1:k})$ користећи Бејзијанску рекурзију [5].
Калманов филтер је оптимално решење за линеарне и Гаусове системе.
Проширени Калманов филтер (EKF) линеаризује нелинеарни модел.
Unscented Калманов филтер (UKF) користи сигма тачке за боље апроксимације.
Честични филтер представља расподјелу скупом честица $\{x_{0:k}^i, \omega_k^i\}$.

4. ИМПЛЕМЕНТАЦИЈА И КОМПРАТИВНА АНАЛИЗА

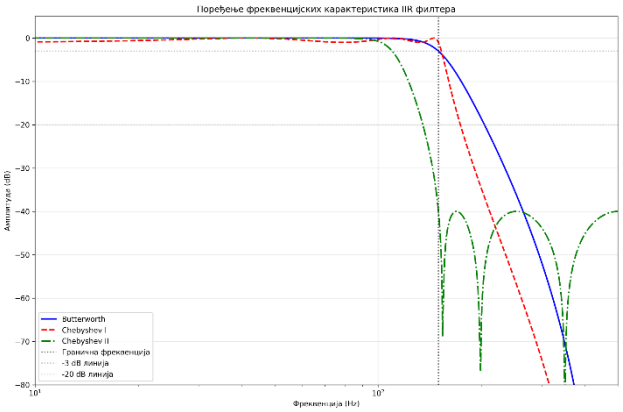
Анализа је спроведена у два корака: анализа унутар једне групе филтера под идентичним условима, затим поређење представника сваке групе на заједничким тест сигналима. Сви филтри су имплементирани у Python програмском језику. Евалуација се спроводи на разноврсном сету сигнала: стационарни синусоидални за IIR филтере, нестационарни chirp сигнали за адаптивне филтере, и нелинеарни динамички системи за Калманове филтере.

4.1. Резултати

Филтри бесконачног импулсног одзива су тестирани на стационарном синусном сигналу са адитивним шумом (SNR = 5 dB), сви филтери 6. реда, гранична фреквенција 150 Hz.

Табела 1. Компаративна анализа филтара са бесконачним одзивом

Тип филтра	RMSE	SNR побољшање (dB)	Вријеме извршавања (ms)
Батерворт	0.1531	5.80	4.01
Чебишев I	0.1431	6.38	0.61
Чебишев II	0.1309	7.16	0.40



Слика 1. Амплитудски одзив сва три филтра

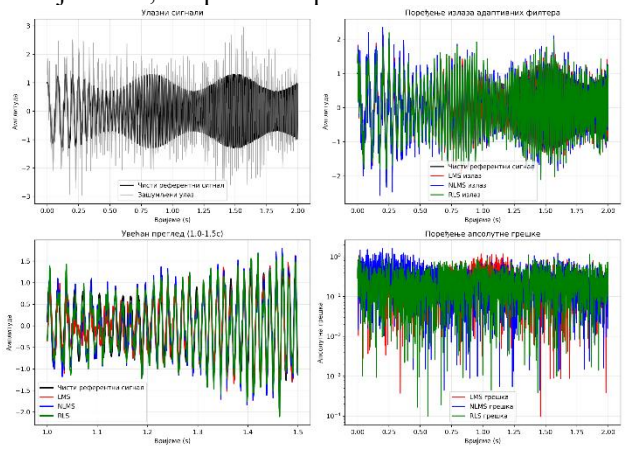
По резултатим из Табеле 1, Чебишев II показује најбоље перформансе (RMSE: 0.131, SNR побољшање: 7.16 dB) уз најкраће вријеме извршавања (0.40 ms). Батерворт има најглаткији одзив, док Чебишев I пружа оштрији прелаз уз таласање, што се види на Слици 1.
Адаптивни филтри су тестирани на нестационарном *chirp* сигналу са сложеном интерференцијом (почетни SNR = 1.28 dB), 32 коефицијента.

Табела 2. Компаративна анализа адаптивних филтара

Алгоритам	Конвергенција (итерације)	MSE	Побољшање SNR (dB)	Вријеме (ms)
LMS	403	0.1073	5.60	0.0117
NLMS	599	0.1116	5.43	0.0093
RLS	61	0.0959	6.09	0.0688

По резултатима из Табеле 2, RLS постиже најбржу конвергенцију (61 итерација) и најмању MSE (0.096) уз највеће рачунско оптерећење (0.069 ms). LMS је најједноставнији (0.012 ms) али са споријом

конвергенцијом (403 итерације). NLMS пружа компромис између стабилности и брзине. На Слици 2 су у временском домену приказани референтни сигнал са шумом и без, и како се сваки од филтера адаптира на тај сигнал, са грешком праћења.

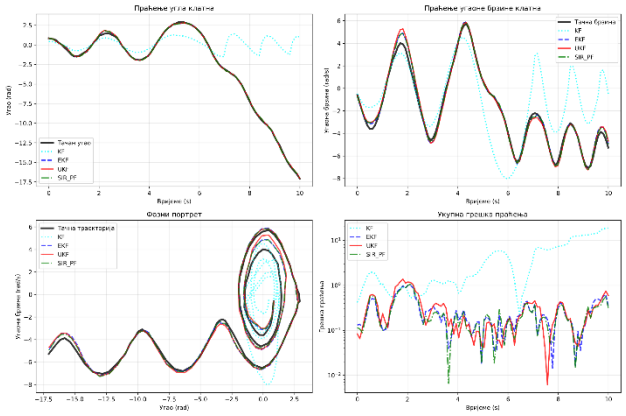


Слика 2. Оригинални и чист референтни сигнал, филтриран сигнал са сва три филера, увећан приказ филтрираних сигнала и грешка праћења

Нелинеарни филтри су тестирани на моделу нелинеарног клатна са пригушењем, процесним и мјерним шум.

Табела 3. Компаративна анализа нелинеарних филтара

Тип	RMSE	Грешка праћења	Вријеме извршавања (ms)	Рачунска захтијевност
KF	4.4011	3.8348	0.107	Ниска
EKF	0.2943	0.2433	0.100	Ниска
UKF	0.4065	0.3232	0.604	Средња
SIR-PF	0.2902	0.2417	41.710	Висока



Слика 3. Праћење угла и угаоне брзине клатна, фазни портрет и грешка праћења

По Табели 3, за нелинеарно клатно, SIR честични филтер је постигао најбољу тачност (RMSE: 0.290) али са највећим рачунским захтјевом (41.7 ms). UKF пружа добар компромис (RMSE: 0.407, време: 0.604 ms), док је стандардни Калманов филтер неадекватан за нелинеарне системе, што се јасно може видјети на Слици 3.

4.2. Упоредни резултати између група филтара

Циљ ове анализе је да покаже јасне области примјене сваке од група филтара. Анализа је спроведена на 3 различита сигнала (синус, *chirp* и Ван дер Пол осцилатор, сва 3 са додатим шумом) и кориштени су филтри какви су дефинисани и у претходним анализама појединачних група филтара. Из резултата у табели 4. Чебишев 2 филтер остварује најбоље перформансе за стационарне сигнале уз јако добру рачунску ефикасност (504725 узорака/секунда), што га чини погодни за филтрирање у реалном времену у индустријским примјенама. RLS адаптивни филтер ради најбоље на нестационарним сигналима са умјереним (21116 узорака/секунда), међутим има лоше перформансе на нелинеарним системима. SIR честични филтер постиже супериорне резултате за нелинеарне динамичке системе, али такође показује и највеће рачунско оптерећење (494 узорка/секунда), што га чини погодним искључиво за офлајн анализу сложених система. Неочекивано је да Чебишев 2 остварује изненађујуће добре резултате на нелинеарном Van der Pol осцилатору (11.74 dB побољшање), што показује да традиционални IIR филтери могу бити довољни за многе индустријске сигнале који се сматрају нелинеарним.

Табела 4. Компаративна анализа између класа филтара на 3 различита сигнала

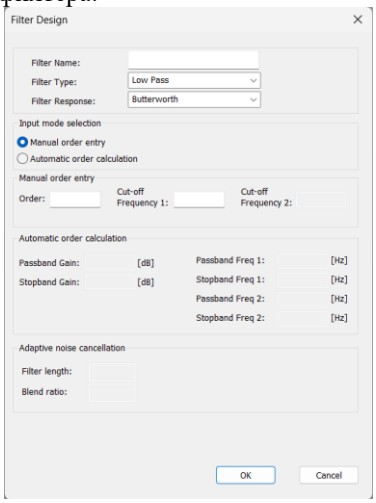
Сигнал	Филтер	MSE	SNR побољшање	Вријеме
Синус (50 Hz)	SIR	1.19e-01	-1.24	5.7084
	Chebyshev	1.59e-02	7.50	0.0079
	RLS	5.89e-01	-8.19	0.2137
Chirp (10-200 Hz)	SIR	9.01e-02	6.27	7.8147
	Chebyshev	3.58e-01	0.27	0.0018
	RLS	9.73e-02	5.94	0.1894
Ван дер Пол осцилатор	SIR	2.13e-02	8.31	8.1008
	Chebyshev	9.67e-03	11.74	0.0024
	RLS	3.05e+00	-13.25	0.2562

5. РЈЕШЕЊЕ У KESTUDIO SCOPE-У

Идеја је била да се првобитно у постојеће рјешење софтверског осцилоскопа додају филтри са бесконачним импулсним одзивом, а након тога а се дода један експериментални филтер, који је комбинација адаптивног и нелинеарног филтра.

5.1. Имплементација

Филтери су имплементирани у C++ за реално време и офлајн обраду. Класични *IIR* филтери користе аналогне прототипове са билинеарном трансформацијом. Развијен је хибридни адаптивни модел који интегрише NLMS и честични филтер, омогућавајући пондерисану комбинацију резултата.
Кориснички интерфејс са Слике 4 омогућава интуитивно подешавање параметара филтера са аутоматским одређивањем реда на основу захтјева за потискивањем сигнала. Хибридни режим дозвољава фино подешавање односа тежине између NLMS и честичног филтера.



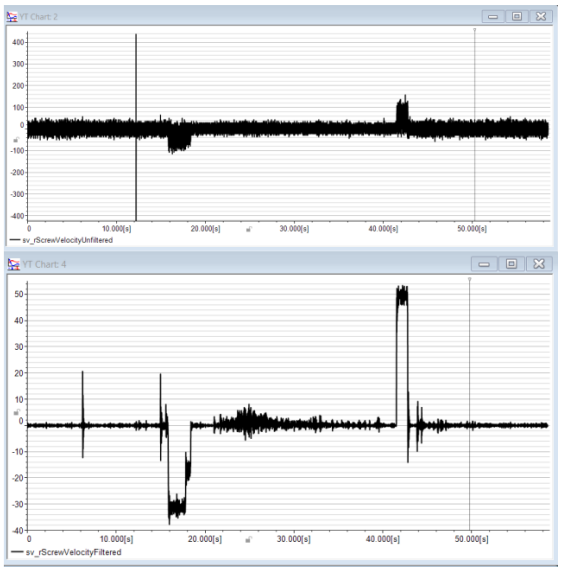
Слика 4. Кориснички интерфејс

5.2. Тестирање

Валидација је извршена на аутентичним индустријским сигналима. Резултати показују успјешно смањење шума без губитка битних динамичких компоненти. Посебно корисно за диференцијацију сигнала где се шум природно појачава. Чебишев II филтер са граничном фреквенцијом 30 Hz ефикасно уклања шум из диференцираног сигнала позиције, омогућавајући тачно мјерење брзине. На Слици 5 се види диференцирани сигнал позиције на горњој слици, са јако великом амплитудом шума, затим филтриран поменутим филтром.

6. ЗАКЉУЧАК

Рад представља систематску компаративну анализу три фамилије филтера за индустријску примјену. *IIR* филтери пружају ефикасно рјешење за стационарне сигнале са Чебишев II као најбољим избором. Адаптивни филтери су погодни за промењиве услове са RLS као најтачнијим, али рачунски захтевним. Нелинеарни филтери су неопходни за сложене системе са честичним филтрима као најфлексибилнијим решењем.
Успјешна интеграција у *KeStudio Scope* потврђује практичну вриједност истраживања. Хибридни приступ омогућава комбиновање предности различитих метода, чинећи систем прилагодљивим широком спектру индустријских примјена.



Слика 5. Сигнал прије и послије филтрирања Чебишев филтром у KeStudio Scope-у

7. ЛИТЕРАТУРА

[1] M. Shouran and E. Elgamli, "Design and implementation of Butterworth filter," *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 9, pp. 7975, 2020.

[2] M. Chavan, R. Agarwala, and M. Uplane, "Comparative study of Chebyshev I and Chebyshev II filter used for noise reduction in ECG signal", Jan. 2008.

[3] J. Okoekhian, M. Guiawa, and I. Onyegbadue, "Design and simulation of adaptive filters in real-time environment," *Int. J. Res. Rev.*, vol. 11, no. 11, pp. 215–238, 2024. doi: 10.52403/ijrr.20241118.

[4] J. Benesty and T. Gänsler, "New insights into the RLS algorithm," *EURASIP J. Adv. Signal Process.*, vol. 2004, no. 3, p. 243857, 2004. doi: 10.1155/S1110865704310188.

[5] A. Kutschireiter, S. C. Surace, and J.-P. Pfister, "The hitchhiker's guide to nonlinear filtering," *J. Math. Psychol.*, vol. 94, p. 102307, 2020. doi: 10.1016/j.jmp.2019.102307.

Кратка биографија:

Анђела Лакић рођена је у Градишци 2000. год. Мастер студије на Факултету техничких наука из области Рачунарства и аутоматике започиње 2023. године.
Контакт: lakic.andjela@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 004.9:519.83
ДОИ: https://doi.org/10.24867/35BE08Mirosavic

Систем за адаптацију стратегије у понављеним неодређеним играма нулте суме са два играча

System for Strategy Adaptation in Repeated Two-Player Zero-Sum Uncertain Games

Ђорђе Миросавић, *Факултет техничких наука, Нови Сад*

Студијски програм – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Тема овог рада је прављење платформе која учи понашање човека у игри папир-камен-маказе, и сходне томе адаптивно бира стратегију помоћу које га побеђује. У првом делу рада биће приказане стратегије које ће се користити, прављења софтверске и хардверске платформе. У другом делу ће бити приказани резултати експеримената.*
Кључне речи (три до пет): *Теорија игара, папир-камен-маказе, Q учење, ESP32*

Abstract – *The topic of this paper is the creation of a platform that learns human behavior in the game rock-paper-scissors and, accordingly, adaptively selects a strategy to defeat them. The first part of the paper will present the strategies to be used, and the development of the software and hardware platform. The second part will present the results of the experiments.*

Keywords: (three to five): *Game theory, rock-paper-scissors, Q learning, ESP32*

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

1. УВОД

Игра папир камен маказе вуче корене још из периода династије Хан (200 година п.н.е.). Касније је игра стигла у Јапан, где се често називала *sansukumi-ken*, што у преводу значи "игра тројице, од којих се свако плаши једнога". Једна од *sansukumi-ken* игара је *Mushi-ken*, у којој су постојале 3 гестикулације руком, где једна побеђује другу. По изгледу, највише подсећа на тренутну игру папир-камен-маказе.

Уколико стриктно посматрамо проблем са теоријске стране, нема велике разлике који знак ћемо изабрати, пошто увек постоји супротан знак који га може победити. И уколико се примени Нешов еквилибријум на овај проблем, добија се да је најбоља стратегија по ком бисмо бирали потезе та да играмо насумично. И уколико имамо 2 играча који играју насумично, онда је средња добит (награда) коју оба играча могу да добију је једнака нули. Односно, након довољно великог броја партија, број победа, ремија и пораза би био изједна- чен. Међутим, човек није најбољи када је у питању насумичност. Постоји доста студија где је посматрано насумично бирање бројева, и када би се бирао број између 1 и 10, најчешће су били бирани бројеви 3 и 7. Људи заправо не могу да генеришу насумичне бројеве [1]. Такође, у једном од радова су показали да могу успешно да идентификују појединца (преко 95% успешнсти) само на основу 300 цифара које је он изабрао. [2] Слична ствар се уочила и у игри папир камен маказе, где постоји тенденција да се чешће бира камен него остали знакови.[4] Сходно томе, у овом раду ће фокус бити на прављењу агента који учи понашање човека у игри папир камен маказе, покушава да пронађеману у његовој игри и искористи је.

2. ПОСТАВКА ПРОБЛЕМА

Систем који ми посматрамо је игра папир-камен-маказе, у ком човек игра против агента. Акције које имају на располагању су папир, камен и маказе. Стање система можемо да дефинишемо на произвољно начина (уређен пар претходна акција играча - агент, претходних N акција игара, исход претходне партије,...). Награде које могу да добију су победа, нерешено или пораз. Правила игре су да папир побеђује камен, камен побеђује маказе, маказе побеђују папир. У случају да су одигране идентичне акције, резултат је нерешен. Када се игра уживо, оба играча показују акцију у исто време. У нашем случају, агент ће своју акцију испоручити пре него што корисник изабере своју, али ће се оба приказати након корисниковог избора. Агент ће имати на располагању више стратегија на основу којих може да игра. Након сваке акције играча, оне се рангирају и бира се она са највећом вероватноћом за победу у наредној партији. Интеракција са агентом је обезбеђена на 2 начина: *Flask* веб апликација и хардверска платформа базирана на *ESP32* контролеру.

3. Q-УЧЕЊЕ

Једна од стратегија коју користи јесте Q-учење. Уколико је агент у стању s одиграо акцију a, и за њу

добио награду r, ажурирање вредности коришћењем алгоритма Q учења би гласило:

$$Q^{+}(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha\left[r + \gamma \cdot \max_{a'} Q(s',a')\right], \quad \alpha,\gamma \in [0,1] \tag{1}$$

Помоћу параметра учења α можемо да подешавамо колико брзо ће алгоритам реаговати на награде које добија. Први члан је претходна вредност акције, а други представља Белманову једначину. Заједно чине аналогију са филтром првог реда, где уколико желимо стабилно учење агента (са мање шума), параметар α бисмо подесили ближе нули. Након реформулације чланова, израз се чешће јавља у наредној форми:

$$Q^{+}(s,a) = Q(s,a) + \alpha\left[r + \gamma \cdot \max_{a'}(s',a') - Q(s,a)\right], \quad \alpha,\gamma \in [0,1] \tag{2}$$

Ова стратегија је нарочито ефикасна уколико за стање узмемо претходну награду коју је добио, а акцију играча интепретирамо помоћу „смера акције“. Смер акције је „понављање“ уколико су тренутна и претходна акција исте. Посматрајући редослед акција, смер „напред“ је уколико акција папир претходи камену, или из камена следе маказе, или из маказа следи папир. Смер „назад“ је обрнуто. Видећемо у резултатима да је овакво дефинисање акција у значајној мери допринела успешности агента.

3.1. Серијско Q учење

Један од начина како можемо да учење учинимо стабилнијим јесте да ажурирање вредности радимо након N партија. Формирамо листу искустава, односно, чувамо стања, изабране акције и награде. Касније у редоследу чувања података сукцесивно применимо ажурирање табеле.

3.2. Инкрементално Q учење

За разлику од обичног Q учења, овде се ажурирање табеле ради тако што формирамо речник где је кључ уређен пар (s, a) из листе искустава. Уколико запис (s, a) не постоји у табели, вредност $\mathrm{Q}(s,a)^{+}$ је средња вредност свих добијених награда за дато (s, a), тј. $Q_{\text{средње}}(s,a)$. У супротном, вредност је дата као:

$$Q^{+}(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha \cdot Q_{\text{средње}}(s,a) \tag{3}$$

3.3. CAPCA (*eng. State Action Reward State Action*) алгоритам

Белманова једначина узима у обзир највећу вредност акције наредног стања. Док CAPCA бира вредност акције која би била изабрана у том стању. Разлика се примећује уколико је политика одлучивања епсилон-похлепна, с обзиром да неће увек бити изабрана вредност акције са највећом вредности.

$$Q^{+}(s,a) = Q(s,a) + \alpha[r + \gamma \cdot Q(s',a') - Q(s,a)] \tag{4}$$

3.4. Двоструко Q учење

Уколико применимо Јенсенову неједнакост над елементом за највећу Q вредност наредног стања (то можемо да урадимо пошто је функција максимума конвексна функција) добија се:

$$\mathrm{E}\left[\max_{a'} Q(s',a')\right] \geq \max_{a'} \mathrm{E}[Q(s',a')] \tag{5}$$

Ово нам говори да ако се много пута нађемо у истом стању, очекивана (средња) вредност будућег стања у које се прелази ће бити увек већа или једнака највећој очекиваној вредности. То говори да прецељујемо (*eng. overestimating)* колико је добро наредно стање у које ћемо отићи. Стога уводимо 2 табеле, где случајним одабиром бирамо коју ажурирамо након партије.

$$Q_1^{+}(\mathrm{s},\mathrm{a}) = (1-\alpha) * Q_1(s,a) + \alpha\left[r + \gamma \max_{a'} Q_2(s',a')\right] \tag{6}$$

$$Q_2^{+}(\mathrm{s},\mathrm{a}) = (1-\alpha) * Q_2(s,a) + \alpha\left[r + \gamma \max_{a'} Q_1(s',a')\right] \tag{7}$$

Вредност акције $\mathrm{Q}(\mathrm{s},\mathrm{a})$ је средња вредност обе табеле.

3.5. Дубоко Q учење

Једно од ограничења Q учења настаје када систем у ком се налазимо има велики број стања. Када се нађе у новом стању, алгоритам нема искуства о том стању и коју акцију би требало да одигра, тако да враћа насумичну. Уколико се приликом тренинга алгоритам не нађе у неком од стања, онда можемо да имамо неочекивано понашање у њима. Стога се уместо табеле користе 2 неуронске мреже. Једна за одређивање тренутне а друга на наредну вредност стања, које се изједначавају периодично. У раду је коришћена потпуно повезана секвенцијална мрежа са једним скривеним слојем. Надоградњу представља двоструко дубоко Q учење, које има значајан потенцијал у побеђивању човека у сложенијим играма. [3, 5]

4. Остале стратегије

4.1. Марковљева стратегија

Стратегија формира матрице прелаза из стања у акцију коју је изабрао играч. Свако поље матрице представља број појављивања акције у датом стању. Акција може да се бира епсилон-похлепном или политиком базираној на вероватноћи појављивања акција у стању.

4.2. Стратегија тражења секвенци

Посматра се секвенца претходних N акција играча. Проласком кроз историју потеза, бележимо колико пута је играч изабрао коју акцију након дате секвенце. Фактором заборављања више уважавамо скорије изабране акције.

4.3. Остале стратегије

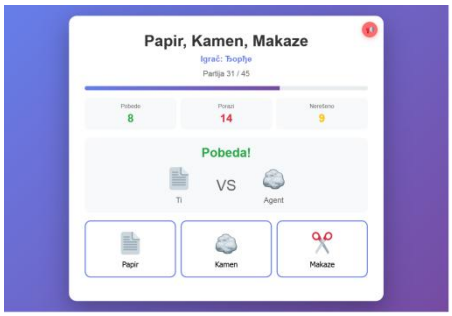
Поред споменутих, још неке од стратегија укључују коришћење неуронске мреже, посматрање

најкоришћеније акције играча, вероватноћа бирања акције, насумична...

За рангирање стратегија коришћен је UBC (*Upper Bound Confidence*) алгоритам. Параметри који се подешавају су фактор заборављања γ, ширина прозора (број посматраних партија) и одабир насумичне стратегије са вероватноћом ε.

5. Аквизиција података

За потребе извођења експеримената и прикупљања података креирана је веб апликација као и хардверска платформа.



Слика 1. Интерфејс веб апликације

Апликација је писана у *Flasku*, док је за бележење података коришћена *NoSQL* база података. Хардверска имплементација је базирана на ESP32 контролеру и SH1107 ОЛЕД екрана коришћењем *MicroPython* језика.

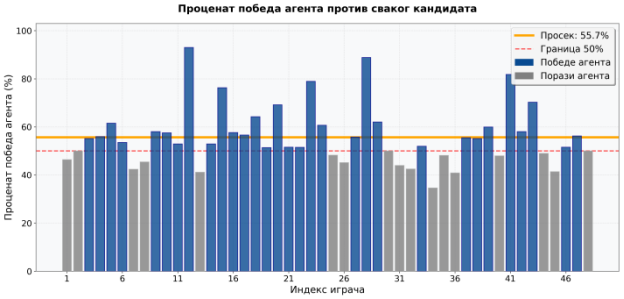
Слика 2. Хардверска платформа

6. Експеримент

У првом експерименту је коришћено 28 стратегија. Од тога је 8 било Q учење (класично и инкрементално). Ту су стања представљала једна и две претходне акције играча, један и два претходна смера акције играча, као и претходна награда играча. Код инкременталног учења, период ажурирања табеле је било 10 и 15 партија. 2 стратегије су биле CAPCA. Једно стање су представљала преходни смер акције, а друго претходна награда. 2 стратегије су биле праћење омиљене акције и смера акције играча, унутар прозора од 10 потеза са фактором заборављања 0.97. Марковљеви процеси су имали 4 стратегије где се пратила претходна акција играча уз фактор заборављања од 0.95, 0.97 и 0.995. Насумична стратегија са вероватноћама је имала 4 стратегије, од којих су 2 пратиле акције, а 2 смер акције играча. И ова случаја је једна имала фактор заборављања 0.95. 7 стратегија је било праћење секвенци, од чега је пола имало фактор заборављања 0.98. Пратиле су акције и смер акције играча, у оквиру прозора ширине 2 и 3 партије. Последња стратегија је била неуронска мрежа са 1 скривеним слојем, 3 неурона унутар њега уз фактором учења 0.04. UBC је имао епсилон-похлепну политику одлучивања са $\varepsilon = 0.05$, фактором заборављања $\gamma = 0.98$, и ширином прозора од 10 партија.
Платформа на којој су играчи играли је била веб апликација. Број партија које је играч требао да одигра је 45. Након тога може да настави игру произвољно дуго с тим да на сваких 30 партија добија питање да ли жели да настави даље. Број кандидата који је учествовао је 48. Укупан број одиграних мечева је 80.

7. Резултати

За почетак посматрамо само први одигран меч играча. Просечан број партија изражен аритметичком средином је 49.06, а медијаном 45.



Слика 3. Приказ исхода мечева

Проценат победа ћемо дефинисати као однос победа и укупног броја победа и пораза. Нерешене резултате нећемо разматрати. На слици 3 смо приказали све мечеве уз забележен проценат победа, а у табели 1 смо изразили исходе свих мечева. Агент је однео победу у 64.5% мечева.

Табела 1. Исход агента у првим одиграним мечевима

Победе	Нерешено	Порази	Укупно мечева
31	3	14	48

Аритметичка средина података свих мечева је $\mu = 55.7$ медијана $\tilde{\mu} = 53.2$, док је стандардна девијација $\sigma = 12.1$. Након уклањања резултата изнад $\mu + 2 * \sigma$, добијамо да је $\mu = 53.56$, $\tilde{\mu} = 52.9$, $\sigma = 9$.

Табела 2. Исход агента у првим одиграним партијама

Победе	Нерешено	Порази	Укупно партија
832	686	712	2230

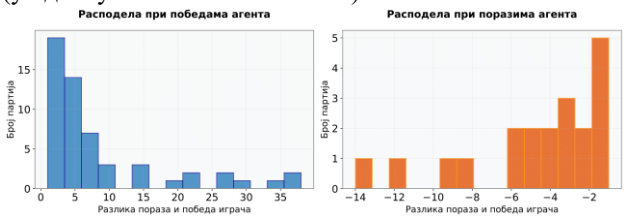
Интервал поверења за средњу вредност на нивоу поверења 95% износи $[50.83\%, 56.3\%]$. Уколико је ниво поверења 99%, онда интервал износи $[49.91\%, 57.22\%]$. Ово значи да уколико имамо велики број мечева трајања 45 партија, агентов проценат победа ће се кретати у датом опсегу. Што значи да би агент практично увек био успешнији од играча.

Посматрајмо сада свих 80 одиграних мечева. Овде је проценат победа био 68.7%.

Табела 3. Исход агента у свим одиграним партијама

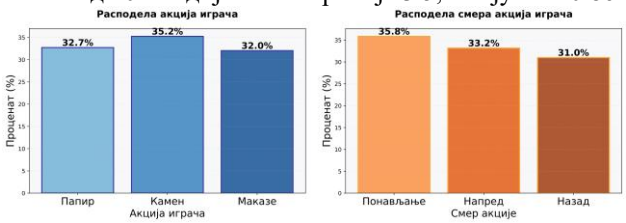
Победе	Нерешено	Порази	Укупно партија
1786	1326	1365	4477

Проценат победа по партијама је већи и износи 56.6% (у односу на 53.8% из табеле 2).



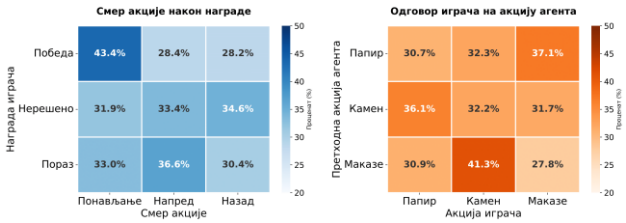
Слика 4. Убедљивост агента при победама и поразима по мечевима

Са слике 4 имамо увид у разлику са којом је агент побеђивао и губио. Када је агент побеђивао, медијана је +5, тј. имао је 5 победа више него пораза, а скјунис метрика је 1.13. Видимо да је постојало пар мечева где је агент имао предност од 20 или више победа, али то се дешавало када је играч више пута продужавао наставак игре. Посматрањем пораза агента, примећујемо да је постојало пар играча који су приметилиману у игри и побеђивали га са разликом већом од 10. Медијана за поразе је 3.5, а скјунис -0.86.



Слика 5. Приказ расподела акција играча у свим партијама

Посматрајући најчешћу акцију коју су играчи бирали, видимо да је то камен, што се поклапа са резултатима осталих великих експеримената који су рађени [4]. Као смер акције, доминантно је понављање претходно одигране акције. Такође, чешће се бира акција која је позиционирана десно односу на претходну акцију, него лево.



Слика 6. Матрице прелаза

На слици 6 смо приказали 2 матрице прелаза. Прва приказује прелаз из награде играча у смер одигране акције. Уколико је играч победио, интересантно је да у 43.4% случајева понавља исту акцију. Матрица десно, приказује одговор играча спрам претходне акције агента. Можемо приметити да је доминантно понашање играње акције која ће победити тренутну акцију агента. Ово је наизраженије код агентове акције Маказе, након које играч у 41.4% бира Камен. У табели 4 смо издвојили стратегије које су донеле највише победа агенту. Праћењем смера акције играча, можемо да приметимо највећи број победа.

Табела 4. Најуспешније стратегије

Назив стратегије	Број победа	Број ремија	Број пораза
Q учење (награда->смер)	152	85	77
Q учење (смер->смер)	148	95	98
Марковљева (награда->смер)	124	51	70
Најкоришћенији смер	123	47	54
Q инкрементално учење (смер->смер)	84	62	57

8. ЗАКЉУЧАК

У овом раду смо показали да је могуће направити систем који ће победити човека у игри папир-камен-маказе. Посматрајући први одигран меч играча, проценат победа агента је 64.5%, док у појединачним партијама 53.5%. Ово је значајан резултат узимајући у обзир да је просечан број одиграних партија (изражен медијаном) био свега 45.

9. ЛИТЕРАТУРА

[1] Figurska M, Stańczyk M, Kulesza K. (2008). *"Humans cannot consciously generate random numbers sequences: Polemic study"*

[2] Schulz, MA., Baier, S., Timmermann, B. (2021). *"A cognitive fingerprint in human random number generation"*

[3] Hasselt, H., Guez, A., & Silver, D. (2015). *"Deep Reinforcement Learning with Double Q-learning"*

[4] Wang, Z., Xu, B. & Zhou, HJ. (2014). *"Social cycling and conditional responses in the Rock-Paper-Scissors game"*

[5] Kumar, D. (2024). *"DDQN: Tackling Overestimation Bias in Deep Reinforcement Learning"*

Кратка биографија:

Ђорђе Миросавић рођен је у Ваљеву 2000. године. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства одбранио је 2025.год.
Контакт:
mirosavicdjordje@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.41
DOI: https://doi.org/10.24867/35BE09Slovic

WEB APLIKACIJA ZA PODRŠKU RADA OFF-GRID SOLARNIH ELEKTRANA I BATERIJSKIH SKLADIŠTA

WEB APPLICATION FOR SUPPORTING OFF-GRID SOLAR POWER PLANTS AND BATTERY STORAGE

Lana Slović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu prikazan je sistem za upravljanje off-grid solarnim elektranama i baterijskim skladištima. Sistem omogućava odabir lokacija za postavljanje solarnih panela i baterija, izračunavanje količine električne energije koju proizvode solarni paneli, prikaz dijagrama potrošnje, proizvodnje energije i nivoa napunjenosti baterija, kao i predikciju vremenskih uslova pomoću metoda mašinskog učenja, na osnovu kojih se procenjuje buduća proizvodnja energije u panelima. Cilj ovog rada je doprinos razvoju rešenja za upravljanje obnovljivim izvorima energije, koja omogućavaju efikasnije korišćenje resursa, smanjenje zavisnosti od fosilnih goriva i povećanje energetske održivosti.*

Ključne reči: *Obnovljivi izvori energije, solarni paneli i baterijska skladišta, veb, mašinsko učenje, predikcije*

Abstract – *This paper presents a system for managing off-grid solar power plants and battery storage facilities. The system enables the selection of locations for installing solar panels and batteries, calculation of the amount of electricity generated by the solar panels, display of consumption diagrams, energy production, and battery charge levels, as well as weather forecasting using machine learning methods to estimate future energy production by the panels. The aim of this paper is to contribute to the development of solutions for managing renewable energy sources, enabling more efficient resource utilization, reducing dependence on fossil fuels, and increasing energy sustainability.*

Keywords: *Renewable energy sources, solar panels and battery storage, web, machine learning, predictions*

1. UVOD

Sve veća potražnja za održivim izvorima energije u poslednjim decenijama podstakla je razvoj tehnologija obnovljivih izvora, poput solarnih elektrana. U Srbiji je uočljiv napredak u ovoj oblasti, što potvrđuje rast broja domaćinstava koja koriste solarne panele i doprinose energetskoj stabilnosti i nezavisnosti kroz primenu obnovljivih izvora energije.

Ovaj rad istražuje razvoj web aplikacije za upravljanje off-grid solarnim elektranama i baterijskim skladištima.

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

Cilj aplikacije je omogućiti korisnicima da u realnom vremenu prate proizvodnju, potrošnju i skladištenje energije, uz analizu i vizualizaciju podataka. Aplikacija koristi savremene tehnologije poput mašinskog učenja, pružajući alate za predikciju i optimizaciju sistema. Krajnji cilj je razvoj softverskog rešenja koje doprinosi efikasnijem korišćenju resursa i širenju primene obnovljivih izvora energije.

2. OSNOVNI POJMOVI

2.1. Obnovljivi izvori energije i solarna energija

U 21. veku, sve veća svest o ograničenjima i ekološkim posledicama korišćenja fosilnih goriva dovela je do razvoja obnovljivih izvora energije (OIE) poput solarnih, vetroelektrana, hidroelektrana i geotermalnih izvora, koji obećavaju čistu energiju i energetsku nezavisnost. Ipak, OIE imaju ograničenja, poput niske gustine energetskih tokova (solarno zračenje u proseku 150–250 W/m², a vetar i voda oko 500 W/m²) i njihove promenljivosti, što zahteva značajne troškove za opremu za prikupljanje i skladištenje energije. Solarni paneli zauzimaju vodeću poziciju među OIE, omogućavajući efikasnu konverziju solarne energije u električnu energiju, toplotu i hlađenje. Iako njihova instalacija ima visoke troškove, tehnologije poput fotonaponskih (PV) pretvarača sa efikasnošću od 15–17% nalaze sve širu primenu. PV sistemi u srednjim geografskim širinama proizvode 900–1500 kWh/kWp godišnje, odnosno 120–200 kWh/m² godišnje, čineći ih ključnim za prelazak na održiviju energetsku budućnost [1].

2.2. Prikupljanje i obrada podataka

Kako bi se dobili verodostojni rezultati, neophodno je prikupiti precizne podatke o vremenskim uslovima na osnovu kojih će se računati proizvodnja energije i vršiti predikcije. Podaci koji se prikupljaju uključuju trenutnu temperaturu, procenat oblačnosti, vreme izlaska i zalaska Sunca, a prikupljaju se periodično, na svakih sat vremena, za željenu lokaciju na osnovu njenih koordinata. Pored prikupljenih podataka, potrebni su i određeni konstantni parametri koji igraju ključnu ulogu u izračunavanju proizvodnje električne energije.
S obzirom na to da se radi o solarnim panelima, tokom noći (između momenta zalaska i momenta izlaska Sunca) količina proizvedene električne energije biće jednaka nuli, dok će se u suprotnom koristiti formule (1), (2) i (3) za izračunavanje trenutne količine električne energije proizvedene u solarnim panelima.

$$T_{celija} = T_{spoljna} + k \cdot (1 - C) \quad (1)$$

$$f(Tcelija) = 1 - \beta \cdot |T_{celija} - T_{referentna}| \quad (2)$$

$$E_{trenutno} = N_{panela} \cdot P_{instalisano} \cdot \eta \cdot (1 - C) \cdot f(T_{celija}) \cdot t \quad (3)$$

Parametri:

• k – koeficijent uticaja oblačnosti na temperaturu solarnih ćelija, upotrebljena vrednost je 5˚C

• β – odražava uticaj temperature na efikasnost, upotrebljena vrednost je 0,005˚C⁻¹

• Treferentno – temperatura pri kojoj se efikasnost solarnih panela ne menja, upotrebljena vrednost je 25˚C

• η – efikasnost solarnih panela, upotrebljena vrednost je 80%

Uslovi zajedničkog funkcionisanja solarnih panela i baterijskih skladišta u odnosu na potrošnju su sledeći:

• Ukoliko solarni paneli proizvedu više energije nego što se potroši, tada se viškom energije puni baterijsko skladište, ukoliko ima kapaciteta.

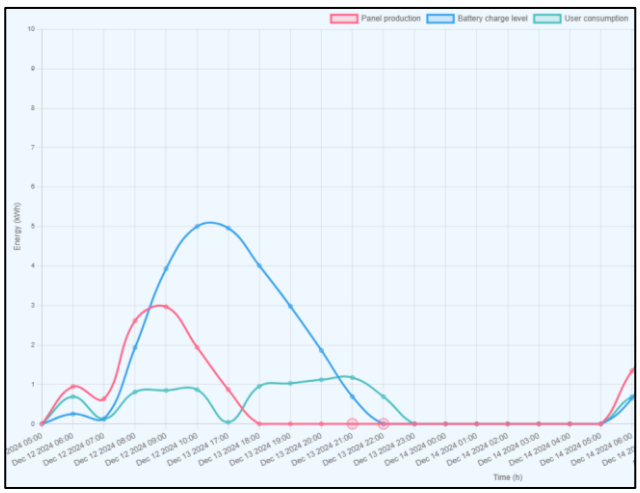
• Ukoliko solarni paneli proizvedu manje energije nego što se troši, tada se neophodna energija crpi iz baterijskih skladišta, ukoliko skladišta poseduju tu količinu energije.

• Ukoliko je baterijsko skladište prazno, a proizvodnja energije je manja od predviđene potrošnje, tada je potrošnja energije jednaka proizvodnji energije.

• Ukoliko je baterijsko skladište puno, a proizvodnja energije je veća od predviđene potrošnje, tada višak proizvedene energije curi.

Uz navedene formule i uslove izračunavanja, dobija se prikaz proizvodnje električne energije, nivoa napunjenosti baterije i potrošnje električne energije za jedan sistem.

Vrednosti izračunate na osnovu navedenih formula i uslova se čuvaju u bazi podataka. Ponašanje sistema se najbolje može ispratiti formiranjem dijagrama na kom se prikazuju sledeće tri vrednosti: trenutna potrošnja energije, trenutna proizvodnja energije i nivo napunjenosti baterije u određenom trenutku. Primer takvog dijagrama je prikazan na slici 1.



Slika 1. *Dijagram proizvodnje, potrošnje energije i nivo napunjenosti baterije*

2.3. Mašinsko učenje i predikcije

Mašinsko učenje se kontinuirano razvija u svetu informacionih tehnologija i dobija na značaju u različitim poslovnim sektorima. Popularno je među svim tehnologijama. To je oblast proučavanja koja omogućava računarima da automatski uče i poboljšavaju svoje performanse na osnovu iskustva. Definiše se kao tehnologija koja se koristi za obučavanje mašina da izvršavaju različite radnje, kao što su predviđanja, preporuke, procene itd., na osnovu istorijskih podataka ili prethodnog iskustva. Mašinsko učenje se stoga fokusira na jačanje kompjuterskih programa uz pomoć prikupljanja podataka iz različitih opažanja. Omogućava računarima da se ponašaju kao ljudska bića, obučavajući ih uz pomoć prethodnog iskustva i prediktivnih podataka [2].

Random Forest je jedan od najpoželjnijih algoritama mašinskog učenja. Slično KNN-u (K Nearest Neighbour) i stablu odlučivanja, on takođe omogućava rešavanje klasifikacionih kao i regresionih problema, ali se preferira kada postoji potreba za rešavanjem složenijih problema i poboljšanjem performansi modela. Algoritam random forest se bazira na konceptu ansambla učenja, što je proces kombinovanja više klasifikatora.

Klasifikator random forest se sastoji od kombinacije više stabala odlučivanja. Ova kombinacija uzima u obzir prosečnu predikciju svih stabala i poboljšava tačnost modela. Veći broj stabala u šumi dovodi do veće tačnosti i sprečava problem preterivanja. Dodatno, zahteva manje vremena za obuku u poređenju sa drugim algoritmima [2].

3. ARHITEKTURA SISTEMA

Višeslojna arhitektura ove aplikacije ima jasno razdvojene slojeve za frontend, backend i bazu podataka. Za dodavanje podataka o geografskim lokacijama i vremenskim svojstvima, aplikacija komunicira sa postojećim serverima kao što su OpenWeatherMap i GoogleMapsAPI.

3.1. Backend sloj

Servisni deo aplikacije koristi *Node.js*, *Express.js* za definisanje ruta i *Python*. Sadrži *mongoose* modele. Sastoji se od mikroservisa koji imaju komunikaciju sa svojim bazama podataka i međusobno komuniciraju preko *REST API*-ja. Takođe, servis prikuplja podatke sa *OpenWeatherMap* .

Mikroservisna arhitektura:

- **UserAuthService**: Za prijavu i registraciju korisnika.
- **UsersService**: Upravljanje operacijama nad korisnicima (blokiranje, pretraga).
- **PanelsService**: Obrada operacija nad panelima (dodavanje, brisanje, proizvodnja energije).
- **BatteriesService**: Upravljanje nad baterijama (dodavanje, brisanje, ažuriranje).
- **ConstantParametersService**: Dodavanje i upravljanje konstantnim parametrima za proizvodnju energije.
- **EnergyConsumptionService**: Upravljanje podacima o potrošnji energije.

- **HistoryDataService**: Čuvanje istorijskih podataka o proizvodnji i vremenu.
- **ForecastPredictionService**: Python mikroservis. Predikcija proizvodnje energije koristeći algoritam mašinskog učenja RandomForest.

3.2. Frontend sloj

Klijentski deo aplikacije razvijen je u *React.js* i sadrži komponente i servise koji komuniciraju sa mikroslužbama na serverskoj strani. Koristi *React-Leaflet* biblioteku za izbor lokacije na mapi i poziva *OpenWeatherMap* za prikaz trenutnih vremenskih parametara.

3.3. Baza podataka

Za skladištenje podataka koristi se *MongoDB*. Svaki mikroservis ima svoju bazu podataka u kojoj se čuvaju informacije o korisnicima, panelima, baterijama, parametrima i istorijskim podacima o proizvodnji, potrošnji i nivou napunjenosti baterije.

S obzirom na to da mikroservisi međusobno komuniciraju preko *REST API*-ja, tim putem se propagiraju promene između njih kako bi se postigla konzistentnost sistema. Primer: UsersService ima funkcionalnost za blokiranje/odblokiranje korisnika, gde menja vrednost polja u bazi koje određuje da li je korisnik blokiran ili ne. Isto to polje je relevantno za UserAuthService, jer se na osnovu njega dozvoljava korisniku prijava na sistem. Iz tog razloga, UsersService nakon promene vrednosti navedenog polja, šalje UsersAuthService servisu informaciju o novoj vrednosti polja i UsersAuthService ažurira podatke u svojoj *MongoDB* bazi.

4. PREDIKCIJE VREMENSKIH USLOVA I IZRAČUNAVANJE BUDUĆE KOLIČINE PROIZVODNJE ELEKTRIČNE ENERGIJE NA OSNOVU NJIH

Mikroservis za predikciju vremenskih uslova predstavlja posebnu Python aplikaciju, koja komunicira sa drugim servisima koristeći Flask API. Koristi napredne tehnike mašinskog učenja, i putem Random Forest algoritma vrši obradu istorijskih podataka o temperaturi i oblačnosti. Model koristi podatke o trenutnim i prethodnim temperaturama, kao i o oblačnosti, da bi učio obrasce koji se javljaju tokom vremena. Da bi se modelu omogućilo da koristi istorijske informacije, podaci o temperaturi i oblačnosti se "laguju". Ovo znači da se dodaju prethodne vrednosti temperature i oblačnosti iz poslednjih 48 časova kao nove karakteristike u skupu podataka. Time se omogućava modelu da prepozna obrasce u podacima koji su se dogodili u prošlosti i da ih koristi za predikciju budućnosti. Na osnovu pripremljenih podataka, model se obučava korišćenjem algoritma mašinskog učenja, poput Random Forest regresora. Ovaj algoritam analizira obrasce u podacima i uči kako da predviđa buduće temperature na osnovu ulaznih karakteristika (lagovanih temperatura i oblačnosti). Obučavanje uključuje deljenje podataka na obučeni i test skup, pri čemu se model optimizuje da bi postigao što tačnije predikcije. Nakon obučavanja, model može da predviđa temperature za narednih 24 časa. Na kraju, rezultati se dobijaju kao niz predviđenih temperatura i oblačnosti za naredni period.

5. IMPLEMENTACIJA PREDIKCIJA VREMENSKIH SERIJA

Implementacija predikcija budućih vremenskih serija obuhvata tri fukcije: za pripremu podataka, treniranje modela i predikciju budućih podataka. Biblioteke neophodne za implementaciju su *numpy* i *scikit-learn*, koje su neizostavan deo rada sa podacima i modelima mašinskog učenja. Podaci se učitavaju iz CSV fajla koji sadrži istorijske informacije, a zatim se pripremaju u funkciji prepare_data prikazane u listingu 1.

```
def prepare _data(data, target_column):
#Dodavanje lagovanih vrednosti za temperaturu i oblačnost
for i in range(1, 49):
       data[f'temperature_lag_{i}']=data['currentOutsideTemperature'].shift(i)
       data[f'cloudiness_lag_{i}']=data['currentCloudinessPercent'].shift(i)

data = data.dropna()
X = data[[f'temperature_lag_{i}' for i in range(1,49)] + [f'cloudiness_lag_{i}' for i in range(1, 49)]]
    y = data[target_column]

    return X, y
```

Listing 1. *Funkcija za pripremu podataka za predikcije*

Nakon pripreme podataka sledi treniranje modela a zatim i predikcija budućih vrednosti na osnovu primrepljenih podataka u fukciji prepare_data. Funkcije train_model i predict_future_data su prikazane u listingu 2.

```
def train_model(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1,random_state=42)

model=RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
try:
   check_is_fitted(model)
   print("Model je uspešno treniran.")
except:
   print("Model nije treniran.")

return model, X_test, y_test
```

```
def predict_future_data(model, data):

last_24_data=data.iloc[1][[f'temperature_lag_{i}' for i in range(1, 49)] + [f'cloudiness_lag_{i}' for i in range(1, 49)]].values.reshape(1, -1)

future_data = []

for _ in range(48):
   predicted_data=model.predict(last_24_data)
   future_data.append(predicted_data[0])
```

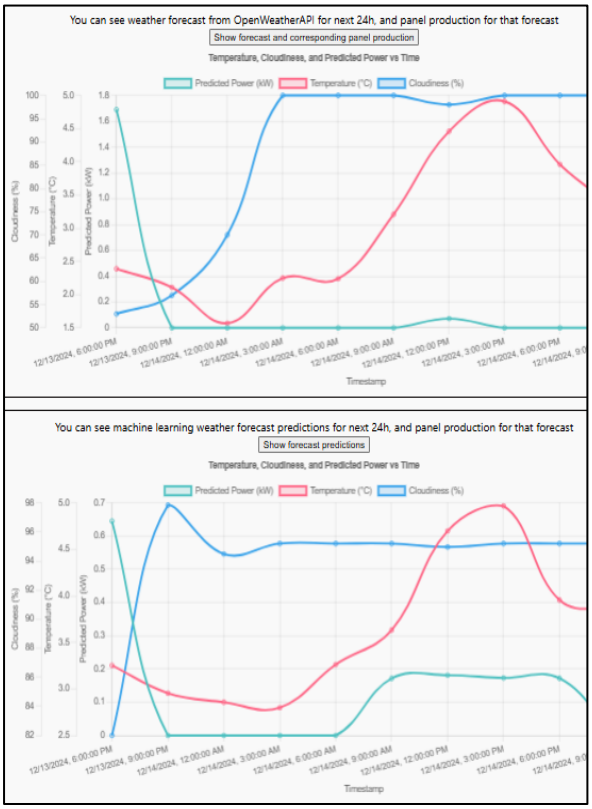
```
    last_24_data = np.roll(last_24_data, shift=-1)
    last_24_data[0, -1] = predicted_data[0]

return future_data
```

Listing 2. *Funkcije za treniranje modela i predikcije budućih vrednosti*

Model kreiran i treniran na ovaj način, tokom evaluacije je pokazao da su vrednosti Root of the Mean of the Square of Errors (RMSE) i Mean of Absolute value of Errors (MAE) manje od 1, što dokazuje tačnost modela. Primer RMSE i MAE za neki od lokacija za koje su se vršile prognoze:

- RMSE: 0.6755029072419337
- MAE: 0.5621307692307664

Nakon treniranja modela, evaluacije modela, i predikcija vremenskih uslova, sledi izračunavanje buduće proizvodnje električne elergije za solarne panele na datoj lokaciji za koju se vršila predikcija. Izračunavanje se vrši na osnovu formula i parametara navedenih u drugom poglavlju, a prikaz rezultata predikcija se vrši putem dijagrama (buduće temperature, stepeni oblačnosti i količina proizvedene električne energije). Na slici 2 je prikazano poređenje vremenske prognoze preuzete sa *OpenWeatherMap API*-ja, izračunate proizvodnje energije na osnovu nje, vremenske prognoze putem mašinskog učenja i Random Forest algoritma, i njoj odgovarajuće izračunate proizvodnje energije.



Slika 2. *Vremenska prognoza preuzeta sa OpenWeatherMap API-ja (gore) i preikcije uz mašinsko učenje (dole)*

6. ZAKLJUČAK

Ovaj rad obrađuje razvoj sistema za upravljanje off-grid solarnim elektranama, kao i predikcije vremenskih uslova na osnovu kojih se dobija buduća količina električne energije koja će biti proizvedena. Uz pomoć prikupljanja podataka, korisničkog interfejsa i tehnologija mašinskog učenja omogućeno je praćenje proizvodnje i potrošnje električne energije, kao i nivoa napunjenosti baterije u sistemu solarnih panela i baterijskih skladišta, uz pružanje pomenutih predikcija. Samim tim, olakšano je praćenje sistema i dalje planiranje proizvodnje električne energije i korišćenja resursa u solarnim elektranama.

Rad doprinosi time što omogućava predviđanje budućih vremenskih uslova uz određenu tačnost predikcija, kao i buduće proizvodnje energije na osnovu tih uslova. Mikroservisna arhitektura sistema omogućava laku modifikaciju i nadogradnju sistema, kao i integraciju sa drugim sistemima, što ga čini fleksibilnim i prilagodljivim. Raznovrsnost tehnologija implementiranih u sistemu pokazala se kao efikasan način za postizanje preciznih rezultata. Rezultati dobijeni korišćenjem sistema ukazuju na visoku upotrebljivost algoritama mašinskog učenja u solarnim elektranama.

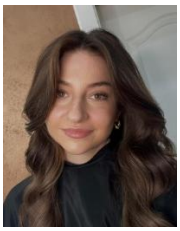
S obzirom na potencijal obnovljivih izvora energije, razvoj sistema za upravljanje ovakvim resursima i predikcije buduće proizvodnje električne energije doprinosi efikasnom korišćenju istih, smanjenju troškova i zavisnosti od neobnovljivih izvora energije. Osim što doprinosi energetskoj održivosti, ovakav sistem može poslužiti kao model za slične aplikacije u drugim sektorima baziranim na obnovljivim resursima.

4. LITERATURA

[1] Rakhmatov A., Primov O., Mamadaliyev M., Tòrayev S., Xudoynazarov U., Ulugmurodov E., Xaydarov S., Razzoqov I., (2024), E3S Web of Conferences, Advancements in renewable energy sources (solar andgeothermal): A brief review, 1-4, doi: 10.1051/e3sconf/202449701009

[2] *Javatpoint,* (2024), приступљено у децембру 2024. године са https://www.javatpoint.com/basic-concepts-in-machine-learning

Kratka biografija:



Lana Slović rođena je u Prijepolju 2001. god. Po završetku Prijepoljske gimnazije, 2019. godine upisuje Fakultet tehničkih nauka u Novom Sadu, studijski program Primenjeno softversko inženjerstvo. Nakon završenih osnovnih akademiskih studija, 2023. godine upisuje master akademske studije iz iste oblasti.
Kontakt: slovicl17 @gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.4
ДОИ: https://doi.org/10.24867/35BE11Sasic

Оптимизација претраге текстуалних дигиталних докумената

An Optimization of Textual Digital Document Search

Наталија Шашић, *Факултет техничких наука, Нови Сад*

Студијски програм – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *Рад истражује употребу Elasticsearch-а за оптимизацију претраге у реалном времену на великим скуповима података, са посебним фокусом на научне радове из ArXiv-а. Истиче напредне функције Elasticsearch-а као што су fuzzy претрага, обрада синонима и скалабилност, које побољшавају ефикасност претраживања и корисничко искуство. У раду се такође разматрају изазови у оптимизацији перформанси претраге и конфигурација, нудећи увид у ефикасне стратегије за управљање великим подацима. Посебна пажња посвећена је анализи резултата претраге у зависности од различитих конфигурација индекса и упита. Истраживање пружа смернице за имплементацију система за интелигентно претраживање у домену научних публикација.*
Кључне речи: *Elasticsearch, напредна претрага, веб апликација, дигитални документи*

Abstract – *The paper explores the use of Elasticsearch for real-time search optimization on large datasets, with a particular focus on scientific papers from ArXiv. It highlights advanced features of Elasticsearch such as fuzzy search, synonym processing, and scalability, which improve search efficiency and user experience. The paper also discusses the challenges in optimizing search performance and configurations, offering insights into effective strategies for managing big data. Special attention is paid to the analysis of search results depending on different index and query configurations. The research provides guidelines for the implementation of intelligent search systems in the domain of scientific publications.*

Keywords: *Elasticsearch, advanced search, web application, digital documents*

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

1. УВОД

Услед великог броја података у дигиталном добу, област проналажења информација никад није била важнија. Обим јавно доступних информација је експоненцијално растао заједно са експанзијом употребе интернета и сталним јачањем пословне и институционалне информационе технологије. Традиционални релациони системи база података суочавају се са ограничењима у обради полу структуираних и неструктуираних докумената, што је омогућило развој *NoSQL* база, попут *Elasticsearch*-а. Ефикасни системи за претрагу комбинују индексирање и претраживање, често користећи инвертовани индекс, метод који је основа многих претраживача, укључујући *Google*. Са порастом корисничких захтева за брзину и тачност, примена напредних алгоритама и дистрибуираних система постала је неопходна.
Elasticsearch је дистрибуирани систем за претрагу и анализу података базиран на библиотеци *Apache Lucene*. Омогућава ефикасну обраду великих количина података, независност од оперативног система и скалабилност. Његова прилагодљивост и *REST API* интерфејс омогућавају лаку интеграцију и конфигурацију. Мастер рад се фокусира на оптимизацију претраге у реалном времену за *ArXiv* колекцију научних радова, укључујући генерисање извештаја и обраду агрегационих упита, користећи Elasticsearch. Његове могућности укључују подршку за напредне алгоритме, као што су индексирање, рад са синонимима и нејасним упитима, чиме се побољшавају перформансе претраге и задовољство корисника.

2. ПРЕГЛЕД АКТУЕЛНОГ СТАЊА У ОБЛАСТИ

У наставку се даје кратак осврт на најзначајније категорије база података и њихове карактеристике, са нагласком на савремене приступе који омогућавају ефикасну претрагу и анализу података.

2.1. Класификација база података

Данас је на располагању мноштво база података, као и система за управљање истим. Традиционални приступ имплементацији базира се на релационим базама, док модернији приступ представља базу података оријентисану на документе. Оба приступа имају своје случајеве употребе, као и предности и недостатке. Литература пореди претраге текстуалног садржаја у свакој категорији, што подупире циљеве рада и проширује знање о савременим техникама претраге.

Релационе базе података чувају податке у табелама које представљају типове ентитета са атрибутима као колонама. Односи између табела дефинишу се преко

примарних и страних кључева, омогућавајући структурисано повезивање.

Нерелационе базе података (*NoSQL*) користе флексибилне шеме за складиштење података у форматима као што су документи, кључ/вредност или графикони. Оне су оптимизоване за специфичне моделе података и модерне апликације, познате по перформансама у раду са великим обимом података.

Базе података базиране на документима складиште полуструктуриране податке у *JSON* формату, омогућавајући флексибилност и једноставно прилагођавање апликацијама. Документи представљају објекте, а колекције докумената се групишу у индексе. Документ-оријентисане базе представљају подскуп *NoSQL* система, али се често издвајају због специфичног модела складиштења и функција претраге.

2.2. Преглед сличне литературе

У оквиру прегледа литературе истраживани су кључни аспекти претраге текстуалних докумената и анализа савремених претраживача и база података. Главни акценат је стављен на поређење перформанси релационих и нерелационих база, као и на анализу претраживача отвореног кода.

„Анализа претраживача отвореног кода“ [1] наглашава важност бесплатних и флексибилних лиценци, стабилности и флексибилности кода, али упозорава на комплексност корисничког интерфејса и могуће додатне трошкове за обуку. Алати *MeiliSearch*, *Typesense*, *Apache Solr* и *Elasticsearch* су поређени у овом раду, а аутори су нагласили да је библиотека иза *Elasticsearch* алата, *Apache Lucene*, најнапреднија на тржишту.

„Поређење перформанси складиштења података за претрагу пуног текста: са структуираним упитним језиком или без?“ [2] анализира перформансе Elasticsearch-а и Solr-а, истичући њихову брзину и поузданост, уз указивање на предности *PostgreSQL*-а за мање скупове података и на ограничења *MongoDB*.

„Свеобухватна студија Elasticsearch-а“ [3] истиче значај инвертованих индекса у *Elasticsearch*-у, као и његову дистрибуирану архитектуру, која обезбеђује поузданост и опоравак од грешака. Аутори су обезбедили детаљан увид у препроцесирање текста које укључује филтере карактера, токенизаторе и филтере токена.

„Управљање дигиталним документима“ [4] је уџбеник који се фокусира на значај релевантности и организације резултата претраге, наглашавајући да корисници траже информације, а не само податке који одговарају упиту.

3. ТРАДИЦИОНАЛНИ ПРИСТУПИ ПРЕТРАЗИ

Системи за претрагу података деле се на централизоване и дистрибуиране, зависно од начина складиштења индекса и обраде упита. Централизовани приступ чува све индексе на једном месту, што поједностављује претрагу, али ограничава скалабилност. Насупрот томе, дистрибуирани индекси омогућавају бржу и скалабилнију претрагу јер су подаци распоређени по више сервера. Модели претраге укључују класичне и алтернативне моделе. Класични модели претраге су Булов, векторски и пробабилистички, који се разликују по приступу рангирању релевантности. Алтернативни модели су *fuzzy*, проширени Булов, модели неуронских мрежа и језички модели, који уводе напредне функције као што су процена сличности, обрада нејасних упита и контекстуализација.

3.1. Традиционални приступи

Индексирање података користи различите структуре, попут *B* стабала, хеш индекса, бит-мапских и *R* стабала. У *Windows*-у, *NTFS* фајл систем ослања се на *B+* стабла за ефикасно управљање подацима.

Релационе базе попут *PostgreSQL*-а користе напредне технике као што је *GIN* индекс за брзу претрагу текста. Инвертовани индекс, који чува листу докумената за сваки термин, широко се примењује у системима за индексирање текста и интернет претраживачима.

У *NoSQL* базама приступи варирају: од брзих упита у кључ-вредност базама, претраге графова алгоритмима *BFS* и *DFS*, до напредне претраге текста у документ оријентисаним базама, које користе инвертоване индексе за сложене филтере и агрегације.

За претрагу великих текстуалних скупова, дистрибуирани индекси, векторска претрага и документ-орјентисане базе су кључни, нудећи скалабилност, прецизност и ефикасност.

3.2. Алати за претрагу текста

Apache Lucene је *open-source* библиотека за брзо и ефикасно индексирање и претрагу великих количина података, заснована на инвертованим индексима. Она омогућава развој сложених претрага и скалабилних система, али захтева напредне програмерске вештине за оптимизацију.

Apache Solr је платформа за претрагу заснована на Lucene библиотеци, дизајнирана за рад са великим количинама структурираних и неструктурираних података. *Solr* додаје функције као што су фасетирана претрага, рангирање и висока доступност, што га чини погодним за примену у великим предузећима и веб апликацијама.

4. МЕХАНИЗМИ ПРЕТРАЖИВАЧА *ELASTICSEARCH*

Elasticsearch је алат за претрагу и анализу података, базиран на *Apache Lucene* технологији. За разлику од *Lucene* библиотеке, која захтева писање *Java* кода, *Elasticsearch* обезбеђује лакши приступ функцијама претраге и индексирања преко *REST* интерфејса. Подржава концепт скоро реалног времена (*NRT*), што значи да измене у подацима постају претраживе у року од једне секунде.

Иако чува документе, *Elasticsearch* није база података у класичном смислу, већ је оптимизован за *full-text* претрагу и анализу великих количина података. Његова структура се ослања на инвертоване индексе за текстуална поља и *BKD* [5] стабла за нумеричке податке. Поред тога, нуди снажан *Query DSL* језик за креирање напредних упита.

4.1. Терминологија

Кластер је скуп чворова (сервера) који сарађују у складиштењу и обради података, где сваки кластер има јединствено име и један мастер чвор. Чвор је сервер који чува податке и обрађује упите. Партиција је јединица складиштења која омогућава хоризонталну скалабилност, а реплика је копија партиције која побољшава доступност и брзину претраге. Индекси су логичке колекције докумената у *JSON* формату, док су типови у старијим верзијама служили за категоризацију докумената. Документи су основне јединице информација у паровима кључ-вредност, а поља могу бити текстуална или других типова као што су бројеви и датуми, с тим да се текстуална поља обрађују инвертованим индексом.

4.2. Претпроцесирање текста

У *Elasticsearch* претрази се текст мора обрадити пре складиштења, а ова обрада се дешава у фази анализе, односно препроцесирања. Текстуална поља пролазе кроз процес анализе пре складиштења. Компоненте анализатора укључују првенствено филтере карактера, који уклањају непотребне симболе, као што су *HTML* тагови. Затим се користе токенизатори који деле текст на мање јединице - токене, а на крају се примењују токен филтери који нормализују токене, уклањају стоп речи и врше лематизацију. Стандардни анализатор (енг. *Standard analyzer*) је један од многих уграђених анализатора текста у *Elasticsearch*-у.

4.3. Мапирања

Сваки индекс има мапирање или шему за начин на који су поља у документима индексирана. Мапирање се дешава након индексирања и дефинише тип података за свако поље, како поље треба да буде индексирано и како треба да буде ускладиштено. Када се документи додају у *Elasticsearch*, постоје две опције за мапирање. Динамичко мапирање је кад *Elasticsearch* аутоматски детектује типове података и креира мапирања, али не даје увек оптималне резултате и често дупло индексира поједина поља да би се она могла користити за операције филтрирања исто као и за претрагу текста. Експицитни приступ чине ручно дефинисана мапирања, препоручена за производну употребу. *Elasticsearch* омогућава ефикасну обраду података и напредну претрагу, али захтева пажљиво конфигурисање ради оптималних резултата.

5. *ELASTICSEARCH* ПРЕТРАГА И СТРАТЕГИЈЕ ОПТИМИЗАЦИЈЕ

Elasticsearch је претраживач текста заснован на технологији *Lucene* који користи токенизацију и стеминг за анализу текста. Подржава специјализоване упите као што су нејасно подударање (*fuzzy query*) и претраге опсега. Упити могу бити комбиновани са филтерима и агрегацијама како би се добила прецизна анализа података.

5.1. Претраживање

Elasticsearch пружа разноврсне функционалности претраге, укључујући претраживање комплетног текста (путем *match* и *match_phrase* упита), тачно подударање (*term* и *terms* упити за структурирана поља), и упите по опсегу (за бројеве или датуме). Подржава и нејасно подударање (*fuzzy*, *wildcard* и *prefix*), булове комбинације за сложене логичке упите, као и геопросторну претрагу по координатама. Додатно, омогућава анализу са терм-вектор упитима и нуди агрегације и аналитику за статистичку обраду резултата претраге. Ове функције чине *Elasticsearch* ефикасним алатом за претрагу и обраду великих количина података.

5.2. Релевантност документа

Elasticsearch процењује релевантност докумената користећи однос *tf* и *df* мера. Учесталост термина у документу (*tf*) и његова реткост у целокупном корпусу (*idf*) одређују значај термина. У пракси, овај процес оптимизује *BM25* алгоритам [6], који ефикасно мери релевантност за текстуалну претрагу. Документи се затим рангирају према добијеним *score* вредностима, уз примену метрика као што су прецизност и поврат, како би се осигурао најрелевантнији редослед резултата.

5.3. Репликација и партиционисање

Elasticsearch користи партиционисање (енг. *sharding*) за расподелу података на различите сервере, тј. чворове, чиме се побољшава скалабилност и брзина обраде упита. Истовремено, примењује репликацију (енг. *replication*), где се копије података чувају на више чворова, обезбеђујући већу поузданост система и континуирану доступност података чак и у случају отказа појединачних чворова.

5.4. Оптимизације перформанси

Оптимизација перформанси у *Elasticsearch*-у обухвата ефикасно индексирање кроз прецизно дефинисање мапирања података, као што је коришћење типа *keyword* за структуиране податке. Такође, укључује оптимизацију упита употребом филтера и функција попут *search_after* за бржу обраду резултата. Додатно, подешавање кластера и хардвера игра кључну улогу, јер правилна конфигурација сервера и дистрибуција ресурса побољшавају перформансе и скалабилност система.

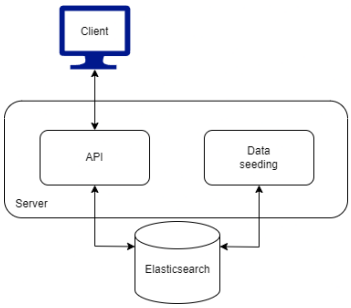
6. АРХИТЕКТУРА СИСТЕМА И МОДЕЛИ ПОДАТАКА

Имплементација веб апликације за претрагу научних радова се састоји од три главна дела: клијентске, серверске апликације и базе података. Подаци се чувају у *JSON* формату, а сви радови су идентификовани преко јединственог *ID*-а. Радови су складиштени у фајл систему.

6.1. Архитектура система

Архитектура, приказана на слици 1, се састоји од четири основна елемента: клијента, серверске апликације, сервера за иницијализацију података и *Elasticsearch* који је коришћен за складиштење и претраживање података. Клијент комуницира са

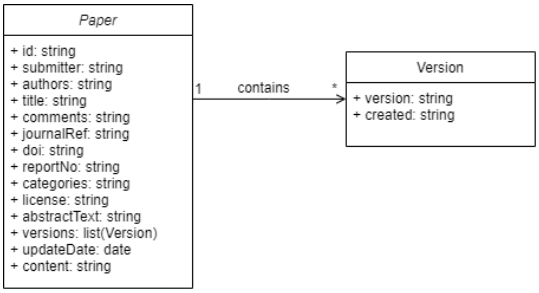
сервером, који обавља различите претраге путем коришћења *QueryDSL Elasticsearch* функционалности.



Слика 1. Архитектура система

6.2. Модел података

ArXiv скуп података садржи метаподатке о научним радовима као што су *ID*, аутори, наслов, апстракт и други који се виде на слици 2. Ови подаци су представљени као *JSON* документи, а садржај радова је доступан као *PDF* путем директног линка. Приликом учитавања података и иницијализације индекса, додато је поље *content* у којем се складишти индексирани текст екстрактован из сваког појединачног рада.



Слика 2. Дијаграм класа

6.3. Коришћене технологије

У развоју апликације коришћени су различити алати као што су *Spring Boot* за серверску страну, *Python* за препроцесирање података и увоз у *Elasticsearch*, и *React* за креирање фронтенд компонената. *Elasticsearch* је коришћен за складиштење и претрагу података, док *Docker* омогућава изолацију апликација у контејнерима ради боље преносивости и ефикасности.

7. СТУДИЈА СЛУЧАЈА

Апликација за претрагу научних радова има за циљ брзу и релевантну претрагу, оптимизовану за високе перформансе. Основне функције укључују претрагу по текстуалним пољима као што су наслов, аутори, апстракт и садржај радова. Такође, подржава претрагу по подносиоцу научног рада и омогућава генерисање хистограма о броју објављених радова у одређеном периоду. Систем користи *Elasticsearch* за пружање релевантних резултата, а све претраге су интерактивне, освежавајући резултате у реалном времену при промени упита. Резултати претраге укључују наглашене делове текста који одговарају упиту и информације о локацији *PDF* документа. Детаљни приказ докумената обухвата метаподатке, као што су идентификатор рада и име часописа.

Оптимизација система укључује неколико кључних аспеката: *bulk* унос који убрзава индексирање, смањујући време за обраду, екстерно чување *PDF* докумената смањује оптерећење кластеру, а ручно мапирање омогућава боље индексирање метаподатака и текста. *Boosting* побољшава релевантност резултата, док оптимална конфигурација са 2 партиције и 1 репликом даје најбоље перформансе. Прекомерне партиције изазивају пад перформанси, а са већим обимом података, захтеви за хистограмима успоравају претрагу.

8. ЗАКЉУЧАК

Ова студија истражује претрагу докумената користећи *Elasticsearch*, са фокусом на предности дистрибуираног система за велике скупове података. Обрађене су технике као што су инвертовани индекси, рангирање и партиционисање, уз интеграцију са *Python*-ом, *Spring Boot*-ом и *React.js*-ом за развој веб апликације. *Bulk* операције су убрзале обраду, а оптимална конфигурација са две партиције и једном репликом дала је најбоље резултате, избегавајући проблеме с *oversharding*-ом. Резултати указују на изузетну флексибилност *Elasticsearch*-а и његову примену у разним индустријама. Будућа унапређења укључују анализаторе за српски језик, аутоматско довршавање упита и семантичко рангирање, чиме би се побољшало корисничко искуство и функционалност система.

9. ЛИТЕРАТУРА

[1] Xinyuan Chang, “The Analysis of Open Source Search Engines”, 2023.

[2] George Fotopoulos, Paris Koloveas, Paraskevi Raftopoulou, Christos Tryfonopoulos, “Comparing Data Store Performance for Full-Text Search: to SQL or to NoSQL?”, 2023.

[3] Nikita Kathare, O. Vinati Reddy, Dr. Vishalakshi Prabhu, “A Comprehensive Study of Elastic Search”, 2020.

[4] Д. Ивановић, Б. Милосављевић, “Управљање дигиталним документима“, 2015.

[5] Octavian Procopiuc, Pankaj K. Agarwal, Lars Arge, Jeffrey Scott Vitter, “Bkd-Tree: A Dynamic Scalable kd-Tree”, 2003 .

[6] Madhusudhan Konda, “Elasticsearch in Action, Second Edition”, 2023.

Кратка биографија:

Наталија Шашић рођена је у Сомбору 1999. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Софтверско инжењерство и информационе технологије одбранила је 2025.год.
Контакт: sasicnatalija4@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

UDK: 007:34
DOI: https://doi.org/10.24867/35BE12Roljic

Компаративна анализа *NER* модела за пресуде на црногорском језику

Comparative Analysis of NER Models for Judgements in Montenegrin

Бранислав Рољић, Стеван Гостојић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИЧКО И РАЧУНАРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *Препознавање именованих ентитета у језицима са ограниченим ресурсима отежано је малим бројем доступних анотираних података и доменски специфичних модела. У овом раду се евалуирају трансформерски и генеративни NER приступи на ручно анотираном корпусу црногорских правосудних пресуда. Методологија спаја лингвистичке увиде и технике дубоког учења како би се модели прилагодили правном домену. Спроведени експерименти анализирају перформансе по типовима ентитета и показују кључне предности и ограничења сваког приступа. Резултати доприносе развоју правно-језичких NLP алата и указују на могуће правце даљег унапређења NER система у условима ограничених ресурса.*

Кључне речи: *NER, правни NER, BERT, LLM*

Abstract – *Named entity recognition in under-resourced languages is hindered by limited annotated data and the absence of domain-adapted models. This study evaluates transformer-based and generative NER approaches on a manually annotated corpus of Montenegrin legal texts. The methodology integrates linguistic insight with deep learning techniques to adapt models to the legal domain. Through comprehensive experiments, we examine performance across entity types, assess generalization, and identify key strengths and limitations of each method. The results support the development of NLP tools for Montenegrin legal language and highlight directions for advancing NER in low-resource legal settings.*

Keywords: *NER, legal NER, BERT, LLM*

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

1. УВОД

Препознавање именованих ентитета (енг. *named entity recognition*, *NER*) у правним текстовима представља изазован задатак, посебно за језике са ограниченим дигиталним ресурсима као што је црногорски. Правни документи карактеришу се формалним стилом, комплексном синтаксом и специфичном терминологијом, што отежава поуздану аутоматску идентификацију ентитета као што су судови, странке и правни акти. Постојећи *NER* модели, иако веома успешни у општим доменима, обично не обезбеђују задовољавајуће резултате у правним корпусима са мањим бројем анотација.

Циљ овог рада је компаративна анализа више приступа заснованих на модерним језичким моделима прилагођеним јужнословенским језицима, као и модела оптимизованих за рад у условима ограничених ресурса. Истраживање обухвата прикупљање и ручну анотацију корпуса црногорских пресуда, дефинисање доменски специфичних категорија ентитета и евалуацију више архитектура.

Вредност овог рада огледа се у систематском поређењу различитих архитектура на истом корпусу, чиме се добија поуздан увид у њихову стабилност, ограничења и применљивост у правном домену. Практична мотивација проистиче из растуће потребе за аутоматизованом анализом правних докумената у Црној Гори, док научни допринос лежи у развоју иницијалних језичких ресурса и метода за *NER* на мање заступљеним језицима.

2. СРОДНА ИСТРАЖИВАЊА

Истраживања у области *NER*-а еволуирала су од раних метода заснованих на лингвистичким правилима и статистичким моделима ка дубоким неуронским мрежама и трансформер архитектурама. Рани *BiLSTM* и *CNN* приступи омогућили су моделовање контекстуалних зависности, али је појава *BERT*-а представљала кључни заокрет захваљујући бидирекционом механизму самопажње и богатим контекстуализованим представама токена. Његови деривати, као што су *DistilBERT* и *RoBERTa*, даље су унапредили резултате у различитим *NER* задацима [1]. Бројне студије показују да трансформери надмашују класичне секвенцијалне моделе у доменски специфичним контекстима. *LegalBERT* значајно премашује *BiLSTM-CRF* у правним текстовима [2], што потврђује предност трансформерске архитектуре у задацима обраде правних докумената. За јужнословенске језике посебно је значајно истраживање [3], које показује да доменски прилагођена токенизација може побољшати *NER* перформансе за 4,5–54,6%, што указује на важност квалитетног сегментирања у морфолошки богатим језицима.

Структурни аспекти *NER*-а такође су предмет анализе: избор анотационе шеме значајно утиче на коначне метрике – једноставни ентитети боље се препознају у *IO* формату, док су *IOBES* шеме погоднije за сложеније структуре [4]. Шири преглед модерних приступа дат је у [1], који истиче тренд ка интеграцији

трансформера, *LLM* модела и техника структурисаног предвиђања као што је *CRF*.
За јужнословенски простор, студија [5] показује да *bcms-BERTuh* и *XLM-R-BERTuh* постижу највише вредности *F1*-мере (до 0,942), што потврђује ефикасност мултијезичне *BCMS* обуке. На правосудним текстовима, рад [6] показује да фино подешавање *BERTuha* на доменски специфичним пресудама даје изузетно високе резултате ($F1 \approx 0.96$), чак и у условима ограничених ресурса.

3. МЕТОД

3.1 Претпроцесирање и анализа података

Како не постоји јавно доступан, анотиран корпус црногорских правних докумената за *NER*, у овом истраживању је конструисан нови корпус од 225 пресуда (7201 ентитет). Документи су прикупљени аутоматизованим *scraping*-ом са портала црногорских судова [7], коришћењем *Playwright* библиотеке [8] ради обраде динамичког једностраничног садржаја (енг. *single-page application*, *SPA*). Након *scraping*-а, вршено је уклањање *HTML*/*CSS* артефаката и нормализација текста у *plain-text* формат погодан за анотацију.
Анотација је извршена у *LabelStudio* [9] окружењу, уз дефинисање 14 ентитетских категорија специфичних за правни домен, заснованих на постојећим правним *NER* системима [2], [6]. Обухваћени су следећи ентитети: назив суда, датум пресуде, број предмета, судија, записничар, тужилац, окривљени, кривично дело, одлука, врста санкције, износ/трајање санкције, материјалноправне одредбе, процесноправне одредбе и трошкови поступка.
Током анотације уочени су бројни изазови, укључујући варијабилне формате ентитета (посебно код бројева предмета и датума), различите обрасце анонимизације и изражену структурну недоследност међу документима.

3.2 Припрема података за моделе

Дужина пресуда у великом броју случајева премашује ограничење од 512 токена које намеће трансформер архитектуре, те је била неопходна примена механизма клизног прозора (енг. *sliding window*) – сегменти дужине 512 токена са вредношћу помераја (енг. *stride*) од 128, што омогућава очување контекста око граница сегмената.
Токенизатор дели речи на подречи, при чему први подтокен добија *BIO* ознаку, а сви остали вредност -100, да би били игнорисани у функцији губитка. У циљу избегавања неважећих *BIO* секвенци, I-ознаке на почетку прозора конвертују се у B-ознаке. Сваки сегмент се затим попуњава (енг. *padding*) и допуњује [*CLS*]/[*SEP*] токенима.
За објективну процену модела и избегавање претренирања примењена је стратификована петострука унакрсна валидација (енг. *stratified 5-fold cross-validation*), уз очување пропорционалне заступљености ентитетских класа у сваком *fold*-у. У свакој итерацији око 180 докумената коришћено је за обуку, док је приближно 45 докумената служило као валидациони скуп.

3.3 NER модели

Циљ методологије је обухватање више комплементарних приступа, од класичних трансформера до генеративних *zero-shot* и *few-shot* модела.

3.3.1 Трансформер модели

BERTuh, као први велики *BCMS* трансформер, је претрениран користећи приступ детекције замењених токена (енг. *replaced token detection*, *RTD*). Овакав приступ омогућава ефикасније учење у нискоресурсним условима, што је посебно важно за црногорски правни домен.
***BERTuh* са тежинама класа** користи модификовану *CrossEntropyLoss* функцију са тежинама пропорционалним учесталости класа, како би се ублажио изражен дисбаланс у корпусу и побољшало препознавање слабо заступљених ентитета.
***BERTuh* са фокалним губитком** – примењује фокални губитак (енг. *focal loss*) како би се смањио утицај лако класификованих примера и појачала осетљивост на тешке и варијабилне ентитете, што доводи до стабилнијих перформанси у правним текстовима.
XLM-R-BERTuh је мултијезичка варијанта *XLM-RoBERTa* модела, додатно обучена на *BCMS* подацима. Претходна истраживања [5] показују да овај модел постиже највише вредности *F1*-мере на *NER* задацима за јужнословенске језике, нарочито код језичких варијација и флективних структура.
BERTuh-CRF **–** ова архитектура комбинује контекстуализоване енкодинге *BERTuh*-а са *CRF* слојем који моделује зависности између ознака у секвенци. *BERTuh* обезбеђује богате представе токена у сложеним правним реченицама, док *CRF* осигурава структурно конзистентне *BIO* секвенце и прецизније обележавање дугачких, вишечланих ентитета, који су чести у правним текстовима.
***BERTuh* са *DAPT-MLM* –** ради бољег хватања специфичне правне терминологије и стилских образаца, модел *BERTuh* је додатно претрениран *MLM* техником на корпусу од 2600 пресуда. Овај поступак омогућава моделовање доменски релевантних дистрибуција речи и побољшава квалитет токенских репрезентација. Као што показује рад [3], *DAPT* доследно унапређује перформансе трансформер модела у задацима *NER*-а специфичним за одређени домен.

3.3.2 Zero-shot и Few-shot модели

GLiNER третира *NER* као *span-level* класификацију и омогућава *zero-shot* препознавање ентитета на основу њихових текстуалних описа, без употребе *BIO* шеме. Модел користи ембединг простор који истовремено представља текст и дефиниције ентитета, што му омогућава да препозна и категорије које нису биле део тренинг скупа, што је посебно значајно у нискоресурсним доменима као што је правни.

PromptNER формулише *NER* као генеративни задатак, где *LLM* добија упутство за издвајање ентитета и производи структуриран *JSON* излаз. Примењени су *zero-shot* и *few-shot* режими, како би се испитала способност *LLM* модела да без додатног обучавања, или уз минималан број примера, обрађују ентитете у специјализованом правном домену.

3.4 Евалуација

Евалуација је изведена на нивоу ентитета, јер делимично препознати ентитети немају практичну вредност у правном домену. За *BIO* секвенце коришћена је библиотека *seqeval* [10], а за *GLiNER* и *PromptNER* прилагођена *span-based* логика.
Мерене су метрике прецизност, одзив и *F1*-мера по класама, као и макро и пондерисани *F1* агрегати.

4. ИМПЛЕМЕНТАЦИЈА

Обука свих модела спроведена је по јединственој методологији која обезбеђује фер поређење, контролисане услове и репродуктивност резултата.

4.1. Рачунарска инфраструктура

Експерименти су реализовани на *cloud* инфраструктури са **NVIDIA RTX A4000 (16GB)**, користећи **Python 3.11**, **PyTorch 2.1** и **HuggingFace Transformers 4.36**. Конзистентна *GPU* конфигурација обезбедила је стабилност и поновљивост обуке свих модела.

4.2. Заједнички хиперпараметри

За све моделе примењени су идентични хиперпараметри: *learning rate*: $\mathbf{3\times10^{-5}}$ (*AdamW*, *weight decay* 0.01), *warmup ratio*: **0.01** (линеарни *schedule*), број епоха: **8** (осим *DAPT-MLM* фазе: 2 епохе), *batch* size: **4**, уз **акумулацију 4 градијента** (ефективни *batch* 16), *mixed precision* (***fp16***) обука, *sliding window*: **512 / stride 128**

4.3. Динамика обуке и конвергенција

Тренинг и валидациони губитак праћени су током обуке, заједно са макро/микро метрикама на сваких 100 корака. Механизам раног заустављања је активиран након три узастопне стагнације *F1*-мере, што је спречило претренирање и обезбедило стабилну конвергенцију. Сви модели су конвергирали у опсегу између шесте и осме епохе.

4.4. Састав тренинг скупа

Применом *sliding window* токенизације добијено је 11.330-12.464 тренинг примера по *fold*-у, док су валидациони скупови садржали 2.747-2.993 примера. Ова расподела одговара 80/20 подели докумената (приближно 180 за тренинг, 45 за валидацију). Корпус је показао изражен дисбаланс класа.

4.5. Стратификована расподела

Стратификација је заснована на комбинацијама ентитетских образаца (16 доминантних шаблона), што је омогућило да сваки *fold* садржи пропорционалан број примера свих типова ентитета.

5. РЕЗУЛТАТИ И ДИСКУСИЈА

У табели 1 су приказани резултати за све варијанте *BERTuh* модела, као и за мултијезички *XLM-R-BERTuh*, те *zero-shot* и *few-shot* приступе.

Табела 1 Поређење резултата различитих NER модела на корпусу црногорских правних пресуда

Модел	Прецизност	Одзив	*F1*-мера
BERTuh	0.8020	0.7656	0.7628
BERTuh + Class Weights	0.4790	0.9213	0.5951
BERTuh + Focal Loss	0.7874	0.7345	0.7319
BERTuh + CRF	0.8109	0.8201	0.8049
BERTuh + DAPT-MLM	0.7980	0.7872	0.7768
XLM-R-BERTuh	**0.8942**	**0.8842**	**0.8858**
GLiNER Large	0.05	0.08	0.07
GLiNER Large v2	0.09	0.07	0.07
GLiNER bi-large (knowledge)	0.19	0.12	0.14
GLiNER multi v2.1	0.19	0.12	0.14
PromptNER Zero-shot	0.5643	0.2910	0.3567
PromptNER Few-shot	0.4848	0.4466	0.4397

Основни *BERTuh* модел представља поуздану референтну основу за *NER* у пресудама. Макро-*F1* износи 0,76, што показује да *BCMS* модели адекватно обрађују синтаксичке и семантичке обрасце карактеристичне за правни језик, упркос ограниченој величини корпуса. Најбоље резултате модел постиже за учестале категорије, док ређе класе остају изазов због ограничене заступљености у подацима.
Примена тежина класа довела је до значајног погоршања резултата. Иако је одзив био веома висок, прецизност је драстично опала, што је резултовало макро-*F1* вредношћу од 0,59. Ово указује да инверзно пондерисање класа није ефикасна стратегија за правне пресуде: због великог дисбаланса ентитета, модел генерише превише лажних позитивних предвиђања.
С друге стране, варијанта *BERTuh*-а са фокалним губитком се показала знатно стабилнијом. Модел је успео да боље контролише утицај тешких примера, уз макро-*F1* $\approx$ 0,73. Фокални губитак се показао као ефикаснија стратегија, јер је селективно појачавао учење на примерима са већим степеном неизвесности, без нарушавања стабилности процеса тренинга.
Увођење *CRF* слоја довело је до побољшања у структурној доследности предвиђања. Модел *BERTuh-CRF* остварује макро-*F1* од 0,80, што указује да експлицитно моделовање транзиција између ознака ублажава грешке на границама ентитета. Ово је нарочито важно за правни домен, где су ентитети често дугачки и вишечлани.
Примена *DAPT-MLM* преттренирања показала је да доменска адаптација има значајну вредност. Модел *BERTuh* са *DAPT-MLM*-ом постиже макро-*F1* од 0,78. Иако ово није највиши резултат међу трансформерима,

доменско преттренирање побољшава репрезентације правних текстова и доприноси стабилнијој конвергенцији у односу на основни модел.

Најбоље резултате остварује *XLM-R-BERTuћ*, са макро-*F1* оценом од 0,89. Мултијезичко *BCMS* преттренирање и снажнија енкодер архитектура *XLM-R* омогућили су моделу да ефикасно обухвати морфолошке и синтаксичке варијације присутне у правним документима. Модел показује ниску варијансу између *fold*-ова и највиши ниво робусности.

Zero-shot приступи су тестирани ради испитивања применљивости модела у условима без анотираних података. *GLiNER* варијанте остариле су веома ниске макро-*F1* вредности (0,07–0,14), са релативно бољим одзивом, али недовољном прецизношћу за практичну употребу у правном домену. Ово је очекивано, јер модели нису прилагођени језичким структурама правосудних пресуда.

PromptNER показује боље перформансе од *GLiNER*-а: *zero-shot* варијанта постиже макро-*F1* од 0,36, а *few-shot* варијанта 0,44. Иако ови резултати и даље знатно заостају за фино подешеним трансформерима, они указују да *LLM* модели имају потенцијал у сценаријима ограничених ресурса, али тренутно не могу заменити трансформерске моделе обучене на доменски специфичном корпусу.

6. ЗАКЉУЧАК

У раду је представљена компаративна анализа више приступа препознавању именованих ентитета у црногорским пресудама, заснована на новоформираном, ручно анотираном корпусу и доменски релевантним категоријама ентитета. Испитани су трансформерски модели *BERTuћ* и *XLM-R-BERTuћ*, као и њихове варијанте са *CRF* слојем, класним тежинама, фокалним губитком и доменском адаптацијом. Резултати показују да најстабилније и најпрецизније решење представља мултијезички *XLM-R-BERTuћ* (≈0.89 макро-*F1*), док *CRF* додатно побољшава конзистентност секвенцијалног означавања. Модел *BERTuћ* са *DAPT-MLM* преттренирањем такође је остварио унапређење у односу на основну варијанту, што потврђује значај доменске адаптације у условима ограничених ресурса. Ови резултати су упоредиви са перформансама достигнутим за српски правни *NER*, што потврђује ефикасност трансформера и доменске адаптације и за мање језике.

Главна ограничења рада односе се на величину корпуса и изостанак подршке за угнежђене ентитете. *Zero-shot* и *few-shot* приступи показују потенцијал, али и даље знатно заостају за моделима обученим на доменским подацима.

Будућа истраживања треба усмерити ка проширивању корпуса (укључујући *active learning* и синтетичке податке), моделовању односа ентитета на нивоу целог документа, подршци за хијерархијске и угнежђене ознаке, као и тестирању архитектура за дуге секвенце. Посебно перспективан смер представља истраживање хибридних модела који би спојили високу прецизност трансформера у означавању секвенци са напредним контекстуалним резоновањем *LLM* модела.

7. ЛИТЕРАТУРА

[1] I. Keraghel, S. Morbieu, и M. Nadif, „Recent Advances in Named Entity Recognition: A Comprehensive Survey and Comparative Study“, 20. Децембар 2024., arXiv: arXiv:2401.10825. doi: 10.48550/arXiv.2401.10825.

[2] H. Darji, J. Mitrović, и M. Granitzer, „German BERT Model for Legal Named Entity Recognition“, у Proceedings of the 15th International Conference on Agents and Artificial Intelligence, 2023, стр. 723–728. doi: 10.5220/0011749400003393.

[3] M. Bogdanović, M. Frtunić Gligorijević, J. Kocić, и L. Stoimenov, „An Analysis of the Training Data Impact for Domain-Adapted Tokenizer Performances—The Case of Serbian Legal Domain Adaptation“, Appl. Sci., том 15, изд. 13, стр. 7491, Јули 2025, doi: 10.3390/app15137491.

[4] I. Ait Talghalit, H. Alami, и S. O. El Alaoui, „Exploring Different Annotation Schemes for Single and Consecutive Named Entity Recognition in the Arabic Biomedical Domain using Transformer Models and Contextual Semantic Embeddings“, Eng. Technol. Appl. Sci. Res., том 15, изд. 2, стр. 21854–21860, Апр. 2025, doi: 10.48084/etasr.10019.

[5] M. Škorić, „New Language Models for Serbian“, Infotheca, том 24, изд. 1, стр. 7–28, 2024, doi: 10.18485/infotheca.2024.24.1.1.

[6] V. Kalušev и B. Brkljač, „Named entity recognition for Serbian legal documents: Design, methodology and dataset development“, 14. Фебруар 2025., arXiv: arXiv:2502.10582. doi: 10.48550/arXiv.2502.10582.

[7] „Sudovi Crne Gore“. Приступљено: 28. Октобар 2025. [На Интернету]. Available at: https://sudovi.me/sdvi/odluke

[8] Playwright. [На Интернету]. Приступљено: 28. Октобар 2025. Available at: https://playwright.dev/

[9] LabelStudio. [На Интернету]. Приступљено: 28. Октобар 2025. Available at: https://labelstud.io/

[10] seqeval. [На Интернету]. Приступљено: 28. Октобар 2025. Available at: https://huggingface.co/spaces/evaluate-metric/seqeval

Кратка биографија:

Бранислав Рољић рођен је 1999. године у Бањалуци. Основне академске студије завршио је 2022. године на Електротехничком факултету у Бањалуци, а Мастер академске студије на Факултету техничких наука у Новом Саду 2025. године.
Контакт:branislavroljic99@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.42
ДОИ: https://doi.org/10.24867/35BE13Mastilovic

AI систем за анализу образовног садржаја са LLM интеграцијом у Service Fabric микросервисима

AI System for Educational Content Analysis with LLM Integration in Service Fabric Microservices

Мастиловић Радослав, *Факултет техничких наука, Нови Сад*

Студијски програм – ПРИМЕЊЕНО СОФТВЕРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *У раду је представљен интелигентни систем за анализу и евалуацију образовног садржаја, реализован као дистрибуирана апликација заснована на Microsoft Service Fabric микросервисној архитектури. Систем обједињује функционалности за пријем, обраду и оцену радова уз интеграцију великих језичких модела (LLM) ради аутоматске анализе и генерисања повратних информација. За трајно чување података користи се MongoDB база, док ReliableDictionary обезбеђује брз приступ у меморији сервиса. Циљ решења је унапређење процеса наставе и евалуације путем аутоматизације и примене вештачке интелигенције у образовном контексту.*

Кључне речи: *Микросервисна архитектура, вештачка интелигенција, Service Fabric, LLM модели,*

Abstract – *This work presents an intelligent system for analyzing and evaluating educational content. The system is built as a distributed application using Microsoft Service Fabric microservices. It can receive, process, and evaluate student work with the help of large language models (LLMs), which create automatic feedback. The system uses MongoDB to store data and ReliableDictionary for fast access in the services. The goal is to improve teaching and evaluation by using automation and artificial intelligence.*

Keywords: *Microservice architecture, artificial intelligence, Service Fabric, LLM models*

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

1. УВОД

Развој савремених информационих технологија омогућио је примену интелигентних система у различитим областима, па тако и у образовању. Све већи број образовних установа тежи дигитализацији процеса учења, провере знања и евалуације студентских радова. Класичан приступ оцењивању често захтева значајно време и ресурсе, док аутоматизовани системи пружају могућност брже, објективније и конзистентније анализе.

Циљ овог рада је развој AI система за анализу образовног садржаја који користи Service Fabric микросервисну архитектуру и интеграцију LLM модела (Large Language Models) ради аутоматске евалуације едукативних радова. Овакво решење омогућава професорима да добију квалитетне повратне информације о студентским радовима, а студентима да разумеју своје грешке и добију конкретне препоруке за побољшање.

Систем је дизајниран као скалабилна, дистрибуирана апликација која омогућава независно функционисање и комуникацију више сервиса. Подаци се трајно чувају у MongoDB бази, док ReliableDictionary обезбеђује кеширање и брз приступ подацима у реалном времену. Кориснички интерфејс је реализован у React окружењу, чиме је омогућено једноставно коришћење и визуелни преглед резултата анализе.

Овај приступ представља корак ка интелигентним системима за подршку настави и учењу, где вештачка интелигенција може да допринесе ефикаснијој, објективнијој и персонализованијој евалуацији образовних садржаја.

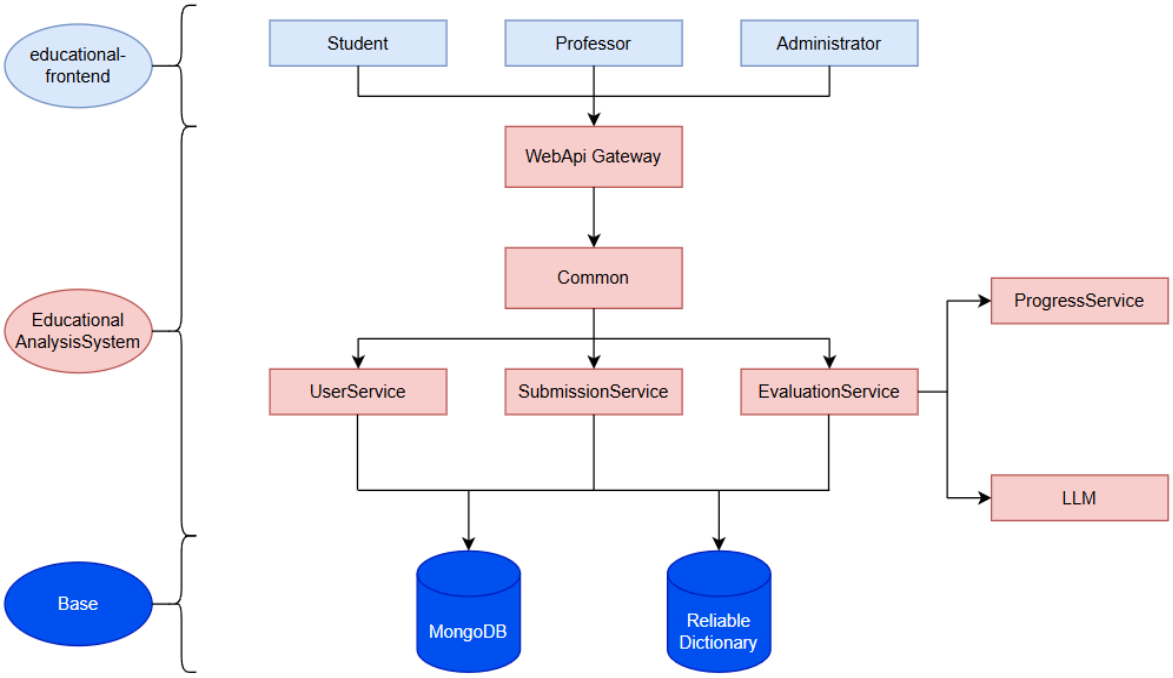
2. ПРЕГЛЕД КОРИШЋЕНИХ ТЕХНОЛОГИЈА

Развој предложеног система ослања се на савремене технологије које омогућавају дистрибуираност, поузданост и примену интелигентних алгоритама у анализи образовног садржаја. Основу архитектуре чини Microsoft Service Fabric, платформа за оркестрацију микросервиса која обезбеђује скалабилност, отпорност на грешке и једноставно одржавање више независних компоненти у оквиру једне апликације. У оквиру Service Fabric окружења реализовани су следећи сервиси: UserService – управља корисницима, улогама и аутентификацијом, SubmissionService – обрађује и чува предате студентске радове, EvaluationService – врши анализу садржаја радова применом LLM модела и генерише повратне информације, ProgressService – израчунава статистике и прати напредак студената током времена, WebApi Gateway – статлес сервис који представља централну тачку комуникације фронтенда са свим осталим сервисима, укључујући подршку за реалновременску комуникацију преко SignalR технологије. За трајно чување података користи се MongoDB, документно оријентисана NoSQL база

података, док ReliableDictionary обезбеђује кеширање у меморији ради убрзаног приступа и бољих перформанси. Интеграција LLM модела (Large Language Models) омогућава аутоматску анализу, препознавање грешака и генерисање конструктивних препорука студентима. Ови модели се повезују са системом преко API интерфејса (Groq и Gemini). Кориснички интерфејс је реализован у React окружењу, уз примену Ant Design библиотеке за визуелне компоненте. Ова комбинација омогућава интуитиван, прегледан и респонзиван интерфејс за студенте, професоре и администраторе.
Целокупно решење прати принципе микросервисне архитектуре, што омогућава лако проширење, интеграцију нових интелигентних модула и независно одржавање сваке компоненте система.

3. ОПИС И СПЕЦИФИКАЦИЈА СИСТЕМА

Систем за аутоматску анализу образовног садржаја заснива се на микросервисној архитектури реализованој у оквиру Microsoft Service Fabric окружења. На Слици 1. приказана је архитектура система, која обухвата више независних, али међусобно повезаних сервиса. Архитектура је подељена на три логичка слоја. Фронтенд слој (educational-frontend) реализован је у React окружењу и представља кориснички интерфејс за три улоге у систему: студента, професора и администратора. Бекенд слој (EducationalAnalysisSystem) обухвата више Service Fabric сервиса који реализују пословну логику: UserService управља корисницима, улогама и приступом; SubmissionService прима и обрађује студентске радове; EvaluationService врши аутоматску анализу садржаја применом LLM модела; ProgressService израчунава статистику и прати напредак студената; док WebApi Gateway омогућава комуникацију фронтенда са осталим сервисима и реалновременску синхронизацију преко SignalR технологије. Слој базе података (Base) обухвата MongoDB за трајно чување података и ReliableDictionary за кеширање и брзи приступ активним подацима. EvaluationService користи LLM (Large Language Model) за анализу студентских радова и генерисање педагошког feedback-a, док ProgressService из добијених резултата израчунава просечне оцене, најчешће грешке и тренд напредовања. Оваквом поделом постигнута је боља изолованост функционалности, могућност паралелног рада и лакше одржавање система. Комбинација Service Fabric-a, MongoDB базе, React интерфејса и LLM интеграције омогућава поуздан, флексибилан и скалабилан систем за интелигентну евалуацију образовних радова.

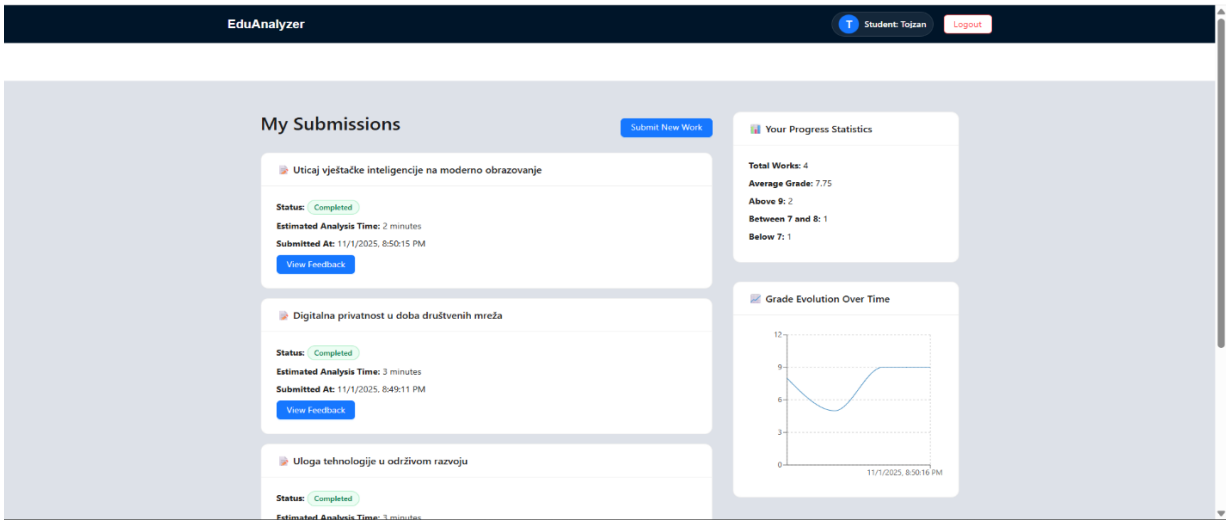


Слика 1. Архитектура система

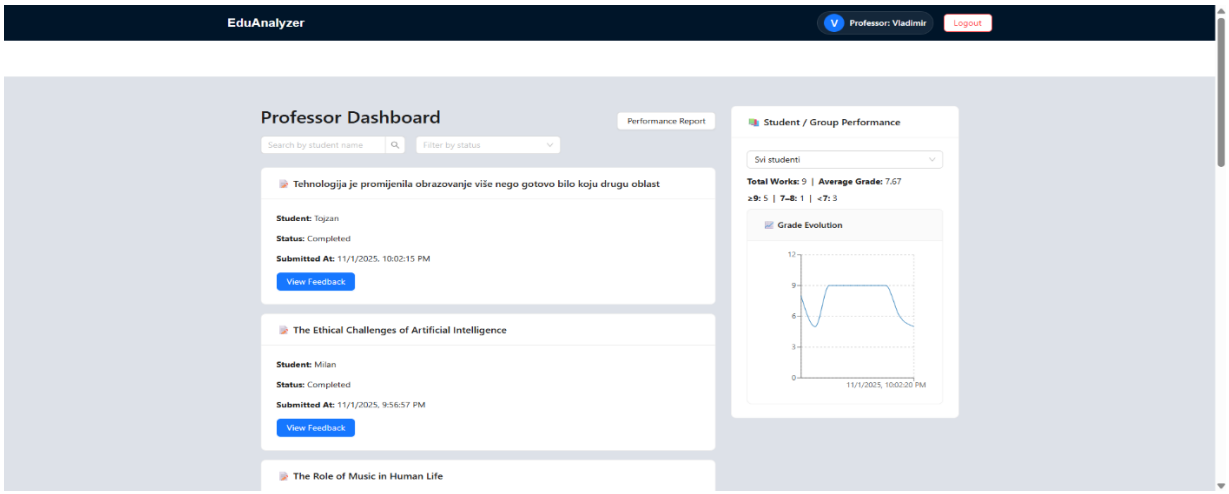
4. ПРИКАЗ ИМПЛЕМЕНТИРАНОГ СИСТЕМА

Имплементирани систем представља функционалну веб апликацију под називом EduAnalyzer, која омогућава студентима, професорима и администраторима интерактиван рад у оквиру заједничке платформе за анализу образовних радова. Кориснички интерфејс је реализован у React окружењу и повезан са Service Fabric микросервисима путем WebApi Gateway слоја. На Слици 2. приказана је контролна табла студента, која пружа преглед свих предатих радова, њихов статус, процењено време анализе и датум предаје. Са десне стране приказује се напредак студента у виду статистике (просечна оцена, број радова изнад или испод задатог прага) и графички омогућава студенту да једноставно прати свој развој и прегледа повратне информације добијене од система. На Слици 3. приказан је професорски интерфејс, који омогућава увид у све студентске радове, филтрирање

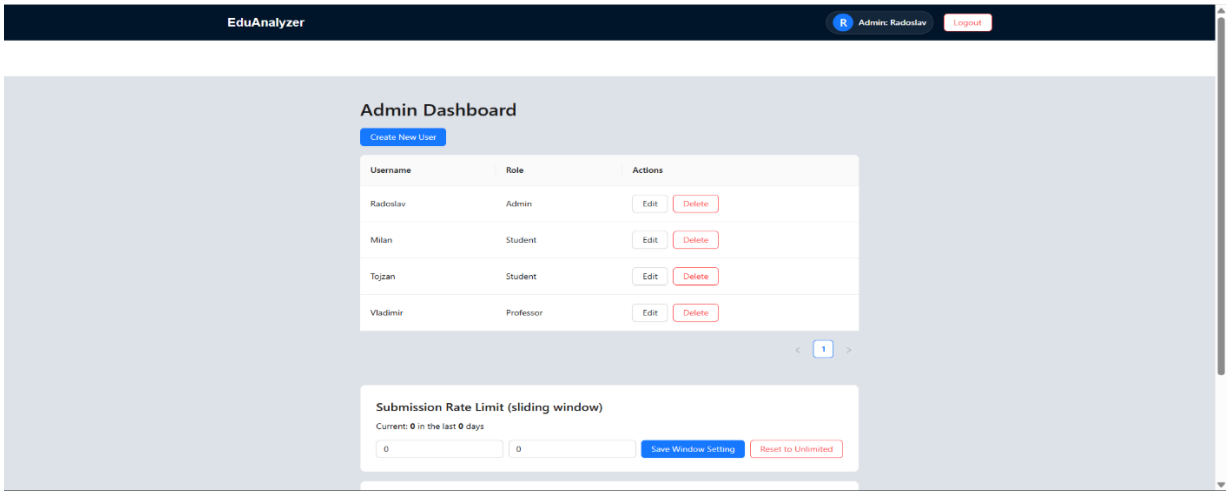
по статусу или имену студента, као и преглед генерисаних анализа и коментара. Поред тога, професор може да иницира поновну анализу рада са новим инструкцијама и да прегледа статистичке податке за појединачне студенте или групе, укључујући графички приказ просека оцена кроз време. На Слици 4. приказана је администраторска контролна табла, која омогућава управљање корисницима система (додавање, уређивање и брисање налога), као и подешавање глобалних правила система. Администратор може дефинисати ограничење броја предаја радова у задатом временском интервалу (sliding window) и подешавати параметре анализе као што су распон оцена и методе процене.



Слика 2. Студент контролна табла



Слика 3. Професор контролна табла



Слика 4. Администратор контролна табла

Оваква организација интерфејса обезбеђује јасно раздвајање улога и једноставну употребу система. Интеракција између корисника и микросервиса одвија се у реалном времену захваљујући SignalR технологији, што омогућава тренутно ажурирање података након анализе или измене. Комбинација интуитивног дизајна и интелигентне позадине омогућава да систем буде користан у академским окружењима као подршка процесу наставе и оцењивања.

5. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

Имплементација система заснива се на Microsoft Service Fabric платформи, која омогућава дистрибуиран рад више независних микросервиса. Бекенд је реализован у .NET окружењу, где сваки сервис има јасно дефинисану улогу: управљање корисницима, пријем и обраду студентских радова, евалуацију садржаја и анализу напредовања. Комуникација између сервиса реализована је преко Service Fabric remoting механизма, док WebApi Gateway служи као посредник између фронтенда и унутрашњих сервиса и обезбеђује реалновременску синхронизацију преко SignalR технологије. База података је реализована у MongoDB систему за трајно чување информација, уз ReliableDictionary који обезбеђује брз приступ активним подацима. За потребе аутоматске анализе садржаја и генерисање педагошког извештаја интегрисани су LLM (Large Language Model) модели путем Groq API интерфејса, који омогућава ефикасно извршавање и обраду великих језичких упита. Поред тога, у тестне сврхе је коришћен и Gemini API ради поређења резултата и процене квалитета анализе. На основу излазних података LLM-а систем генерише оцену, повратне информације и препоруке за студента, које се затим користе у сервису за анализу напретка (ProgressService).
Фронтенд део апликације реализован је у React окружењу и пружа интуитиван интерфејс за све корисничке улоге — студента, професора и администратора. Систем је тестиран у локалном Service Fabric кластеру, где је потврђена стабилност, поузданост и могућност скалабилног рада у вишекорисничком окружењу.

6. ЗАКЉУЧАК

У овом раду представљен је интелигентни систем за аутоматску анализу и евалуацију образовних радова, заснован на микросервисној архитектури и Microsoft Service Fabric окружењу. Комбинацијом више независних сервиса, MongoDB базе података и React фронтенда постигнут је стабилан и скалабилан систем који омогућава студентима, професорима и администраторима ефикасну интеракцију и надзор над процесом оцењивања. Интеграцијом LLM модела путем Groq API-ја, систем добија могућност разумевања и анализе садржаја студентских радова, чиме се унапређује квалитет повратних информација и омогућава бржа и објективнија евалуација. Развијени систем представља основу за даљи развој интелигентних образовних платформи које комбинују микросервисну архитектуру и вештачку интелигенцију ради подршке наставном процесу, а посебно аутоматизацији анализе, праћењу напретка и индивидуализацији учења.

7. ЛИТЕРАТУРА

[1] Microsoft Service Fabric. Приступ: 20.10.2025. Кључне речи: Service Fabric, микросервисна архитектура, Microsoft. Доступно на: https://learn.microsoft.com/en-us/azure/service-fabric/.

[2] MongoDB. Приступ: 20.10.2025. Кључне речи: MongoDB, NoSQL, база података. Доступно на: https://www.mongodb.com/.

[3] React – The library for web and native user interfaces. Приступ: 21.10.2025. Кључне речи: React, JavaScript, фронтенд развој. Доступно на: https://react.dev/.

[4] Groq API. Приступ: 22.10.2025. Кључне речи: Groq, LLM, вештачка интелигенција. Доступно на: https://groq.com/.

[5] OpenAI ChatGPT. Приступ: [20.10.2025 – 02.11.2025]. Кључне речи: ChatGPT, OpenAI, AI помоћник. Доступно на: https://chat.openai.com/.

[6] GitHub Repository: Educational Analysis System – Service Fabric Implementation. Приступ: [20.10.2025 – 02.11.2025]. Кључне речи: GitHub, код, Service Fabric, микросервисна апликација. Доступно на: https://github.com/MastilovicRadoslav/CRUIS_Auto_Grader.

Кратка биографија:

Радослав Мастиловић рођен је 2001. године у Требињу. Основно и средње образовање завршио је у Гацку, након чега је уписао Основне академске студије на Факултету техничких наука у Новом Саду, смер Примењено софтверско инжењерство. Његова интересовања обухватају развој софтверских система, микросервисне архитектуре и примену вештачке интелигенције.
Контакт: radoslav930@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 007:004.056
ДОИ: https://doi.org/10.24867/35BE14Stanisic

Аутентификација и ауторизација базиране на сесијама и токенима за постојеће информационе системе

Session and Token Based Authentication and Authorization for Legacy Information Systems

Живко Станишић, *Факултет техничких наука, Нови Сад*

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Рад представља интеграцију OAuth 2.0 и OpenID Connect у постојеће апликације са сесијама, уз AWS Cognito систем за управљање корисницима и токенима. Предложени хибридни модел са .NET доказом концепта описује архитектуру, верификацију/опозив токена и креирање сесије, омогућавајући постепену миграцију ка cloud стандардима без реконструкције.*
Кључне речи: *OAuth 2.0, OpenID Connect, AWS Cognito, сесије, .NET*

Abstract – *This thesis presents the integration of OAuth 2.0 and OpenID Connect into legacy session-based applications, using AWS Cognito for user and token management. The proposed hybrid model, supported by a .NET proof of concept, describes the system architecture, token verification and revocation, and session creation, enabling a gradual migration to cloud standards without full system reconstruction.*

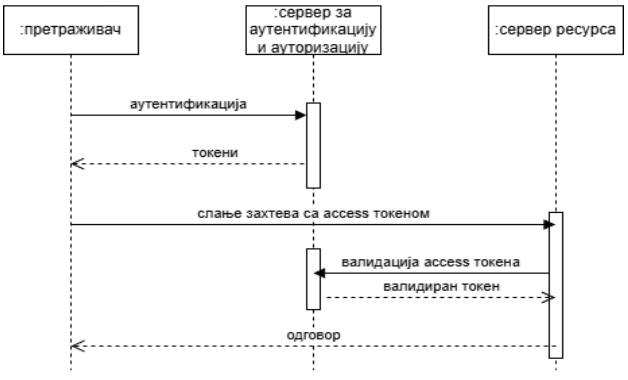
Keywords: *Auth 2.0, OpenID Connect, AWS Cognito, sessions, .NET*

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

1. УВОД

Уз ширење пословања на интернету јављају се изазови за старе системе, па предузећа често примењују хибрид *legacy* система са микросервисима у облаку [1]. У овом решењу аутентификација/ауторизација се ослања на *AWS Cognito* и *OAuth* токене [2]. Недостајући подаци се држе у серверској сесији повезаној са *access* токеном, а *refresh* токени у *DynamoDB* се могу опозвати при промени креденцијала/суспензији налога. Тако се испоштује *OAuth* и уз минималне измене постојећег кода добија практично, економично и лако прошириво решење за остатак система. У овом раду се приказује једно хибридно решење за аутентификацију и ауторизацију које омогућава интеграцију савремених *cloud* механизама са постојећим апликацијама које користе сесије. Циљ рада је да се прикаже како се употребом *OAuth 2.0* и *OpenID Connect* стандарда може проширити постојећи сесијски систем без потпуне реконструкције. На тај начин постиже се модернизација постојећих апликација и њихово усклађивање са савременим безбедносним стандардима. На слици 1 приказан је дијаграм секвенци протока у протоколу *OAuth 2.0*, који илуструје комуникацију између три главне компоненте система: претраживача, сервера за аутентификацију и ауторизацију и сервера ресурса.

1. **Аутентификација корисника** - процес почиње када корисник преко претраживача шаље захтев серверу за аутентификацију и ауторизацију, где се врши провера његовог идентитета.
2. **Издавање токена** - након успешне аутентификације, сервер враћа клијенту један или више токена (најчешће *access* и *refresh* токени) [4].
3. **Слање захтева ка серверу ресурса** - претраживач затим шаље *HTTP* захтев серверу ресурса, при чему у заглављу захтева шаље и добијени *access* токен.
4. **Валидација токена** – сервер ресурса прослеђује *access* токен серверу за аутентификацију и ауторизацију ради провере његове исправности и важења.
5. **Одговор ка клијенту** - ако је токен валидан, сервер ресурса враћа тражени одговор, а у супротном приступ је одбијен.



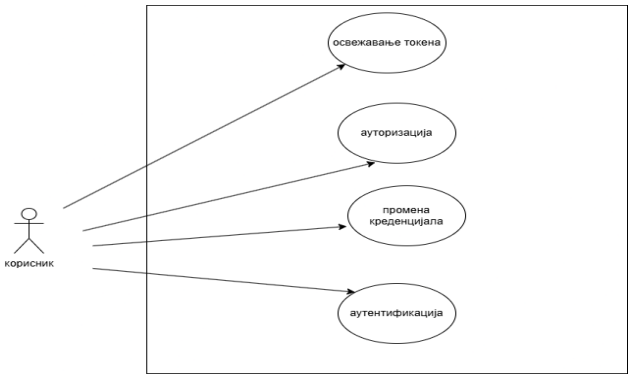
Слика 1. Дијаграм секвенци OAuth 2.0

2. ПРЕГЛЕД ПОСТОЈЕЋИХ РЕШЕЊА

Комбиновање сесија и токена задржава проверену интеракцију у претраживачу и обезбеђује интероперабилност са спољним провајдерима идентитета и *API*-јима у оквиру исте архитектуре. Обрасци се углавном разликују по месту чувања токена и где се терминише сесија, а основни циљ остаје исти [5]. Неки од образаца су: *Cookie Sessions and OpenID Connect*, *Legacy Monolith with Sessions and Federated Login*, *Backend-for-Frontend*, *API Gateway Token-to-Session Bridge* и *SaaS: App Session and Per-Provider OAuth Connections* [6][7][8].

3. МОДЕЛ СИСТЕМА

Поглавље укратко представља најчешће случајеве моделоване *UML* дијаграмима, са актерима: ауторизациони сервер, сервер ресурса и актера који их повезује. На слици 2 приказан је дијаграм случајева коришћења најчешћих сценарија коришћења система за аутентификацију и ауторизацију. Главни актер је корисник, који може да изврши више основних операција у оквиру система као што су: аутентификација, ауторизација, освежавање токена и промена креденцијала.

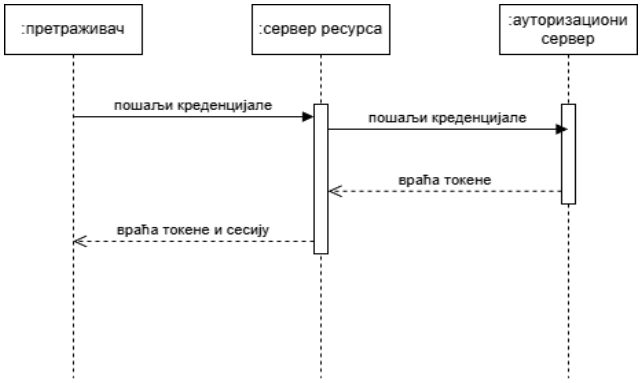


Слика 2. Приказ најчешћих случајева коришћења

3.1. Аутентификација

Аутентификација у овом решењу подразумева проверу валидности креденцијала и издавање *ID, access* и *refresh* токена у *JWT* формату [9]. Корисник шаље *email* и лозинку серверу ресурса, који преко ауторизационог сервера валидира креденцијале, добија токене, креира сесију и све заједно враћа претраживачу. На слици 3 приказан је дијаграм секвенци процеса аутентификације који илуструје размену података између претраживача, сервера ресурса и ауторизационог сервера.

1. **Слање креденцијала** - корисник преко претраживача уноси своје креденцијале који се шаљу ка серверу ресурса.
2. **Прослеђивање ка ауторизационом серверу** – сервер ресурса прослеђује креденцијале ауторизационом серверу који је задужен за проверу идентитета.
3. **Издавање токена** - након успешне провере ауторизациони сервер враћа токене серверу ресурса.
4. **Враћање токена и сесије клијенту** – сервер ресурса прослеђује добијене токене и информације о сесији назад претраживачу, чиме корисник постаје аутентификован и може приступати заштићеним ресурсима.

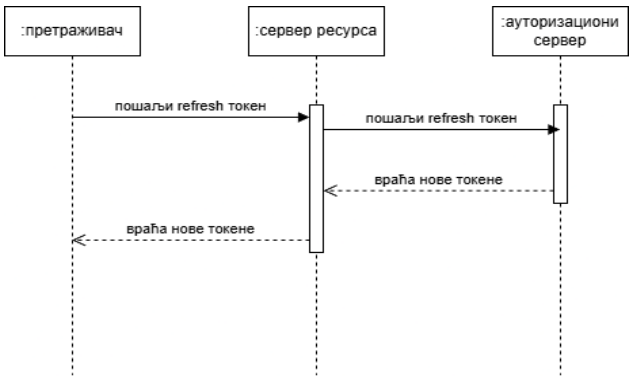


Слика 3. Дијаграм секвенци аутентификације

3.2. Освежавање токена

ID и *access* токени су кратког века, док *refresh* токен важи знатно дуже у складу са потребама пословања. Пре истека краткотрајних токена, клијент преко сервера ресурса шаље *refresh* токен ауторизационом серверу, који по валидацији издаје нове *ID* и *access* токене. На слици 4 приказан је дијаграм секвенци процеса освежавања токена, који описује како клијент добија нове токене.

1. **Слање *refresh* токена** - претраживач шаље *refresh* токен серверу ресурса како би добио нове токене без поновне пријаве.
2. **Прослеђивање ка ауторизационом серверу** – сервер ресурса прослеђује *refresh* токен ка ауторизационом серверу ради валидације.
3. **Издавање нових токена** - након успешне провере *refresh* токена, ауторизациони сервер враћа нове токене серверу ресурса.
4. **Враћање нових токена клијенту** - сервер ресурса прослеђује нове токене претраживачу, чиме се корисникова сесија продужава без потребе за поновним уносом креденцијала.



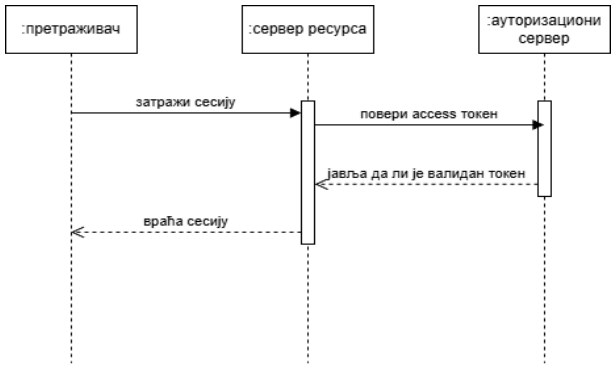
Слика 4. Дијаграм секвенци освежавање токена

3.3. Ауторизација

Ауторизација у овом решењу је сложенија јер се у постојећи сесијски систем уводи ауторизациони сервер. Сесија се логички дели тако да *access* токен

преузима улогу кључа за ауторизацију, а *ID* токен замењује сесијски објекат на клијенту. Због еволуције наслеђене апликације серверска сесија је дубоко укорењена у код, па клијент више нема потребу за сесијом док сервер и даље зависи од ње, што намеће хибридно повезивање сервера ресурса и ауторизационог сервера. На слици 5 приказан је дијаграм секвенци процеса ауторизације, који илуструје ток комуникације између претраживача, сервера ресурса и ауторизационог сервера током провере приступа.

1. **Захтев за сесију** - корисник преко претраживача шаље захтев серверу ресурса ради приступа заштићеним подацима.
2. **Провера *access* токена** – сервер ресурса шаље добијени *access* токен ка ауторизационом серверу ради провере његове валидности.
3. **Одговор о валидности токена** - ауторизациони сервер враћа резултат провере, тј. да ли је токен валидан и да ли корисник има потребна права приступа.
4. **Враћање сесије клијенту** - уколико је токен исправан, сервер ресурса враћа кориснику активну сесију и дозвољава приступ ресурсима.

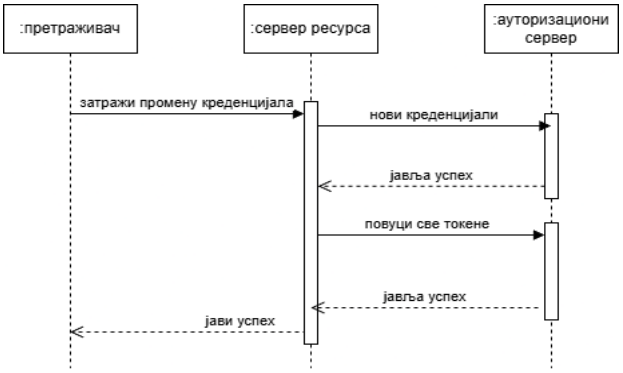


Слика 5. Дијаграм секвенци ауторизације

3.4. Промена креденцијала

Промена креденцијала може довести до озбиљних пропуста ако се не обраде сви сценарији, нарочито кад корисник није улогован. Нападач може остати пријављен, злоупотребити још важећи *access* токен за креирање сесије или освежити приступ помоћу *refresh* токена. Решење је инвалидирање свих токена након промене креденцијала где сервер ресурса шаље идентификатор корисника ауторизационом серверу, који повлачи токене. Исти поступак важи и за деактивацију корисника. На слици 6 приказан је дијаграм секвенци промене креденцијала који приказује кораке комуникације између претраживача, сервера ресурса и ауторизационог сервера током ажурирања података за пријаву.

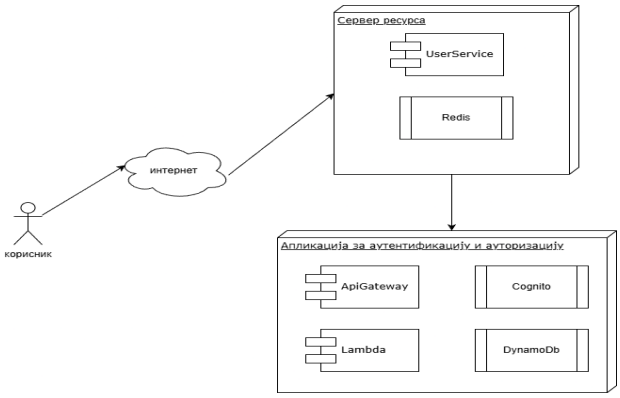
1. **Захтев за промену креденцијала** - корисник преко претраживача шаље захтев серверу ресурса за промену креденцијала.
2. **Прослеђивање нових креденцијала** – сервер ресурса шаље нове креденцијале ауторизационом серверу ради ажурирања података.
3. **Потврда успеха** - ауторизациони сервер враћа поруку о успешној промени креденцијала.
4. **Повлачење постојећих токена** - након успешног ажурирања, сервер ресурса шаље захтев за поништавање свих постојећих токена који су били повезани са старим креденцијалима.
5. **Коначна потврда успеха** - ауторизациони сервер потврђује повлачење токена, након чега сервер ресурса обавештава клијента да је процес успешно завршен.



Слика 6. Дијаграм секвенци промене креденцијала

4. ИМПЛЕМЕНТАЦИЈА

Имплементација се састоји из два дела: *TypeScript/Node.js* апликације за аутентификацију и ауторизацију на *AWS*-у (*Cognito, DynamoDB, API Gateway, Lambda*) која примењује микросервисну архитектуру и решава скалабилност и једноставне ресурсне *.NET* апликације са сесијама у *Redis*-у. На слици 7 приказан је *deployment* дијаграм целог система. Корисник преко интернета приступа серверу ресурса који чине сервиси, као што је *UserService*, који је део *.NET Web API*-ја и *Redis* база података намењена за чување сесија. Сервер ресурса има приступ апликацији за аутентификацију и ауторизацију, која користи поменуте *Cognito, DynamoDB, API Gateway* и *Lambda* сервисе [10][11].



Слика 7. Приказ *deployment* дијаграма

У наставку је дат једноставан пример који приказује два повезана елемента процеса креирања сесије са ауторизацијом. На слици 8 приказана је

имплементација функције *action* која врши проверу ваљаности *access* токена у процесу ауторизације.

```
export async function action(
    request: ValidateAccessTokenRequest
): Promise<ValidateAccessTokenResponse> {
    const provider = new CognitoIdentityProvider({
        region: validateAndGetRegion(),
    });
    const user = await provider.getUser({
        AccessToken: request.accessToken,
    });

    if (!!user.Username && user.Username !== '') {
        return { isValid: true };
    }

    return { isValid: false };
}
```

Слика 8. Приказ акције ауторизације

На слици 9 приказана је метода *CreateSession*, која омогућава креирање нове сесије за већ аутентификованог корисника. Ова метода позива горе поменуту акцију да провери да ли је *access* токен валидан.

```
public async Task<UserSession> CreateSession()
{
    var user = _httpContextAccessor.HttpContext?.User;
    if (user == null || !user.Identity?.IsAuthenticated == true)
    {
        throw new UnauthorizedException();
    }

    var email = user.FindFirst("username")?.Value;
    var systemUser = _repo.GetByEmail(email)
        ?? throw new KeyNotFoundException("User not found.");
    var authHeader = _httpContextAccessor.HttpContext?.Request
        .Headers[HeaderNames.Authorization].FirstOrDefault();

    var token = authHeader != null &&
        authHeader.StartsWith("Bearer ", StringComparison.OrdinalIgnoreCase)
        ? authHeader["Bearer ".Length..].Trim()
        : authHeader;

    var accessTokenResponse = await _authServiceClient
        .ValidateAccessToken(new ValidateAccessTokenRequest
    {
        AccessToken = token
    });

    if (accessTokenResponse == null || !accessTokenResponse.IsValid)
    {
        throw new UnauthorizedException();
    }

    return await _sessionService.CreateAsync(systemUser);
}
```

Слика 9. Приказ методе за креирање сесије

5. ЗАКЉУЧАК

Рад описује хибридни систем аутентификације и ауторизације (*OAuth 2.0* и *OpenID Connect*) који опслужује апликацију са сесијама, ослања се на *AWS* (*Cognito, DynamoDB, Lambda*) и интегрише се са *.NET/Redis* ресурсном апликацијом. Структура обухвата теоријски увод, постојеће хибридне обрасце, моделе и функционалности система, технологије и њихову конфигурацију, као и примере имплементације и токове аутентификације, ауторизације, промене креденцијала. Кључно ограничење је везаност за једног *cloud* провајдера и зависност од сесијског кода. Даљи рад подразумева енкапсулацију и провајдер-агностичан дизајн ради лакше подршке новим апликацијама.

6. ЛИТЕРАТУРА

[1] Sam Newman. *Building Microservices* (2nd ed.). O'Reilly Media, 2021.
[2] RFC 9700 (BCP 240): *OAuth 2.0 Security Best Current Practice,* 2025.
[5] Fett, D., Küsters, R., Schmitz, G. *A Comprehensive Formal Security Analysis of OAuth 2.0.*
[6] Richer, J., Sanso, A. *OAuth 2 in Action.* Manning Publications, 2017.
[7] Madden, N. *API Security in Action.* Manning Publications, 2020.
[8] Siriwardena, P., Dias, N. *Microservices Security in Action.* Manning Publications, 2020.
[9] RFC 7519 — JSON Web Token (JWT), IETF, 2015.
[10] Michael Wittig, Andreas Wittig. *Amazon Web Services in Action (3rd ed.).* Manning Publications, 2023.
[11] Mark J. Price. *C# 12 and .NET 8 – Modern Cross-Platform Development.* Packt, 2024.

Кратка биографија:

Живко Станишић рођен је 11.08.1995. године у Сомбору. Завршио је основне академске студије 30.10.2019. на Факултету техничких наука у Новом Саду, одсек Рачунарство и аутоматика. Уписао је мастер академске студије. **Контакт**: zivkostanisic@pm.me

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.3
DOI: https://doi.org/10.24867/35BE15Vujakovic

SISTEM ZA UPRAVLJANJE PAMETNIM KUĆAMA PUTEM IOT TEHNOLOGIJA

SMART HOME MANAGEMENT SYSTEM VIA IOT TECHNOLOGIES

Jelena Vujaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad prikazuje razvoj i implementaciju naprednog rešenja za upravljanje pametnim kućama zasnovanog na Internetu stvari (IoT), cloud infrastrukturi i višeplatformskoj korisničkoj aplikaciji. Sistem integriše uređaje, senzore i servise u jedinstveno inteligentno okruženje koje omogućava praćenje i upravljanje ambijentalnim uslovima u realnom vremenu. Arhitektura sistema je modularna i sastoji se od tri glavne komponente: IoT uređaja zasnovanog na ESP32 mikrokontroleru sa DHT22 senzorom, serverless pozadinskog sistema implementiranog pomoću Azure Functions i MongoDB baze podataka, te Flutter aplikacije dostupne na više platformi (Android, iOS, web).*

Ključne reči: *IoT, CPU, Edge Node, Računarstvo u oblaku.*

Abstract – *This paper presents the development and implementation of an advanced smart home management system based on the Internet of Things (IoT), cloud infrastructure, and a multiplatform user application. The system integrates devices, sensors, and services into a unified intelligent environment that enables real-time monitoring and control of ambient conditions. The system architecture is modular and consists of three main components: an IoT device based on an ESP32 microcontroller with a DHT22 sensor, a serverless backend implemented using Azure Functions and a MongoDB database, and a Flutter application available across multiple platforms (Android, iOS, and web).*

Keywords: *IoT, CPU, Edge Node, Cloud Computing.*

1. UVOD

Pametne kuće predstavljaju jednu od najdinamičnijih oblasti savremene tehnologije, nudeći korisnicima mogućnost da unaprede kvalitet života kroz automatizaciju, optimizaciju potrošnje energije i povećanje bezbednosti. Razvoj IoT (Internet of Things) rešenja za pametne kuće omogućava povezivanje različitih uređaja i senzora, čime se stvara inteligentno okruženje koje reaguje na promene u realnom vremenu i prilagođava se potrebama korisnika.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Lukić, docent

U tom kontekstu, sistem je osmišljen kao napredno, skalabilno i fleksibilno rešenje koje integriše najnovije tehnologije iz oblasti IoT-a, cloud computinga i multiplatformskih korisničkih interfejsa.

Sistem koristi Azure Functions kao serverless backend, što omogućava automatsko skaliranje resursa i optimizaciju troškova, dok MongoDB obezbeđuje fleksibilno skladištenje podataka bez stroge šeme. Real-time komunikacija je omogućena korišćenjem SignalR tehnologije, koja obezbeđuje trenutna ažuriranja korisnicima bez potrebe za ručnim osvežavanjem. Arhitektura sistema je modularna i obuhvata tri glavne komponente: Flutter klijentsku aplikaciju dostupnu na Android, iOS i web platformama, C# backend implementiran kao skup Azure Functions, i IoT uređaj baziran na Arduino tehnologiji opremljen DHT22 senzorom. Poseban naglasak je stavljen na bezbednost i zaštitu podataka kroz implementaciju sigurnih komunikacionih protokola i autentikacije korisnika. HomeR omogućava korisnicima potpunu kontrolu nad pametnim uređajima, uz mogućnost daljinskog upravljanja i praćenja u realnom vremenu, što doprinosi komforu i energetskoj efikasnosti doma. Sistem je projektovan sa mogućnošću proširenja na dodatne senzore i pametne uređaje, čime se može prilagoditi specifičnim potrebama korisnika i budućim tehnološkim trendovima. Ovaj rad ima za cilj da pruži sveobuhvatan pregled razvoja, implementacije i primene Sistema za upravljanje pametnim kućama, predstavljajući korak ka budućnosti u kojoj će pametne kuće biti standard moderne tehnologije.

2. PREGLED SLIČNIH SISTEMA

U oblasti upravljanja pametnim kućama putem IoT tehnologija postoji nekoliko značajnih komercijalnih rešenja koja dele slične ciljeve sa predloženim sistemom. Google Nest ekosistem integriše pametne termostate, sigurnosne kamere i glasovne asistente kroz centralizovanu platformu za optimizaciju energetske potrošnje i bezbednost doma. Amazon Alexa platforma omogućava upravljanje širokim spektrom pametnih uređaja kroz glasovne komande i jedinstveni interfejs. Apple HomeKit sistem fokusira se na privatnost podataka, pružajući lokalno upravljanje uređajima sa end-to-end enkripcijom. Samsung SmartThings Hub predstavlja otvorenu platformu koja podržava različite komunikacione protokole (Zigbee, Z-Wave, WiFi) i integraciju uređaja različitih proizvođača. Međutim, većina postojećih rešenja oslanja se na zatvoren

ekosistem specifičnih proizvođača, što ograničava fleksibilnost i povećava troškove implementacije. Predloženi sistem se razlikuje po otvorenoj, modularnoj arhitekturi zasnovanoj na standardnim IoT tehnologijama i Azure cloud platformi, omogućavajući kombinovanje uređaja različitih proizvođača uz manje troškove i veću prilagodljivost.

3. OPIS KORIŠĆENIH TEHNOLOGIJA

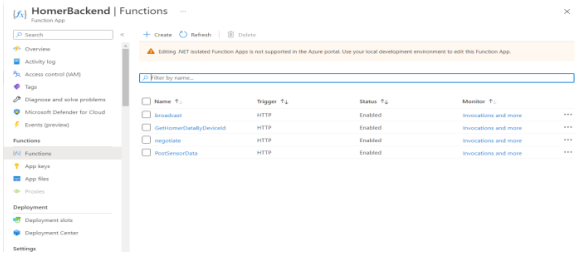
Implementacija sistema za upravljanje pametnim kućama zahteva integraciju različitih tehnologija koje omogućavaju efikasno prikupljanje, obradu i vizualizaciju podataka. Internet of Things (IoT) tehnologije predstavljaju osnovu sistema, omogućavajući povezivanje fizičkih uređaja sa internetom radi prikupljanja i razmene podataka, pri čemu su korišćeni ESP32 mikrokontroleri sa WiFi modulima koji obezbeđuju pouzdanu komunikaciju između senzora i cloud infrastrukture, dok DHT22 senzori omogućavaju precizno merenje temperature i vlažnosti okruženja uz Arduino IDE kao razvojno okruženje za programiranje mikrokontrolera u C++ programskom jeziku. Microsoft Azure cloud platforma pruža skalabilnu infrastrukturu za backend sistem, uključujući Azure Functions za računarstvo bez servera, Azure IoT Hub za upravljanje IoT uređajima, i Azure SignalR Service za komunikaciju u realnom vremenu, omogućavajući automatsko skaliranje resursa, visoku dostupnost i napredne sigurnosne mehanizme potrebne za pouzdano funkcionisanje sistema. MongoDB baza podataka koristi se za skladištenje senzorskih podataka, omogućavajući fleksibilno upravljanje nestrukturiranim podacima bez potrebe za fiksnom šemom, što je posebno pogodno za IoT aplikacije gde se struktura podataka može menjati tokom vremena. Flutter framework i Dart programski jezik omogućavaju razvoj multiplatformske korisničke aplikacije koja funkcioniše na Android, iOS, web, Windows, macOS i Linux platformama, pri čemu Flutter pruža bogat skup alata za kreiranje korisničkih interfejsa, dok Dart omogućava efikasno asinhrono programiranje potrebno za komunikaciju u realnom vremenu. SignalR tehnologija obezbeđuje dvosmernu komunikaciju u realnom vremenu između servera i klijentskih aplikacija, omogućavajući korisnicima da budu trenutno obavešteni o promenama u sistemu bez potrebe za ručnim osvežavanjem aplikacije, dok RESTful API arhitektura omogućava standardizovanu komunikaciju između različitih komponenti sistema, pružajući jasno definisane krajnje tačke za prikupljanje i distribuciju podataka između IoT uređaja, Cloud servisa i korisničkih aplikacija.

3.1 LoRa uređaj: Ivični uređaj za prikupljanje podataka

LoRa uređaj, implementiran kroz ESP32 mikrokontroler sa LoRa modulom, predstavlja ključnu komponentu sistema za prikupljanje podataka na ivici mreže (engl. Edge computing), omogućavajući efikasno prikupljanje senzorskih informacija iz okruženja uz minimalnu potrošnju energije. Ovaj uređaj funkcioniše kao krajnji čvor u IoT arhitekturi, opremljen DHT22 senzorom za precizno merenje temperature i vlažnosti. Kombinacija 32-bitnog dual-core Xtensa LX6 procesora koji radi na frekvenciji do 240 MHz, zajedno sa 520 KB SRAM memorije i 4 MB Flash memorije, omogućava lokalnu obradu podataka pre njihovog prosleđivanja centralnom sistemu, čime se smanjuje mrežno opterećenje i povećava efikasnost celokupnog sistema. Ugrađeni 0.96-inčni OLED displej pruža vizuelnu povratnu informaciju o statusu uređaja i kvalitetu signala, dok napredni sistem upravljanja napajanjem omogućava dugotrajan rad na baterijama kroz različite režime niske potrošnje energije, uključujući mod dubokog spavanja sa potrošnjom manjom od 10 μA. Softverska implementacija realizovana je kroz Arduino IDE razvojno okruženje korišćenjem C programskog jezika, pri čemu se podaci prikupljaju periodično (na svakih 60 sekundi), formatiraju u JSON strukture sa vremenskim oznakama sinhronizovanim sa NTP serverom, i asinhrono šalju ka Azure backend infrastrukturi putem sigurnih HTTP POST zahteva, omogućavajući kontinuirano praćenje ambijentalnih uslova i real-time reakcije sistema na promene u okruženju.

3.1 Pozadinska i Azure Infrastruktura

Pozadinska infrastruktura sistema za upravljanje pametnim kućama implementirana je pomoću Microsoft Azure Cloud platforme. Arhitektura je zasnovana na računarskom modelu bez održavanja servera korišćenjem Azure funkcija. Sistem je organizovan kao višeslojna arhitektura koja se sastoji od prezentacionog sloja implementiranog kroz Azure Functions HTTP krajnje tačke za komunikaciju sa IoT uređajima i klijentskim aplikacijama, servisnog sloja koji sadrži poslovnu logiku kroz HomerService klasu za obradu i validaciju podataka, te sloja za pristup podacima realizovanog putem MongoDB NoSQL baze podataka koja omogućava fleksibilno skladištenje nestrukturiranih senzorskih informacija bez potrebe za fiksnom šemom. U projektu je korišćen C# programski jezik za implementaciju poslovne logike, pri čemu su kreirana tri ključna Azure Functions endpoint-a: PostSensorData funkcija za prijem JSON podataka sa ESP32 uređaja putem HTTP POST zahteva, GetDeviceInformation funkcija za preuzimanje informacija o uređaju na osnovu identifikatora, te broadcast i negotiate funkcije za realizaciju SignalR komunikacije u realnom vremenu. MongoDB je integrisana kroz generički MongoDBRepository<T> repozitorijum koji omogućava centralizovano upravljanje kolekcijama, sa specifičnim repozitorijumima HomerDataRepository za senzorske podatke i DeviceDataRepository za informacije o uređajima, dok WeatherRepository služi za komunikaciju sa spoljašnjim vremenskim servisima. Azure SignalR servis obezbeđuje dvosmerni prenos podataka u realnom vremenu između servera i klijentskih aplikacija kroz currentSensorData kanal, omogućavajući korisnicima trenutna ažuriranja bez ručnog osvežavanja, dok Azure Key Vault čuva bezbedne konfiguracije i pristupne ključeve. Deployment pozadinskog koda realizuje se automatski iz Visual Studio razvojnog okruženja ka Azure cloud platformi kroz kontinuiranu integraciju resursa, što omogućava sistemu dinamičko skaliranje, visoku dostupnost preko ugovorenih garancija kvaliteta usluge i naprednu sigurnost kroz Azure Active Directory sistem za potvrđivanje identiteta autentifikaciju.



Slika 1. Prikaz ključnih funkcionalnosti backend sloja za upravljanje senzorskim, uređajskim i vremenskim podacima u HomeR sistemu

3.2 Korisnički Interfejs HomeR Sistema: Multiplatformska Aplikacija za Pametnu Kuću

Korisnički interfejs predstavlja ključnu komponentu sistema, omogućavajući intuitivno i efikasno upravljanje pametnim domom kroz naprednu višeplatformsku aplikaciju. Razvijen korišćenjem Flutter okvira i Dart programskog jezika, interfejs omogućava korisnicima da pristupe funkcionalnostima sistema sa bilo kog uređaja - mobilnih telefona, tablet računara, stolnih računara ili direktno kroz veb preglednike. Aplikacija je projektovana kao centralna kontrolna tabla koja omogućava praćenje podataka sa senzora u realnom vremenu, upravljanje pametnim uređajima na daljinu, kao i prilagođavanje korisničkog iskustva prema individualnim potrebama.

Arhitektura korisničkog interfejsa zasniva se na modularnim komponentama koje omogućavaju jasno odvajanje funkcionalnosti i lako proširenje sistema, uključujući komponente za vizualizaciju podataka o temperaturi i vlažnosti, prikaz spoljašnjih vremenskih uslova, kao i upravljačke module za različite pametne uređaje u domaćinstvu. Posebnu karakteristiku predstavlja implementacija dvosmerne komunikacije sa pozadinskim sistemom, gde se koristi kombinacija poziva ka programskom interfejsu aplikacije za inicijalno preuzimanje podataka i SignalR tehnologija za trenutna ažuriranja bez potrebe za osvežavanjem aplikacije. Višeplatformska priroda aplikacije omogućava korisnicima kontinuiran pristup funkcionalnostima sistema bez obzira na njihovu trenutnu lokaciju ili tip uređaja koji koriste, pri čemu se aplikacija automatski prilagođava različitim veličinama ekrana i operativnim sistemima, zadržavajući konzistentnost korisničkog iskustva na svim platformama.

Slika 2. Prikaz rada front-end aplikacije na Mac OS emulatoru za vizuelizaciju senzorskih i uređajskih podataka u HomeR sistemu

4. SPECIFIKACIJA

Cilj ovog projekta je razvoj naprednog sistema za upravljanje pametnim kućama koristeći Internet stvari tehnologije, pri čemu je sistem implementiran korišćenjem Azure cloud platforme koja omogućava skalabilno skladištenje podataka, serverless obradu i komunikaciju u realnom vremenu između IoT uređaja i korisničkih aplikacija. Projekt obuhvata implementaciju višeplatformske aplikacije koja omogućava praćenje ambijentalnih uslova u domu na osnovu podataka dobijenih od senzora za temperaturu i vlažnost, koristeći ESP32 mikrokontroler sa DHT22 senzorom za prikupljanje podataka, Azure Functions za serverless obradu i MongoDB za fleksibilno skladištenje informacija. Ključni aspekt sistema predstavlja obrada senzorskih podataka u realnom vremenu i omogućavanje korisnicima da prate uslove u svom domu putem intuitivne višeplatformske aplikacije razvijene u Flutter razvojnom okruženju, uz integraciju SignalR servisa za trenutnu komunikaciju između pozadinskog sistema i korisničke aplikacije. Ovakav pristup omogućava automatska ažuriranja korisničkog interfejsa bez potrebe za ručnim osvežavanjem, čime se obezbeđuje kontinuirano praćenje parametara okruženja i efikasno upravljanje pametnim domom kroz centralizovanu kontrolnu tablu dostupnu na različitim platformama i uređajima.

5. IMPLEMENTACIJA

Sistem je razvijen kroz nekoliko ključnih faza, od inicijalizacije infrastrukture za Internet stvari do prikupljanja i distribucije podataka sa senzora, uz korišćenje fundamentalnih koncepata računarstva u Cloudu kao što su funkcije bez servera, nerelacione baze podataka i komunikacija u realnom vremenu. Sistem koristi višeslojnu arhitekturu koja omogućava komunikaciju između različitih komponenti unutar ekosistema pametne kuće, uključujući uređaje za Internet stvari, pozadinski sistem i korisničke aplikacije, pri čemu ovaj model omogućava razvoj aplikacija koje efikasno razmenjuju podatke preko protokola za prenos hiperteksta i SignalR tehnologije, koristeći generisane interfejse za razmenu informacija između servisa. Na početku, senzori prikupljaju podatke o temperaturi i vlažnosti u prostoru i te informacije prosleđuju sistemu putem dokumenata u notaciji objekata definisanih u okviru programskog interfejsa aplikacije, dok se podaci dalje obrađuju kroz faze obrade podataka bez servera i upravljanja stanjem aplikacije, osiguravajući stabilno i pouzdano funkcionisanje sistema. Azure funkcije su zadužene za upravljanje životnim ciklusom zahteva, dok baza podataka kontroliše skladištenje i preuzimanje podataka u različitim stanjima, uključujući inicijalno čuvanje i kasniju analizu informacija. U okviru razvoja projekta za upravljanje pametnim kućama pomoću senzora za Internet stvari, servisni interfejsi se definišu za ključne komponente koje omogućavaju komunikaciju između različitih delova sistema, pri čemu prva komponenta predstavlja senzorski uređaj koji prikuplja podatke o ambijentalnim uslovima pomoću senzora za temperaturu i vlažnost. Servisni interfejs za senzorski uređaj omogućava slanje podataka o temperaturi i vlažnosti ka komponenti za obradu podataka korišćenjem zahteva za slanje podataka, dok senzorski uređaj periodično prikuplja podatke i koristi definisani interfejs da prosledi informacije procesoru

podataka radi dalje obrade. Procesor podataka predstavlja ključnu komponentu koja obrađuje podatke o ambijentalnim uslovima i donosi odluke o daljim akcijama u sistemu, sa definisanim servisnim interfejsom koji omogućava prijem podataka sa senzora za Internet stvari i obezbeđuje komunikaciju između senzora i modula za obradu podataka, uključujući pravovremenu analizu promena u okruženju i evidentiranje upozorenja kada vrednosti prelaze definisane pragove. Takođe, servisni interfejsi su definisani i za korisnički interfejs i komunikaciju u realnom vremenu, koje upravljaju korisničkim interakcijama i procesima unutar sistema, omogućavajući kontrolu nad prikazom podataka, ažuriranje korisničkog interfejsa u realnom vremenu i omogućavanje dvosmerne komunikacije u skladu sa potrebama sistema, čime svaki deo sistema ima jasno definisane komunikacione kanale što omogućava efikasan i pouzdan rad aplikacije za upravljanje pametnim domom.

5.1 SignalR i integracija sa pozadinskim sistemom

Tehnologija za komunikaciju u realnom vremenu predstavlja ključnu komponentu za uspostavljanje dvosmerne komunikacije između pozadinskog sistema i korisničke aplikacije, omogućavajući trenutno ažuriranje podataka bez potrebe za ručnim osvežavanjem interfejsa, pri čemu je ova tehnologija implementirana kao servis u okviru Cloud platforme koji pruža skalabilnu infrastrukturu za upravljanje velikim brojem istovremenih konekcija i efikasnu distribuciju podataka ka svim povezanim klijentima. Integracija servisa za komunikaciju u realnom vremenu sa pozadinskim sistemom ostvaruje se kroz Cloud funkcije koje služe kao posrednici između baze podataka i klijenata, gde se podaci sa senzora automatski prosleđuju svim aktivnim korisnicima čim stanu na raspolaganje, dok se arhitektura komunikacije zasniva na centralnom čvoru koji upravlja konekcijama i distribucijom poruka prema registrovanim klijentima. Kada senzorski uređaj pošalje nove podatke pozadinskom sistemu putem programskog interfejsa aplikacije, Cloud funkcija prima podatke, obrađuje ih kroz poslovnu logiku i zatim koristi servis za komunikaciju u realnom vremenu da emituje ažurirane informacije ka svim povezanim klijentskim aplikacijama, pri čemu se ovaj proces odvija asinhrono omogućavajući sistemu da istovremeno obrađuje mnogostruke zahteve i održava visoke performanse bez blokiranja drugih operacija. Implementacija klijenta u korisničkoj aplikaciji realizuje se kroz uspostavljanje trajne konekcije sa pozadinskim sistemom, gde aplikacija inicijalno šalje zahtev za dobijanje pristupnog tokena i podešavanja, nakon čega se registruje za praćenje određenih kanala komunikacije kao što je kanal za senzorske podatke, vremensku prognozu ili informacije o stanju uređaja. Klijentska strana implementira slušaoce događaja koji reaguju na pristigle poruke sa servera, automatski ažuriraju korisnički interfejs i obaveštavaju korisnika o relevantnim promenama u sistemu, dok je bezbednost komunikacije bezbeđena kroz implementaciju sigurnih protokola za uspostavljanje konekcije, potvrdu identiteta korisnika i šifrovanje podataka tokom prenosa. Pozadinski sistem proverava sve dolazne zahteve, kontroliše dozvole korisnika za pristup određenim podacima i implementira mehanizme za sprečavanje neovlašćenog pristupa, dodatno omogućavajući podešavanje različitih nivoa pristupa čime se osigurava da korisnici primaju samo one informacije zaoje imaju odgovarajuće dozvole, dok je optimizacija performansi postignuta kroz implementaciju mehanizama za upravljanje konekcijama, automatsko ponovno povezivanje u slučaju prekida mreže i inteligentno grupiranje poruka radi smanjenja mrežnog saobraćaja.

6. ZAKLJUČAK

Razvoj sistema za upravljanje pametnim kućama putem tehnologija Interneta stvari predstavlja značajan korak ka digitalizaciji savremenih domaćinstava, omogućavajući korisnicima potpunu kontrolu nad uslovima u okruženju kroz implementaciju napredne arhitekture koja kombinuje senzorske uređaje, Cloud infrastrukturu i višeplatformske korisničke aplikacije. Ključni doprinos ovog rada ogleda se u uspešnoj integraciji ivičnog uređaja sa senzorima za temperaturu i vlažnost, Cloud servisa za obradu podataka i korisničke aplikacije razvijene u Flutter razvojnom okruženju, pri čemu je korišćenje funkcija bez servera omogućilo kreiranje skalabilne arhitekture koja se automatski prilagođava opterećenju sistema. Modularnost sistema predstavlja najvažniju karakteristiku ostvarenog rešenja, omogućavajući lako proširenje funkcionalnosti i dodavanje novih komponenti bez značajnih promena postojeće arhitekture, dok je testiranje u realnim uslovima potvrdilo pouzdanost, stabilnost i performanse rešenja. Implementirani sigurnosni mehanizmi, uključujući šifrovanje komunikacije i kontrolu pristupa, obezbeđuju zaštitu privatnosti korisnika, dok značaj rada se ogleda u demonstraciji mogućnosti kreiranja ekonomski isplativog rešenja koje koristi dostupne tehnologije i otvorene standarde. Pravci za buduća istraživanja uključuju integraciju veštačke inteligencije za prediktivnu analizu podataka, proširenje na dodatne tipove senzora i implementaciju naprednih algoritma za optimizaciju potrošnje energije, čime ovaj rad predstavlja solidnu osnovu za dalja istraživanja u oblasti pametnih kuća i tehnologija Interneta stvari.

7. LITERATURA

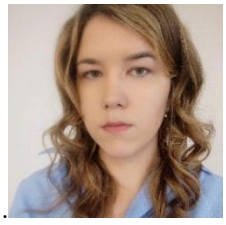
[1] Erl, T., Puttini, R. and Mahmood, Z., 2013. Cloud computing: concepts, technology & architecture. Pearson Education.

[2] Cao, J., Zhang, Q. and Shi, W., 2018. Edge computing: a primer. Springer.

[3] https://learn.microsoft.com/en-us/azure/azure-functions/ (pristupljeno u novembru 2025.)

[4] https://learn.microsoft.com/en-us/azure/iot-hub/ (pristupljeno u novembru 2025.

Kratka biografija:



Jelena Vujaković rođena je u Somboru 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Embeded sistemi i algoritmi, odbranila je 2025.god.

kontakt:
jelena.vujakovic7@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.43
ДОИ: https://doi.org/10.24867/35BE16Bulaja

Имплементација Elasticsearch клијента у ZIO екосистему

Implementation of Elasticsearch client in ZIO ecosystem

Димитрије Булаја, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Овај рад описује израду и имплементацију ZIO Elasticsearch библиотеке, односно типски безбедног Elasticsearch клијента, природно интегрисаног у ZIO екосистем, употребом програмског језика Scala, ZIO окружења и других ZIO изворних библиотека.*

Кључне речи (три до пет): *ZIO, Elasticsearch, Scala, функционално програмирање*

Abstract – *This paper describes the development and implementation of the ZIO Elasticsearch library, a type-safe Elasticsearch client naturally integrated into the ZIO ecosystem, using the Scala programming language, the ZIO environment, and other ZIO-native libraries.*

Keywords: (three to five): *ZIO, Elasticsearch, Scala, functional programming*

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

1.УВОД

У свету дистрибуираних система и микросервисних архитектура, неопходна је технологија која омогућава брзу претрагу, анализу и обраду великих количина података, уз висок ниво доступности, капацитета, скалабилности и поузданости. Један од најзаступљенијих софтвера који испуњава ове захтеве јесте *Elasticsearch* [1]. *Elasticsearch* је моћан претраживачки и аналитички алат, изграђен на *Lucene* библиотеци [2], који омогућава ефикасну обраду и анализу различитих врста података.

Паралелно, *ZIO* екосистем [6], развијен у програмском језику *Scala,* омогућава типски безбедан и једноставан рад са ефектима, асинхроним операцијама, конкурентношћу и са руковањем грешкама, кроз јасан и функционалан стил програмирања.

До скоро није постојало адекватно нативно, идиоматско *ZIO* решење за интеграцију са *Elasticsearch*-ом. Због тога су програмери били приморани да развијају прилагођене интеграције, што је често водило ка комплекснијим системима, већем простору за грешке и смањеној поузданости.

Полазећи од те празнине, развијена је библиотека *ZIO Elasticsearch*, клијент који спаја *Elasticsearch* сервер са *ZIO* ефектним моделом. Библиотека обезбеђује типску сигурност приликом моделовања упита, агрегација, као и приликом стримовања података. Дизајнирана је да омогући једноставну употребу, високе перформансе, скалабилност и робусно управљање грешкама. Такође, обезбеђена је нативна интеграција са *ZIO* екосистемом.

Рад је организован у неколико поглавља. Након уводне мотивације, пружа се увид у теоријске основе и коришћене технологије. Затим је приказана имплементација библиотеке и њена употреба кроз малу демонстративну апликацију. На крају, у оквиру закључка, сумирани су резултати и могући правци будућег развоја.

2.ТЕОРИЈСКЕ ОСНОВЕ И КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ

У овом поглављу биће описани теоријски концепти, дефиниције и технологије које представљају основу за имплементацију библиотеке. Разумевање ових појмова је неопходно како би се јасно схватила логика, архитектура и решења која су примењена током развоја *ZIO Elasticsearch* библиотеке.

2.1.Функционално програмирање

Функционално програмирање представља једну од програмских парадигми у развоју софтвера, код које је главна идеја употреба функција као основне јединице кода. Поставља се акценат на чисте функције, имутабилност и декларативни стил програмирања, тачније представља својеврсну рестрикцију на то како програм треба нешто да уради, а не на то шта тај програм треба да уради. Овај приступ олакшава тестабилност, предвидљивост и паралелизацију, што је од суштинског значаја у дистрибуираним системима, док конструкције као што су функције вишег реда и богати типски системи повећавају изражајност и безбедност кода [3][5].

2.2.Scala и JVM

Програмски језик *Scala* осмишљен је као обједињење објектно-оријентисаног и функционалног програмирања, које пружа програмерима могућност да користе оба стила у складу са потребама апликације. Захваљујући чињеници да се *Scala* извршава на *Java* виртуелној машини, омогућена је интеграција са постојећим *Java* екосистемом. Ова компатибилност омогућава приступ великом броју постојећих

библиотека и алата, што је допринело широј примени *Scala*-е у развоју модерних система.
Scala нуди концизну синтаксу и снажан типски систем, што омогућава писање изражајнијег и безбеднијег кода. Такође, *Scala* пружа могућност развоја апликација које захтевају паралелизам и високе перформансе, а омогућава и креирање апстракција, које смањују дуплирање кода и поједностављују одржавање софтвера [4].

2.3.*ZIO* екосистем

Основни концепт *ZIO* библиотеке јесте ”ефекат” (тип *ZIO Effect*), који енкапсулира програмске операције попут читања, писања или управљања грешкама. Сем тога, обезбеђује снажан модел за управљање ресурсима и омогућава програмирање без споредних ефеката, што додатно олакшава развој апликација [3].
ZIO тип репрезентује се као *ZIO[R, E, A]*, где су *R*, *E* и *A* редом: окружење, грешка и успешан исход. Оваква структура омогућава потпуно раздвојено управљање успешним и неуспешним исходима, чиме се знатно поједностављује обрада грешака [6].
Још један од кључних аспеката *ZIO* библиотеке јесте комбиновање и трансформација ефеката (путем метода *map*, *flatMap* и *zip*). Поред тога, *ZIO* обезбеђује и једноставно управљање паралелизмом, што значи да se више ефеката могу истовремено извршавати без потребе за ручним управљањем конкурентношћу [7].
Иако је првобитно био усмерен само на управљање ефектима, *ZIO* временом прераста оквире библиотеке и еволуира у комплетан екосистем. Тај екосистем укључује разне библиотеке које омогућавају лакшу интеграцију са екстерним системима и базама података, рад са разним врстама података и изградњу дистрибуираних апликација [6].

2.4.Elasticsearch и *Lucene*

Elasticsearch је дистрибуирани софтвер отвореног кода који омогућава претрагу и анализу података. Развијен је у *Java* програмском језику и заснива се на *Lucene* библиотеци. Омогућава брз приступ информацијама у реалном времену кроз *REST API* и *JSON* формат.
Захваљујући свом дистрибуираном складишту са шардовима и репликама, овај софтвер програмерима пружа могућност рада са разноврсним типовима података. Кластер се може хоризонтално скалирати, а сви чворови унутар кластера су потпуно независни и самостални. Оваквим дизајном елиминише се ”*single point of failure*” проблем и задржавају се високе перформансе.
Један од основних коцепата *Elasticsearch*-а је структура података ”инвертовани индекс”, коју *Elasticsearch*, помоћу *Lucene* библиотеке, користи за ефикасну и брзу претрагу текста [8].
Lucene је библиотека за индексирање и претрагу текста, коју је развио *Apache Software Foundation* [9]. Ради побољшања перформанси претраживања, *Lucene* библиотека пре него што крене са индексирањем, врши претпроцесирање података, а потом те претпроцесиране податке складишти у инвертованом индексу. Ова библиотека такође подржава и различите типове претрага, од једноставних до веома комплексних упита и филтера.

2.5.Fluent *API*

Fluent API је дизајн шаблон који омогућава ланчано позивање метода над истим објектом, где свако ланчање враћа нову инстанцу и тиме одржава читљив, типски безбедан код, без сувишних променљивих. Често се користи у *Query Builder*-има, конфигурационим фајловима и спецификацијама пословне логике, где је потребна јасна семантика при конструисању сложених објеката.

2.6.Streaming у *Elasticsearch*-у

Streaming омогућава да се велики скупови резултата обрађују постепено, уместо да се цео сет учита одједном у меморију, што је кључно у дистрибуираним системима који раде у реалном времену. *Elasticsearch* нуди два механизма за *streaming*. Први механизам је *scroll*, он отвара серверски контекст и преко *scrollId* идентификатора враћа резултате део по део, све док се контекст експлицитно не затвори [10]. Други механизам се назива *search_after,* користи *stateless* приступ, који захтева сортиране резултате, док у свакој враћеној страници резултата шаље и вредности потребне за наставак претраге. У пракси се *search_after* често комбинује са *Point in Time* (*PIT*), који ”замрзава” индекс у тренутном стању. *PIT* гарантује да ће све странице резултата одражавати исту верзију података, чак и у случају да се у међувремену упишу нови документи, или бришу постојећи [11].

3.ИМПЛЕМЕНТАЦИЈА

Библиотека *ZIO Elasticsearc*h омогућава идиоматски *ZIO* приступ *Elasticsearch*-у, пружајући реактивни модел рада са асинхроним операцијама, као и сигурно руковање ресурсима и грешкама. Библиотека подржава верзије 7 и 8 *Elasticsearch*-а, као и развој апликација у *Scala* програмском језику верзија 2 и 3.
Изворни код библиотеке је организован у складу са стандардном *SBT* структуром пројекта, где се изворни фајлови налазе унутар путање *src/main*, а тестови унутар путање *src/test*.
Архитектура библиотеке заснива се на принципима функционалног програмирања у *ZIO* екосистему са акцентом на тестабилност, декларативност и *Separation of Concerns* [12]. У основи библиотеке се налази апстракција *ElasticRequest*, која моделује сваки захтев ка *Elasticsearch*-у и апстракција *Executor*, која извршава захтеве и одговорна је за комуникацију са *Elasticsearch* сервером. Кроз *ZIO* слојеве (*ZLayer* [14]) омогућена је једноставна примена *Dependency injection*-а [13] и изузетно флексибилно конфигурисање окружења.
Сви захтеви се моделују као класе, које наслеђују апстрактни тип *ElasticRequest[A],* где *A* представља енкапсулирани тип који се враћа приликом извршавања захтева. Корисници захтеве не креирају директно, већ преко јавног *API*-ја, а након креирања, могу над њима позивати и одговарајуће методе које постављају опционе параметре.
Executor апстракција дефинише како се захтеви извршавају и како се обрађују резултати, а *HTTPExecutor* представља конкретну имплементацију,

која преко *STTP* клијента комуницира са *Elasticsearch* сервером. Апстракција *Elasticsearch* (листинг 1) служи као главна улазна тачка за кориснике, а све функције унутар те апстракције су дефинисане помоћу *ZIO* ефеката.

```
trait Elasticsearch {
  def execute[A](request: ElasticRequest[A]): Task[A]

  def stream(request: SearchRequest): Stream[Throwable,
Item]

  def stream(request: SearchRequest, config: StreamConfig):
Stream[Throwable, Item]

  def streamAs[A: Schema](request: SearchRequest):
Stream[Throwable, A]

  def streamAs[A: Schema](request: SearchRequest, config:
StreamConfig): Stream[Throwable, A]
}
```

Листинг 1. *Elasticsearch* апстракција

Један од кључних концепата у библиотеци јесте моделовање упита и агрегација према *Elasticsearch* серверу. *ElasticQuery* објекат пружа јавни *API*, који садржи функције за креирање различитих врста упита, док *ElasticAggregation* објекат садржи јавни *API* са функцијама за креирање различитих врста агрегација. Због тога што се упити и агрегације дефинишу као класе унутар *Scala* кода, обезбеђује се типска сигурност, боља читљивост и једноставна композиција сложенијих упита и агрегација из једноставнијих. Свака инстанца упита или агрегације може се обогатити додатним, опционим параметрима ланчаним позивањем метода над том инстацом.
Уместо да се ослања на ”ниске типове”, за представљање имена индекса или вредности рутирања, библиотека *ZIO Elasticsearch* користи специјализоване типове. Поред тога што омогућавају бољу контролу, типску сигурност и већу безбедност, специјализовани типови уводе и додатне валидације за сваки тип које се извршавају већ приликом компилације кода. На листингу 2 налази се дефиниција типа *IndexName*, који представља име индекса у *Elasticsearch*-у, дефинисан је над типом *String*, уз придружене валидације које одговарају правилима *Elasticsearch*-а.

```
object IndexName extends NewtypeCustom[String] {
  protected def validate(name: String) =
    IndexNameValidator.validate(name)
}
type IndexName = IndexName.Type
```

Листинг 2. Дефиниција *IndexName* типа за верзију 3 програмског језика *Scala*

Поред класичног рада са индексима и документима, библиотека омогућава и извршавање више операција над документима у оквиру истог захтева, помоћу *bulk API*-ја *Elasticsearch*-а. Овај механизам значајно убрзава рад, јер се више операција групише и шаље истовремено. У *ZIO Elasticsearch* библиотеци, за коришћење овог механизма предвиђена је функција *bulk* (листниг 3), која прихвата произвољан број захтева типа *BukableRequest* (апстракција коју наслеђују само захтеви који смеју бити укључени у *bulk* операцију).

```
final def bulk(requests: BulkableRequest[_]*): BulkRequest =
  Bulk(requests = Chunk.fromIterable(requests), index =
None, refresh = None, routing = None)
```

Листинг 3. Функција за креирање *Bulk* захтева

Постоје три врсте захтева чијим извшавањем корисници могу да врше претрагу и агрегирање: *SearchRequest*, *AggregateRequest* и *SearchAndAggregateRequest*. Сваки од захтева омогућава пагинацију, сортирање по пољима или скриптованој логици, филтрирање, додавање нових агрегација и *highlighting* за истицање делова текста.
Библиотека *ZIO Elasticsearch* подржава стриминг докумената кроз *ZStream* [15], користећи један од два доступна механизма у *Elasticsearch*-у: “*search_after*” и “*scroll*”. Оба приступа омогућавају постепено преузимање резултата претраге у виду стрима, чиме је омогућено обрађивање података без претходног учитавања целог скупа у меморију.
Сви захтеви ка *Elasticsearch* серверу у оквиру библиотеке *ZIO Elasticsearch* извршавају се преко *HttpExecutor*-а. Он омогућава слање сваког од постојећих захтева и обраду одговора са сервера у типизованом облику. Свака функција из *HttpExecutor*-а формира *HTTP* захтев, поставља заглавља, шаље сам захтев (преко *HTTP Client*-а) и декодира одговор у очекивани тип помоћу *JSON* декодера. У случају грешака, *HTTP* одговори се мапирају у доменске изузетке, омогућавајући јасну обраду грешака.

4.ДЕМОНСТРАЦИЈА КОРИШЋЕЊА БИБЛИОТЕКЕ

Да би се приказало коришћење библиотеке *ZIO Elasticsearch* у пракси, развијена је једноставна *backend* апликација која симулира рад са *GitHub* репозиторијумима. Апликација омогућава чување информација о репозиторијумима (назив, организација, број звездица…), претрагу по различитим критеријумима, као и креирање, измену и брисање докумената у *Elasticsearch* индексу. Користи се *REST* интерфејс написан у *Scala* програмском језику, док се библиотека *ZIO Elasticsearch* користи за комуникацију са *Elasticsearch* кластером.
За покретање апликације, довољно је прво покренути локални *Elasticsearch* сервер и након тога позвати команду: *./sbt “~example/reStart”*, која ће заправо и подићи саму апликацију.

```
def createAll(repositories: Chunk[GitHubRepo]): Task[Unit]
=
  for {
    routing <- routingOf(organization)
      <- elasticsearch.execute(
         ElasticRequest
           .bulk(repositories.map { repository =>
             ElasticRequest.create[GitHubRepo](Index,
DocumentId(repository.id), repository)
           }: _*).routing(routing)
       )
  } yield ()
```

Листинг 4. Попуњавање *Elasticsearch* индекса

Приликом покретања апликације, прво се, уколико постоји, аутоматски обрише постојећи *Elasticsearch* индекс, након чега се индекс изнова креира и попуњава подацима коришћењем *bulk* операције са угњежденим *create* операцијама (листинг 4). Основни слој за рад са базом представља класа *RepositoriesElasticsearch*. Она користи *ZIO Elasticsearch* библиотеку за креирање и извршавање различитих типова захтева. Овај слој, уз помоћ библиотеке, омогућава креирање појединачних докумената, креирање више докумената одједном, ажурирање докумената, брисање по идентификатору, добављање свих докумената, добављање појединачних докумената и претрагу са пагинацијом.
Претрага је реализована кроз функцију *search* (листинг 4), којој се сем жељеног упита, прослеђују и параметри *from* і *size*, који се користе за пагинацију. Функција извршава захтев *Search* добијен *search* функцијом из библиотеке, односно из *ElasticRequest* класе. Као и код добављања докумената, над повратном вредности извршавања захтева, позива се метода ”*.documentAs[GitHubRepo]*” из библиотеке, која омогућава да документи који се добављају буду очекиваног типа, а не генерички.

```
def search(query: ElasticQuery[_], from: Int, size: Int):
Task[Chunk[GitHubRepo]] =
  elasticsearch.execute(ElasticRequest.search(Index,
query).from(from).size(size)).documentAs[GitHubRepo]
```

Листинг 4.2: Функција *search*

Ова демонстративна апликација показује како се *ZIO Elasticsearch* може употребити као потпуно типски безбедан и реактиван слој за рад са *Elasticsearch*-ом, без директног писања *Elasticsearch DSL*-а.
Апликација је свесно поједностављена, али и као таква добро илуструје пуни животни циклус рада са *Elasticsearch*-ом: од креирања индекса, преко уписивања и ажурирања података, до нарпедних претрага са разним критеријумима. Оваква архитекрура се може директно применити и на реалне пројекте, било да је реч о претраживању садржаја или аналитичким апликацијама.

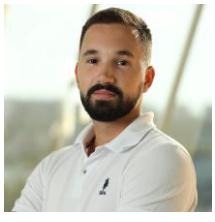
5.ЗАКЉУЧАК

У овом раду приказана је израда библиотеке *ZIO Elasticsearch*, чији је циљ поједностављено и функционално интегрисање *Elasticsearch* претраживачког система са *ZIO* екосистемом. Полазећи од потребе да се корисницима омогући елегантан и типски безбедан начин рада са *Elasticsearch*-ом, развијено је решење које комбинује снагу *ZIO* ефектног модела са богатим могућностима претраге, агрегирања и стримовања података.
Током рада најпре су размотрени кључни концепти и технологије које чине основу библиотеке, након чега је уследила детаљна анализа имплементације. Практична демонстрација библиотеке потврдила је да се *ZIO Elasticsearch* може ефикасно користити у реалним апликацијама, уз очувану читљивост и предвидивост кода. Због функционалног приступа и подршке за *ZIO* екосистем, библиотека олакшава обраду великих количина података, омогућава једноставно управљање грешкама и обезбеђује бољу тестабилност у поређењу са класичним приступима.
Иако је библиотека већ у употребљивом стању, постоји простор за даље унапређење. Будући развој може обухватити проширивање подршке за будуће верзије *Elasticsearch*-а, додавање још напреднијих типова упита и агрегација, као и израду детаљније документације и туторијала који ће допринети лакшем усвајању од стране шире заједнице.
Закључно, развој библиотеке *ZIO Elasticsearch* представља значајан корак ка интеграцији савремених претраживачких система са функционалним програмирањем у *Scala* програмском језику. Овај рад не само да демонстрира техничке могућности и добијене резултате, већ поставља и темеље за будућа истраживања и практичну примену у домену обраде и анализе великих података.

6.ЛИТЕРАТУРА

[1] https://www.elastic.co/ (Последњи приступ Октобар 2025)
[2] https://lucene.apache.org/ (Последњи приступ Октобар 2025)
[3] Paul Chiusano, Runar Bjarnason: “Functional Programming in Scala”
[4] Martin Odersky: “Programming in Scala”
[5] John Hughes: “Why Functional Programming Matters”
[6] https://zio.dev/ (Последњи приступ Октобар 2025)
[7] John A. De Goes, Adam Fraser, Milad Khajavi: “ZIONOMICON”
[8]https://codingexplained.com/coding/elasticsearch/understanding-the-inverted-index-in-elasticsearch/ (Последњи приступ Октобар 2025)
[9] https://www.apache.org/ (Последњи приступ Октобар 2025)
[10]https://www.elastic.co/docs/reference/elasticsearch/rest-apis/paginate-search-results#scroll-search-context/ (Последњи приступ Октобар 2025)
[11]https://www.elastic.co/docs/reference/elasticsearch/rest-apis/paginate-search-results#search-after/ (Последњи приступ Октобар 2025)
[12] https://www.geeksforgeeks.org/software-engineering/separation-of-concerns-soc/ (Последњи приступ Октобар 2025)
[13] https://zio.dev/reference/di/ (Последњи приступ Октобар 2025)
[14] https://zio.dev/reference/contextual/zlayer/ (Последњи приступ Октобар 2025)
[15] https://zio.dev/reference/stream/zstream/ (Последњи приступ Октобар 2025)

Кратка биографија:



Димитрије Булаја рођен је у Зрењанину 1998. године. Дипломирао је на Факултету техничких наука 2021. године и тада уписује мастер академске студије на истом факултету.
Контакт: dbulaja98@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.4
ДОИ: https://doi.org/10.24867/35BE17Glisic

Миграција наспрам развоја *Cloud-Native* апликација

Migration versus Cloud-Native application development

Цвијетин Глишић, *Факултет техничких наука, Нови Сад*

Студијски програм – ПРИМЕЊЕНО СОФТВЕРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *У овом раду обрађене су основне карактеристике cloud computing-*а*, као и начини и приступи миграције и развоја апликација у cloud окружењу. Посебан фокус рада је на два приступа cloud-у: „replatform" миграцији, која подразумева подизање постојеће (legacy) апликације у cloud уз минималне измене ради искоришћења одређених погодности cloud платформе и cloud-native приступу, који обухвата развој нове апликације посебно намењене за рад у cloud окружењу и коришћење свих предности које cloud омогућава.*

Кључне речи: *cloud, cloud-computing, миграција, cloud native, replatform*

Abstract – *This paper explores the fundamental characteristics of cloud computing, as well as the various approaches to migrating and developing applications in cloud environments. The primary focus is on two key cloud strategies: the "replatform" migration, which involves deploying a legacy application to the cloud with minimal modifications to leverage selected cloud benefits, and the cloud-native approach, which entails building a new application specifically designed to operate in the cloud and to fully utilize all advantages offered by cloud platforms.*

Keywords: *cloud, cloud-computing, migration, cloud native, replatform*

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

1. УВОД

У савременом добу, *cloud-computing* је постао битан за развој и имплементацију апликација, омогућавајући организацијама већу флексибилност, скалабилност и ефикасност. Овај рад истражује два приступа у усвајању *cloud* технологија: миграцију постојећих апликација на *cloud*, без измене архитектуре и логике саме апликације познату као „*replatform*", и развој *cloud-native* апликација, дизајнираних да у потпуности искористе предности cloud окружења. Док „*lift and shift*" приступ омогућава брзу транзицију уз минималне измене постојеће инфраструктуре, *cloud-native* развој захтева већа почетна улагања, али нуди супериорну еластичност, отпорност и ефикасност у дугорочној перспективи [1].

Кроз упоредну анализу ових приступа, рад има за циљ да прикаже њихове предности, изазове и применљивост у различитим контекстима, пружајући смернице за доношење што бољих одлука у процесу преласка на *cloud*. За конкретан експеримент имплементирана је *Talent Acquisition Manager* апликација. Ова апликација, због своје природе и захтева за скалабилношћу, представља релевантан случај за анализу перформанси и прилагодљивости у *cloud* окружењу, и на њеном примеру су извучени закључци у контексту теме овог рада.

2. *CLOUD-NATIVE* ПРИСТУП

Cloud-native представља архитектурални стил развоја софтвера, где су апликације од темеља пројектоване и оптимизоване за рад у *cloud* окружењу. Овај приступ карактерише коришћење технологија које омогућавају максимално искоришћавање предности *cloud* платформи, попут скалабилности, еластичности и отпорности на грешке. *Cloud-native* апликације се обично заснивају на микро-сервисној архитектури, где је апликација подељена на мање, независне компоненте које међусобно комуницирају путем дефинисаних интерфејса. Оваква структура омогућава брзо прилагођавање променама, коришћење различитих, међусобно некомпатибилних технологија у истом систему, лакше скалирање појединачних делова апликације и ефикасније коришћење ресурса.

Кључне карактеристике cloud-native приступа укључују коришћење контејнера (нпр. *Docker*), оркестрацију контејнера (нпр. *Kubernetes*), аутоматизовано управљање инфраструктуром путем алата попут *CI/CD* (*Continuous Integration/Continuous Deployment*). Ове технологије омогућавају апликацијама да буду високо доступне, отпорне на кварове и способне да се аутоматски прилагођавају променама у оптерећењу. *Cloud-native* апликација може аутоматски скалирати своје ресурсе у зависности од броја корисника, смањујући трошкове током периода мање активности.

Међутим, *cloud-native* развој доноси и бројне изазове. Захтева велика почетна улагања у погледу времена и финансија. Поред тога, прелазак на микро-сервисну архитектуру може донети и додатну сложеност у управљању апликацијама, као на пример потребу за напредним мониторингом и координацијом између

сервиса. Упркос томе, *cloud-native* приступ је веома чест код нових апликација које захтевају висок степен скалабилности и флексибилности јер омогућава организацијама да у потпуности искористе сав потенцијал *cloud-computing*-а [2].

3. МИГРАЦИЈЕ НА *CLOUD*

Миграција постојећих апликација на *cloud* представља процес премештања постојећих софтверских решења са локалне инфраструктуре на *cloud* платформу уз минималне или никакве измене у њиховој основној архитектури. Овај приступ је посебно привлачан за организације које желе брзо искористити предности *cloud*-а, попут смањења трошкова за физичку инфраструктуру и побољшања доступности, без потребе за значајним преобликовањем апликација. Седам највише примењиваних и најпознатијих типова миграције су:

1. *Rehost (Lift and Shift)* – Померање апликације на *cloud* по „*as is*“ принципу, без икаквих архитектуралних промена и додатних компоненти.
2. *Relocate* – Погодно за апликације које су већ у виртуелним окружењима, попут *docker* контејнера или *kubernetes* кластера. Принцип је исти као код *rehost* приступа, али се мигрира цело окружење.
3. *Replatform* – Апликације се делимично адаптирају *cloud* окружењу, али без промене логике и архитектуре саме апликације. Ово може да подразумева прелазак на *cloud* базу података, имплементацију *HPA* (*horizontal pod autoscaling*) на нивоу целе апликације, имплементацију сервиса за праћење метрика, итд.
4. *Refactor/rearchitect* – Подразумева редизајнирање апликације, уноси видљиве промене у архитектури а често и у логици апликације. Ово је тип миграције који организације користе како би постојећу апликацију што више приближиле cloud-native стилу.
5. *Repurchase (Drop and Shop)* – Представља замену постојеће (*legacy*) апликације неким *SaaS* (*Software-as-a-Service*) решењима. Овај приступ је поготову изводљив код апликација које обрађују неке стандардизоване процесе.
6. *Retire* – Одбацивање застарелих апликација или делова апликација, ради усмеравања ресурса на битније процесе и делове система.
7. *Retain/revisit* – Задржавање одређених апликација или делова система у локалном (*on-premise*) окружењу, обично због безбедносних, регулаторних или техничких ограничења, уз план да се њихова миграција поново размотри у будућности [3][4].

4. *TALENT ACQUISITION MANAGER* АПЛИКАЦИЈА

4.1. Идеја и инспирација

С обзиром да у последње време све више људи проналази запослење путем интернета, те чињенице да се појављује све више апликација које помажу људима да пронађу посао на овај начин и повежу се с људима из исте или сличне струке, створена је идеја да софтверско решење уз овај рад буде апликација тог типа.

4.2. Коришћене технологије

У реализацији софтверског решења за овај рад коришћене су технологије и алати који су често заступљени у раду са cloud окружењем:

- *ASP.NET Core* – радно окружење отвореног типа развијено од стране компаније *Microsoft*, за развој *web* и *cloud* апликација.
- *gCloud* – cloud платформа развијена од стране компаније *Google*. Укључује рачунарске ресурсе, складиштење података, аналитiku и машинско учење. Ова платформа омогућава организацијама сигурно, скалабилно и високо перформантно окружење за покретање апликација, управљање радним оптерећењима и обраду великих количина података.
- *Insomnia – REST API* клијент, за организовање, складиштење и извршавање *API* захтева.
- *RabbitMQ* – софтвер за асинхрону комуникацију између сервиса. Има улогу средњег слоја приликом слања порука са једног сервиса на други.
- *gRPC* – технологија високе поузданости и перформанси за синхрону комуникацију између сервиса. Користи *RPC* протокол за размену порука.
- *JSON Web Token* – отворени стандард који дефинише правила преноса информација између страна у облику *JSON* објекта.
- *Docker* – отворена платформа за развој, слање и покретање апликација. *Docker* омогућава одвајање софтвера од инфраструктуре у којој је направљен, што омогућава слање и покретање у другим инфраструктурама и олакшава пуштање софтвера у продукцију.
- *Kubernetes* – систем отвореног типа за аутоматизацију процеса менаџмента, скалирања и пуштања у продукцију контејнеризованих апликација. Групише контејнере који заједно чине логичку јединицу, систем или апликацију, што је од посебног значаја у контексту микро-сервиса.
- *Horizontal Pod Autoscaler (HPA)* – компонента *Kubernetes* платформе која омогућава динамичко прилагођавање броја инстанци (*pod*-ова) на основу метрика оптерећења.
- *Prometheus* – отворени систем за прикупљање, чување и обраду метрика из различитих

компоненти дистрибуираних система. Развијен је првенствено за потребе мониторинга cloud и *container-based* апликација, а данас представља стандардно решење за надзор *Kubernetes* окружења.

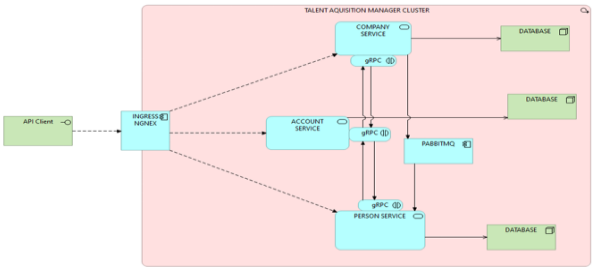
- *Grafana* – отворена платформа намењена за визуализацију, анализу и праћење метрика и података из различитих извора. Најчешће се користи у комбинацији са *Prometheus*-ом. Главна улога *Grafana*-е је да прикупљене метрике и логове прикаже у виду интерактивних графова, табела и *dashboard*-а, чиме се олакшава праћење перформанси система и правовремено уочавање проблема.
- *Google Cloud SQL* – managed релациона база података у *cloud*-у.
- *K6* – отворени алат за *load* и *performance testing*.

4.3. Архитектура апликације

За потребе рада, креиране су две верзије апликације, логички потпуно исте и пружају потпуно исте могућности. Једна верзија је микро-сервисна, која је помогла да се на практичном примеру испита *cloud-native* приступ *cloud* окружењу, а друга монолитна, која је помогла да се на практичном примеру истражи „*replatform*“ приступ миграцији постојеће апликације.

4.4. Микро-сервисна архитектура апликације

Архитектура се састоји од три микро-сервиса: *AccountService*, *CompanyService* и *PersonService*. Поред ових компоненти, саставни део апликације чини и *Ingress NGINX* контролер, који је задужен за балансирање оптерећења и даље рутирање *API* захтева према сервисима.
Сваки од сервиса има своју *Google Cloud SQL* базу података која трајно складишти податке потребне за рад сервиса. Поред горе наведених компоненти, саставни део апликације чини и *RabbitMQ* сервис за асинхрони пренос порука. Сваки од сервиса користи *gRPC* технологију за синхрону комуникацију са другим сервисима (*слика 1*).



Слика 1. Приказ архитектуре микро-сервисне апликације

AccountService је микро-сервис који обавља сву пословну логику везану за корисничке налоге корисника ове апликације.
CompanyService и *PersonService* представљају два одвојена микро-сервиса унутар нашег система. Ови сервиси су специјализовани за ефикасно управљање профилима корисника на нашој апликацији.

4.5. Монолитна архитектура апликације

Ова верзија апликације је настала спајањем сва три сервиса микро-сервисне верзије у један заједнички монолитни сервис, који омогућава све логичке функционалности микро-сервисне апликације. Оваквим приступом из архитектуре се избацује доста компоненти микро-сервисне верзије.
Три базе података су замењене једном, која треба да чува све податке. *Ingress* контролер систему више није потребан јер сав саобраћај сада улази у једну тачку, тј. заједнички сервис. За комуникацију између различитих логичких целина користе се апликацијски интерфејси, што уклања потребу за *gRPC* синхроном и *RabbitMQ* асинхроном комуникацијом.
Монолитна архитектура одбацује *cloud-native* приступ и представља добар пример *legacy* апликације, што омогућава симулацију миграције на *cloud*, а посебно „*Replatform“* приступ.

5. ПОСТАВЉАЊЕ АПЛИКАЦИЈЕ У *CLOUD* ОКРУЖЕЊЕ

За потребе овог рада коришћен је *gCloud*, платформа компаније *Google*. Обе архитектуре (микро-сервисна и монолитна) постављене су у засебне кластере, како би испитивања оптерећења и потрошње ресурса на њима била релевантнија. Обе апликације користе *e2-medium node pool*, где свака инстанца унутар *pool*-а пружа 2 виртуелна процесорска језгра и 4GB RAM меморије.
Код микро-сервисне архитектуре, pod сваког сервиса је иницијално покренут у једној инстанци, а максимална вредност хоризонталног скалирања код сваког сервиса је постављена на три. Подови се скалирају када оптерећење ресурса којима располаже *pod* (CPU језгара и RAM меморије) достигне 80%. Унутар кластера је постављен *RabbitMQ message broker*, већ поменути *HPA* (*horizontal pod autoscaling*), као и алати за праћење и визуализацију метрика, *Prometheus* и *Grafana*. За складиштење података користи се *Google Cloud SQL*, који за потребе овог система садржи три базе података.
Код монолитне архитектуре, апликација је иницијално покренута на једној инстанци, а максимална вредност хоризонталног скалирања је постављена на три. Као и код под-ова у оквиру кластера микро-сервисне апликације, под-ови се скалирају ако оптерећење ресурса којима располаже pod достигне 80%. За разлику од кластера микро-сервисне архитектуре, кластер монолитне апликације не садржи *RabbitMQ* инстанцу, и систем користи само једну базу података.
Кластер микро-сервисне апликације користи видно више ресурса када је систем у стању без оптерећења или стању лаког оптерећења. Тада микро-сервисни кластер користи четири инстанце *e2-medium node*-а, док монолитни кластер користи три. Треба узети у обзир да на оба кластера по један *node* нема скоро никакво оптерећење, тако да је реална искоришћеност два *node*-а за монолитну, а три *node*-а за микро-сервисну архитектуру. Овој, на изглед превеликој оптерећености ресурса, доста доприносе и инстанце које раде мониторинг постојећег система и праве

метрике, логове или воде рачуна о хоризонталном скалирању система.

6. ОПТЕРЕЋЕЊЕ СИСТЕМА И РЕЗУЛТАТИ ТЕСТИРАЊА

За потребе тестирања коришћен је K6 софтверски алат, за load и performance тестирање web апликација. За потребе овог рада изведени су load (реално оптерећење) и stress (високо оптерећење) тестови. У овим тестовима, сваки виртуелни корисник прати кораке редоследом: креирање налога, креирање профила на налогу, измена профила, брисање профила, брисање налога.

6.1. *Load* тестови система

Оба система су тестирана за случај нормалног оптерећења. За ово тестирање коришћено је 40 виртуелних корисника који су у исто време, у трајању од 15 минута, користили апликацију, извршавајући горе наведене кораке. Добијени су резултати видљиви на слици 2.

Metric	Monolithic	Microservices
Total requests	147,833	108,801
Average request duration	93.47 ms	188.28 ms
Median (p50)	63.48 ms	69.1 ms
p90 latency	175.42 ms	301.57 ms
p95 latency	222.54 ms	526.92 ms
Max response time	30.1 s	21.9 s
Error rate	0.13%	0.00%
Average iteration duration	1.7 s	2.31 s

Слика 2. Резултати *load* тестирања система

6.2. *Stress* тестови система

Оба система су такође тестирана и за случај неочекиваног преоптерећења. За ово тестирање коришћено је до 200 виртуелних корисника који су у исто време, 16 минута, користили апликацију, извршавајући горе наведене кораке. Број корисника је временом постепено повећаван са 40 на 200.

Metric	Monolithic	Microservices
Total requests	207,094	102,135
Average request du.	398.7 ms	991.88 ms
Median (p50)	178.89 ms	77 ms
p90 latency	860.26 ms	495.32 ms
p95 latency	1.04 s	2.97 s
Max response time	2 min 40s	46.68 s
Error rate	0.07%	0.008%
Average iteration dura.	3.89 s	7.94 s

Слика 3. Резултати *stress* тестирања система

Добијени су резултати видљиви на слици 3.

7. ЗАКЉУЧАК

На конкретном примеру мале апликације, на основу резултата тестова, упоређене су перформансе при реалном и високом оптерећењу. Резултати показују да „*replatform*“ миграција, симулирана кроз монолитну архитектуру са једним сервисом, једном базом података и *load balancer*-ом, пружа бољу ефикасност у контексту мале до средње апликације. Монолитна верзија обрадила је значајно више захтева са нижом просечном латенцијом и краћим временом итерације виртуелних корисника, што указује на мањи *overhead* код монолитне верзије.

С друге стране, *cloud-native* приступ, имплементиран кроз микро-сервисну архитектуру, са више сервиса, *gRPC* комуникацијом и *RabbitMQ*-ом, показао је бољу отпорност на грешке и бољу медијану латенције. Међутим, већа просечна латенција и варијабилност потврђују познати *overhead* микро-сервиса, посебно у малим системима где мрежна комуникација и координација превазилазе потребе.

Cloud-native приступ, иако супериоран у скалабилности и агилности за велике, дистрибуиране системе, показује мање предности у овом сценарију, посебно уз ограничене ресурсе и не тако логички и функционално сложеним системом.

8. ЛИТЕРАТУРА

[1] Fahmideh, Low, Beydoun & Daneshgar , “Cloud Migration Process: A Survey Evaluation Framework and Open Challenges“, 2016.

[2] https://aws.amazon.com/what-is/cloud-native/ (pristupljeno u oktobru 2025.)

[3] www.netapp.com/blog/aws-cvo-blg-strategies-for-aws-migration-the-new-7th-r-explained/ (pristupljeno u oktobru 2025.)

[4] www.techtarget.com/searchcloudcomputing/tip/The-Rs-of-cloud-migration-How-to-choose-the-right-method (pristupljeno u oktobru 2025.)

Кратка биографија:

Цвијетин Глишић рођен је 2000. године у Бијељини. Дипломирао је 2023. године, на смеру Примењено софтверско инжењерство.

Контакт: cvijetinglisic@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.7
ДОИ: https://doi.org/10.24867/35BE18Vukovic

Једно решење IoT система за испитивање утицаја вештачког осветљења на човекову продуктивност

IoT System for Investigating the Impact of Artificial Lighting on Human Productivity

Павле Вуковић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСКА ТЕХНИКА И РАЧУНАРСКЕ КОМУНИКАЦИЈЕ

Кратак садржај – *Циљ рада је дизајн и реализација IoT система за контролу осветљења у затвореном простору који отклања ограничења решења коришћених у научним истраживањима. На основу прегледа литературе потврђен је утицај светлости на продуктивност и здравље и идентификовани су недостаци постојећих система. Као основа система је изабран софтвер паметне куће. Он је потом проширен имплементацијом дефинисане паметне логике. Систем је пројектован, реализован и тестиран према унапред задатим критеријумима, при чему резултати показују да их у потпуности испуњава и да успешно превазилази уочене мане. Главно ограничење је скалабилност: тренутна верзија је погодна за мање просторе, док је за примену у великим постројењима потребно даље скалирање.*

Кључне речи: *IoT, вештачко осветљење, продуктивност, циркадијални ритам, Home Assistant*

Abstract – *The aim of this thesis is the design and implementation of an IoT system for indoor lighting control that overcomes the limitations of solutions currently used in scientific research. Based on a literature review, the impact of light on human productivity and health is confirmed and shortcomings of existing systems are identified. The system is built on a smart home software platform and extended through the implementation of defined smart logic. The system was designed, implemented and tested against predefined criteria and the results show that it fully meets these criteria and successfully addresses the shortcomings of existing systems. The main limitation of the current solution is scalability: current solution is appropriate for small indoor spaces. It requires scaling to be applicable in industrial environments.*

Keywords: *IoT, artificial lighting, productivity, circadian rhythm, Home Assistant*

НАПОМЕНА: Овај рад проистекао је из мастер рада чија менторка је била др Јелена Ковачевић, ванр. проф.

1. УВОД

Многа научна истраживања су показала несумњиви утицај осветљења на продуктивност и здравље човека. Анализом такве научне литературе уочене су значајне мане система осветљења коришћених у експериментима. Ти системи најчешће захтевају сложену инсталацију и употребу, не омогућавају брзу и лаку промену својстава параметара осветљења и имају ограничен или непостојећи степен аутоматизације. Поред тога, инсталације су углавном фиксне и не пружају флексибилност у погледу промене распореда извора светлости у просторији [2–9]. Мотив овог рада је дизајн и реализација бесплатног IoT система за контролу осветљења у затвореном простору који елиминише наведена ограничења. Предложени систем није намењен искључиво научним истраживањима, већ има примену и у домаћинствима, као и у пословним и индустријским окружењима.

У центру предложеног система се налази већ готов софтвер паметне куће. Будући да на тржишту већ постоји велики број решења који омогућавају услуге паметне куће, имплементација оваквог софтвера није оправдана. Након анализе могућности које нуде водећа решења, као основа система је изабран Home Assistant. Почетни софтвер се проширује имплементацијом паметне логике која дефинише аутоматизације осветљења. Логика је дефинисана у складу са научном литературом. Дефинисани су и критеријуми које систем мора да испуни, након чега је реализовани систем тестиран како би се утврдило да ли и у којој мери задовољава постављене критеријуме.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1. IoT системи

IoT обухвата међусобно повезане уређаје који аутономно размењују податке и управљају системом. Најчешће се разликују системи паметних кућа и индустријски IoT. Типичан IoT систем чине централни уређај, сензори, актуатори, корисничка спрега и, по потреби, облак за сложенију обраду података.

Паметне куће

Системи паметних кућа примењују се у стамбеним просторима и заснивају се углавном на „лаким“ бежичним комуникационим протоколима и технологијама (MQTT, Zigbee, Z-Wave, BLE, Wi-Fi, Matter). Они поједностављују интеграцију уређаја уз релативно ниску цену и довољну поузданост за сценарије где временски одзив није критичан. На

тржишту постоји више платформи које пружају услуге софтвера паметне куће.

Индустријски IoT

Индустријски IoT се примењује у индустријским и пословним окружењима где су поузданост, сигурност и рад у реалном времену од кључног значаја. Због тога се најчешће користе комуникациони протоколи засновани на Ethernet протоколу. Системи су скупљи, ажурирања ретка, а свака инсталација је прилагођена конкретној примени, тј. нема универзалних готових решења.

2.2. Избор софтвера

У оквиру рада се не развија потпуно ново решење, већ се врши избор и надограђује већ постојећи софтвер паметне куће. На тржишту постоји неколицина таквих софтвера: Home Assistant, SmartThings, Amazon Alexa, Google Home, Apple HomeKit и на десетине других. Изабрани софтвер мора да буде: бесплатан, подржава велики број модерних комуникационих протокола и уређаја, нуди механизме аутоматизације, има активну заједницу и често се ажурира. Једини који је испунио све критеријуме је Home Assistant. Поред наведених захтева, он нуди и неке додатне алате о којима ће бити речи у даљем току рада.

2.3. Параметри светлости

Фотометрија проучава светлост у односу на људско око и дефинише, између осталог, следеће физичке величине везане за интензитет светлости: количину светлости, светлосни флукс (лумен), јачину светлости (кандела), луминацију и осветљеност (лукс). За боју светлости, то су: температура боје (CCT), у Келвинима, која описује да ли је светло „топло“ или „хладно“ и генерални индекс рендеровања боје (CRI), који указује на веродостојност приказа боја [8]. Видљиви спектар је део електромагнетног спектра који је видљив голим оком. Типично људско око реагује на таласне дужине од 380 до 740 нм.

2.4. Циркадијани ритам

Циркадијални ритмови регулишу циклусе будности и сна, метаболизам и друге физиолошке процесе. Главни „сат“, који координише све ове процесе се налази у супракијазматичном нуклеусу (SCN) у хипоталамусу, а информације о светлу добија путем специјализованих ћелија у ретини ока (ipRGC), осетљивих на светлост. Светлост преко ipRGC утиче на лучење кортизола и мелатонина: кортизол је повишен ујутру и подстиче будност, док мелатонин расте увече и припрема организам за сан. Одређени светлосни услови могу померити фазу циркадијалног ритма унапред или уназад. [1][9][10].

3. ПРЕГЛЕД НАУЧНЕ ЛИТАРАТУРЕ

Жеља је да се уоче карактеристике светлосних система коришћених у досадашњим истраживањима како би се уочиле њихове мане које ће потом бити отклоњене. Такође, сагледава се утицај светлости на продуктивност и здравље човека и идентификују захтеви за дизајн паметне логике. Претрага је извршена путем сервиса Google Scholar, коришћењем комбинација кључних речи light + productivity и light + circadian rhythm.

Анализа светлосних система коришћених у истраживањима показује да они имају бројна ограничења [2–9]:

- сложену поставку и употребу;
- потребу за техничком подршком;
- ограничене могућности брзе промене својстава осветљења;
- доминацију флуоресцентних лампи као извора светлости;
- употребу застарелих или тешко управљивих технологија;
- фиксне инсталације које је тешко премештати током студије.

Предложени систем је дизајниран тако да адресира све наведене недостатке.

Када је реч о утицају светла на продуктивност и здравље човека, изведени су следећи закључци [1-10]:

- светлост јаког интензитета побољшава продуктивност;
- аутономија корисника у контроли светла на радном месту је важна;
- на продуктивност утиче више међусобно повезаних фактора па је тешко изоловати појединачне утицаје;
- LED извори светла су пожељнији од флуоресцентних;
- важно је да светло испуњава очекивања корисника;
- ноћу се препоручују нижи нивои осветљености како би се ублажио негативан акутни утицај светла на сан;
- изложеност светлу може значајно да помери фазу циркадијалног ритма, унапред или уназад, а циркадијални поремећаји су повезани са више негативних последица по здравље, посебно код ноћних радника;
- квалитетан светлосни систем треба да обезбеди довољну осветљеност за визуелне задатке, добру униформност и балансирану дистрибуцију светла у простору, минималан одсјај, добар индекс рендеровања боје и одсуство треперења.

Ови закључци ће бити коришћени за дефинисање паметне логике чијом се имплементацијом демонстрира функционалности реализованог система.

4. КОНЦЕПТ СИСТЕМА

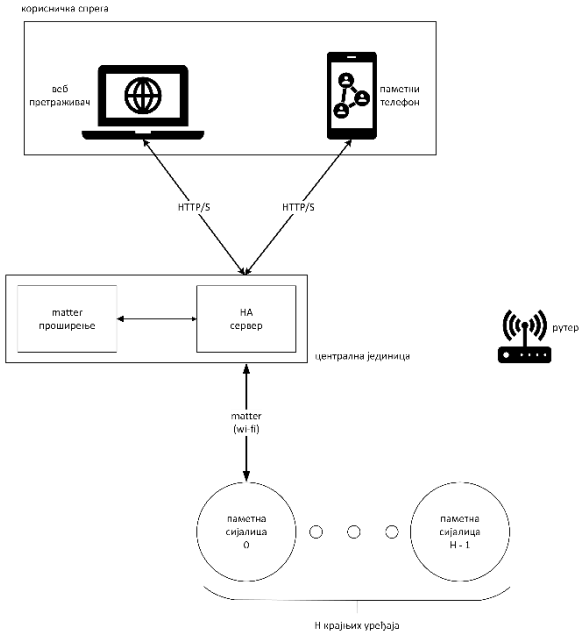
Слика 1 приказује концепт система. Систем чине:

- корисничка спрега (рачунар, паметни телефон и сл.) - спона корисника и система;
- централни уређај – повезује све компоненте и извршава логику система;
- крајњи уређаји (паметне сијалице, Moes MWB-TDA9) - сензорски и актуаторски уређаји;
- рутер - обезбеђује локалну мрежу и омогућава комникацију у систему.

4.1. Корисничка спрега

То је спона корисника и система. Може јој се приступити преко веб претраживача или кроз

наменску апликацију за паметни телефон. Кроз њу је могуће додавати уређаје, посматрати њихова стања, ручно управљати стањима уређаја и имплементирати аутоматизације. Паметни телефон има додатну улогу у иницијалном повезивању паметних сијалица у систем, путем наменске HA мобилне апликације.



Слика 1. Концепт система

4.2. Централни уређај

Централни уређај је Raspberry Pi 4 Model B са Ubuntu Linux оперативним системом. Home Assistant сервер се извршава у Docker окружењу и комуницира са корисничком спрегом преко HTTP(S) протокола. Са крајњим уређајима комуницира путем Matter протокола, док се подржавају и други уобичајени комуникациони протоколи, што омогућава једноставно проширење система. Подршку за Matter комуникациони протокол обезбеђује Matter проширење HA сервера, које се такође извршава у Docker окружењу.

4.3. Крајњи уређаји

Крајњи уређаји су паметне сијалице које истовремено имају улогу и сензора и актуатора. Могуће је читати и мењати њихова својства (интензитет, боја и температура светла), било ручно, преко интерфејса или аутоматски, на основу дефинисане паметне логике. Сијалице се у систем повезују уз помоћ паметног телефона који преко BLE везе преноси податке о Wi-Fi мрежи на коју је потребно да се повежу. Након иницијалног повезивања, даља комуникација се одвија између централног уређаја и сијалица путем Matter протокола. У систему су коришћене сијалице: Moes MWB-TDA9. У ову сврху је могуће користити и сијалице које користе друге популарне протоколе, нпр. Zigbee.

4.4. Критеријуми исправности система

Систем се сматра исправним ако су испуњени следећи критеријуми:

- робусност – систем мора стабилно да ради током дужег периода без прекида;
- преносивост и лакоћа подешавања – инсталација и конфигурација морају бити довољно једноставни да их могу обављати и корисници без техничке позадине (нпр. истраживачи).
- исправност крајњих уређаја – својства паметних сијалица морају се тачно читати и мењати, а временски одзив при промени стања треба да буде реда величине једне секунде како би се очувао добар кориснички доживљај.
- исправност паметне логике – све дефинисане аутоматизације морају се извршавати у складу са спецификацијом и на очекиван начин.

Систем такође треба да реши све проблеме које имају тренутно коришћени системи, о којима је било речи у поглављу 1. Испуњеност ових критеријума проверава се у поглављу посвећеном тестирању система.

5. ПОСТАВКА СИСТЕМА

Иницијална поставка обухвата следеће кораке:

- инсталацију Ubuntu ОС на Raspberry Pi и његово повезивање на локалну мрежу;
- омогућавање даљинског приступа уређају путем SSH-а, како би се даљи приступ обављао са персоналног рачунара;
- доделу статичке IP адресе Raspberry Pi уређају, ради поузданог приступа;
- инсталацију Docker и Docker Compose алата;
- покретање Docker контејнера: Home Assistant сервера и Matter интеграције као проширења за подршку Matter протоколу;
- приступ HA веб интерфејсу и пролазак кроз кратку иницијалну конфигурацију.

6. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

Дефинисана су два режима рада у којима је могуће манипулисати осветљењем:

- кориснички режим – мануелна контрола својстава сијалица;
- аутоматски режим – аутоматска контрола на основу паметне логике.

6.1. Кориснички режим

Када су паметне сијалице интегрисане у HA окружење, корисник добија могућност директне контроле њихових својстава преко корисничке спреге. Из основног приказа могуће је укључивање и искључивање појединачних сијалица или група, док детаљнији приказ омогућава:

- подешавање параметара светлости;
- увид у мета-податке и историју промена.

Према [2], аутономија корисника у подешавању осветљења значајно утиче на задовољство и перформансе, па систем задржава пун мануелни режим као важну компоненту. HA интерфејс је високо конфигурабилан, те је могуће дефинисати своје верзије приказа постојећих стања паметних сијалица. У овом раду се користи подразумевани приказ који је био довољан за демонстрацију функционалности.

6.2. Аутоматски режим

У аутоматском режиму HA самостално управља паметним сијалицама на основу унапред дефинисане паметне логике. За то се могу користити HA алати:

- аутоматизације;
- сцене;
- скрипте.

Након евалуације алата, аутоматизације су се показале као најпотпунија опција. Свака аутоматизација има следеће елементе:

- окидач – догађај који покреће аутоматизацију;
- услов – логички услов који мора бити испуњен да би се акција извршила;
- акција – низ корака који мењају стање уређаја.

Свака аутоматизација је скрипт фајл у .yaml формату, написан у складу са HA документацијом. Аутоматизације је могуће креирати на неколико начина, а изабрано је да се то врши на следећа два начина.

- за лице без техничког знања - кроз кориснички спрегу HA окружења изабрати Подешавање → Аутоматизације и сцене → Креирај нову аутоматизацију. Корисник добија једноставан мени чијим праћењем може креирати једноставне аутоматизације. Овај приступ је веома једноставан, али нуди ограничене могућности аутоматизација.
- за лице са техничким знањем – Кроз кориснички спрегу HA окружења изабрати Подешавање → Аутоматизације и сцене → Креирај нову аутоматизацију → ... → Измени у YAML. Сада се добија могућност креирања аутоматизације у .yaml формату. Овај приступ нуди све функционалности које аутоматизације нуде. С друге стране, компликован је и захтева техничко знање.

За потребе рада, дефинисане су и имплементиране следеће паметне логике:

Аутоматизација буђења

Непосредно пре времена буђења (нпр. тридесет минута), интензитет светла у спаваћој соби се постепено повећава од минималне до жељене вредности, симулирајући излазак сунца. На тај начин се ублажавају негативни ефекти наглог буђења алармом и приближава природни начин буђења. Овим се комбинују добре карактеристике спавања у потпуно замраченој просторији [10] и „природног“ буђења, без аларма.

Аутоматизација рада

HA је повезан са Google календаром корисника. То је могуће коришћењем Google календар интеграције за HA. Када у календару почне догађај који означава време за рад (нпр. садржи кључну реч „work“ у наслову), светло у радном простору се поставља на висок интензитет погодан за повећање продуктивности [1][2]. По завршетку догађаја, светло се враћа у основно стање.

Аутоматизација припреме за сан

Неколико сати пре спавања, систем аутоматски смањује интензитет осветљења и подешава топлију температуру боје у животном простору, а у време одласка на спавање искључује светла. Овим се смањују акутни негативни утицаји јаког светла у сатима пре спавања [9][10].

Поред ових аутоматизација, корисник, нпр. истраживач може лако дефинисати своје аутоматизације, те не мора мануелно мењати стања извора светлости током свог експеримента, већ се то може вршити аутоматизовано.

6.3. Коришћење постојеће конфигурације

Како би систем био лакши за коришћење код корисника без техничке позадине (нпр. истраживача), могуће је коришћење већ припремљене конфигурације HA окружења. HA омогућава креирање резервне копије која садржи:

- дефинисане аутоматизације;
- упарене уређаје;
- интеграције (нпр. Matter, Google календар);
- подешене изгледе картица и др.

Ову архиву је могуће учитати при поставци новог система, чиме се добија већ конфигурисано окружење. Након тога је потребно само поново упарити физичке уређаје (нпр. паметне сијалице) и, по потреби, поново одобрити поједине интеграције. На тај начин се знатно скраћује време поставке и смањује потреба за техничком подршком.

7. ТЕСТИРАЊЕ СИСТЕМА

Следи низ тестова спроведених над системом ради провере испуњености критеријума дефинисаних у одељку 4.4.

Тест робусности система

Централни уређај био је под непрекидним напоном и у продукционом режиму рада током периода од више недеља, односно месеци. За то време систем није бележио прекиде у раду нити неочекивано понашање. Овим је систем прошао стрес-тест дуготрајног рада и потврдио довољан ниво робусности за планирану примену.

Тест преносивости и лакоће подешавања система

Преносивост система и лакоћа подешавања проверени су коришћењем механизма резервних копија. Полазећи од ручно конфигурисаног система, креирана је резервна копија која је затим учитана у новоинсталирано HA окружење. Након учитавања резервне копије, све конфигурације (аутоматизације, сцене, интеграције, изглед картица, историја стања и др.) биле су доступне. Једино је било неопходно да се физички уређаји (паметне сијалице) поново упаре са системом. Ово потврђује да је могуће припремити „шаблон“ система који знатно поједностављује поставку за кориснике без техничког знања (нпр. истраживаче).

Тест исправности рада крајњих уређаја

Исправност рада паметних сијалица тестирана је у два корака:

- кроз HA корисничку спрегу, визуелно се проверава да ли измена стања (осветљеност, боја, температура) даје очекивани ефекат.
- коришћењем HA REST API и HTTP клијента (Postman). Спроведен је низ тестова над

својствима: осветљеност, температура, боја и комбинована промена више својстава одједном.

За свако својство је извршен пар HTTP захтева од стране Postman клијента ка REST серверу: упис нове вредности својстава; читање стања својстава. Додатно је вршено мерење укупног времена потребног да се ове операције изврше. У ту сврху су коришћени Postman алати: колекције, скрипте и окружење (енгл. collections, scripts, environment). Процедура је поновљена 5 пута за свако својство.

Табела 1 показује резултате из којих се види да се својство осветљености мења очекивано (уз предвидљива ограничења), а уочени су тек спорадични случајеви кашњења REST одговора. Просечно измерено време одзива (упис + читање) било је 357,8 ms, што значи да је стварно време промене стања уређаја још краће (узети у обзир време које доприноси HTTP, као и да се у временски одзив рачуна и провера промене стања – други HTTP захтев). Резултат је значајно испод прихватљивог прага (~1s из поглавља 4.4) и не нарушава кориснички доживљај. Резултати су приказани за параметар јачине осветљености. На исти начин тестирана су и остала својства сијалица, са сличним резултатима, па се закључује да промена својстава паметних сијалица коришћењем HA окружења ради очекивано.

Табела 1. Резултати теста исправности рада крајњих уређаја за параметар осветљености

број теста	својство	жељена вредност	постављена вредност	дужина трајања теста, у ms	очекивано
1	осветљеност	1	1	348	да
2	осветљеност	255	255	350	да
3	осветљеност	300	255	364	да
4	осветљеност	0	null	373	да
5	осветљеност	100	null	није релевантно – тест неуспешан	не
6	осветљеност	100	100	354	да

Тест исправности рада паметне логике

Исправност рада паметне логике тестирана је подешавањем контролисаних сценарија у којима долази до активације аутоматизација од интереса и праћењем понашања система кроз HA алате: Преглед, Историју и Праћење. Преглед нуди приказ тренутног стања уређаја, Историја нуди приказ стања уређаја у прошлости а Праћење нуди детаљан приказ редоследа којим су се извршавале аутоматизације.

Аутоматизација буђења тестирана је скраћивањем трајања на 1 минут и постављањем броја корака на 30. Окидач је коригован како би се аутоматизација одмах извршила. Очекивано је да се на сваке 2 секунде (1мин / 30 корака) почевши од тренутка окидања аутоматизације, осветљеност повећава за по 3.3% (100% / 30 корака). Визуелни утисак, као и алати Преглед, Историја и Праћење су потврдили да се аутоматизација понаша очекивано.

Аутоматизација рада тестирана је креирањем посебног догађаја у Google календару (са кључном речи „work“ у наслову) и посматрањем реакције система. Догађај је креиран непосредно пре извршења теста, те се аутоматизација одмах извршава. Поново је визуелно и уз алате Преглед, Историја и Праћење потврђено да аутоматизација ради очекивано.

Аутоматизација припреме за сан тестирана је померањем времена окидача на термин близак тестном и смањењем размака између „пригушивања“ и гашења светла. Визуелно и уз алате Преглед, Историја и Праћење потврђено је да се параметри светла мењају онако како је дефинисано.

Закључак на основу извршених тестова

Тест робусности система је показао да је систем довољно стабилан да функционише током дужег временског периода. Тест преносивости и лакоће подешавања система је показао да систем решава проблеме компликоване поставке. Коришћење бежичних комуникационих протокола решава проблем промене конфигурације извора осветљења. Тест исправности рада крајњих уређаја је показао да систем решава проблем брзе и лаке промене својстава светлости. Тест аутоматизација доказује да је систем способан и за извршавање аутоматизованих радњи, те се тиме додатно олакшава истраживачима, који не морају мануелно манипулисати светлошћу. Избор извора осветљења је решиоману коришћења флуоресцентних уместо LED извора. Коришћене су најмодерније доступне технологије чиме се решава проблем коришћења застарелих технологија. На основу овога се може закључити да систем задовољава све задате критеријуме (поглавље 4.4) и решава све мане које су имали досадашњи системи (поглавље 1).

8. ЗАКЉУЧАК

Циљ рада је дизајн и реализација IoT система за контролу осветљења у затвореном простору са намером да превазиђе ограничења светлосних система који се тренутно користе у научним истраживањима. Прегледом релевантне научне литературе потврђен је утицај светлости на продуктивност и здравље човека, као и недостаци до сада коришћених светлосних система. Као основа система је изабран софтвер паметне куће који је проширен имплементацијом аутоматизација. Систем је дизајниран, реализован и тестиран у складу са претходно дефинисаним критеријумима. Резултати тестирања показују да систем у потпуности испуњава постављене критеријуме и успешно отклања мане постојећих решења. Систем има и своја ограничења. Највеће ограничење је скалабилности система. У тренутном облику, систем је погодан за примену у мањим затвореним просторима и потребно га је скалирати како би био користан и у великим индустријским постројењима. Даљи рад може бити усмерен управо на решавање овог проблема.

9. ЛИТЕРАТУРА

[1] Van Bommel, Ir WJM, Ir GJ Van Den Beld, and Ir MHF Van Ooyen. "Industrial lighting and productivity." Philips Lighting, The Netherlands 20 (2002): 8.

[2] Juslén, Henri. "Lighting and productivity in the industrial working place." Proceedings of Fifteenth international symposium. 2006.
[3] Juslen, Henri, Marius Wouters, and Ariadne Tenner. "The influence of controllable task-lighting on productivity: a field study in a factory." Applied Ergonomics 38.1 (2007): 39-44.
[4] Juslén, Henri. "Influence of the colour temperature of the preferred lighting level in an industrial work area devoid of daylight." Ingineria Iluminatului 8.18 (2006): 25-36.
[5] Juslén, H. T., M. C. H. M. Wouters, and A. D. Tenner. "Lighting level and productivity: a field study in the electronics industry." Ergonomics 50.4 (2007): 615-624.
[6] Juslén, Henri T., Jos Verbossen, and Marius CHM Wouters. "Appreciation of localised task lighting in shift work—a field study in the food industry." International Journal of Industrial Ergonomics 37.5 (2007): 433-443.
[7] Juslén, H. T., and E. Kremer. "Localised lighting for efficient use of energy and better performance—Field Study in the factory." CIE midterm meeting and international lighting Congress, Leon 2005, Proceedings. 2005.
[8] Kralikova, Ruzena, Miriama Piňosová, and Beata Hricová. "Lighting quality and its effects on productivity and human healts." Int. J. Interdiscip. Theory Pract 10 (2016): 8-12.
[9] Davis, Lourdes K., et al. "Health effects of disrupted circadian rhythms by artificial light at night." Policy Insights from the Behavioral and Brain Sciences 10.2 (2023): 229-236.
[10] Tähkämö, Leena, Timo Partonen, and Anu-Katriina Pesonen. "Systematic review of light exposure impact on human circadian rhythm." Chronobiology international 36.2 (2019): 151-170.

Кратка биографија:

Павле Вуковић рођен је у Зрењанину 2001. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Рачунарска техника и рачунарске комуникације - Софтвер за потрошачку електронику одбранио је 2025.год.
Контакт:
pavlevukovic68@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.41
ДОИ: https://doi.org/10.24867/35BE19Momcilovic

Интеграција мапа у Home Assistant и омогућавање тока података ка њима

Integration of Maps in Home Assistant and Enabling Data Flow to Them

Миладин Момчиловић, *Факултет техничких наука, Нови Сад*

Студијски програм – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *Рад описује пројектовање и имплементацију система за визуализацију локације и праћење кретања IoT ентитета у реалном времену унутар Home Assistant платформе. Фокус рада је на успостављању поузданог тока података од мобилних уређаја до Map Dashboard-а коришћењем MQTT протокола као комуникационог стандарда. Предложено рјешење интегрише OwnTracks апликацију као извор података и користи хибридну архитектуру са локалним и удаљеним брокером како би се омогућио сигуран пријем података изван локалне мреже, превазилазећи мрежна ограничења (NAT). Практична примјена система демонстрирана је кроз приказ позиција на мапи и имплементацију аутоматизација које реагују на улазак и излазак корисника из дефинисаних гео-зона. Посебан значај рјешења огледа се у очувању приватности корисника кроз локалну обраду и складиштење осјетљивих података о кретању.*

Кључне речи: *Home Assistant, MQTT, MQTT Bridge, IoT, OwnTracks*

Abstract – *This paper describes the design and implementation of a system for real-time location visualization and tracking of IoT entities within the Home Assistant platform. The focus of the paper is on establishing a reliable data flow from mobile devices to the Map Dashboard using the MQTT protocol as a communication standard. The proposed solution integrates the OwnTracks application as a data source and employs a hybrid architecture with both a local and a remote broker to enable secure data reception from outside the local network, effectively overcoming network limitations (NAT). The practical application of the system is demonstrated through the display of positions on a map and the implementation of automations triggered by users entering and exiting defined geo-zones. A key significance of the solution lies in preserving user privacy through the local processing and storage of sensitive movement data.*

Keywords: *Home Assistant, MQTT, MQTT Bridge, IoT, OwnTracks*

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

1. УВОД

Развој система заснованих на *Internet of Things* (IoT) концепту довео је до значајног повећања количине података који се прикупљају, обрађују и приказују у реалном времену [1, 2]. Поред класичних сензорских мјерења, све већу улогу имају и геопросторни подаци, који омогућавају праћење кретања уређаја и корисника, као и имплементацију просторних аутоматизација у паметним окружењима.

Home Assistant [3] представља једну од најраспрострањенијих *open-source* платформи за локалну аутоматизацију паметних система, са снажном подршком за интеграцију различитих IoT уређаја и сервиса. Иако платформа нуди уграђене механизме за визуализацију локацијских података, интеграција геопросторних информација из хетерогених извора, посебно у сценаријима који укључују уређаје ван локалне мреже, представља технички изазов.

Овај рад се бави пројектовањем и имплементацијом система који омогућава поуздан ток локацијских података ка *Home Assistant* платформи и њихову визуализацију на мапи. Предложено рјешење заснива се на употреби MQTT [4] протокола као комуникационог слоја и хибридне архитектуре која комбинује локални и удаљени MQTT брокер, повезане механизмом MQTT моста [5].

Фокус рада је на мапирању локацијских података на ентитете типа *device_tracker* и њиховом приказу путем *Map Dashboard* компоненти, уз примјену TLS енкрипције и локалне обраде података ради очувања безбједности и приватности корисника. Поред визуализације, систем омогућава и имплементацију просторних аутоматизација заснованих на уласку и изласку уређаја из дефинисаних зона.

2. АРХИТЕКТУРА СИСТЕМА

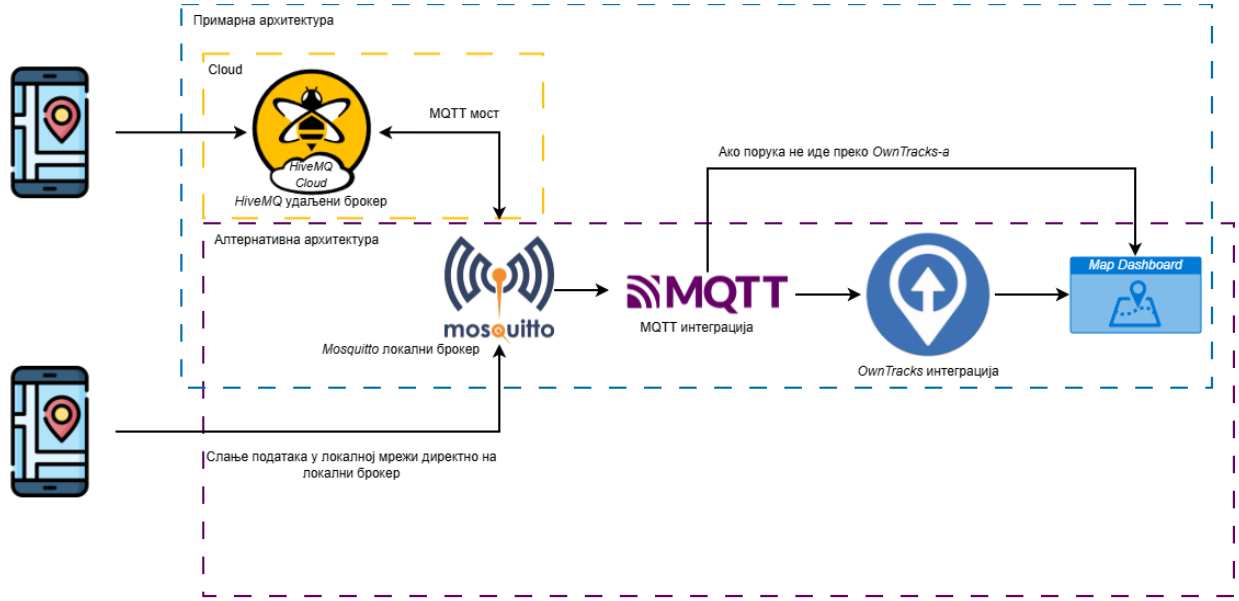
Предложени систем је пројектован као дистрибуирано рјешење које интегрише локалну обраду података са модерним комуникационим протоколима. Архитектура се састоји од три кључна сегмента: апликативног слоја (*Home Assistant*), комуникационог слоја (MQTT топологија) и слоја аквизиције података (клијентски уређаји). Интеракција између ових компоненти приказана је на Слици 1.

2.1. Home Assistant платформа

Home Assistant представља централну компоненту система и служи као основни механизам за прикупљање, обраду и визуализацију геопросторних података. У оквиру предложене хибридне архитектуре, он функционише као локална инстанца која осигурава да се логика аутоматизације извршава унутар приватне мреже корисника.
Пристигли подаци се унутар платформе мапирају на ентитете типа *device_tracker*. Ови ентитети представљају дигиталну апстракцију физичких уређаја и садрже кључне атрибуте попут географских координата, прецизности сигнала и статуса присутности. За интеракцију са корисником користи се *Map Dashboard* компонента, док интерни механизми платформе омогућавају креирање напредних аутоматизација које се окидају на основу просторних догађаја (нпр. улазак или излазак из дефинисане зоне).

2.3. Извори локацијских података

Слој аквизиције података чине мобилни уређаји са инсталираном апликацијом *OwnTracks* [8]. Ова апликација користи интегрисане GPS сензоре за генерисање телеметријских података, које потом шаљу у формату стандардизованих MQTT порука.
Структура ових порука (садрже координате, прецизност, ниво батерије) је оптимизована за аутоматску обраду унутар *Home Assistant* платформе. Овакав приступ омогућава флексибилну интеграцију различитих клијената, уз стриктно поштовање приватности корисника кроз локално складиштење историје кретања.



Слика 1. Дијаграм компоненти система и ток података

2.2. Комуникациони слој

Комуникациона инфраструктура система заснована је на MQTT протоколу и реализована кроз спрегу локалног и удаљеног брокера, као што је илустровано на дијаграму компоненти (Слика 1).
Локални брокер (*Mosquitto* [6]): Представља чвориште интеграције са *Home Assistant*-ом, задужено за дистрибуцију порука унутар локалне мреже (LAN).
Удаљени брокер (*HiveMQ* [7]) и MQTT Мост: За комуникацију са уређајима на јавним мрежама користи се удаљени брокер. Веза између њега и локалног система остварена је путем *MQTT Bridge* механизма. Овај механизам омогућава локалном брокеру да аутоматски и транспарентно преузима поруке са удаљеног сервера.
Кључна предност овакве конфигурације је што *Home Assistant* инстанца није директно изложена интернету, чиме се значајно повећава безбједност система. Интегритет и повјерљивост података током преноса кроз јавну мрежу гарантовани су примјеном TLS енкрипције на транспортном слоју.

3. ИМПЛЕМЕНТАЦИЈА РЈЕШЕЊА

Реализација система спроведена је у окружењу заснованом на *Home Assistant OS* платформи (верзија 16.3), која се извршава на *Raspberry Pi 4* уређају. Процес имплементације обухватио је конфигурацију серверске инфраструктуре, успостављање сигурних комуникационих канала и развој логике за аутоматизацију.

3.1. Конфигурација хибридног MQTT окружења

Успостављање комуникационог слоја започето је на страни локалног система инсталацијом *Mosquitto* брокера као додатка (*add-on*) унутар *Home Assistant* платформе. Након основне локалне конфигурације, приступило се креирању удаљене инфраструктуре иницијализацијом кластера на *HiveMQ Cloud* платформи. На кластеру су дефинисани јединствени акредитиви за приступ и омогућен TLS протокол на

порту 8883, чиме је креирана сигурна улазна тачка за податке са интернета.

Кључни корак имплементације представљало је повезивање локалног и удаљеног брокера конфигурацијом *MQTT Bridge* механизма. Ради модуларности и лакшег одржавања, конфигурација моста није уписивана директно у главну конфигурацију, већ је креирана засебна датотека *bridge.conf*. У њој је дефинисана директива која успоставља везу ка удаљеном кластеру, укључујући путању до *Root CA* сертификата (*Let's Encrypt ISRG Root X1*) неопходног за верификацију идентитета сервера. На овај начин остварује се једносмјерна TLS аутентификација, при чему клијент верификује идентитет удаљеног MQTT брокера. Правило мапирања **topic owntracks/# in 1** осигурава да се све поруке са овом темом аутоматски реплицирају са удаљеног на локални брокер уз QoS 1 ниво услуге, чиме се гарантује испорука чак и у случају краткотрајних прекида мреже и чиме се омогућава пренос података изван локалне мреже без потребе за директним излагањем *Home Assistant* инстанце интернету.

3.2. Интеграција клијентских апликација

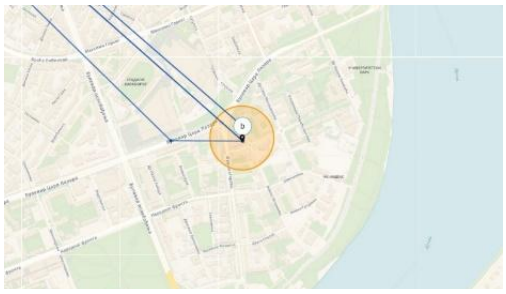
За прикупљање података одабрана је мобилна апликација *OwnTracks* због своје флексибилности и подршке за MQTT. На клијентским уређајима (iOS и Android) апликација је конфигурисана у режиму "Тихо" (*quiet*), који шаље податке само када се иницирају од стране корисника, како би се смањило слање непотребних промјена локација.

У параметрима конекције унесена је адреса удаљеног брокера и активирана TLS енкрипција. Сваки уређај је добио јединствен идентификатор теме (*topic*), што *Home Assistant*-у омогућава да аутоматски препозна извор података. Захваљујући *Discovery* механизму платформе, није било потребно ручно дефинисати ентитете; систем је аутоматски креирао одговарајуће device_tracker објекте на основу прве примљене поруке.

Детектовани *device_tracker* ентитети се приказују на *Map Dashboard*-у као што можемо видјети на Слици 2.

3.3. Аутоматизација

Апликативни слој рјешења заокружен је имплементацијом логике за обраду просторних догађаја. Прво су у систему дефинисане гео-зоне (Home, Fakultet) са одговарајућим радијусима детекције (Слика 2).



Слика 3. Приказ локације на *Map Dashboard*-у

Затим је развијена универзална аутоматизација која прати промјене стања свих праћених ентитета. Логика аутоматизације динамички пореди претходну и тренутну локацију корисника како би детектовала три типа догађаја: Улазак у зону, Излазак из зоне и Промјена зоне. На основу детектованог догађаја, систем генерише информативну поруку и шаље *push* нотификацију путем *Home Assistant Companion* сервиса као што је приказано на Слици 3, истовремено биљежећи догађај у системски дневник (*Logbook*) ради касније анализе и ревизије кретања.

Слика **Error! No text specified style in document.**2. *Push* нотификација на телефону

4. ДИСКУСИЈА РЕЗУЛТАТА

Тестирање система показало је да је латенција (кашњење) при преносу поруке од мобилног уређаја до приказа на мапи у кући мања од 2 секунде у већини случајева, што задовољава потребе за праћењем у реалном времену.

Посебан акценат стављен је на поузданост система у условима нестабилне мрежне конекције. Примјеном QoS 1 (*Quality of Service*) нивоа услуге, обезбијеђено је да се поруке не губе чак ни у случају привременог прекида интернет везе код куће. У таквим ситуацијама, удаљени *HiveMQ* брокер привремено складишти пристигле поруке ("*queueing*") и аутоматски их испоручује локалном *Mosquitto* брокеру чим се *Bridge* конекција поново успостави. Ово гарантује континуитет података и тачност историје кретања, која се трајно архивира искључиво на локалном *Home Assistant* серверу, чиме је приватност корисника у потпуности заштићена.

5. ЗАКЉУЧАК

У овом раду представљено је рјешење за интеграцију и визуализацију геопросторних података у оквиру *Home Assistant* платформе, засновано на употреби MQTT протокола и хибридне брокерске архитектуре. Комбинацијом локалног и удаљеног MQTT брокера, повезаних механизмом MQTT моста, омогућен је поуздан ток локацијских података ка систему, независно од мрежног окружења у којем се уређаји налазе.

Предложени приступ омогућава мапирање локацијских података на ентитете типа device_tracker и њихову визуализацију путем *Map Dashboard* компоненти, без потребе за развојем посебних визуализационих модула. Поред приказа података, систем подржава и имплементацију просторних

аутоматизација заснованих на догађајима уласка и изласка уређаја из дефинисаних зона, чиме се додатно проширује функционалност платформе.
Посебан значај рада огледа се у очувању безбједности и приватности корисника, које се постижу примјеном TLS енкрипције и локалном обрадом и складиштењем осјетљивих података. Представљено рјешење је флексибилно, прошириво и примјењиво у различитим IoT сценаријима, те може послужити као основа за даљи развој напреднијих система за анализу кретања, подршку већем броју корисника и интеграцију додатних извора геопросторних података.

6. ЛИТЕРАТУРА

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A survey on enabling technologies, protocols, and applications” IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions” Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, Sept. 2013.

[3] “Home Assistant documentation”, https://www.home-assistant.io/docs/. (приступљено у септембру 2025)

[4] Nordic Semiconductor Academy, “Lesson 4: MQTT protocol” Cellular IoT Fundamentals. https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-4-cellular-fundamentals/topic/lesson-4-mqtt-protocol/. (приступљено у септембру 2025)

[5] S. Cope, “Mosquitto MQTT bridge – usage and configuration” Steve’s Internet Guide, Oct. 2024. http://www.steves-internet-guide.com/mosquitto-bridge-configuration/. (приступљено у септембру 2025)

[6] “Mosquitto: An open source MQTT broker”, https://mosquitto.org/documentation/. (приступљено у септембру 2025)

[7] “HiveMQ Cloud: Fully managed MQTT broker”, https://www.hivemq.com/mqtt-cloud-broker/. (приступљено у септембру 2025)

[8] "OwnTracks Booklet (Documentation)", https://owntracks.org/booklet/. (приступљено у септембру 2025)

Кратка биографија:

Миладин Момчиловић рођен је 25.01.2001. године у Љубовији. Факултет техничких наука у Новом Саду уписао је 2019. године. Основне студије завршио у року 2023. године и након тога уписао мастер студије на истом факултету смјер Софтверско инжењерство и информационе технологије.
Контакт: miladin.momcilovic@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.4
ДОИ: https://doi.org/10.24867/35BE20Kuzminac

Софтверско решење за обраду, предикцију и визуализацију података о земљотресима

A Software Solution for the Processing, Prediction and Visualization of the Earthquake Data

Нина Кузминац, Владимир Димитриески, *Факултет техничких наука, Нови Сад*

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Данас, услед све веће количине података о сеизмичким активностима, јавља се потреба за системима који ће омогућити благовремено, поуздано и разумљиво праћење и анализу земљотреса. Циљ овог рада је омогућавање анализе сеизмичких података, прогнозе магнитуде и праћења земљотреса у реалном времену уз помоћ развијеног софтверског решења. Стога, имплементирано је софтверско решење које обједињује пакетну и обраду токова података, примену машинског учења и интерактивну визуализацију ради анализе историјских података, праћења актуелне сеизмичке активности, предикције магнитуде и обавештавања о потенцијално опасним догађајима.*

Кључне речи: *пакетна обрада података, обрада токова података, визуализација података, предикција магнитуде земљотреса*

Abstract – *Nowadays, due to the increasing amount of data on seismic activities, there is a growing need for systems that will enable timely, reliable, and comprehensible monitoring and analysis of earthquakes. The aim of this work is to enable seismic data analysis, magnitude prediction and earthquake monitoring in real time with the help of a developed software solution. Therefore, a software solution was implemented that combines batch and stream processing, application of machine learning and interactive visualization for the purpose of analyzing historical data, monitoring current seismic activity, magnitude prediction and alerting about potentially dangerous events.*

Keywords: *batch processing, real-time processing, data visualization, earthquake magnitude prediction*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Владимир Димитриески, ван. проф.

1. УВОД

Земљотреси представљају један од најзначајнијих природних феномена који директно угрожавају људске животе, инфраструктуру и економску стабилност читавих региона.

Развојем глобалних мрежа сеизмолошких станица и савремених технологија за акумулацију информација, количина података о сеизмичкој активности расте из године у годину. Истовремено, повећава се и потреба за њиховим ефикасним складиштењем, обрадом и анализом.

Последњих година, све већи значај добијају технике машинског учења. Примењене на историјске сеизмичке податке, оне омогућавају изградњу модела који могу да предвиде одређене карактеристике новооткривених земљотреса или препознају обрасце који нису уочљиви класичним статистичким приступима.

Важан аспект представља и визуализација сеизмичких података, како историјских, тако и оних који пристижу у реалном времену. Разумевање механизама настанка земљотреса, као и идентификација образаца у подацима, од пресудног је значаја за унапређивање система раног упозоравања и планирање инфраструктуре отпорне на потресе.

Данас, све више људи има потребу да разуме и прати сеизмичке активности, због њиховог потенцијално опасног утицаја, што захтева брз и поуздан увид у податке, без оптерећења детаљима начина њихове обраде. Због тога постоји потреба за решењем које омогућава анализу историјских података, праћење актуелних догађаја, процену потенцијалне магнитуде у реалном времену и правовремено упозоравање у случају повећаног сеизмичког ризика, кроз интуитиван и разумљив приказ података.

Циљ овог рада је омогућавање анализе сеизмичких података, прогнозу магнитуде и праћење земљотреса у реалном времену уз помоћ развијеног софтверског решења. Комбинацијом анализе историјских података и обраде података који пристижу у реалном времену, софтверско решење представљено у овом раду настоји да обезбеди дубљи увид у обрасце појављивања земљотреса и омогући бржу процену њихове потенцијалне снаге.

2. АНАЛИЗА ПОСТОЈЕЋИХ СИСТЕМА

Анализа сеизмичких података представља кључну компоненту у праћењу сеизмичке активности, процени

ризика и унапређењу процедура заштите. У наставку поглавља представљени су неки од најзначајнијих софтверских решења који се користе у пракси за обраду података о земљотресима, анализу таласних облика и управљање мрежама сеизмолошких станица.

SeisComP је један од најраспрострањенијих софтверских система за аквизицију, процесирање и дистрибуцију сеизмичких података у реалном времену [1]. Првенствено се користи у професионалним сеизмолошким центрима. Подржава аутоматску детекцију и локацију земљотреса, рад са више станица, транспорт података у реалном времену. Такође, омогућава стабилан рад националних и регионалних служби, што га чини стандардом у сеизмолошким институцијама.

ObsPy је *Python* радни оквир отвореног кода намењен читању, обради и анализи сеизмичких података [2]. Подржава све кључне формате сеизмичких података и интеграцију са глобалним сервисима попут *IRIS*-а, *GFZ*-а или *ORFEUS*-а. Пружа развојне алате за филтрирање сигнала, корекције инструмента, *STA/LTA* детекцију и визуализацију. *ObsPy* је кључан алат у истраживању и изради научних и оперативних апликација.

SEISAN, систем за сеизмичку анализу, је скуп програма и једноставна база података за анализу земљотреса из аналогних и дигиталних података [3]. Садржи *GUI* и служи за одређивање сеизмичког момента, локација, анализу спектра итд. Намењен је пост-сеизмичкој анализи, а често се користи у научним институцијама. Корисницима омогућава уређивање догађаја и статистичке извештаје.

USGS је глобално најпознатија платформа за праћење земљотреса, пружајући *ComCat* каталог, *API*-је, интерактивне мапе земљотреса и *ShakeMap* карте померања тла [4]. Поред тога, *USGS* развија *PAGER* систем за процене фаталних случајева и економских губитака након потреса. Платформа је референтни извор за глобална обавештења и податке у реалном времену.

EMSC је међународна организација која обједињује податке из бројних сеизмолошких мрежа и прикупљање извора путем заједнице [5]. Њихова апликација *LastQuake* користи аутоматско препознавање повећања активности корисника (посете сајту, мобилна апликација, друштвене мреже) како би детектовала потресе у реалном времену [6]. Пружа брза обавештења, атрактивне мапе, листе догађаја и пријаве примећених земљотреса од стране становништва.

GEOFON (*GFZ Potsdam*) корисницима пружа приступ опсежној међународној мрежи трајних сеизмолошких станица и богатој архиви записа о сеизмолошким таласима, укључујући податке у реалном времену и архивске записе, као и алате и веб-сервисе за преузимање и анализу тих података. Такође, омогућава аутоматско одређивање основних параметара земљотреса и дистрибуцију параметара догађаја научној заједници, центрима за рано упозоравање и другим релевантним институцијама.

Наведена софтверска решења и интернет платформе представљају основу савремене сеизмолошке праксе, али су углавном усмерене на прикупљање, основну анализу и приказ података. Развој сопственог решења произилази из потребе за дубљом аналитиком, интеграцијом великих историјских скупова података и обрадом догађаја у реалном времену, као и применом техника машинског учења. Предложено софтверско решење, описано у овом раду, обједињује обраду историјских и података у реалном времену, кориснички прилагођене упите, предикцију магнитуде и интерактивну визуализацију. На тај начин не надовезује се само на постојеће изворе података, већ омогућава напредну анализу, аутоматизовано доношење закључака и благовремено информисање корисника, што га издваја као аналитички и предиктивни алат у односу на постојећа решења.

3. АРХИТЕКТУРА СОФТВЕРСКОГ РЕШЕЊА

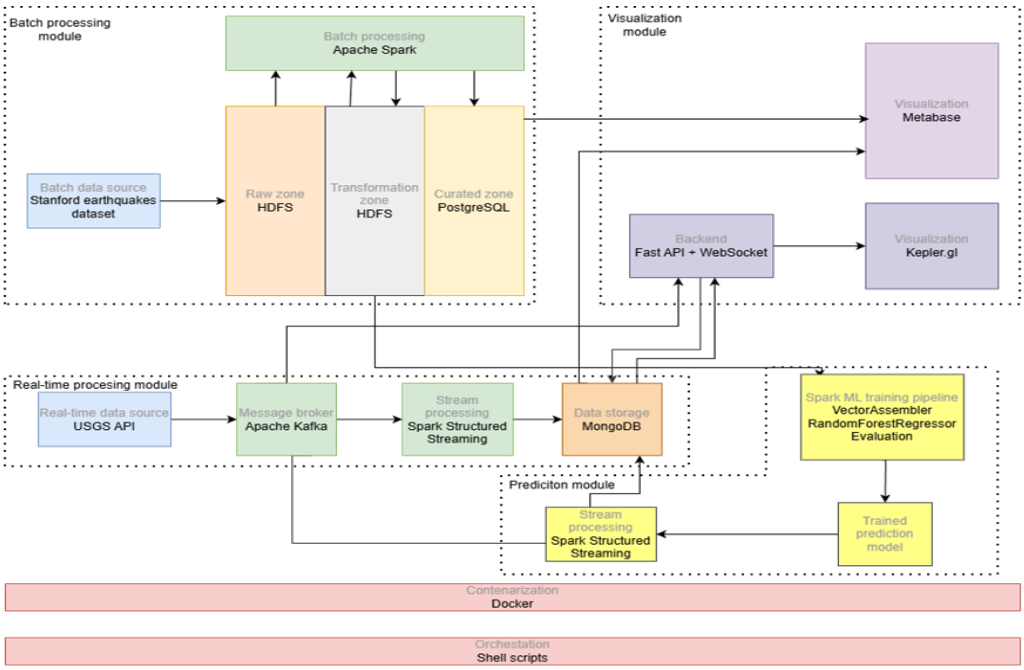
Софтверско решење за обраду, предикцију и визуализацију сеизмичких података заснива се на скупу међусобно интегрисаних компоненти које заједно формирају јединствено софтверско решење за обраду података у пакетном режиму и у режиму обраде у реалном времену, предикцију магнитуде и визуализацију резултата. Архитектура софтверског решења приказана је на слици 1 и организована је у четири функционалне целине: пакетну обраду података, обраду токова података, модул за предикцију магнитуде и визуализациони модул.

Ове целине функционишу као засебне, али повезане компоненте, које размењују податке путем заједничких складишта и комуникационих механизама. На тај начин софтверско решење омогућава да се историјски подаци, текући догађаји и предиктивни резултати интегришу у кохерентан процес континуиране анализе.

3.1. Пакетна обрада података

Прва компонента платформе представља модул за пакетну обраду података (енгл. *Batch processing module*) и представљена је у горњем левом углу на слици 1. Пакетна обрада обухвата систем за управљање великим историјским скуповима података о сеизмичкој активности. У овом делу софтверског решења врши се централизовано прикупљање, складиштење и трансформација података ради формирања стабилне аналитичке основе.

Историјски подаци преузимају се из спољних извора и смештају у *HDFS*, у оквиру сирове зоне која служи као трајни репозиторијум неизмењених оригиналних датотека. Ова зона обезбеђује накнадну реконструкцију било ког корака обраде. Над подацима се затим извршава обрада у оквиру окружења *Apache Spark*. *Spark* је задужен за скалабилну трансформацију, укључујући структурирање података, припрему временских и геолокационих атрибута и формирање аналитичких скупова. Резултати се складиште у зони трансформација *HDFS*-а, где добијају уједначен формат погодан за аналитичке упите и даљу употребу у модулу машинског учења. Коначни обрађени подаци уписују се у базу података *PostgreSQL* која представља уређену зону, који у архитектури обавља улогу централизоване аналитичке базе података.



Слика 1. Архитектура софтверског решења

3.2. Обрада токова података

Другу функционалну целину чини модул за обраду података у реалном времену (енгл. *Real-time processing module*) и представљен је испод модула за пакетну обраду података на слици 1. У оквиру овог модула, подаци се преузимају у реалном времену из *USGS API*-ја и представљају текуће сеизмичке догађаје који се глобално ажурирају у кратким интервалима. Пристижући догађаји прво се прослеђују преко платформе *Kafka*. *Kafka* омогућава хоризонталну скалабилност, високу пропусност и гарантовану испоруку порука, што софтверско решење чини отпорним на прекиде и нестабилности у спољним сервисима.

Spark Structured Streaming је интегрисан као конзумент токова *Kafka*. *Spark Structured Streaming* у архитектури обавља улогу механизма за континуирано структурирање података и њихово прилагођавање за даље коришћење у складишту и модулима предикције. Резултати добијени након извршавања упита над трансформисаним подацима се уписују у базу података *MongoDB*, која служи као складиште полуструктурираних података у реалном времену. Ова компонента архитектуре омогућава да софтверско решење реагује на нове догађаје у року од неколико секунди, што је неопходно за апликације чија је сврха праћење сеизмичке активности у реалном времену.

3.3. Визуализација података

Визуализациони модул (енгл. *Visualization module*), приказан у горњем десном углу на слици 1, представља кориснички оријентисан део архитектуре и интегрише резултате пакетне обраде и обраде токова података у јединствен интерфејс. Овај модул обухвата два комплементарна подсистема: аналитичку визуализацију историјских података и динамичку геопросторну визуализацију догађаја у реалном времену. Аналитички део је заснован на платформи *Metabase*, која се ослања на *PostgreSQL* као извор података. У овој компоненти користе се агрегирани и историјски подаци настали током пакетне обраде, уз могућност формирања контролних табли, статистичких прегледа и временских анализа. Други подсистем реализује визуализацију у реалном времену путем алата *Kepler.gl,* односно праћење актуелне сеизмичке активности путем интерактивне мапе. Визуализација се заснива на подацима о земљотресима који тренутно пристижу у реалном времену, као и на подацима који су раније примљени у реалном времену и били сачувани у бази података *MongoDB*. Подаци које серверски слој прими у реалном времену чувају се у бази података како би обезбедили историјски преглед догађаја насталих у реалном времену, што омогућава да корисник приликом отварања *Kepler.gl* мапе има комплетан увид у раније и актуелне земљотресе, без приказа празне мапе.

Комбинацијом ова два визуализациона механизма софтверско решење пружа истовремено увид у дугорочне трендове и тренутна дешавања, чинећи визуализацију кључним делом целокупне архитектуре.

3.4. Модул за предикцију магнитуде

Модул за предикцију магнитуде (енгл. *Prediction module*), приказан у доњем десном углу на слици 1, представља интелигентни део архитектуре који се ослања на машинско учење ради предикције магнитуде земљотреса. Он је структуриран као комбинација процеса обучавања и процеса извођења предикције. Обучавање се изводи периодично, где библиотека *Spark ML* користи припремљене историјске податке за изградњу модела. Одабране колоне се комбиновањем у вектор припремају за улаз у модел користећи *VectorAssembler*, чиме се испуњавају формални услови библиотеке *Spark ML*. На тај начин се свака

појединачна евиденција своди на структуру погодну за обучавање и тестирање. За предикцију је изабран модел *RandomForestRegressor* јер омогућава учење нелинеарних односа између атрибута и добро подноси шум и нестандардну расподелу података типичну за сеизмичке догађаје.

Након завршетка обучавања, модел се процењује помоћу метрика *RMSE*, *MAE* и R^2. Ове мере омогућавају да се оцени колико добро модел реконструише стварну магнитуду и у којој мери објашњава варијансу присутну у подацима. Ниске вредности *RMSE* и *MAE*, добијене приликом евалуације модела, указују на добру тачност и стабилност предикција магнитуде земљотреса, док вредност коефицијента детерминације R^2 показује да модел објашњава већину варијабилности у подацима.

Обучени модел се затим чува у централизованом складишту модела које је доступно осталим компонентама софтверског решења.

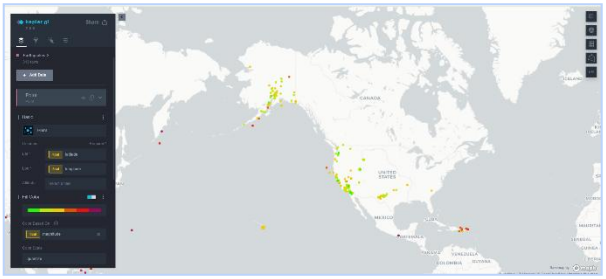
У реалном времену, модул за предикцију магнитуде учитава већ обучени модел и примењује га на нове догађаје који пристижу кроз платформу *Kafka*. Предикције се затим уписују у базу података *MongoDB* и постају доступне визуализационим и обавештавајућим подсистемима. Овај модул такође обухвата логику за генерисање упозоравајућих сигнала када предикција пређе унапред дефинисан праг.

Оваква организација омогућава да софтверско решење комбинује историјске податке, догађаје у реалном времену и машинско учење у јединствен, аутоматизован и скалабилан предиктивни механизам.

4. ЗАКЉУЧАК

Примарни циљ развијеног софтверског решења је омогућавање анализе сеизмичких података, прогнозе магнитуде и праћења земљотреса у реалном времену. Софтверско решење обезбеђује интегрисану обраду историјских и текућих података, као и упозоравање у случају потенцијално опасних сеизмичких активности. Коришћењем платформи *Spark* и *Kafka* и компоненте *Spark Structured Streaming* омогућено је континуирано прикупљање, обрада и трансформација података, уз скалабилну и флексибилну архитектуру. Модел машинског учења, обучен над историјским подацима, примењује се над новим догађајима у реалном времену, уз сталну процену тачности предикције.

У случају прекорачења дефинисаног прага прогнозиране магнитуде, систем шаље обавештење путем електронске поште. Резултати пакетне обраде приказују се кроз *Metabase*, док се подаци који пристижу у реалном времену визуализују на мапи у алату *Kepler.gl,* као што је приказано на слици 2. Сваки земљотрес је приказан као тачка на мапи, где различите боје указују на разлике у магнитуди и највећа концентрација земљотреса јавља се дуж тектонских плоча. Резултати постигнути током развоја софтверског решења показали су да је могуће интегрисати пакетну обраду и обраду токова података са моделом машинског учења и обезбедити платформу која у реалном времену обрађује земљотресе, предвиђа њихову магнитуду и пружа корисне и правовремене информације кроз визуализационе алате.



Слика 2. Визуализација земљотреса који се догађају у реалном времену уз помоћ *Kepler.gl*

Иако софтверско решење показује висок ниво функционалности и стабилности, постоје ограничења која треба имати у виду. Прецизност модела машинског учења зависи од квалитета и обима историјских података, док кашњења у долазним догађајима могу утицати на правовременост предвиђања. Такође, инфраструктурни захтеви окружења *Spark* могу бити значајни у условима велике фреквенције података.

Потенцијална проширења софтверског решења укључују увођење напреднијих модела машинског учења, побољшани *feature engineering* и интеграцију података из више сеизмичких извора. Такође, софтверско решење се може проширити механизмима за геопросторну анализу, предвиђањем накнадних потреса и интеграцијом са постојећим платформама за цивилну заштиту, чиме би се унапредио квалитет раног упозоравања.

5. ЛИТЕРАТУРА

[1] ”SeisComP seismological software.” [Online]. Available: https://www.seiscomp.de/. [Accessed: 01-Dec-2025].

[2] “ObsPy Documentation (1.4.2).” [Online]. Available: https://docs.obspy.org/. [Accessed: 01-Dec-2025].

[3] “SEISAN.” [Online]. Available: https://seisan.info/. [Accessed: 01-Dec-2025].

[4] “Earthquake Hazards Program | U.S. Geological Survey.” [Online]. Available: https://www.usgs.gov/programs/earthquake-hazards/. [Accessed: 01-Dec-2025].

[5] “EMSC – European-Mediterranean Seismological Centre.” [Online]. Available: https://www.emsc-csem.org/. [Accessed: 01-Dec-2025].

[6] “LastQuake.” [Online]. Available: https://m.emsc.eu/. [Accessed: 01-Dec-2025].

Кратка биографија:

Нина Кузминац, рођена је 21. јануара 2002. године. Након завршених основних академских студија, уписала се на мастер академске студије на Факултету техничких наука, студијски програм Рачунарство и аутоматика.

Контакт: kuzminacn@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 621.311:620.9
ДОИ: https://doi.org/10.24867/35BE21Radovanovic

Соларна електрана у Змајеву и моделовање соларне електране у софтверском окружењу Матлаб

Solar power plant in Zmajevo and modeling of solar power plant in Matlab

Лазар Радовановић, *Факултет техничких наука, Нови Сад*

Студијски програм – ЕНЕРГЕТИКА, ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ

Кратак садржај –. *У овом мастер раду је обрађена тема соларне електране у Змајеву инсталисане снаге 192,64 kW са прикључењем на мрежу. Затим је ова електрана измоделована у софтверском програму Матлаб-у и прикључена у модел дистрибутивне мреже. У моделу дистрибутивне мреже IEEE 13 са прикљученом соларном електраном су одрађене симулације и тестирано је понашање соларне електране током квара у мрежи.*

Кључне речи: *Фотонапонска електрана, обновљиви извори енергије*

Abstract – *In this master's thesis, the topic of a 192.64 kW solar power plant in Zmajevo connected to the grid is examined. The power plant was then modeled in the MATLAB software program and integrated into a distribution network model. In the IEEE 13 distribution network model with the connected solar power plant, simulations were performed and the behavior of the solar power plant during a network fault was tested*.

Keywords: *Solar Power Plant, renewable energy.*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Александар Станисављевић, вандредни професор

1. УВОД

Без електричне енергије не би био могућ данашњи начин живота људи. Енергија је природни ресурс. Електрична енергија се може користити на различите начине и у различитим областима људског деловања, од кућних апарата, индустрије, саобраћаја, пољопривреде, администрације, итд. Скоро да не постоји ни један уређај који за свој рад не захтева електричну енергију у неком облику. Основне величине којима се описује електрична енргија су: напон, струја, снага, и фреквенција. Електрична енергија се може испољити на различите начине и користити за различите намене. Практично се може трансформисати у све друге видове енергије, механичку, топлотну, светлосну, хемијску, итд. Користи се у свим областима људског деловања. У индустрији за напајање електромоторних погона, електричних пећи, за осветљење, за напајање електронских уређаја, телекомуникационих система, рачунарске опреме, електричних возила, итд.

Потрошња електричне енергије је динамичка величина, јер се непрестано мења током дана, недеље, месеца и године. Та променљивост је последица укључења или искључења постојећих потрошача. Због сезонских промена климе (температуре, брзине ветра, падавина), као и одређених дневних и сезонских карактеристика рада великих потрошача електрине енергије, постоје сезонске варијације потрошње на нивоу електроенергетског система. Тако је на пример потрошња већине индустријских потрошача независна од годишњег доба, док је потрошња за расвету, грејање, и генерално у домаћинствима, у знатној мери зависна од годишњег доба.

Пораст људске популације на Земљи узрокује повећање потреба за енергијом, као и повећано трошење енергетских ресурса, и зато се енергетика развија константно, са великим развојем обнољивих извора, и константим повећањем броја електрана које користе обновљиве изворе задњих деценија.

2. ПОЛОЖАЈ ЕЛЕКТРАНЕ У ЕЕС-у

Електроенергетски систем (ЕЕС) је комплексан и динамички систем. Његова је улога да сигурно, поуздано и економично снадбева потрошаче електричном енергијом. Са растом интерконекција прелаза границе појединих држава прерастају у континеталне системе. Специфичност проучавања ЕЕС-а се базира се на јакој ослоњености на реалан живот и животне потребе, и на потребу за применом модерних научних и технолошких достигнућа у циљу успешног функционисања овог система. С обзиром на велика улагања при изградњи и експлоатацији енергетских капацитета и одржавању ЕЕС-а .

Примарни разлог постојања ЕЕС-а је подмирење потреба потрошача за електричном енергијом. За данашњи развој човечанства можемо захвалити управо електричној енергији, па из те чињенице произлази да је ЕЕС најутицајнији и најраспрострањенији. Савремени развој ЕЕС-а укључује коришћење обновљивих извора енергије, попут соларних и ветроелеткрана. Тема овог рада је соларна електрана у Змајеву, и њено моделовање у програмском окружењу Матлаб.

3. ОБНОВЉИВИ ИЗВОРИ ЕНЕРГИЈЕ

Обновљиви извори енергије су природни извори енергије који се стално обнављају у природи и који не исцрпљују ресурсе планете. За разлику од фосилних горива (угаљ, нафта, гас), њихова употреба много мање загађује животну средину и доприноси смањењу емисије гасова стаклене баште.
Све више земаља прелази или повећава капацитете обновљивх изворе енергије – изворе који се у природи стално обнављају и не исцрпљују током употребе. То су извори који постоје захваљујући природним процесима као што су сунчево зрачење, кретање воде и ветра, као и биолошки циклуси. Њихова употреба доприноси одрживом развоју, јер обезбеђују енергију без угрожавања потреба будућих генерација.
Улагање у обновљиве изворе енергије има више циљева:

- смањење зависности од увоза енергената,
- побољшање енергетске безбедности,
- очување природних ресурса,
- смањење загађења и ублажавање климатских промена,
- развој нових технологија.

Током последњих деценија, технолошки напредак и смањење цена учинили су обновљиве изворе све доступнијим и економски исплативијим. Данас они постају кључни елемент у транзицији ка чистијој и одрживој енергетици.
Неки од типови обновљивих извора су:

- Соларна енергија
- Енергија ветра
- Хидроенергија
- Биомаса

У овом раду је нагласак на енергији добијеној из соларне енергије.

4. ТИПОВИ ОБНОВЉИВИХ ИЗВОРА

4.1. Соларне електране

Соларне електране претварају енергију сунчевог зрачења у електричну или топлотну енергију. Основни елемент су соларни панели (фотонапонске ћелије) који директно претварају светлосну енергију у електричну. Панели производе једносмерну струју (DC) која се помоћу инвертера претвара у наизменичну струју (AC) погодну за дистрибутивну мрежу.
Електране могу бити:

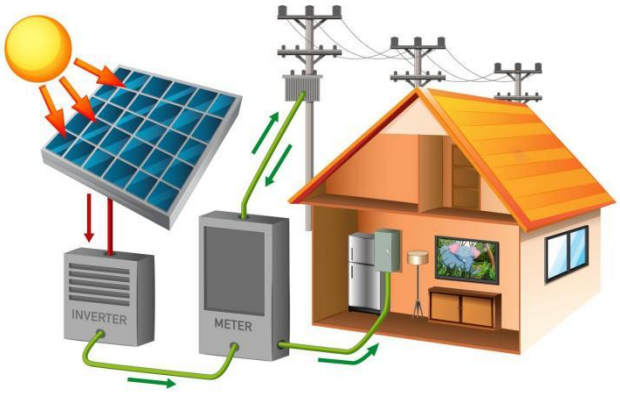
- Дистрибуиране (на крововима или на земљи) – намењене домаћинствима и малим потрошачима.
- Велики соларни паркови – на отвореним површинама.

На слици 1. приказан је пример соларне електране на крову објекта [1].

Слика 1. Соларна електрана на крову [1]

5. ПРИЦИП РАДА СОЛАРНЕ ЕЛЕКТРАНЕ

Соларна електрана представља савремени енергетски систем који користи обновљиви извор енергије (сунчево зрачење) за производњу електричне енергије. Захваљујући напретку технологије фотонапонских (PV) ћелија и електронских уређаја за конверзију, соларне електране су постале један од најбрже растућих извора чисте енергије у свету.
Основна идеја рада соларних електрана заснива се на фотонапонском ефекту, односно способности полупроводничких материјала да под утицајем сунчевих зрака ослобађају електроне и тиме стварају електричну струју. На тај начин, сунчева енергија се директно претвара у електричну без покретних делова, што систем чини поузданим, тихим и еколошки прихватљивим. На слици 2. приказан је графички приказ елемената соларне електране [2].



Слика 2. Графички приказ елемената соларне електране [2]

6. СОЛАРНА ЕЛЕКТРАНА У ЗМАЈЕВУ

Опрема фотонапонског система се монтира на металну носећу конструкцију смештену на поменутој локацији, на крову објекта, док се фотонапонски модули монтирају под углом у односу на раван земље да би се обезбедила максимална апсорпција сунчевог зрачења, а за дату локацију водећи при томе рачуна да су јужно оријентисани.
Електрична енергија се у соларној електрани производи методом конверзије неакумулираног сунчевог зрачења у једносмерну струју преко фотонапонских панела од поликристалног силицијума на бази полупроводничке технологије [3].

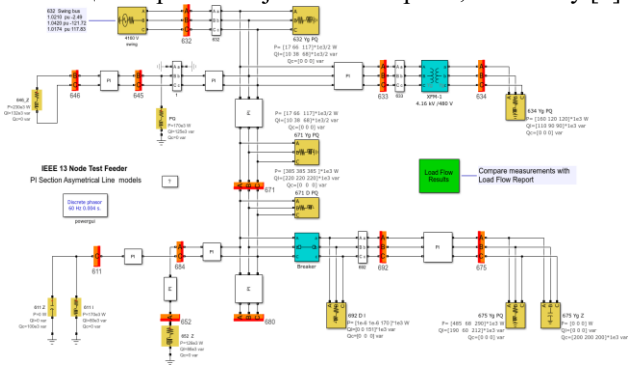
Слика 3. Соларна електрана у Змајеву

Соларна електрана поседује правилно распоређених 5 инвертора произвођача „INVT Solar Technology" типа XG 30KTR номиналне излазне снаге 30 kW са максималним улазним напоном инвертора 1100 V и типа XG 33KTR номиналлне излазне снаге 33 kW са максималним улазним напоном инвертора 1100 V. Поред инвертора поседује и 344 фотонапонских соларних панела типа Cortex OP560M55-P4 сваки појединачне снаге 560 W, димензије 2384x1096x35 mm, што даје инсталисану снагу од 192640 W. Напонски ниво на коју се прикључује електрана је 400 V. Кабловски водови наизменичног развода су типа 2xPP00-A 4x120 mm^2 где је трајно дозвољена струја 450 A, док каблови једносмерног развода су типа N2XY-J 5x16 mm^2 где је трајно дозвољена струја 102 A.

7. СИМУЛАЦИЈЕ У ПРОГРАМУ МАТЛАБ

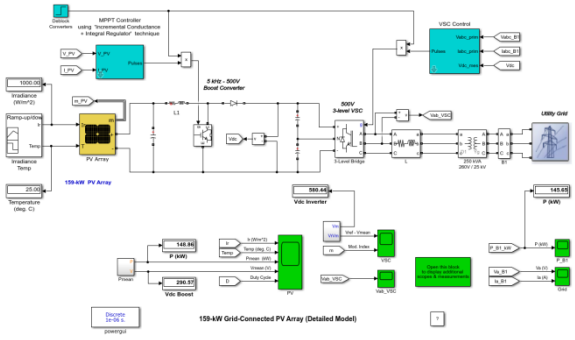
У оквиру Симулинк-а, дела софтверског пакета Матлаб, налази се модел једне мање дистрибутивне мреже. Креиран је посебан модел 1992. године од стране организације IEEE. У оригиналном пројекту постојале су 4 верзије ове мреже, са 13, 34, 37 и 123 чвора. IEEE 13 мрежа састоји се од 13 чворова. Направљена је по узору на северноамеричке мреже које су знатно другачије од европских, пре свега због постојања великог броја потрошача који су прикључени на једну или две фазе. Корен мреже представљен је swing генератором номиналног напона 4.16 kV. Чворови су повезани надземним и кабловским водовима који могу бити трофазни, двофазни или монофазни. На мрежу су прикључена три типа потрошача.

На слици 4. приказана је IEEE 13 мрежа, описана у [4].



Слика 4. IEEE 13 мрежа [4]

На слици 5. приказан је модел соларне електране моделоване у програмском окружењу Матлаб [4].

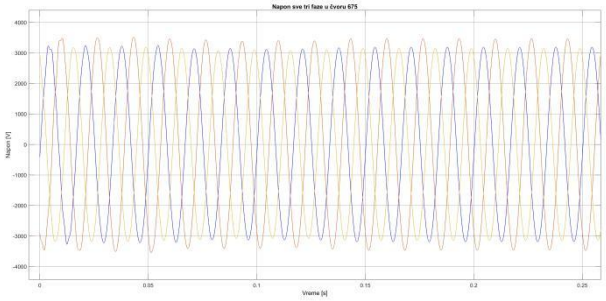


Слика 5. Модел соларне електране моделоване у програмском окружењу Матлаб [4]

Да би смо правилно измоделовали електрану користи се модел који је пре промена које смо унели имао снагу од 100 kW. Код модела коришћеног у овом раду, да би одговарао електрани у Змајеву, прилагођено је неколико битних параметара као што су: тип фотонапонскоg панела, тип инвертора као и трансформатор. Такође је повећан број стрингова, а што се тиче инвертора и трансформатора повећана је снага ових елемената, да би одговарале моделованом систему.

7.1. Симулација у режиму без квара

Основни циљ рада је испитивање утицаја дистрибуираних извора на напонске прилике и хармонијска изобличења у условима једнополног и двополног кратког споја. Прво ће бити одрађена симулација без квара да сагледамо какви су резултати добијени пре и после квара.



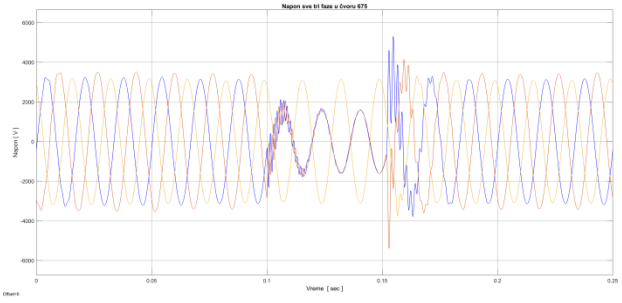
Слика 6. Напони три фазе у режиму без квара

На слици 6. приказани су напони све три фазе добијени симулацијом у режиму без квара. Извршено је мерење у чвору 675. Можемо приметити да су напони синусног облика и благо несиметрични, што је последица несиметрије ИЕЕЕ13 тест мреже.

7.2. Симулација при двополном кратком споју са земљом

Након посматрања рада соларне електране у дистрибутивном систему у уравнотеженом режиму, симулиран је рад током трајања двополног квара са земљом, и резултати су приказани на слици 7. Ова врста кратког споја настаје у тренутку када се две фазе трофазног система истовремено споје са земљом, док трећа фаза остаје исправна.

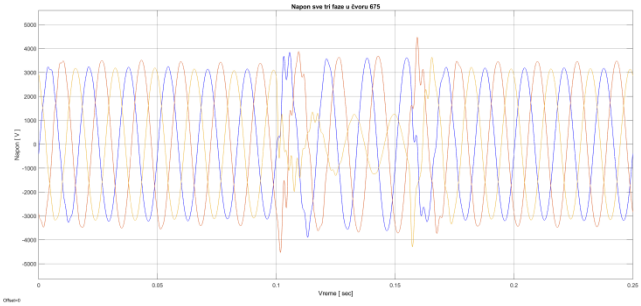
На слици 7. приказан је график напона све три фазе добијени симулацијом при двополном кратком споју дужине трајања од 0.5 секунди односно, квар је почео у 0.1 и завршио се у 0.15 секунди. Квар је настао у дистрибутивној мрежи, у чвору 632. Извршено је мерење у чвору 675.



Слика 7. Напони све три фазе у чвору 675 при двополном кратком споју са земљом

7.3. Симулација при једнополном кратком споју са земљом

Овај квар настаје када се једна фаза трофазног система директно споји са земљом, док су преостале две фазе исправне и остају под напоном.
На слици 8. приказан је график напона све три фазе добијени симулацијом при једнополном кратком споју дужине трајања од 0.5 секунди, квар почиње у 0.1 и завршава се у 0.15 секунди. Квар је симулиран у чвору 645. Извршено је мерење у чвору 675.



Слика 8. Напони све три фазе у чвору 675 при једнополном кратком споју

8. ЗАКЉУЧАК

У овом раду извршена је анализа соларне електране у Змајеву као и њено моделовање у програмском окружењу Матлаб. У симулацији соларне електране су посматрани случајеви двополног кратког споја са земљом и једнополног кратког споја са земљом у моделу соларне електране прикључене у дистирбутинву мрежу. Ови кварови су постављени на различите локације у мрежи, а затим је анализиран њихов утицај на напоне и хармонике у неколико кључних тачака дистрибутивног система. Да би се предвидело понашање система у случају квара, потребно је све прецизно измоделовати и симулирати, и то је циљ овог рада.
Као што видимо из резутата симулација, када прикључимо електрану и пустимо симулацију у рад без прикљученог квара примећујемо да су напони скоро симетрични, наравно уз мало одступања прве фазе, што је узроковано самом несиметријом ИЕЕЕ 13 тест система.
Међутим, у друга два случаја примећујемо значајне промене током трајања квара, у посматраним напонима у мрежи током квара. Амплитуда напона је значајно већа од номиналне вредности током трајања транзијентног режима приликом завршетка квара.

9. ЛИТЕРАТУРА

[1] Соларна електрана на крову доступно на: https://hermi.co.rs/reference/studija-slucaja/soncna-elektrarna-na-tovornih-skladiscih-luke-koper

[2] Графички приказ елемената соларне електране доступно на: https://www.poduzetnickicentar-aktiva.com/vas-vodic-kroz-solarne-elektrane/

[3] Гојко Влашки, „Пројекат за извођење (ПЗИ), Фотонапонска електрана активне снаге 159 kW AC на крову објекта, са испоруком вишка електричне енергије у дистрибутивни систем, у Змајеву, на катастарској парцели 3275/1 К.О. Змајево“, Змајево, 2025.

[4] Д. Ђукетић, Александар Станисављевић, В. Катић, ”Заштитна опрема и њено моделовање у дистрибутивној тест мрежи“, Зборник радова Факултет техничких наука, Год. 38, Бр. 9, 2023, pp.1077-1296

Кратка биографија:

Лазар Радовановић је рођен у Новом Пазару 1999. године. Основне студије уписао на ФТН-у 2018. године из области Електротехника и рачунарство, и завршио 2023.године. Исте године уписао мастер студије на ФТН у Новом Саду – смер Дистрибуирани енергетски ресурси.

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.8
ДОИ: https://doi.org/10.24867/35BE22Batanovic

Анализа вибрација помоћу GNSS пријемника

Vibration Analysis Using GNSS Devices

Јелена Батановић, *Факултет техничких наука, Нови Сад*

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Овај рад представља могућности коришћења GNSS (Global Navigation Satellite System) пријемника за анализу вибрација. Циљ овог рада је испитати да ли се помоћу GNSS уређаја вибрације могу у некој мери квалитетно мерити. У раду је креиран модел неуронских мрежа помоћу мерења добијених са GNSS пријемника и акцелерометра прикупљених у експерименту у ком су симулиране вибрације дефинисаних амплитуда и фреквенција.*

Кључне речи: *GNSS, акцелерометар, неуронске мреже, вибрације*

Abstract *– This paper presents the possibilities of using GNSS receivers for vibration analysis. The aim of this paper is to investigate whether vibrations can be measured to some extent using GNSS devices. In this paper, a neural network model is created using measurements obtained from GNSS receivers and an accelerometer, collected during an experiment in which vibrations of defined amplitudes and frequencies were simulated.*

Keywords: *GNSS, accelerometer, neural networks, vibrations*

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

1. УВОД

Вибрације представљају осцилаторна кретања тела или система око равнотежног положаја. Њихово разумевање и правилна анализа од великог су значаја за нашу безбедност, јер омогућавају благовремено откривање потенцијалних опасности и издавање упозорења, како у природи, тако и на објектима и машинама [1,2].

У овом раду испитује се могућност GNSS (Global Navigation Satellite System) пријемника за мерење карактеристика вибрација, фреквенције и амплитуде. Креирани су модели неуронски мрежа обучени подацима мерења GNSS уређаја, а као референтне вредности коришћени су подаци добијени акцелерометром. Мерења са оба уређаја прикупљена су у експерименту где су генерисане вибрације тачно дефинисаних амплитуда и фреквенција на креираној макети.

2. ПРЕГЛЕД СТАЊА У ОБЛАСТИ

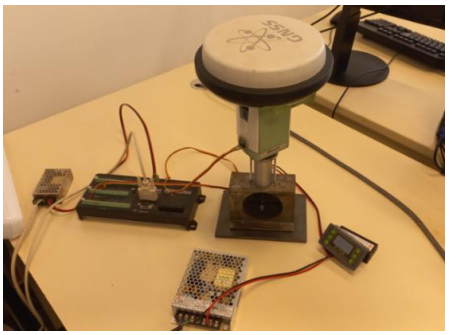
Ово поглавље пружа увид у методологије и радове различитих аутора у овој области. Посебан акценат је стављен на анализу вибрација у дијагностици кварова. Радови [3,4] се баве коришћењем средње квадратне вредности (енгл. Root Mean Square - RMS) и екстремних (енгл. peak) вредности у анализи вибрација и детекцији квара. У раду [5] анализа сигнала са акцелерометра и значајних фреквенција коришћена је као средство за детекцију и разумевање дефекта лежаја. Систематичан преглед коришћења анализе вибрација у дијагностици кварова представљен је у чланку [6]. У њему су описани аквизиција података, коришћени сензори, обрада сигнала са издвајањем значајних карактеристика и примена вештачке интелигенције у анализи кварова. Могућности GPS (Global Positioning System) уређаја за одређивање вибрација анализиране су у студији [7], где су праћене динамичке карактеристике зграде већих фреквенција. Интеграција два уређаја, GPS и акцелерометра, описана је у радовима [8,9], где је приказано да ови уређаји надопуњују један другог и да заједничким коришћењем дају најбоље резултате.

3. АНАЛИЗА И ПРИКУПЉАЊЕ ПОДАТАКА

Истраживање је спроведено на припремљеној макети на којој су генерисане вибрације са тачно дефинисаним амплитудама и фреквенцијама. Подаци су прикупљани помоћу GPS уређаја и акцелерометра.

3.1. Уређаји коришћени у пројекту

Макета из експеримента (слика 1.) је направљена у виду конструкције са клипом и ексцентром, са различитим полупречницима плочица: 1,5 cm, 1,75 cm и 2 cm.



Слика 1. Макета коришћена у експерименту

За покретање је коришћен електромотор, са неопходним напајањем и контролером електромотора. Додатно, за прикупљање података, се користи уређај за снимање и контролу података (Data Logger) Campbell Scientific CR1000, са неопходним напајањем и серијском RS-232 комуникацијом ка рачунару.
Акцелерометри мере убрзање вибрација и широко се користе у индустрији, машинству и истраживањима због своје осетљивости, прецизности и великог фреквентног опсега. За потребе овог пројекта коришћен је двоосни акцелерометар ADXL203 приказан на слици 2.

Слика 2. Изглед акцелерометра [10]

GNSS/GPS уређаји коришћени у пројекту су Leica GS15 (слика 3.) пријемник (ровер), уз њега и Leica CS15 контролер и Trimble R8s (слика 4.), који је коришћен као референтна станица (база) постављена на пар метара од макете. Примењена метода позиционирања је кинематичко позиционирање у накнадној обради (енгл. Post Processing Kinematics) и представља метод GNSS позиционирања који се заснива на фазним мерењима носиоца сигнала и омогућава постизање високе прецизности.

Слика 3. Leica Viva GS15 [11]

Слика 4. Trimble R8 [12]

3.2. Подаци

Подаци коришћени за анализу карактеристика GPS-а су прикупљани у оквиру два засебна мерења. Први експеримент обухвата мерења која су спроведена тако што су подаци прикупљани истовремено помоћу оба уређаја, GPS-а и акцелерометра, како би били временски синхронизовани и како би се могло једноставније анализирати понашање и карактеристике сигнала у истим временским тренуцима. Овај начин обухватао је поставку GPS-а на макету, испод ког је био постављен акцелерометар. Оптерећење, које је последица поставке GPS пријемника, је утицало на квалитет снимака добијених са акцелерометра, тј. на вибрације које он мери, стварајући велике шумове, те су снимци првог експеримента изузети из анализе. Експеримент је поновљен, и други пут су вибрације праћене у различитим тренуцима. На овај начин подаци нису временски синхронизовани, али су добијене вредности поузданије.
Подаци добијени мерењем GPS уређаја су обрађени у програму Leica GeoOffice. Овај програм омогућава решавање фазних неодређености и елиминисање сметњи из сигнала, чиме су добијене координате са фиксним фазним решењем и центиметарском тачношћу. Координата од значаја за овај систем је висина, те је рачуната њена средња вредност и она одузимана од сваког појединачног мерења како би се добио померај по висини, који је коришћен у даљој обради. Резултати мерени акцелерометром представљају убрзање, дато у мерној јединици g, која представља гравитационо убрзање. У даљој обради коришћено је убрзање у m/s^2 добијено помоћу формуле (1):

$$g = 9{,}81\ \mathrm{m/s^2} \qquad (1)$$

Мерења поменутих сензора су извршена у интервалима од по 0,05 секунди, што одговара фреквенцији узорковања од 20 Hz. У складу са Никвист-Шеноновом теоремом (2), фреквенција узорковања директно одређује највећу фреквенцију сигнала која се може поуздано анализирати.

$$fs \geq 2fmax \qquad (2)$$

f_s је фреквенција узорковања, а f_{max} максимална фреквенција сигнала. Како је фреквенција узорковања ових података 20 Hz, на основу ове теореме закључено је да све фреквентне компоненте сигнала до 10 Hz могу бити прецизно анализиране, што и обухвата фреквенције из експеримента које су у опсегу од 3 Hz до 5 Hz.

4. МОДЕЛ ПРЕДИКЦИЈЕ

Након примарне обраде и сређивања сирових података са сензора, спроведена је додатна припрема и адаптација података коришћењем окружења и доступних алата програмског језика Matlab. Поступак припреме података обухвата анализу и прилагођавање сигнала добијених са GPS уређаја и акцелерометра, како би били међусобно упоредиви, а потом и проналазак значајних образаца и карактеристика унутар ових сигнала који су релевантни као улази у неуронску мрежу.

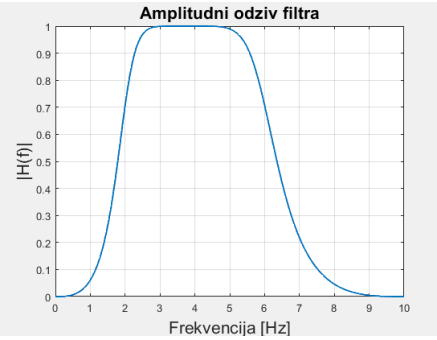
4.1. Matlab

Matlab представља програмско окружење које обухвата различите апликације за визуелизацију и тестирање, алате за генерисање кода и мноштво библиотека прилагођених областима као што су

управљање системима, машинско учење, обрада сигнала, роботика и многе друге.

4.2. Припрема улазних података

Анализа података отпочета је итерацијом кроз датотеке (1 датотека за снимак са акцелерометром и 5 датотека са GPS снимцима). Свако мерење обрађује податке тачно дефинисане амплитуде и фреквенције.
У првом експерименту, сваки од снимака је посматран као један улазни податак и креирана је једна неуронска мрежа са два излаза: амплитудом и фреквенцијом. На овај начин обраде података, улазни скуп представља матрицу величине 3 × 42 (3 карактеристике и 45 фајлова, од којих су 3 коришћена за накнадно тестирање мреже, а 42 као улази приликом тренинга). Фајлови представљају снимке комбинација фреквенција од 3 Hz, 4 Hz и 5 Hz и амплитуда од 1,5 cm, 1,75 cm и 2 cm. Да би се постигао што већи скуп улазних података, у другом експерименту је коришћена обрада сигнала по прозорима. Сваки од улазних скупова података подељен је на прозоре од 5 секунди уз преклапање од 50 %.
Након читања података са GPS уређаја и акцелерометра, над њима је примењен Butterworth филтер (слика 5.) који пропушта одређени опсег вредности сигнала. У овом раду коришћени филтер има дефинисани опсег од 2 Hz до 6 Hz и трећег је реда.



Слика 5. Butterworth-ov филтер, амплитудни одзив

Над припремљеним GPS подацима примењена је двострука деривација, како би се од позиције добило убрзање.

4.3. Обрада података

Карактеристике које дефинишу матрицу улаза првог експеримента су: доминантна фреквенција, амплитуда на доминантној фреквенцији (максимална амплитуда) и RMS параметар. Одређивањем максималне вредности сигнала добијена је доминантна амплитуда, док је доминантна фреквенција одређена њеним индексом. RMS (корен средње квадратне вредности) представља меру просечне енергије сигнала.
Други експеримент чине две неуронске мреже. Карактеристике од значаја за анализу амплитуде које су коришћене у раду су: максимална вредност омотача сигнала, средња вредност омотача сигнала, RMS, опсег од врха до врха сигнала и спектрална снага у задатом опсегу. Омотач сигнала представља глатку криву која прати промену амплитуде сигнала кроз време, а добијен је применом Хилбертове трансформације. Као значајне карактеристике за одређивање амплитуде сигнала коришћени су максимум и средња вредност омотача. Приликом креирања мреже за анализу фреквенција коришћена је поред поменутих: RMS вредности, доминантне фреквенције и спектралне снаге у задатом опсегу и додатна карактеристика, тежиште спектра. Жељене вредности, таргети којима се тежи су доминантна фреквенција (Hz) и амплитуда на доминантној фреквенцији (m).
Након припреме улазних података за неуронске мреже, у оба експеримента, подаци су нормализовани. Вештачка неуронска мрежа је обучавана функцијом Бајесове регуларизације пропагацијом уназад. Додатна два експеримента представљају обучавање неуронске мреже са прозорима краће дужине - 1,25 секунди, и са фиксним вредностима као таргетима.

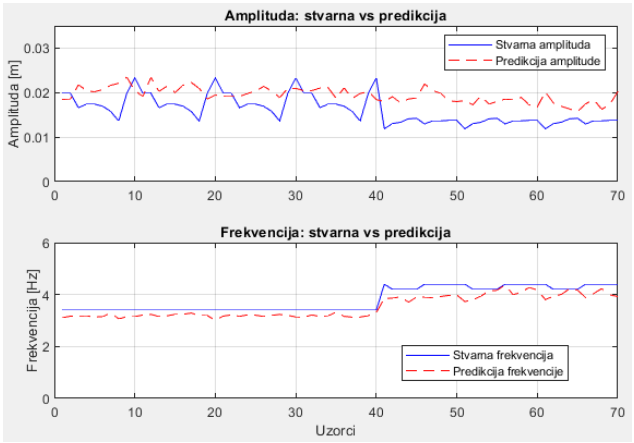
5. РЕЗУЛТАТИ И ДИСКУСИЈА

Неуронске мреже су тестиране над подацима који нису коришћени током обуке. Резултати примене неуронске мреже из првог експеримента при симулираној амплитуди од 2 cm и фреквенцији од 5 Hz, приказани су у Табели 1. Резултати за ову вредност амплитуде су у околини од око 2,27 cm, док је вредност из снимка акцелерометра 2,244 cm. За дату фреквенцију резултати су у околини вредности од 4,6 Hz, док је акцелерометар измерио 4,685 Hz.

Табела 1. Резултати употребе неуронске мреже из првог експеримента

	Добијене вредности	**Таргети**
Амплитуда (m)	0,02236	0,02244
Фреквенција (Hz)	4,554	4,685
Амплитуда (m)	0,02274	0,02244
Фреквенција (Hz)	4,55	4,685
Амплитуда (m)	0,02304	0,02244
Фреквенција (Hz)	4,723	4,685

Резултати добијени применом неуронских мрежа из другог експеримента уз употребу прозора величине 5 s са преклапањем од 50 %, приказани су на слици 6.

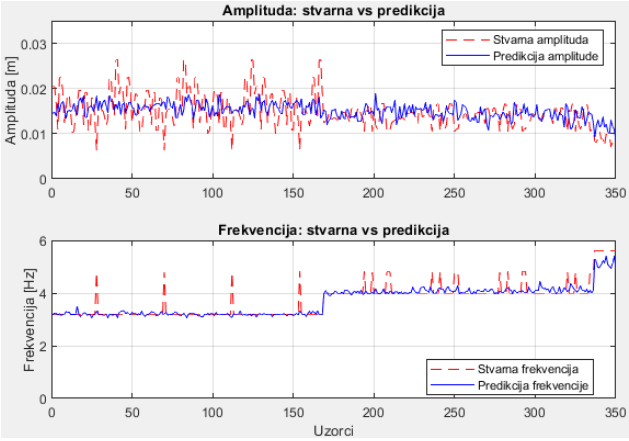


Слика 6. Резултати неуронских мрежа из другог експеримента

Приказ је подељен у два графика. Горњи график представља резултате неуронске мреже за одређивање амплитуде, а доњи за одређивање фреквенције.

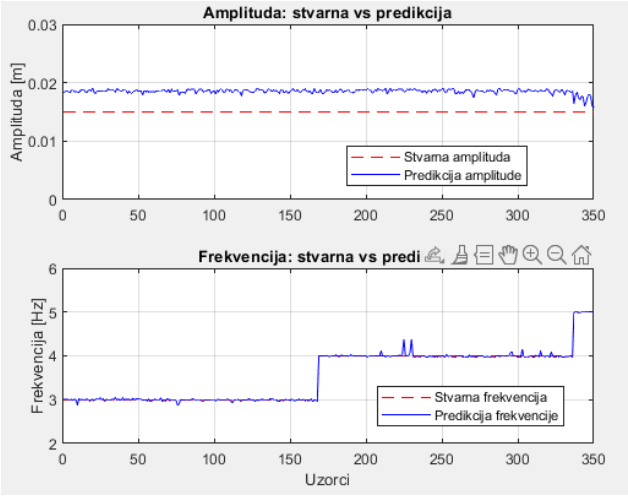
Црвеном испрекиданом бојом приказане су предвиђене вредности, док су плавом означени таргети мреже. У приказаних 70 узорака вредност симулиране амплитуда је 1,5 cm, док се симулирана фреквенција мења и првих четрдесет узорака износи 3 Hz, а наредних тридесет 4 Hz. Резултујуће вредности амплитуде варирају између 0,015 m до 0,023 m, али и таргети варирају. Резултујућа фреквенција је око 3 Hz за вредност таргета од 3,5 Hz и око 4 Hz за вредности од око 4,3 Hz.

Применом прозора величине 1,25 секунде, вредност амплитуде која је добијена као резултат у првом делу сигнала, варира у опсегу од 0,015 m до 0,02 m, при чему је ближа вредности од 0,015 m, док је у последњим узорцима сигнала нешто нижа од 0,015 m. Овакве вредности амплитуде су приближније симулираним вредностима него у случају са прозорима од 5 секунди. Опсег резултујућих вредности је мањи. Резултати неуронске мреже за одређивање фреквенције, такође, доста боље прате симулиране вредности у односу на претходни случај. Резултати су приказани на слици 7.



Слика 7. Резултати неуронских мрежа при дефинисаним прозором од 1,25 s

Као додатни експеримент, за таргете мреже унете су фиксне вредности уместо вредности добијених са акцелерометра. Резултати су приказани на слици 8.



Слика 8. Резултати неуронских мрежа из четвртог експеримента

Таргети неуронске мреже за одређивање фреквенције су прослеђене вредности од 3 Hz, 4 Hz и 5 Hz, а за одређивање амплитуде вредности од 0,015 m, 0,0175 m и 0,02 m. Резултати овог експеримента (слика 8.) указују на то да када је у питању мрежа за одређивање фреквенције, резултати су добри и прате понашање циљних вредности тј. предвиђају вредности од 3 Hz, 4 Hz и 5 Hz, док неуронска мрежа за одређивање амплитуде приказује нешто већа одступања и за вредности таргета од 0,015 m, добијена је вредност од око 0,0185 m.

6. ЗАКЉУЧАК

На основу спроведене анализе и експеримената, закључак је да интеграција података са GNSS/GPS уређаја и акцелерометра, уз примену неуронских мрежа омогућава добру процену карактеристика вибрација. Мрежа за одређивање фреквенције је показала поуздану процену доминантне фреквенције и добру усклађеност предвиђаних и стварних вредности на тестираним примерима. Резултати мреже за одређивање амплитуде прате реално понашање система, уз одређена одступања те нису довољно поуздани, али то указује на могућност примене овог приступа у пракси и добру основу за даља истраживања.

7. ЛИТЕРАТУРА

[1] T. Irvine, "*An Introduction to Shock and Vibration Response Spectra*", Revision E, 2023.

[2] J.T. Broch et al., "*Mechanical Vibration and Shock Measurements*", Nærum, Brüel & Kjær, 1984.

[3] J. Igba et al., "Analysing RMS and peak values of vibration signals for condition monitoring of wind turbine gearboxes", *Renewable Energy*, Vol. 91, pp. 90-106, June 2016.

[4] M. Minescu et al., "Fault detection and analysis at pumping units by vibration interpreting encountered in extraction of oil", *Journal of the Balkan Tribological Association*, Vol. 21, pp. 372-384, 2015.

[5] P.D. McFadden, J.D. Smith, "Model for the vibration produced by a single point defect in a rolling element bearing", *Journal of Sound and Vibration*, Vol. 96, pp. 69-82, September 1984.

[6] M.H.M. Ghazali, W. Rahiman, "Vibration Analysis for Machine Monitoring and Diagnosis: A Systematic Review", *Shock and Vibration*, Vol. 2021, August 2021.

[7] P. Psimoulis, et al., "Potential of Global Positioning System (GPS) to measure frequencies of oscillations of engineering structures", *Journal of Sound and Vibration*, Vol. 318, pp. 606-623, June 2008.

[8] X. Meng et al., "Detecting bridge dynamics with GPS and triaxial accelerometers", *Engineering Structures*, Vol. 29, pp. 3178-3184, November 2007.

[9] X. Li et al., "Full-scale structural monitoring using an integrated GPS and accelerometer system", *GPS Solutions*, Vol. 10, pp. 233-247, 2006.

[10] https://www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/eval-adxl203.html#eb-overview (pristupljeno u januaru 2025.)

[11] https://cpec.leica-geosystems.com/producto/base-rover-gs15-cs15-controller/ (pristupljeno u novembru 2025.)

[12] https://geomaticslandsurveying.com/wp-content/uploads/2018/11/trimble-r8s-datasheet.pdf (pristupljeno u decembru 2025.)

Кратка биографија:

Јелена Батановић рођена је у Шапцу 2000. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства - Аутоматика и управљања одбранила је 2025. год.
Контакт:
jelena.batanovic@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.934
ДОИ: https://doi.org/10.24867/35BE23Vitkovic

Анализа перформанси Whisper модела у превођењу српског говора на енглески језик

Performance analysis of the Whisper model in translating Serbian speech into English

Милана Витковић, *Факултет техничких наука, Нови Сад*

Студијски програм – ЕНЕРГЕТИКА, ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ

Кратак садржај – *Овај рад анализира методе аутоматског превођења српског говора на енглески језик, фокусирајући се на Whisper моделе и велике језичке моделе. Поређени су директни end-to-end приступ и каскадни систем који раздваја препознавање говора и превођење. Евалуација је спроведена на више тест скупова користећи BLEU, METEOR и WER метрике. Резултати показују да каскадни приступ са Whisper транскрипцијом и преводом помоћу великих језичких модела даје боље перформансе од директног превођења Whisper модела. Дообука Whisper модела за транскрипцију на српском значајно побољшава квалитет транскрипције и превода. Велики језички модели показују робусност чак и уз грешке у транскрипцији, што указује на њихов потенцијал у овој области. Као будући кораци препоручује се даља дообука и интеграција језичких модела ради унапређења квалитета превођења.*

Кључне речи (три до пет): *аутоматско превођење говора, whisper модел,*

Abstract – *This paper analyzes methods for automatic speech translation from Serbian to English, focusing on Whisper models and large language models (LLMs). It compares a direct end-to-end approach with a cascade system that separates speech recognition and translation. Evaluation was conducted on multiple test sets using BLEU, METEOR, and WER metrics. Results show that the cascade approach, combining Whisper transcription and LLM translation, outperforms direct Whisper translation. Fine-tuning Whisper for automatic speech recognition in Serbian significantly improves transcription and translation quality. LLMs demonstrate robustness even with transcription errors, highlighting their potential in this domain. Future work includes further fine-tuning and integration of language models to enhance translation accuracy.*

Keywords: (three to five): *automatic speech translation, whisper model*

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

1. УВОД

Аутоматско превођење говора (енг. *automatic speech translation*, AST) представља једно од најдинамичнијих истраживачких подручја у оквиру технологија говора и природног језика, подстакнуто напретком дубоког учења и развојем великих мултијезичких модела. Традиционални системи за превођење говора најчешће се заснивају на каскадној архитектури која обухвата два одвојена корака: препознавање говора (АПГ; енг. *automatic speech recogntion*, ASR) и машинско превођење (МП; енг. *machine translation*, MT). У овом приступу, говорни сигнал се најприје транскрибује у текст, након чега се добијена транскрипција преводи на циљни језик. Иако ефикасни, каскадни системи су подложни проблемима пропагације грешака, јер свака грешка настала у фази транскрипције директно утиче на квалитет финалног превода [1, 2].

Развој *end-to-end* неуронских архитектура омогућио је да се превођење говора у текст врши у оквиру једног модела. Ови модели, који се у великој мјери ослањају на трансформер архитектуру [3], истовремено уче акустичке и лингвистичке представе, што резултује већом робусношћу и бољом кохерентношћу превода у односу на традиционални приступ. Ипак, овакви модели захтијевају велике количине паралелних говорно-текстуалних података, који су често недоступни за језике са ограниченим ресурсима, као што је српски [1-3].

С обзиром на ове изазове, важно је анализирати како се постојећи АПГ и МП системи, како каскадни, тако и *end-to-end*, понашају на задацима превођења српског говора. Ово је посебно значајно у сценаријима гдје недостају велики паралелни корпуси и гдје се модели ослањају на мултилингвалну предобуку или *zero-shot* способности. *Zero-shot* приступ означава способност модела да извршава задатке превођења за језике или језичке парове који нису били директно обухваћени током обуке. Ова способност се заснива на мултилингвалној предобуци и заједничким семантичким репрезентацијама које омогућавају пренос знања између различитих језика. У том контексту, овај рад испитује перформансе различитих система, *Whisper* модела [4], *Google*

Translate-а и савремених модела великих језика (енг. *Large Language Model*, LLM). Посебна пажња посвећена је упоређивању директног превођења говора и каскадног приступа који комбинује транскрипцију помоћу *Whisper*-а и превођење коришћењем великих језичких модела. Циљ рада је да се идентификују предности и ограничења сваког од наведених приступа.

2. WHISPER МОДЕЛ

Whisper представља савремени модел за аутоматско препознавање и превођење говора, развијен од стране компаније *OpenAI*. Обучен је на око 680.000 часова аудио података на 99 различитих језика, прикупљених у условима слабог надгледаног учења, што му омогућава изузетну робусност и стабилне перформансе на различитим језицима, акцентима и условима снимања. Модел постоји у више конфигурација (*tiny*, *base*, *small*, *medium*, *large*, *large-v2*), које се разликују по укупном броју параметара. Мањи модели су доступни у енглеској и вишејезичној варијанти, док су највећи модели тренирани искључиво у вишејезичном режиму. Вишејезични модели подржавају и препознавање говора и директно превођење говора на енглески језик. Као *end-to-end* систем, *Whisper* интегрише препознавање језика, транскрипцију, превођење и сегментацију говора унутар једне архитектуре, што га чини једноставнијим за употребу у односу на класичне АПГ системе који се ослањају на више одвојених компоненти [4].

2.1. Архитектура модела

Whisper је заснован на кодер–декодер трансформер архитектури, која се показала изузетно успјешном у задатку секвенцијалног моделовања. Улазни аудио сигнал се најприје ресемплује на 16 kHz и претвара у 80-канални логаритамски мел спектрограм, који представља стандардна обележја у обради говорног сигнала [5]. Овај спектрограм се нормализује и служи као улаз у кодер, чији први слојеви користе плитку конволуциону структуру [6] са GELU (енг. *Gaussian Error Linear Unit*) активацијом [7] како би се смањила временска резолуција и припремио сигнал за серију трансформер блокова. У тим блоковима, уз примјену синусоидалних позиционих ембединга [8] и резидуалних веза [9], модел гради високоапстрактне и контекстуализоване репрезентације аудио садржаја.

Декодер функционише као ауторегресивни језички модел који, на основу информација из кодера и претходно генерисаних токена, производи текст корак по корак. Он се ослања на механизам пажње који омогућава ефикасно повезивање звучног контекста са језичким обрасцима. *Whisper* подржава више различитих задатака унутар једне архитектуре, па се одређени задатак моделу задаје путем специјалних токена. Свака обрада почиње токеном <|*startoftranscript*|>, након чега се додаје токен језика (нпр. <|*sr*|> или <|*en*|>), затим токен који одређује да ли модел треба да транскрибује (<|*transcribe*|>) или преводи (<|*translate*|>), као и токен <|*notimestamps*|> уколико се не предвиђају временске одреднице. На тај начин модел може флексибилно да прелази између различитих функција као што су препознавање говора, превођење, идентификација језика или детекција тишине [4].

Текст се на излазу генерише коришћењем *byte-level* BPE (енг. *Byte Pair Encoding*) токенизације [10, 11], која је прилагођена да равноправно обрађује и енглеске и вишејезичне моделе. Ова врста токенизације омогућава ефикасно кодирање различитих писама и језика, што омогућава да *Whisper* стабилно функционише и на српском језику [4].

3. ЕВАЛУАЦИОНЕ МЕТРИКЕ

Процјена квалитета система за аутоматско превођење говора захтијева употребу поузданих и уједно осјетљивих мјера које могу да обухвате лексичку, структурну и семантичку тачност превода. У овом раду примијењене су три најзаступљеније метрике у области машинског превођења и препознавања говора: BLEU, METEOR и WER. Наведене метрике међусобно се допуњују и омогућавају свеобухватну евалуацију различитих модела и приступа.

BLEU (енг. *Bilingual Evaluation Understudy*) [12] мјери сличност између генерисаног превода и референтног превода на основу преклапања n-грама. Основна претпоставка је да ће квалитетан превод садржати значајан степен преклапања са референтним преводом у виду појединачних ријечи и фраза различите дужине. Општи облик формуле гласи [12]:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \qquad (1)$$

гдје p_n представља процењат одговарајућег броја n-грама садржаног у кандидату у поређењу са референтним преводом, w_n тежинске коефицијенте (у пракси се обично рачуна као $1/N$), а BP фактор казне за краткоћу, који умањује резултат уколико је генерисани превод значајно краћи од референце.

За разлику од BLEU метрике, која даје предност лексичком преклапању, **METEOR** (енг. *Metric for Evaluation of Translation with Explicit ORdering*) тежи да обухвати семантичку сличност између превода и референце. У процесу поређења узима у обзир тачно поклапање ријечи, лематизоване облике, синониме, фразну непрекидност и редослијед ријечи [13].

Метрика се заснива на хармонијској средини прецизности и одзива, на коју се примјењује казнени фактор који умањује резултат уколико су поклапајуће ријечи распоређене у већем броју неповезаних сегмената. На тај начин METEOR кажњава фрагментоване преводе, односно преводе у којима је редослијед ријечи или структура значајно нарушена. Формално, метрика се изражава као [13]:

$$METEOR = F_{mean} \cdot (1 - Penality), \qquad (2)$$

гдје F_{mean} представља хармонијску средину прецизности и одзива, а $Penalty$ казну која расте са степеном фрагментације превода.

Ова метрика се сматра посебно погодном за језике богате морфологијом, као што је српски, јер боље препознаје сродност између различитих облика исте ријечи и мање је осјетљива на варијације у синтаксичком редослиједу.

WER (енг. *Word Error Rate*) је стандардна метрика за оцјену квалитета препознавања говора, али се може примијењивати и на анализу структурне тачности превода. Она мјери разлику између генерисаног текста и референтног текста, на основу броја извршених операција уређивања [14]:

$$WER = \frac{S + D + I}{N}, \quad (3)$$

гдје S представља замјене ријечи (substitutions), D брисања ријечи (deletions), I уметања ријечи (insertions) и N укупан број ријечи у референтном тексту [14].

4. ПРЕВОЂЕЊЕ СРПСКОГ ГОВОРА У ЕНГЛЕСКИ ТЕКСТ ПОМОЋУ WHISPER МОДЕЛА

Аутоматско превођење говора са српског језика представља изазован задатак због језичких специфичности и релативно ограничене заступљености српског у великим мултилингвалним корпусима. Иако *Whisper* спада у најнапредније моделе за препознавање и превођење говора, количина података на српском била је ограничена, што може утицати на перформансе у сценаријима без додатне дообуке.

Међутим, захваљујући свом опсежном мултилингвалном предтренингу и способности да учи опште акустичке и језичке обрасце, *Whisper* је у стању да обавља транскрипцију и превођење српског говора у *zero-shot* режиму. Управо то омогућава његово тестирање и поређење са другим системима, иако је српски био ограничено заступљен (28 сати за транскрипцију и 136 сати за превођење) у оригиналним тренинг подацима [4].

Постојећа литература показује да модели обучени на универзалним, доминантно енглеским и високоресурсним корпусима често имају потешкоћа са језицима који имају богату флексију, слободнији редослијед ријечи и специфичне морфосинтаксичке структуре, што су управо карактеристике српског језика. Због тога се у бројним студијама истиче значај адаптације модела (дообука на језички специфичним подацима или доменска адаптација), јер она значајно побољшава тачност, флуентност и робусност система за аутоматско превођење говора за нискоресурсне језике као што је српски [15-17].

4.1. Поређење постојећих метода превођења

У првој фази истраживања извршено је поређење више приступа превођењу говора са српског на енглески језик. Анализирани су *Whisper-small* (оригинални модел), *Whisper-medium* (оригинални модел), *Google Translate* (текст → текст), *GPT-5 mini* (текст → текст).

У овој поставци Whisper функционише у режиму директног превођења говора (енг. *speech-to-text translation*): модел као улаз добија аудио запис на српском језику, а затим директно генерише енглески текст, без посредног АПГ корака на српском. Овај *zero-shot* приступ омогућава процјену способности модела да преводи са језика који је био ограничено заступљен у тренинг подацима.

Евалуација је спроведена над три различита тест скупа:

- Јужне вести – 50 реченица из домена новинарског, спонтаног говора [1]
- Common Voice – 30 кратких диктираних реченица са различитим говорницима [2]
- FLEURS – 30 реченица из стандардизованог мултилингвалног корпуса [3].

Сваки скуп представља различите стилове говора (спонтани новинарски, диктирани, контролисани), што омогућава процјену робусности модела у више реалистичних сценарија.

Табела 1. Поређење различитих метода превођења – Јужне вести (50 реченица)

Model	BLEU	METEOR	WER
Whisper-small	13.88	33	80
Whisper-medium	21.74	44	72
Google Translate	29.33	59.89	68.64
GPT-5 mini	47.05	72.18	45.86

Табела 2. Поређење различитих метода превођења – Common Voice (30 реченица)

Model	BLEU	METEOR	WER
Whisper-small	18.28	32	86
Whisper-medium	24.78	47	71
Google Translate	41.97	67.19	43.70
GPT-5 mini	52.40	77.33	36.25

Табела 3. Поређење различитих метода превођења – Fleurs (30 реченица)

Model	BLEU	METEOR	WER
Whisper-small	22.46	44	78
Whisper-medium	36.47	62	56
Google Translate	66.79	88.14	24.88
GPT-5 mini	82.60	95.14	11.10

Перформансе су мјерене коришћењем BLEU, METEOR и WER метрика (Табеле 1, 2 и 3) у односу на референтни превод који је ручно креиран, прилагођен слободном и природном говору. Овај приступ омогућава објективнију и реалнију процјену квалитета машинских превода у контексту стварне употребе, избјегавајући претјерано дословне или неприродне преводе као референту вриједност. Резултати показују да оригинални *Whisper* модели постижу умјерен квалитет директног превођења, при

[1] https://www.clarin.si/repository/xmlui/handle/11356/1679?locale-attribute=en

[2] https://datacollective.mozillafoundation.org/datasets/cmflnuzw7y2kjfhed2p152qe5

[3] https://huggingface.co/datasets/google/fleurs/tree/main/data/sr_rs

чему *Whisper-medium* у свим случајевима надмашује *Whisper-small*. Ово је очекивано с обзиром на већи моделски капацитет и боље мултилингвалне репрезентације.

С друге стране, *Google Translate* и *GPT-5 mini* остварују значајно боље резултате, што се може приписати огромним паралелним корпусима на којима су обучени и њиховој способности да прецизно моделују структуру енглеског и српског језика у текстуалном домену. Ова разлика потврђује да директно *zero-shot* превођење говора представља изазов за српски, нарочито у одсуству језички специфичне обуке.

Иако *Whisper* не достиже перформансе специјализованих текстуалних система, важно је нагласити да један модел истовремено извршава препознавање говора и превођење, чиме представља ефикасну основу за даљи развој *end-to-end* система за аутоматско превођење говора на нискоресурсним језицима попут српског.

4.2. Превођење у два корака

У другој фази истраживања примјењен је каскадни приступ превођењу у два корака. Прво, *Whisper* модел је као улаз добијао аудио запис на српском језику и извршавао транскрипцију на српски. Након тога, добијени српски транскрипт је коришћен као улаз за превођење на енглески језик помоћу великог језичког модела, конкретно, *ChatGPT-4o*. Тиме је омогућена независна процјена утицаја квалитета транскрипције на коначни превод.

Тестиране су сљедеће варијанте *Whisper* модела: *Whisper-small* (оригинални), *Whisper-medium* (оригинални), *Whisper-small* (дообучен за транскрипцију на српском), *Whisper-medium* (дообучен за транскрипцију на српском).

За дообуку *Whisper-small* модела за транскрипцију на српском коришћен је *Common Voice* корпус, који садржи аудио снимке и валидиране транскрипције на српском језику. Модел је трениран у трајању од 5 епоха, уз стандардне параметре оптимизације. За дообуку *Whisper-medium* модела за транскрипцију на српском језику коришћена је комбинација тренинг скупа Јужне вести и корпуса *ParlaSpeech*. Модел је дообучен у трајању од 3 епохе. У оба случаја, током тренинга пратиле су се перформансе модела, а као метрика за одабир најбоље епохе коришћен је WER, при чему је модел са најнижом вриједношћу WER у евалуационој фази сачуван као финална верзија.

Табела 4. Поређење различитих модела за превођење у 2 корака – Јужне вести тест скуп

Model	BLEU	METEOR	WER
Whisper-small	43.58	70.39	47.41
Whisper-medium	50.36	75.9	40.87
Whisper-small finetuned	43.83	71.53	46.3
Whisper-medium finetuned	51.11	76.84	37.99

Перформасе на тест скупу „Јужне вести“ су приказане у табели 4. Анализа показује да каскадни приступ значајно надмашује директно превођење *Whisper* модела из прве фазе. *Whisper-medium* модели доследно постижу боље резултате од *Whisper-small*, што је у складу са већим капацитетом и бољом мултилингвалном репрезентацијом.

Дообука *Whisper* модела за српски АПГ резултира побољшаним WER показатељима, што директно утиче на бољи квалитет превода мјерен BLEU и METEOR метрикама. Нарочито је видљив раст METEOR резултата, који је важан за српски језик јер боље одражава морфолошка и семантичка подударања. Такође, велики језички модел показује висок степен робусности у превођењу чак и када транскрипција *Whisper* модела садржи грешке.

Овим се потврђује да је за српски језик тренутно најпогоднији каскадни АПГ → МП приступ, док директно *end-to-end* превођење још увијек није довољно зрело за високу тачност.

5. ЗАКЉУЧАК

У овом раду испитани су различити приступи аутоматском превођењу српског говора на енглески језик. Резултати показују да директно превођење *Whisper* моделима има ограничен квалитет, док каскадни приступ који комбинује *Whisper* за транскрипцију и велики језички модел (*ChatGPT-4o*) за превод даје значајно боље резултате. Дообука *Whisper* модела за српски АПГ смањује грешке у препознавању, што позитивно утиче на квалитет превода, посебно мјере METEOR која је погодна за српски језик.

Велики језички модели показују висок ниво робусности у превођењу, чак и када транскрипције садрже грешке, што истиче потенцијал за даљу интеграцију ових технологија. Тренутно, каскадни АПГ → МП приступ представља најпоузданије рјешење за српски језик у условима ограничених ресурса.

Као сљедећи кораци у истраживању препоручују се: даља дообука *Whisper* модела на већим и разноликијим српским говорним корпусима, експериментисање са различитим великим језичким моделима, развој *end-to-end* модела прилагођених специфичностима српског језика, као и креирање српско–енглеског паралелног корпуса који садржи транскрипте на српском и њихове одговарајуће енглеске преводе. Овај корпус би омогућио циљану дообуку и адаптацију модела за задатак превођења говора, што би значајно допринијело побољшању прецизности, природности и робусности система за нискоресурсне језике.

ЛИТЕРАТУРА

[1] A. B´erard, O. Pietquin, C. Servan, and L. Besacier, “Listen and translate: A proof of concept for end-to-end speech-to-text translation,” in NIPS Workshop on

End-to-End Learning for Speech and Audio Processing, 2016.

[2] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, “Sequence-to-sequence models can directly translate foreign speech,” in Interspeech. ISCA, 2017, pp. 2625–2629.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.

[4] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” arXiv preprint arXiv:2212.04356, 2022.

[5] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and trends® in signal processing*, *7*(3–4), 197-387.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016, ch. 9: “Convolutional Networks”.

[7] Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.

[8] Press, O. and Wolf, L. Using the output embedding to improve language models. In Proceedings of the 15th Conference of the European Chapter of the Associa tion for Computational Linguistics: Volume 2, Short Papers, pp. 157–163, Valencia, Spain, April 2017. As sociation for Computational Linguistics. URL https: //aclanthology.org/E17-2025.

[9] Child, R., Gray, S., Radford, A., and Sutskever, I. Gen erating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509, 2019.

[10] Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909, 2015.

[11] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

[12] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[13] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evalua-tion with improved correlation with human judgments,” in Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, 2005, pp. 65–72.

[14] A. Ali and S. Renalds, “Word Error Rate Estimation for Speech Recognition: e-WER,” in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pp. 20–24 Melbourne, Australia, July 15 - 20, 2018.

[15] V. Timmel, C. Paonessa, M. Vogel, D. Perruchoud and R. Kakooe, “Fine-tuning Whisper on Low-Resource Languages for Real-World Applications,” *arXiv preprint* arXiv:2412.15726, Apr. 2025.

[16] R. Ma, M. Qian, Y. Fathullah, S. Tang, M. Gales and K. Knill, “Cross-Lingual Transfer Learning for Speech Translation,” *arXiv preprint*, arXiv:2407.01130, Feb. 2025.

[17] P. Peng, B. Yan, S. Watanabe and D. Harwath, “Prompting the hidden talent of web-scale speech models for zero-shot task generalization,” *arXiv preprint*, arXiv:2305.11095v3, Aug. 2023.

Кратка биографија:

Милана Витковић је рођена 20.10.2001. године у Требињу. Основне академске студије је завршила у октобру 2024. године на Факултету техничких наука у Новом Саду. Мастер студије на Факултету техничких наука из области обраде сигнала уписала је 2024. године.
Контакт:
milanavitkovic2001@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.9
ДОИ: https://doi.org/10.24867/35BE24Djordjevic

Децентрализовани приступ развоју друштвених мрежа коришћењем Урбит оперативног система

Decentralized development approach of social networks using Urbit operating system

Михајло Ђорђевић, *Факултет техничких наука, Нови Сад*

Студијски програм – СОФТВЕРСКО РЕИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *Ovaj rad razmatra primenu decentralizacije u razvoju društvenih mreža, sa posebnim fokusom na Urbit tehnološki stek. Opisani su osnovni principi kriptografije i decentralizovanog identiteta, kao i uloga UrbitOS-a i UrbitID sistema u obezbeđivanju sigurne komunikacije. Rad takođe prikazuje pojednostavljenu arhitekturu decentralizovane društvene mreže zasnovane na Urbit platformi i razmatra prednosti i ograničenja ovog pristupa u odnosu na tradicionalne centralizovane modele.*
Кључне речи: *decentralizacija, društvene mreže, komunikacija, Urbit, kriptografija*

Abstract – *This paper explores the application of decentralization in social network development, with a particular focus on the Urbit technology stack. The core principles of cryptography and decentralized identity are outlined, along with the roles of UrbitOS and the UrbitID system in enabling secure communication. Additionally, the paper presents a simplified architecture of a decentralized social network built on the Urbit platform and discusses the advantages and limitations of this approach compared to traditional centralized models.*

Keywords: *decentralization, social networks, communication, Urbit, cryptography*

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

1. УВОД

Друштвене мреже су први пут популаризоване раних 2000-их, када је *MySpace* први пут постао познат. Он је омогућавао корисницима директну међусобну комуникацију преко интернет мреже као и способност сваког корисника да опишу себе путем свог профила. Из ових основних принципа настале су касније, популарније социјалне мреже као *Facebook* и *Twitter* које са својим функционалностима и лакоћом коришћења постала део свакодневнице већине људи на интернету.

Међутим, све велике функционишу на истом принципу централизације - тј. да се подаци свих њихових корисника налазе у рукама компанија које руководе овим платформама. Овакав приступ подиже питања у вези дигиталне приватности. Компаније прикупљају велике количине података о својим корисницима (које често и продају без сагласности корисника), контролишу алгоритме који диктирају који се садржај приказује и диктирају услове коришћења.
Децентрализација софтвера нуди решење овим бројним проблемима. Она представља имплементацију софтвера која се не заснива на монолитном серверу који управља пословном логиком, већ се ослања на саме кориснике - тј. сваки корисник је и сервер другим корисницима. Сваки корисник држи само своје податке, што им даје независност и потпуну контролу над својим подацима.
Овај рад нуди кратак увид у предности и мане децентрализованог приступа као и опис једног решења.

2. ДЕЦЕНТРАЛИЗАЦИЈА

Традиционалне апликације које се свакодневно користе – друштвене мреже, платформе за размену порука, сервиси за електронску пошту – функционишу кроз централизовану инфраструктуру. То значи да се сви подаци и интеракције корисника обрађују и складиште на серверима у власништву компанија попут *Mete* (*Facebook*), *Google*-а или *Twitter*-а. Овај модел има више проблема: централизована контрола над садржајем, експлоатација корисничких података, рањивост на хакерске нападе, као и могућност цензуре.
Управо због ових слабости, последњих година добија на значају концепт децентрализованих апликација (*dApps*) – софтверских решења која функционишу без централне тачке контроле, често ослањајући се на блокчејн технологију и криптографију као основне стубове безбедности и поверења.
Mastodon и *Bluesky* су међу најпознатијим примерима децентрализованих друштвених мрежа. Код обе платформе, уместо једнг централног ауторитета који

управља свим корисницима и подацима, мрежу чини више независних сервера (често названих инстанце или подови) које могу водити појединци, групе појединаца или организације. Овакав модел смањује ризик од цензуре и монопола, јер ниједан појединачни ентитет нема потпуну контролу над подацима.

2.1. Предности и мане

Овакав приступ нуди корисницима бројне погодности. Пре свега, сваки корисник добија потпуну контролу над својим садржајем на интернету. Ова особина је кључна за дигиталну приватност. Тиме се може и заобићи цензура коју би иначе диктирале приватне компаније (или пак, диктаторска влада). Са таквом дигиталном слободом произилази и слобода изражавања која је иначе веома ретка у централизованим платформама.
Међутим, чак ни овај приступ није без својих мана. Пре свега, постаје теже гарантовати конзистентно корисничко искуство. Администрација корисникових података изискује техничко знање и искуство истог корисника. Поврх тога, постаје теже контролисати спам, и модерација садржаја се пребацује на локалне администраторе. Све ово има за последицу да постаје теже регулисати илегалан садржај.

3. КРИПТОГРАФИЈА

Пракса на којој се заснива сигурност свих дигиталних система, поготово децентрализованих, се зове криптографија. То је пракса развијања и коришћења алгоритама за заштиту и прикривање пренесених информација тако да их могу читати само они који имају дозволу (ауторизацију) и могућност да их дешифрују. Другим речима, криптографија прикрива комуникације тако да им неовлашћене стране не могу приступити [1]. Организације попут Националног института за стандарде и технологију (НИСТ) развијају криптографске стандарде за безбедност података.
Криптографија се заснива на четири главна принципа: поверљивост, интегритет, непорецивост и аутентичност.
Шифроване информације морају бити приступачне само особама којима су намењене и ником другом (поверљивост).
Шифроване информације не могу бити измењене током складиштења или преноса између пошиљаоца и примаоца без да било какве измене буду откривене (интегритет).
Пошиљалац шифрованих информација не може порицати своју намеру да пошаље информације (непорецивост).
Идентитети пошиљаоца и примаоца, као и порекло и одредиште информација су потврђени (аутентичност).

3.1. Криптографске методе

Једне од најзаступљенијих криптографских метода су методе базиране на асиметричној криптографији (парови јавних и приватних кључева) - које служе сигурној размени података - и хеш функције, тј. дигитални потписи, који служе обезвеђивању интегритета података.

3.2. Јавни и приватни кључеви

Асиметрична криптографија (која се такође назива криптографија јавног кључа) користи један приватни кључ и један јавни кључ. Подаци који су шифровани јавним и приватним кључем захтевају и јавни кључ и приватни кључ примаоца да би били дешифровани [1].
Криптографија јавног кључа омогућава безбедну размену кључева преко небезбедног медијума без потребе за дељењем тајног кључа за дешифровање јер се јавни кључ користи само у процесу шифровања, али не и дешифровања. На овај начин, асиметрично шифровање додаје додатни слој безбедности јер приватни кључ појединца никада није дељен.
Овакав приступ је безбедан и робустан (нуди поверљивост, непорецивост и аутентичност података) али је притом и скупа процесорска операција

3.3. Хеш функције

Дигитални потписи и хеш функције се користе за аутентификацију и обезбеђивање интегритета података. Дигитални потпис креиран помоћу криптографије обезбеђује непорецивост, гарантује да пошиљалац поруке не може порицати аутентичност свог потписа над подацима [2].
Хеш функције, попут Сигурног хеш алгоритма 1 (SHA-1), могу трансформисати улазне податке у низ знакова фиксне дужине, који је јединствен за оригиналне податке. Ова хеш вредност помаже у провери интегритета података чинећи рачунарски немогуће да два различита уноса произведу исту хеш вредност.

4. *URBITOS* И *URBITID*

Urbit представља један технолошки стек на којем се могу имплементирати децентрализовани системи (па и друштвене мреже), и главни је фокус овог рада. Два главна ентитета овог стека су *UrbitOS* и *UrbitID*. *UrbitOS* имплементира приступ заснован на идентиету (*UrbitID*), где сваки корисник има идентитет заснован на јавном и приватном кључу. Овај идентитет омогућава корисницима да комуницирају и извршавају трансакције са потпуним поверењем. Кроз криптографске технике, *UrbitOS* елиминише потребу за традиционалним аутентификацијама, као што су лозинке или биометријски подаци, који су склони нападима и злоупотребама [3].
Мрежа *UrbitOS*-а, као децентрализована платформа, користи ове сигурносне технологије за изградњу отпорности на нападе и за очување приватности. Са сваким корисником који има контролу над својим подацима и комуникацијама, систем омогућава високи ниво сигурности, док истовремено омогућава корисницима да задрже потпуну контролу над својим информацијама.
UrbitOS је софтверски стек који чине виртуелна машина, програмски језик (*hoon*) дизајниран да подржи развој апликација на *UrbitOS* и кернел дизајниран да покреће ове апликације. Сваки уређај са

оперативним системом налик *UNIX*-у може користити *UrbitOS* [3].

UrbitOS кернел, звани Арво, само је један од модула који служе да олакшају развој апликација. Остали модули имају различите функционалности који све заједно омогућавају сигурну и ефикасну комуникацију између корисника било којих *Urbit* апликација. Они такође чине посао програмера, који развија те апликације, лакшим - јер имплицитно рукују складиштењем података, сигурном комуникацијом, подизањем локалног сервера, итд.

UrbitID је инфраструктура јавног кључа опште намене заснована на *Ethereum* блокчејну (дистрибуирана база података, тј. књига дигиталних идентитета), и користи се за *Urbit* идентитете [4]. Пружа систем оскудних и непромењљивих идентитета који су криптографски безбедни.

Постоје три важне ствари које треба разумети о целокупном дизајну *UrbitID* система: оскудност, децентрализација и контрола.

Оскудност се постиже тиме што је број могућих *UrbitID* ограничен на 2^{32} (тј. на око 4 милијарде) идентитета. Поврх тога, пошто је заснован на *Ethereum*-у, идентитет такође кошта материјално, што значи да га људи мање користе за спам или зарад злоупотребе. Када се два корисника сретну са *UrbitID*, обе стране знају да друга има неки удео у игри (чак и без откривања личних података у било ком смеру).

Децентрализација се постиже тако што се *UrbitID* се дистрибуирају преко стабла спонзорства. Сваки спонзор издаје фиксиран број адреса. Пошто постоји много спонзора, постоји много начина да се добије *UrbitID* - не само један централни ауторитет. Када се једном добије ИД, постаје трајно власништво корисника.

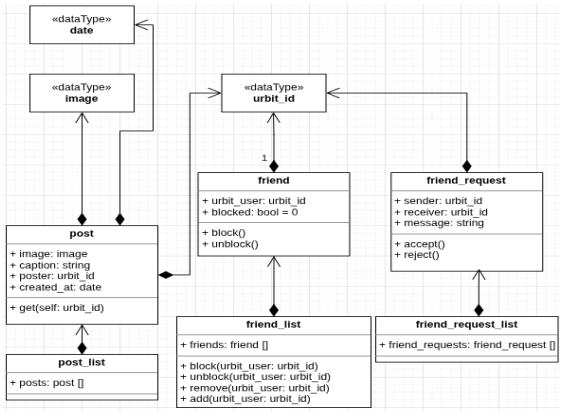
Под контролом се мисли на то да галаксије (тј. врх стабла спонзорства) формирају сенат који може да надгради логику *UrbitID* система већинским гласом.

Стабло спонзорства *UrbitID* није намењено да буде друштвени систем ни на који начин. Интеракције између људи и заједница на *Urbit* мрежи су директне (*peer-to-peer*), потпуно органске и потпуно неконтролисане од стране хијерархије адреса. *UrbitID* је једноставно аутентификациони субстрат на коме се могу градити системи за репутацију и комуникацију [4]. Због овога су се разматрали преименовања звезда и галаксија у "инфраструктурне" и "управљачке" чворове.

4.1. Развој друштвене мреже на *UrbitOS* платформи

UrbitOS је децентрализова платформа, тако да не није могуће развити социјалну мрежу, или било коју другу апликацију на традиционалан начин (нпр. *Facebook*, *Instagram*, итд). Прво, да би се уопште користио *Urbit*, потребан је криптографски индентитет, што у принципу значи да је сваки корисник већ аутентификован као посебна јединка. Самим тим, имплементација аутентификације у социјалној мрежи је изостављива. Друго, пошто сваки корисник има потпуну контролу над својим подацима, имплементација ауторизације је такође већински изостављива. И треће, пошто сваки корисник чува искључиво само своје податке, не постоји потреба за (централизованом) базом података. Ове три олакшице чине развој друштвене мреже веома директним и поједностављеним подухватом.

Имајући ово у виду тај ограниченији скуп функционалности као и непостојање потребе за ауторизацијом, аутентификацијом и базом података, читав систем се може представити веома једноставним класним диаграмом:



Слика 1. Класни диаграм свих неопходних компоненти у једној децентрализованој друштвеној мрежи

Слика 1 показује идеју најједноставнијег облика друштвене мреже, заснован на *Urbit* технолошком стеку. Корисник има своју листу пријатеља, своју листу објава и захтеве за пријатељство. Објаве других корисника (пријатеља) се добављају помоћу њихових *UrbitID*. Захтеви за пријатељство се шаљу директном претрагом *UrbitID*. Корисник у сваком тренутку мозе да приступи захтевима који су му пристигли. "Пријатељство" се успоставља када се захтев прихвати, и тада се пријатељ додаје у листу пријатеља. Из листе се пријатељ мозе избацити и/или блокирати.

И коначно, пошто се сва стања се чувају локално на машини корисника, прјатељи им могу приступити само кад их затраже. Овакав принцип рада се пропагира и на компликованије и модерније функционалности садашњих друштвених мрежа (чет, слике, видеи, сторији, итд).

Дакле, децентрализоване апликације се суштински другачије развијају од монолитних, и *Urbit* постоји као једно од могућих решења над којим се могу имплементирати.

5. ЗАКЉУЧАК

Способности *UrbitOS*-а (његових модула и *hoon* програмског језика), чине развој децентрализованих апликација (као и друштвених мрежа) брзим и једноставним. Захваљујући модулима *UrbitOS*-а, програмер не мора да брине о размени кључева између корисника, нити о сигурности комуникације.

Међутим, ни овај приступ није без својих мана. Пре свега, *hoon* програмски језик за развој апликација је компликован, тешко се савладава и нечитак је. Чак би и искусан програмер морао провести пар недеља учећи га, што га чини неприступачним. Дебагер је такође нечитак и тешко се користи што успорава развој апликација. И поврх свега, терминологија коју

Urbit користи за своје модуле и програмерске појмове је неконвенционалан што додатно отежава учење технолошког стека и самим тим развој апликација. Све заједно, *urbit* оперативни систем чини нову примитиву развоја децентрализованих апликација.

6. ЛИТЕРАТУРА

[1] Johannes A. Buchmann, *Introduction to Cryptography*, vol. 335. New York: Springer, 2004.

[2] R. L. Rivest, "Cryptography," in Algorithms and Complexity, Elsevier, 1990, pp. 717-755.

[3] ‘Urbit — Leave the internet behind’. Приступљено: Окт. 09, 2025. [Online]. Доступно: https://urbit.org/overview/urbit-os

[4] ‘Urbit — Leave the internet behind’. Приступљено: Окт. 09, 2025. [Online]. Доступно: https://urbit.org/overview/urbit-id

Кратка биографија:

Михајло Ђорђевић рођен је у Београду, општини Савски Венац,, 2000. год. Мастер рад на Факултету техничких наука из области Софтверског инжењерства и информационих технологија машине одбранио је 2025.год.

Контакт:
mihajlodj@protonmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.8
ДОИ: https://doi.org/10.24867/35BE25Sarenac

EMS систем за надзор и анализу стања и перформанси производних линија у пиварама

EMS system for monitoring and analysis of state and performance of production lines in breweries

Борјан Шаренац, *Факултет техничких наука, Нови Сад*

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Предмет овог рада је развој и имплементација EMS (Efficiency Monitoring System) система за надзор и анализу стања и перформанси производних линија у пиварској индустрији.*
Кључне речи: *анализа застоја, ефикасност производње, PLC/HMI, производне линије*

Abstract – *The subject of this paper is the development and implementation of an EMS system for monitoring and analysis of the state and performance of production lines in the brewing industry.*

Keywords: *stoppage analysis, production efficiency, PLC/HMI, production lines*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Владимир Бугарски, доцент

1. УВОД

Савремене производне линије у пиварској индустрији карактеришу високи капацитети, велики број међусобно повезаних машина и изражена динамика рада. У таквом окружењу и краткотрајни застоји, неправилности у раду или пад ефикасности појединих машина могу имати значајан утицај на укупну продуктивност линије. Због тога се јавља потреба за систематским надзором рада производних линија, прикупљањем релевантних података и њиховом анализом у реалном времену.

Класични аутоматизациони системи, засновани на PLC (Programmable Logic Controllers) контролерима и локалним HMI (Human Machine Interface) панелима, омогућавају поуздано управљање процесом, али не пружају довољно информација о узроцима застоја, губицима у ефикасности и стварним перформансама линије. Из тог разлога у савременим пиварама све већу примену налазе EMS (Efficiency Monitoring System) системи, чији је циљ праћење рада машина, анализа застоја и процена ефикасности производње.

У овом раду приказан је развој и имплементација EMS система за надзор и анализу стања и перформанси производних линија у пиварама, са посебним осврtом на интеграцију са постојећим PLC и HMI системима и примену Weihenstephan стандарда за аквизицију података.

2. ОПИС EMS СИСТЕМА

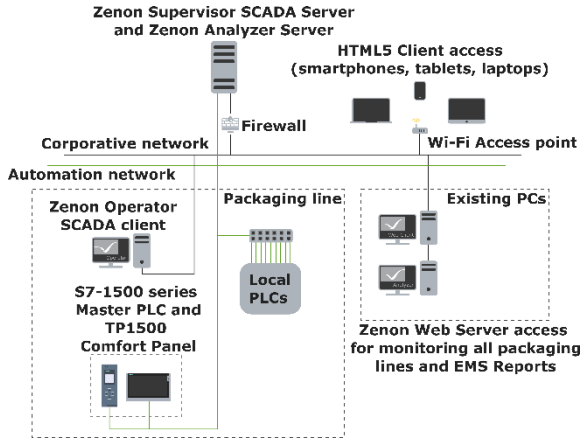
Развијени EMS систем представља надоградњу постојећег аутоматизационог система производних линија за пуњење, реализовану на начин који не утиче на основне управљачке и сигурносне функције машина. Систем је пројектован као независан слој за надзор и анализу, који користи већ доступне сигнале из PLC контролера, чиме се избегавају интервенције у постојећој логици управљања и минимизује ризик по стабилност производног процеса.

Основна намена EMS система је континуирано прикупљање података о стању рада машина, производним циклусима, застојима и њиховим узроцима, као и количини и динамици производње. Прикупљени подаци се обрађују у реалном времену у централном контролеру, а затим визуализују путем HMI и SCADA (Supervisory Control And Data Acquisition) интерфејса, што омогућава оператерима и инжењерима јасан и транспарентан увид у тренутне и историјске перформансе производне линије [1].

Посебан акценат у развоју система стављен је на стандардизацију података о раду машина применом Weihenstephan стандарда, чиме је обезбеђена униформна интерпретација стања и догађаја на различитим машинама. Оваквим приступом створена је поуздана основа за анализу ефикасности линије, идентификацију критичних тачака и доношење одлука усмерених ка смањењу застоја и унапређењу укупних перформанси производње.

2.1. Архитектура система

Архитектура система заснована је на децентрализованим локалним PLC контролерима, који прикупљају сигнале са појединачних машина, којима уједно и управљају, и централном Master PLC-у, који врши обједињавање података, логичку обраду и дистрибуцију ка HMI и SCADA нивоу. Комуникација између контролера реализована је путем индустријске PROFINET мреже и OPC UA протокола [2], што обезбеђује поуздан и временски детерминисан пренос података. На слици 1 је приказана мрежна архитектура EMS система.



Слика 1: Мрежна архитектура EMS система

2.2. Weihenstephan стандард

За стандардизацију података о раду машина примењен је Weihenstephan стандард, који дефинише јасну структуру сигнала за стања машине, производњу, застоје и узроке застоја [3]. Овим приступом омогућена је униформна интерпретација података са различитих машина и линија, као и једноставна проширивост система. Weihenstephan стандард дефинише велики број параметара које треба да поседује систем за валидну и прецизну анализу ефикасности, а за EMS систем је најбитнији параметар који означава тренутно стање машине. Тај параметар се дефинише као INTEGER и на основу његове вредности, Weihenstephan стандард препознаје стање машине. Емпиријски се показало да се може дефинисати неколико основних стања којима се слична стања могу приписати, те су та основна стања дефинисана у табели 1.

Табела 1: Стања машина по Weihenstephan стандарду

Вредност	Стање	Опис
0	Недефинисано	Стање није валидно
1	Заустављено	Машина има електричну енергију, али је у стационарном стању
8	Недостатак	Детектован недостатак материјала
16	Загушење	Детектовано загушење материјалом
32	Недостатак споредне линије	Детектован недостатак материјала на секундарној линији
64	Загушење споредне линије	Детектовано загушење материјалом на споредној линији.
128	Оперативно	Машина обавља своју предвиђену функцију
1024	Грешка опреме	Квар у самој машини
4096	Сигурносни стоп	Активирана заштитна опрема (прекидач хитног заустављања или сигурносна врата)

2.3. First Fault Cause механизам

За сваку машину на производној линији развијен је прецизан и поуздан механизам детекције под називом „First Fault Cause“. First Fault Cause је механизам који дефинише First Fault и First State параметре. Сврха овог механизма је да омогући тачну идентификацију главног узрока застоја на машини, а касније и на целој производној линији. У сложеним производним системима, у једном кратком временском интервалу може се генерисати велики број алармних и статусних порука од којих већина представља последице, а не узроке застоја. Из тог разлога неопходно је да систем забележи управо оног првог узрочника који је довео до преласка машине из оперативног у неисправно стање.
У тренутку застоја на машини, First Fault Cause механизам:

1. *Детектује прву грешку (First Fault)*
   Ово је иницијални аларм који је директно довео до тога да машина напусти стање рада. Иако се касније може појавити много других аларма, порука или последичних грешака, само је ова прва грешка релевантна за дијагностику.
2. *Бележи прво стање машине (First State)*
   Ово је прво стање у које је машина прешла након напуштања нормалног оперативног режима (нпр. грешка опреме или недостатак). Систем тиме јасно дефинише контекст застоја и омогућава јединствено тумачење узрока.
3. *Закључава прву грешку и прво стање*
   Након иницијалне детекције, First Fault и First State се не мењају, без обзира на то шта се касније дешава са машином. Машина може да генерише десетине накнадних аларма, али они више не утичу на евидентирани главни узрок застоја.
4. *Памти их док се машина не врати у оперативни режим*
   Тек када машина поново пређе у стање рада, механизам се ресетује и спреман је за нови циклус.

Јако битна напомена је да First Fault Cause механизам ради искључиво са подацима који су стандардизовани Weihenstephan стандардом.

3. ROOT CAUSE АЛГОРИТАМ

Root Cause је алгоритам који представља централни механизам EMS система за детекцију главног узрока застоја производне линије. Он је заснован на догађајима и покреће се у тренутку када систем региструје застој централне машине, односно пуњача.

3.1. Алгоритам заснован на догађајима

Алгоритам функционише на принципу обраде структура догађаја, односно реагује на догађаје и покреће потребне акције: анализу, снимање података, израчунавање узрока, евидентирање прве грешке, обраду аларма, слање података SCADA-и и слично. Основни податак којим се врши анализа система је стање машине. Да би Root Cause алгоритам посматрао нека стања као потенцијалне узрочнике застоја, неопходно је да се испуни временски критеријум догађаја приказан у изразу (**Error! Reference source not found.**):

$$Ts_m < Te_f \text{ ^ } Te_m > Ts_f \qquad (1)$$

где је:

- Ts_m- почетак стања машине,
- Ts_f- почетак стања пуњача (заправо застој)
- Te_m- крај стања машине и
- Te_f- крај стања пуњача.

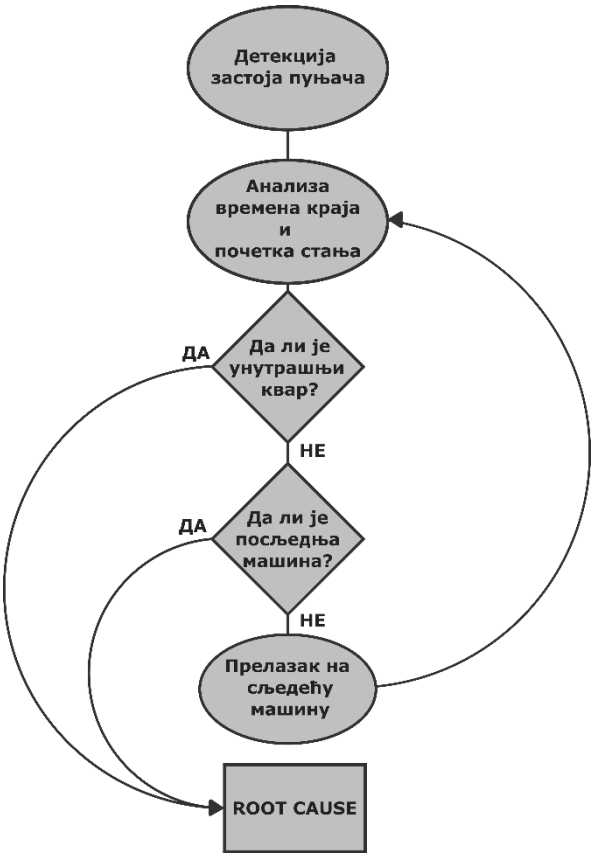
Због великих бафера на производним линијама, потребно је обезбедити пропагацију времена почетка и краја стања и у ту сврху је омогућена конфигурација времена пропагације на свакој машини унутар софтвера Master PLC-а.

3.2. Ток алгоритма

Root Cause алгоритам функционише по следећим корацима:

1. *Детекција застоја пуњача.* Када пуњач пређе из оперативног стања у неко друго стање, генерише се догађај „застој пуњача“ и активира алгоритам.
2. *Анализа времена почетка и краја.* Ова фаза се директно ослања на First Fault механизам. Ова фаза има за циљ да верификује да ли је застој стабилан, односно осигурава да се не ради о краткотрајном импулсу и да изолује тачан тренутак појаве првог стања и прве грешке.
3. *Провера да ли је застој узрокован унутрашњим кваром машине.* Овај корак важи и за пуњач и за све наредне машине. Проверава се да ли је узрок унутрашњи квар у који спадају грешке саме машине и сигурносне грешке. Уколико јесте, пуњач се проглашава за Root Cause. Уколико није, наставља се алгоритам. Када алгоритам поново дође у овај корак, само са аспекта неке друге машине, исто питање важи, и уколико постоји унутрашњи квар, та машина се проглашава за Root Cause. У спољашње кварове спадају „недостатак“ и „загушење“.
4. *Провера да ли је у питању последња машина у анализи.* Уколико застој није интерни, EMS мора да утврди да ли је пуњач ушао у загушење или у недостатак због неке друге машине. Алгоритам се креће уназад ако је пуњач у недостатку или унапред ако је пуњач у загушењу. Затим се поставља питање да ли је машина која се посматра задња у низу. Ако јесте, онда је та машина Root Cause. Ако није, алгоритам прелази на следећу машину. Морамо напоменути да се код задњих машина, као што су дуваљка или депалетизер и оматалица, као Root Cause узима и стање недостатка или загушења јер у тим ситуацијама та стања представљају валидне кандидате.
5. *Прелазак на следећу машину.* Алгоритам у овом кораку врши адаптацију претраге тако да посматрана машина постаје референтна, односно следеће машине и застоји се посматрају у односу на њу.

На слици 2 је приказан дијаграм Root Cause алгоритма.



Слика 2: Дијаграм Root Cause алгоритма

4. РЕЗУЛТАТИ И ЗАКЉУЧАК

У раду је приказан развој и имплементација EMS система за надзор и анализу стања и перформанси производних линија у пиварској индустрији. Интеграцијом EMS система са постојећим PLC и HMI решењима омогућено је прикупљање и анализа релевантних података без нарушавања основних управљачких функција система.

Применом Weihenstephan стандарда и алгоритма за анализу узрока застоја омогућена је поуздана идентификација критичних тачака у раду производне

линије и стварање основе за унапређење ефикасности и смањење губитака. Развијени систем представља флексибилно решење које се може проширивати и прилагођавати различитим типовима производних линија у пиварама.

Даљи развој система може бити усмерен ка интеграцији са Manufacturing Execution System (MES) и Enterprise Resource Planning (ERP) системима, као и примени напредних аналитичких и предиктивних метода за оптимизацију производње у смислу динамичког усклађивања брзина машина у зависности од стања линије.

Као доказ о иновацији овог система служи његова имплементација у Molson Coors пиварској корпорацији. EMS систем је заменио претходни систем у овој корпорацији. Разлог због ког је EMS успешно постао незамењив део Molson Coors-a је његова једноставност, поузданост, брзина обраде и широк опсег параметара које прати, анализира, калкулише и уређује у извештаје. Неки од параметара којима располаже су: Mean Time Between Stops (MTBS), Mean Time To Recovery (MTTR), најчешћи застоји, најпроблематичније машине, нето и бруто производња, аларми целокупне линије, стања свих машина и линије, и други. На основу ових података се даље врше калкулације и генеришу извештаји на нивоу смене, дана, недеље, месеца или године, што је од изузетног значаја за руководство пиваре.

Још један битан фактор у аутентификацији EMS система је оптимизација рада производне линије. Овај систем је обезбедио лак приступ прегледу стања линије са било ког уређаја унутар корпоративне мреже пиваре и тиме олакшао, убрзао и побољшао посао руководиоца. Самим детектовањем проблематичних сегмената производне линије се отвара могућност превентивног одржавања, ремонта или замене компонената, чиме се перформансе производње побољшавају.

Као конкретан пример побољшања рада линије служе подаци из две пиваре у којима је имплементиран EMS систем: Пивара Требјеса у Никшићу, Црној Гори и Kamenitza Brewery у Хаскову, Бугарској. У Никшићу ће се посматрати подаци на линији за пуњење PET амбалаже, а у Хаскову за пуњење буради (KEG) [4]. У табели 2 су приказани подаци из поменутих пивара и њихове промене након имплементације EMS система.

Табела 2: Побољшање параметара производне линије употребом EMS система

	PET	KEG
MTBS	+19%	+14%
MTTR	-9%	-5%
Број застоја	-12%	-8%
Учесталост микро застоја	-22%	-20%
Лажне детекције	-28%	-37%
Време реакције оператера	-11%	-10%
Време дијагностике квара	-37%	-35%

5. ЛИТЕРАТУРА

[1] L.P.P. Muchiri, "Performance measurement using Overall Equipment Effectiveness (OEE)", *International Journal of Production Research,* Vol. 46(13), pp. 3517-3535, July 2008.

[2] W. Mahnke, S.-H. Leitner, M. Damm, "*OPC Unified Architecture*", Ladenburg, Springer, 2009.

[3] H. Weisser, "*Standard Specifications for Data Acquisition Systems in Beverage Bottling Plants*", Technische Universität München, 2010.

[4] http://indasautomation.com/solutions/efficiency-monitoring-system/ (pristupljeno u decembru 2025.)

Кратка биографија:

Борјан Шаренац рођен је у Љубовији 2000. год. Мастер рад на Факултету техничких наука из области Рачунарство и аутоматика – Аутоматика и управљање системима одбранио је 2025. год.
Контакт: borjan0410@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.41
ДОИ: https://doi.org/10.24867/35BE27Jovic

Генератор кода за имплементацију иницијалне верзије микросервисне апликације

Code Generator for Implementing the Initial Version of a Microservice Application

Никола Јовић, Факултет техничких наука, Нови Сад

Студијски програм – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *У овом раду представљен је алат за аутоматско генерисање иницијалне верзије микросервисне апликације на основу YAML спецификације. Рад обухвата анализу постојећих решења у области аутоматизованог генерисања микросервиса, као и опис архитектуре и имплементације алата реализованог као плагин за IntelliJ IDEA развојно окружење. Приказане карактеристике и функционалности алата указују на његову применљивост у иницијалној фази развоја микросервисних апликација.*

Кључне речи: *генератор кода, микросервисна апликација, YAML*

Abstract – *This paper presents a tool for automated generation of the initial version of a microservice application based on the YAML specification. The paper includes an analysis of existing solutions in the field of automated microservice generation, as well as a description of the architecture and implementation of the tool developed as a plugin for the IntelliJ IDEA development environment. The presented features and functionalities of the tool indicate its applicability in the initial phase of microservice application development.*

Keywords: *code generator, microservice application, YAML*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је била др Гордана Милосављевић, редовни професор.

1. УВОД

Развој микросервисних апликација представља један од кључних приступа у савременом софтверском инжењерству који омогућава независан развој појединачних компоненти система, бољу скалабилност и једноставније одржавање [1]. Креирање микросервисне апликације у почетној фази развоја често подразумева велики број понављајућих корака, као што су подешавање конфигурације пројекта и имплементација основних слојева апликације. Ови задаци могу значајно успорити развој и повећати могућност настанка грешака услед ручног писања кода.

Наведени проблеми се превазилазе употребом генератора кода који омогућавају убрзавање развоја и унапређење стандардизације кода. Узимајући у обзир наведене изазове, циљ овог рада је развијање новог алата који би пружао аутоматизовано генерисање почетне микросервисне апликације и тиме програмерима омогућио бржи, једноставнији и ефикаснији процес развоја микросервисних апликација.

2. ПРИКАЗ СТАЊА У ОБЛАСТИ

Алати за генерисање кода и аутоматизован развој представљају иновативан корак ка унапређењу процеса развоја софтвера. У оквиру академских истраживања развијени су алати засновани на формалним моделима, као што су *FSMicroGenerator* и *Zynerator*. *FSMicroGenerator* користи модел коначних аутомата за формално описивање понашања система, након чега се из модела генерише изворни код микросервисне апликације [2]. Овај приступ обезбеђује јасну структуру и доследност, али је ограничен у погледу проширивости и применљивости на комплексније системе. *Zynerator* је заснован на архитектури вођеној моделом и омогућава трансформацију апстрактних доменских модела у изворни код [3]. Он обезбеђује добру поновну употребу и стандардизацију, али његова примена захтева додатно познавање моделских концепата.

У индустрији постоје решења која програмерима нуде практичне платформе са високим степеном аутоматизације, као што су *JHipster* и *Micronaut Launch*. *JHipster* омогућава генерисање комплетне микросервисне апликације [4], али уз велику сложеност и зависност од великог броја технологија. *Micronaut Launch* пружа брзо генерисање појединачних сервиса који прате праксе *Micronaut* фрејмворка и имају добре перформансе [5], али не подржава централизовано генерисање потпуне микросервисне апликације.

Анализа постојећих решења указује на потребу за новим алатом који ће комбиновати једноставност употребе, практичну применљивост и лаку интеграцију у развојно окружење, уз могућност генерисања микросервисне апликације на основу претходно дефинисане спецификације. Овим би се

убрзао развој, смањио понављајући рад и могућност појаве грешака.

3. ПРОЈЕКТОВАЊЕ АЛАТА *MICROGENERATOR*

Пројектовани алат треба да обезбеди брзо, једноставно и поуздано генерисање почетне верзије микросервисне апликације на основу формално дефинисаног модела. Основу рада алата представља декларативна *YAML* спецификација. У оквиру ње се дефинишу основни метаподаци микросервиса, конфигурациони параметри, зависности, подешавања базе података и доменски модели.

3.1. Функционалности алата

Алат омогућава избор и учитавање *YAML* датотеке, врши валидацију њене структуре и садржаја, као и аутоматско генерисање *Spring Boot* пројекта за сваки дефинисани микросервис. На основу описаних доменских модела алат генерише основне компоненте апликације, укључујући ентитете, репозиторијуме, сервисе и *REST* контролере са ендпоинтима за креирање, читање, измену и брисање података.

Поред функционалних, алат мора испуњавати и нефункционалне захтеве који обухватају: да је имплементиран као *IntelliJ IDEA* плагин, да је структура креираних датотека и класа стандардизована и усклађена са праксама *Spring Boot* екосистема и да је дизајниран тако да се у будућности може надоградити и унапредити новим шаблонима и функционалностима.

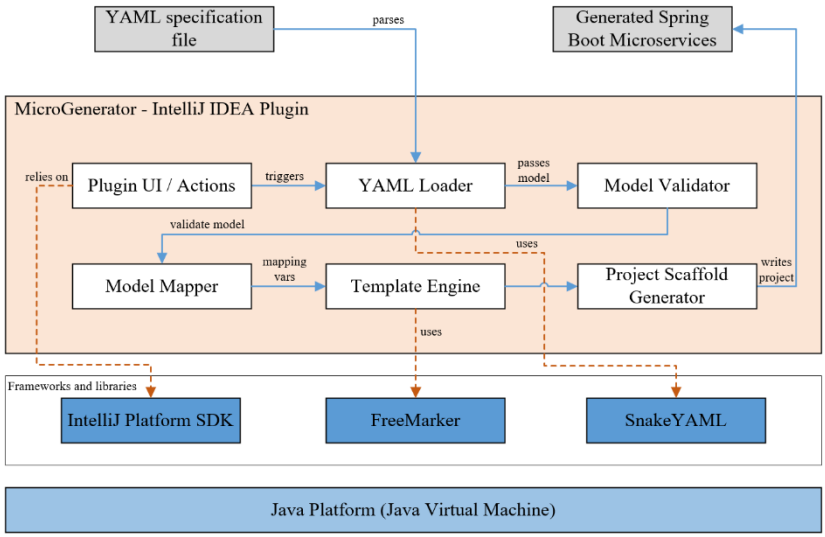
3.2. Архитектура алата

Дизајн архитектуре алата заснован је на принципима слојевите организације, одвајања одговорности и јасној подели улога за сваку компоненту у систему. Централни део архитектуре чине модули који реализују ток обраде, од корисничког интерфејса, преко учитавања и валидације *YAML* спецификације, до трансформације у интерни модел и генерисања изворног кода.

Кориснички интерфејс представља улазну тачку за покретање процеса генерисања. Он прикупља параметре и прослеђује их одговарајућем модулу на даљу обраду. Оваква интеграција у развојно окружење елиминише потребу за инталацијом посебног екстерног алата. Након избора *YAML* датотеке, активира се модул задужен за учитавање спецификације који парсира садржај и трансформише га у интерну репрезентацију. У случају детектованих грешака кориснику се приказује порука да је потребно исправити улазну спецификацију.

Циљ модула за трансформацију је претварање садржаја интерне репрезентације у структуре података које представљају домен микросервиса. Генерисање изворног кода се реализује коришћењем механизма шаблона. Шаблони дефинишу структуру свих делова микросервиса: модела, репозиторијума, сервисног слоја, контролера и конфигурација.

Модул за генерисање кода, након што се шаблони попуне подацима, добијени текстуални садржај записује у датотеке на диску. Ова компонента је одговорна за креирање целокупне структуре директоријума *Spring Boot* пројекта, креирање *Java* класа у одговарајућим пакетима, формирање *Maven* конфигурације и креирање конфигурационих датотека.



Слика 1. Архитектура алата *MicroGenerator*

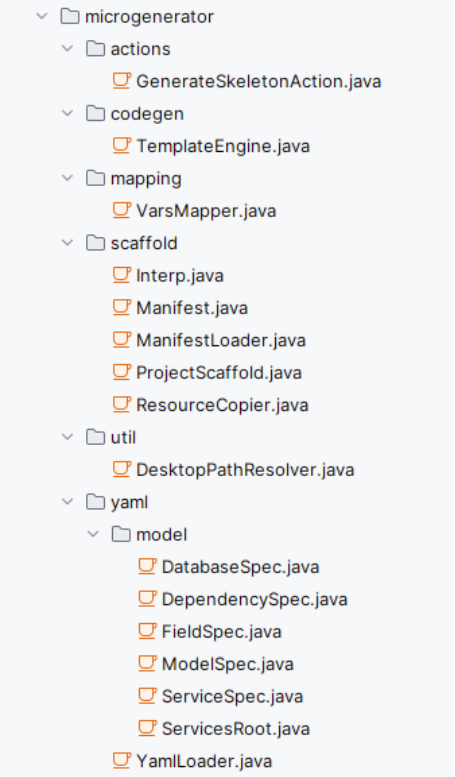
4. ИМПЛЕМЕНТАЦИЈА

Имплементација предложеног решења реализована је кроз развој алата у виду плагина за развојно окружење *IntelliJ IDEA*. Овакав приступ омогућава директну интеграцију у окружење које програмери већ користе, чиме се алат природно уклапа у постојећи радни процес, омогућава његова примена без нарушавања уобичајеног тока развоја и избегава потреба за инсталацијом додатног софтвера.

4.1. Структура пројекта и имплементација модула

Структура пројекта организована је модуларно, са јасном поделом одговорности између појединих компоненти. Улазну тачку представља корисничка акција која се покреће из развојног окружења. Ова

компонента преузима контекст активног пројекта, иницира избор *YAML* спецификације и покреће даљи ток обраде, и на тај начин је интеракција корисника са алатом сведена на мали број једноставних корака.



Слика 2. Структура пакета алата *MicroGenerator*

Учитавање и парсирање спецификације реализовано је у оквиру посебног модула задуженог за рад са улазним моделом. За обраду *YAML* формата коришћена је библиотека *SnakeYAML*, која омогућава директно пресликавање структуре спецификације у типизиране *Java* објекте [6]. Оваквим приступом обезбеђена је типска сигурност и поједностављена валидација улазних података. Основне провере, постојање обавезних секција и исправност структуре, извршавају се приликом самог учитавања спецификације чиме се спречава рад са неконзистентним улазом.

Након успешног учитавања, интерни модел се трансформише у структуру података погодну за механизам шаблона. Овај корак је реализован у посебном слоју за мапирање који има улогу да из улазне спецификације преузме све параметре неопходне за генерисање кода. Тиме је постигнуто раздвајање логике интерпретације од логике шаблона, чиме постигнута је боља одрживост алата.

Генерисање изворног кода засновано је на употреби *FreeMarker* шаблона, који дефинишу структуру генерисаних *Java* класа и конфигурационих датотека [7], док се конкретне вредности уносе на основу параметара припремљених у претходној фази. Шаблони су организовани тако да постоје они који се генеришу за сваки доменски модел појединачно и они који се генеришу једном по микросервису. Део датотека се не генерише из шаблона, већ се копирају као статички ресурси у пројекат.

Креирање физичке структуре микросервисне апликације реализовано је у посебном модулу који је задужен за оркестрацију процеса генерисања директоријума и уписа датотека на диск. Структура пројекта је дефинисана у конфигурационом манифесту, који описује које се датотеке генеришу, које се копирају и где се смештају у оквиру пројекта. Овакав приступ омогућава да генерисана микросервисна апликација буде у потпуности усклађена са *Maven* и *Spring Boot* конвенцијама и да је спремна за покретање.

4.2. *YAML* спецификација

YAML спецификација представља централни улазни артефакт алата и служи за декларативни опис микросервисне апликације. Спецификација омогућава дефинисање једног или више микросервиса, њихових зависности, конфигурације базе података и доменских модела. Доменски модели се описују именима, називима табела и поља, на основу чега се аутоматски генеришу одговарајући *JPA* ентитети. Подржано је и моделовање релација између ентитета, који се током генерисања пресликавају на одговарајуће *JPA* асоцијације.

4.3. Изазови током имплементације

Током имплементације уочено је више техничких и концептуалних изазова. Један од кључних је био дефинисање *YAML* спецификације која је довољно флексибилна да опише различите микросервисе, а истовремено довољно једноставна за коришћење. То је превазиђено увођењем јасно дефинисане хијерархијске структуре и типизираних моделских класа.

Важан имплементациони изазов је био избегавање уношења пословне логике у шаблоне за генерисање кода. Уколико би шаблони садржали сложене услове и трансформације, њихово одржавање и проширивање би било отежано. Проблем је решен увођењем засебног слоја за мапирање података, који припрема све неопходне параметре пре фазе генерисања.

Обезбеђивање доследне и исправне структуре генерисане микросервисне апликације је постигнуто увођењем конфигурационог манифеста који дефинише структуру пројекта, распоређивање датотека и начин њиховог креирања.

5. АНАЛИЗА РЕЗУЛТАТА

Анализа резултата развоја алата *MicroGenerator* усмерена је на процену његових квалитативних атрибута и поређења са постојећим решењима за аутоматизовано генерисање микросервисних апликација.

5.1. Квалитативни атрибути алата *MicroGenerator*

Један од кључних квалитативних атрибута алата је употребљивост, која произилази из његове директне интеграције у развојно окружење. Корисник може да покрене процес генерисања без напуштања окружења у коме већ ради што поједностављује радни ток и смањује потребу за додатним алатима. Употреба *YAML* спецификације као улазног формата омогућава декларативан опис микросервисне архитектуре, чиме се смањује сложеност конфигурационих корака.

Проширивост алата остварена је кроз модуларну архитектуру и јасну поделу одговорности између компоненти за обраду спецификације, мапирање података и генерисање кода.
Оваква организација омогућава додавање нових шаблона, увођење подршке за додатне технологије или проширење улазне спецификације без значајних измена постојећег кода.
Одрживост решења обезбеђена је кроз модуларну организацију и коришћење шаблона за генерисање кода. Измене у начину рада алата могу се реализовати без негативног утицаја на остале компоненте система.
Перформансе алата посматрају се у односну на његову улогу у иницијалној фази развоја микросервисних апликација, при чему је процес учитавања спецификације и генерисања кода довољно ефикасан и не утиче негативно на стабилност развојног окружења.

5.2. Поређење са постојећим алатима

У поређењу са постојећим алатима, *MicroGenerator* заузима специфичну позицију између академских прототипова и индустријских решења. За разлику од алата заснованих на формалним моделима, као што су *FSMicroGenerator* и *Zynerator*, предложено решење не захтева употребу специјализованих моделских језика, већ се ослања на *YAML* као широко прихваћен формат.
У односу на комплексна индустријска решења, попут *JHipster*-а, *MicroGenerator* нуди једноставнији приступ усмерен на аутоматизацију почетне фазе развоја, без увођења великог броја конфигурационих корака. За разлику од *Micronaut Launch*-а који омогућава генерисање појединачних сервиса, *MicroGenerator* подржава централизовано генерисање комплетне микросервисне апликације.
MicroGenerator није директна замена за постојеће алате, али његово коришћење декларативног описа микросервисне архитектуре и интеграција у развојно окружење издвајају га као алат за убрзање почетне фазе развоја микросервисних апликација.

6. ЗАКЉУЧАК

У овом раду представљен је процес пројектовања, имплементације и евалуације алата *MicroGenerator*, намењеног аутоматизованом генерисању иницијалне верзије микросервисне апликације на основу *YAML* спецификације.
Кроз анализу постојећих академских и индустријских решења уочена су ограничења која се односе на сложеност употребе, потребу за формалним моделима или недостатак централизованог приступа генерисању микросервисних апликација. На основу тога развијен је алат који комбинује декларативни приступ опису архитектуре, једноставност употребе и директну интеграцију у развојно окружење *IntelliJ IDEA*. Имплементација алата као плагина омогућила је практичну примену у реалним развојним сценаријима.
Анализа резултата показује да *MicroGenerator* успешно испуњава постављене циљеве, јер смањује понављајући рад, повећава стандардизацију кода и убрзава развој у почетној фази изградње микросервисних апликација. Представљено решење пружа основу за даља унапређења, што може бити проширење улазне спецификације или подршка за напредне инфрастуктурне компоненте, чиме се отвара простор за будући развој алата и његову примену.

7. ЛИТЕРАТУРА

[1] N. Dragoni, S. Giallorenzo, A. Lluch Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, “Microservices: Yesterday, Today, and Tomorrow,” Present and Ulterior Software Engineering, Springer, 2017, pp. 195-216. doi: 10.1007/978-3-319-67425-4_12.

[2] S. Khalfaoui, H. Khalfaoui, and A. Azmani, “Microservices-Driven Automation in Full-Stack Development: Bridging Efficiency and Innovation With FSMicroGenerator,” IEEE Access, vol. 13, pp. 125131-125156, 2025. doi: 10.1109/ACCESS.2025.3589285.

[3] Y. Zouani, and M. Lachgar, “Zynerator: Bridging Model-Driven Architecture and Microservices for Enhanced Software Development,” Electronics, vol. 13, no. 12, art. 2237, 2024. doi: 10.3390/electronics13122237.

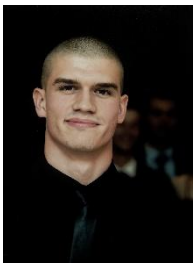
[4] JHipster, “JHipster Official Website.” Available: https://www.jhipster.tech (accessed: Oct. 10, 2025).

[5] Micronaut, “Micronaut Launch.” Available: https://micronaut.io/launch (accessed: Oct. 10, 2025).

[6] SnakeYAML, “SnakeYAML Project.” Available: https://bitbucket.org/snakeyaml/snakeyaml (accessed: Nov. 30, 2025).

[7] Apache, “FreeMarker.” Available: https://freemarker.apache.org/index.html (accessed: Nov. 30, 2025).

Кратка биографија:



Никола Јовић рођен је у Лозници 1999. године. Основне академске студије завршио је на Војној академији у Београду. Мастер рад на Факултету техничких наука у Новом Саду, смер Софтверско инжењерство и информационе технологије, одбранио је 2026. године.

Контакт:jovnikola99@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 621.39
ДОИ: https://doi.org/10.24867/35BE28Milanovic

GPS напади са фокусом на GPS spoofing напад на UAVs (дронове) уз анализу open source GPS пријемника

GPS attacks with a focus on GPS spoofing attacks on UAVs (drones), including the analysis of open-source GPS receiver

Александра Милановић, *Факултет техничких наука, Нови Сад*

Студијски програм – ЕНЕРГЕТИКА, ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ

Кратак садржај – *Глобални навигациони сателитски системи (GNSS) широко се користе у навигацији беспилотних летелица, али су због отворене природе сигнала изразито подложни нападима. Овај рад се бави GPS spoofing нападима на UAV системе, анализом структуре GPS сигнала и принципима успешног spoofing напада. Разматрају се савремене методе детекције spoofing напада, као и анализа open-source GNSS SDR пријемника. На крају су анализирани и објашњени добијени експериментални резултати, а рад може да буде полазна тачка за комплекснија истраживања ових напада.*

Кључне речи (три до пет): *UAV, дрон, spoofing напад, GPS пријемник*

Abstract – *Global Navigation Satellite Systems (GNSS) are widely used in UAV navigation but are highly vulnerable to attacks due to the open nature of their signals. This thesis focuses on GPS spoofing attacks on UAV systems, analyzing the structure of GPS signals and the principles behind successful spoofing. Modern spoofing detection methods are reviewed, and the analysis of an opensource GNSS SDR receiver is presented. Finally, the obtained experimental results are analyzed and discussed. The thesis can serve as a starting point for more complex research into these attacks.*

Keywords: (three to five): *UAV, drone, spoofing attack, GPS receiver*

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

1. УВОД

Глобални навигациони сателитски системи (GNSS) постали су основни механизам који се користи приликом одређивање положаја, као и за навигацију и временску синхронизацију у савременом технолошком окружењу. Њихова улога превазилази класичне навигационе примене и данас обухвата и финансијске системе, енергетске мреже, телекомуникационе инфраструктуре, системе транспорта и целокупан спектар аутономних платформи [1]. Посебно значајну примену GNSS је нашао у области беспилотних ваздухопловних система (UAV), односно дронова, где сигнал навигационог система представља не само извор информације о положају, већ и један од основних елемената стабилизације и управљања летелицом.

UAV платформа, без обзира на степен аутономије, обично користи GNSS у комбинацији са инерцијалним сензорима (IMU) и барометром. GNSS обезбеђује апсолутне координате и временску референцу, док IMU даје информације о угаоним и линеарним брзинама. Контролни алгоритми унутар пилотског рачунара (flight controller) непрестано упоређују податке из ових извора. Уколико GNSS информација постане нетачна или непоуздана, долази до нарушавања равнотеже у систему управљања и тада, UAV може да пређе у нестабилан режим лета, да скрене са задате путање или чак да постане потпуно неконтролисан [2].

2. ГЛОБАЛНИ НАВИГАЦИОНИ САТЕЛИТСКИ СИСТЕМИ (GNSS)

GNSS представља општи појам који обухвата сателитске системе за пружање услуга позиционирања, навигације и времена (PNT, односно енг. Positioning, Navigation & Time), заснованих на сигналима који се емитују из свемира. Ради смањења зависности од једног система, више држава развија и одржава сопствене GNSS системе, који се разликују управо по области покривености коју обухватају, као и по својим оперативним могућностима.

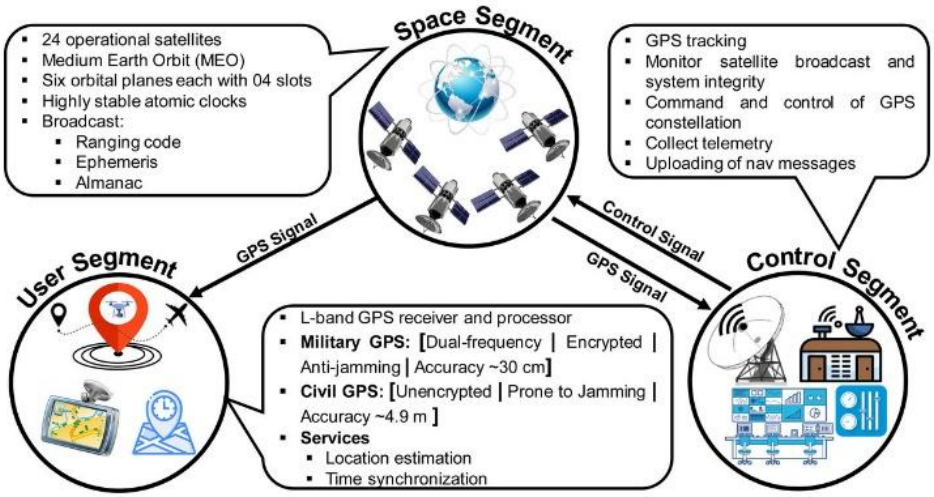
2.1. Преглед GNSS система

GNSS представља заједнички термин који се користи за више независних сателитских навигационих система развијених у различитим земљама. Најзначајнији су следећи: GPS (Сједињене Америчке Државе), GLONASS (Руска Федерација), Galileo (Европска Унија), BeiDou (Народна Република Кина).

2.2. GPS сегменти

UAV, за разлику од већине уређаја, не само да чита GNSS податке, већ их активно користи за стабилизацију управљања, па је тачност GNSS-а директно повезана са безбедношћу лета [3].

Управо због тога, може се рећи да UAV представља једну од најосетљивијих платформи на GPS spoofing нападе. На Слици 1 следи јасан приказ свих GPS сегмената, односно свемирског (Space Segment), корисничког (User Segment) и контролног сегмента (Control Segment) [1]:



Слика 1. GPS сегменти [1]

3. C/A PRN КОД И КОРЕЛАЦИОНО ОТКРИВАЊЕ

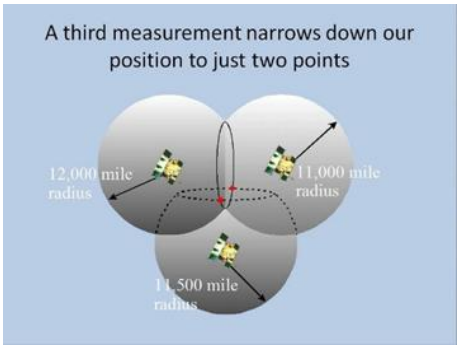
GPS L1 C/A сигнал је CDMA сигнал – сваки сателит има свој PRN код. PRN код је низ од 1023 чипа (chips), који се понавља и то на сваких 1 ms. Пријемник врши корелацију између примљеног сигнала и локалне реплике кода. Корелација показује колико су сигнали „поклопљени“. Када је корелациони максимум висок, пријемник закључује да је сателит „пронађен“ и након тога он улази у фазу праћења. Прецизније, све је изражено на основу следеће формуле (1):

$$R_{fd}(\tau, f_d) = \sum_n x[n] \cdot c[n-\tau] \cdot e^{-j2\pi f_d n t_s} \qquad (1)$$

Где је $x[n]$ примљени RF сигнал, $c[n]$ локална копија PRN кода, τ представља кашњење (кодна офсет-вредност), а f_d је доплерова фреквенцијска промена. **Spoofer мора да направи сигнал са истим PRN кодом**, иначе пријемник уопште неће размотрити сигнал као легитиман сигнал.

Иако у практичној GPS навигацији учествују најмање четири сателита, геометријска суштина поступка се може најјасније приказати на примеру три сфере, што је приказано на Слици 2 [5]. Свака сфера представља могуће положаје UAV-а у односу на један сателит, односно сва места у простору где псеудо-растојање може бити испуњено.

Пресек три сфере одређује потенцијалну позицију корисника у 3D простору. У стварном GPS-у, међутим, време емисије сигнала није савршено познато, што захтева и додатно увођење четвртог псеудо-растојања. Како UAV слепо верује математичком моделу, свака контролисана промена псеудо-растојања директно помера израчунату позицију.

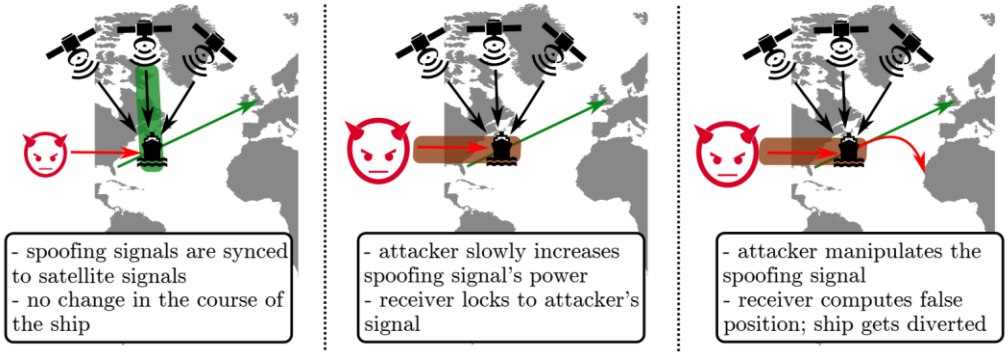


Слика 2. Пресек три сфере: геометријска интерпретација трилатерације [5]

4. ВРСТЕ GPS НАПАДА И SPOOFING НА UAV СИСТЕМИМА

GPS нападима називамо све активности које имају за циљ да омету, наруше или преусмере процес позиционирања корисника. Начелно се напади деле у две основне групе: ометање (jamming) и превара (spoofing). Док jamming има за циљ да потпуно онемогући пријем сигнала, spoofing покушава да пријемнику подметне лажне информације, али тако да сигнал и даље изгледа привидно исправан. Управо је spoofing критичан када је реч о аутономним летелицама (UAV), јер оне у навигацији и стабилизацији лета не користе визуелну референцу Земље, већ искључиво рачунају положај из матричних и временских улазних података [2][3].

Уколико се лажни сигнал појави као **континуитет** онога што је пријемник већ пратио, пријемник ће постепено прећи на лажни сигнал без губитка праћења [2]. Ово се назива **seamless takeover**. Напад је приказан на Слици 3 [4]:



Слика 3. Seamless takeover [4]

4.1. Фазе извођења напада у пракси

Spoofer не „управља дроном“, он заправо управља његовим мерењима и дрон онда сам управља погрешно. Ово је оно што чини spoofing опаснијим од jamminga. GPS spoofing не почиње наглим деловањем. Напротив, најуспешнији напади одвијају се по строго планираном редоследу који је раније објашњен у поглављу 4 детаљније, што сумирано, кратко и јасно приказује Табела 1:

Табела 1. Фазе GPS spoofing напада

ФАЗА	НАЗИВ	ОПИС
1	Поравнање (Alignment)	Спуфер слуша легитимни сигнал и усклађује PRN код, носилац и доплер.
2	Преузимање (Takeover)	Спуфер повећава снагу и полако помера кодну фазу, тада UAV почиње да прати лажни пик.
3	Вођење, манипулисање (Steering)	Спуфер наводи UAV путању ка жељеним координатама.

Суштина напада је у другом кораку, односно у уласку у петље праћења. Ово је тачка где UAV више не зна да ли прати прави или лажни сигнал [4].

5. МОДЕРНЕ МЕТОДЕ ДЕТЕКЦИЈЕ GPS SPOOFING НАПАДА

Иако је GPS spoofing у основи напад који делује на физичком слоју RF сигнализације, савремени GNSS пријемници развијају механизме за откривање неправилности који се ослањају на анализу корелаторских излаза, праћење (tracking) петљи (DLL/PLL), анализу временских серија и, у неким случајевима, интеграцију са инерцијалним сензорима. За UAV платформе, посебно у аутономном режиму, поуздана детекција је критична јер се GNSS често користи као примарни извор апсолутне позиције и времена [2][3]. Три представничка приступа који су често цитирани у академској литератури су следећи: **SPREE (MobiCom 2016)** као пример детекције засноване на помоћним корелационим пиковима [4], **SemperFi (NDSS 2022)** као anti-spoofing пријемнички приступ усмерен на UAV платформе [1], и **A-CoRLiAn, Aggregated Correlation Residue Likelihood Analysis**, као метод заснован на корелационим резидуалима и likelihood одлучивању са агрегирањем одлука преко више сателита, за коју је у [6] приказана висока детекција и за emerging TEXBAT ds7 сценарио. Кључна надоградња код A-CoRLiAn је агрегирање одлука више сателита, што побољшава свеукупну робусност. У [6] је наведено да агрегирани приступ обезбеђује „одрживо упозорење“ за ds7 током 289 секунди и да достиже 99.66% стопу детекције за ds7, при чему се наглашава да је побољшање последица агрегирања одлука више сателита.

6. АНАЛИЗА OPEN-SOURCE GNSS SDR ПРИЈЕМНИКА

За реализацију експеримента изабран је отворени пројекат GNSS-SDR, који је написан у C++ и заснован је на GNU Radio инфраструктури. GNSS-SDR нуди модуларну архитектуру, у којој су блокови за аквизицију, праћење, декодирање навигационе поруке и PVT (Position, Velocity, Time) имплементирани као засебне компоненте које се конфигуришу преко текстуалних .conf фајлова. Ово омогућава да се лако мења извор сигнала (физички SDR пријемник или снимљени фајл), бирају различите имплементације аквизиције и праћења, као и да се, по потреби, убаце неки нови, сопствени блокови за анализу или детекцију spoofing-а.

Као оперативни систем коришћена је дистрибуција Ubuntu 22.04 LTS, на стандардном x86-64 лаптопу, што представља типично и једноставно окружење. Не користи се физички SDR хардвер (попут RTL-SDR или USRP уређаја), већ се као улаз користи већ снимљени GNSS сигнал у облику I/Q узорака. На тај начин се максимално поједностављује хардверски део поставке, добија потпуно контролисан и познат сценарио и омогућава да све варијације у будућим експериментима потичу искључиво од софтверских измена (нпр. увођење spoofing-а или различитих алгоритама детекције) [7], [8] .

Важно је осигурати то да су све библиотеке (GNU Radio, VOLK, Boost и др.) присутне у верзијама

компатибилним са GNSS-SDR-ом. GNSS-SDR се ослања на библиотеку VOLK (Vector-Optimized Library of Kernels), која садржи оптимизоване имплементације основних векторских операција (множење, конволуција, ротација комплексних бројева и сл.). VOLK може да користи различите SIMD инструкционе сетове процесора (SSE, AVX, AVX2, AVX-FMA), а на свакој платформи је потребно једнократно профилисање како би се за сваки тип операције изабрала најбржа имплементација. [8] Профилисање се врши командама volk_profile и volk_gnsssdr_profile, које покрећу серију тестова за различите имплементације сваког језгра и бележе у конфигурационе фајлове који SIMD пут је најбржи на конкретном процесору. На овај начин значајно се убрзава аквизиција и праћење сигнала, посебно када се обрађују велике количине I/Q података.

6.1. Преузимање референтног GNSS снимка (CTTC SPAIN)

За потребе валидирања SDR пријемника изабран је јавни сет података који садржи 100 секунди снимака GPS L1 C/A сигнала, снимљених у истраживачком центру CTTC у Шпанији. Сет података се преузима са званичног хостинга пројекта GNSS-SDR. [9] Прво је креиран посебан радни директоријум, а затим, снимак се преузима помоћу wget алата. Иако архива садржи оригинални конфигурациони фајл, он је писан за старије верзије GNSS-SDR-а и не садржи нови глобални параметар GNSS-SDR.internal_fs_sps. Због тога је формиран нови конфигурациони фајл my-first-GNSS-SDR-receiver.conf, који је логички заснован на оригиналном, али је морао бити и прилагођен верзији 0.0.20 и новој путањи до .dat фајла. Кључни елементи конфигурационог фајла су GNSS-SDR секција, SignalSource секција, SignalConditioner блок који врши конверзију из short у комплексни број као и ресемплинг и Acquisition, Tracking, TelemetryDecoder, Observables и PVT блокови који заједно чине комплетан ланац пријемника: и то од проналажења сателита, преко праћења кода и носиоца, па све до израчунавања позиције, брзине и времена.

6.2. Анализа логова и добијених резултата

Лог GNSS-SDR-а показао је да се израчуната позиција креће око: географске ширине $Lat \approx 41.2748° N$, географске дужине $Long \approx 1.9877° E$, висине између приближно *60 m* и *80 m* над референтним елипсоидом. Ова локација одговара подручју у околини Барселоне (Шпанија), што је у складу са описом CTTC dataset-а. Позиције у времену показују постепене, мале варијације, што је очекивано за статичног пријемника са неизбежним шумом и мањим варијацијама у геометрији сателита током *100 секунди*. Нема драстичних скокова у координатама, нити прекида у PVT решењу, GNSS-SDR успешно прати више канала, а навигационе поруке се успешно и прилично уредно декодирају. Пријављене вредности *C/N0* (Carrier–to–Noise density ratio) крећу се углавном у опсегу *44–48 dB-Hz*, што представља солидне услове пријема за L1 C/A сигнал. Ово значи да софтверски пријемник има довољно добар однос сигнал/шум. Поред текстуалног лога на стандардном излазу, у директоријуму ~/gnss-test GNSS-SDR генерише и више типова излазних фајлова. KML и NMEA фајлови су посебно корисни јер омогућавају интуитивну проверу да ли је позиција „на правом месту“, независно од интерних детаља GNSS-SDR-а.

7. ЗАКЉУЧАК

У овом раду разматрана је безбедност GPS навигације са фокусом на, веома популарне у данашње време, GPS spoofing нападе на беспилотне летелице (UAV). Савремени дронови у огромној мери зависе од GNSS позиционирања, а при томе користе исте, историјски креиране сигнале и протоколе који уопште нису дизајнирани са безбедношћу на уму. GPS пријемник, гледано „изнутра“, не види сателите као физичке објекте на небу, већ као скуп корелационих врхова и навигационих фрејмова који долазе у релативном времену. Цела архитектура оригинално је дизајнирана за поузданост и робусност, али не и за аутентичност, што отвара простор за различите облике spoofing-а. UAV-ови су специфично атрактивне мете, јер, они често лете аутономно, без сталне линије вида са оператором, и у великој мери верују GNSS-у приликом навигације, стабилизације и што се тиче failsafe логике.

SPREE, SemperFi и A-CoRLiAn показују три различита приступа и то од мониторинга корелационих пикова (SPREE), преко „вишка структуре“ у пријемнику и елиминације сумњивих грана (SemperFi), до статистичке анализе резидуалних корелација на реалним TEXBAT сценаријима (A-CoRLiAn).

Резултати анализе су показали стабилно позиционирање у области CTTC центра у Шпанији, са *географском ширином* око *41.27°N* и *дужином* око *1.99°E*, *висином* у опсегу *60–80 m* и C/N_0 вредностима око *44–48 dB-Hz*. Број коришћених сателита кретао се између пет и седам, а PVT решење није показивало драматичне скокове нити некакве прекиде. Описано је како се, полазећи од отворених алата и доступних сетова података, може изградити истраживачко окружење. У том смислу, рад представља полазну тачку за сваки даљи наставак истраживања на раскрсници GNSS технологија, сајбер-безбедности и аутономних летелица.

8. ЛИТЕРАТУРА

[1] Sathaye, H., LaMountain, G., Closas, P., & Ranganathan, A. “SemperFi: Anti-Spoofing GPS Receiver for UAVs.” *Network and Distributed System Security Symposium (NDSS), 2022.*

[2] Psiaki, M. L., & Humphreys, T. E. “GNSS Spoofing and Detection.” *Proceedings of the IEEE*, *2016.*

[3] Khan, S. Z., Mohsin, M., & Iqbal, W. “On GPS Spoofing of Aerial Platforms: A Review of Threats, Challenges, and Countermeasures.” *PeerJ Computer Science, 2021.*

[4] Ranganathan, A., Ólafsdóttir, H., & Capkun, S. “SPREE: A Spoofing Resistant GPS Receiver.” *MobiCom*, *2016.*
[5] Askari, H., & Barekat, F. “Intersection of Three Spheres.” *ResearchGate, 2017.*

[6] Jiang, W., Wang, D., Zeng, Z., Liu, J., & Xu, B. “Advanced GNSS Spoofing Detection: Aggregated Correlation Residue Likelihood Analysis.” *Remote Sensing*, *2024.*

[7] Fernández-Prades, C., Arribas, J., Closas, P., Avilés, C., & Esteve, L. “GNSS-SDR: An Open Source Tool for Researchers and Developers.” *Proc. ION GNSS 2011, Portland, OR, 2011.*

[8] GNSS-SDR Project, “GNSS-SDR – An open-source Global Navigation Satellite Systems software-defined receiver.” https://gnss-sdr.org

[9] GNSS-SDR Project, “GNSS-SDR – An open source software-defined GNSS receiver.” SourceForge, https://sourceforge.net/projects/gnss-sdr/

Кратка биографија:

Александра Милановић рођена је у Руми 1996. год. Дипломски рад на Факултету техничких наука, из области саобраћаја и телекомуникација, са фокусом на управљање пројектима, одбранила је 2022. год. Након тога, одлучила се за мастер академске студије из области енергетике, електронике и телекомуникација.

Контакт:
alekscacam@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.4
ДОИ: https://doi.org/10.24867/35BE29Unkovic

Управљање сертификатима у модерним индустријским системима

Certificate Management in Modern Industrial Systems

Јелена Унковић, *Факултет техничких наука, Нови Сад*

Студијски програм – ПРИМЕЊЕНО СОФТВЕРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *Овај рад обрађује концепт управљања дигиталним сертификатима у индустријским системима, са фокусом на хибридна рјешења која комбинују локалну и cloud инфраструктуру. Кроз теоријске основе, архитектуре, аутоматизацију и мониторинг, приказује се како се сертификатима може ефикасно управљати у сложеним техничким окружењима. Студија случаја производне фабрике и имплементација система за управљање сертификатима илуструју практичну примјену концепта. Као закључак, рад доноси свеобухватну анализу предности хибридног приступа у погледу безбједности и оперативности.*

Кључне ријечи: *дигитални сертификати, хибридно рјешење, cloud, on-premise, SCADA*

Abstract – *This thesis explores the concept of digital certificate management in industrial systems, with a focus on hybrid solutions that combine on-premises and cloud infrastructure. Through theoretical foundations, architectural models, automation, and monitoring, it demonstrates how certificates can be efficiently managed in complex technical environments. A case study of a metal processing factory and the implementation of a custom Certificate Management solution illustrate the practical application of the concept. In conclusion, the thesis provides an analysis of the security and operational advantages of the hybrid approach.*

Keywords: *digital certificates, hybrid solution, cloud, on-premise, SCADA*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Бојан Јелачић, доцент

1. УВОД

Индустријски контролни системи (*Industrial Control Systems – ICS*), укључујући систем за надзор и прикупљање података (*Supervisory Control and Data Acquisition - SCADA*) и савремену индустријску мрежу паметних уређаја (*Industrial Internet of Things - IIoT*), представљају основу модерне производње и инфраструктуре. Са све већом повезаношћу *ICS* система са мрежама информационих технологија (*Information Technology - IT*) и интернетом расту и сајбер ризици. За разлику од класичне *IT* безбједности, *ICS* нагласак ставља на доступност и интегритет, док застарјела опрема и дуг животни циклус отежавају примјену савремених механизама. Традиционални локални модели (*on-premise*) нуде контролу, али су ограничени у скалабилности, док рјешења у рачунарском облаку (*cloud*) доносе аутоматизацију уз изазове усклађености. Хибридни приступ, који комбинује локалну инфраструктуру и *cloud* сервисе, намеће се као оптимално рјешење. Циљ рада је приказати савремене приступе управљању дигиталним сертификатима (*certificate management*) у индустријским системима, са фокусом на хибридна рјешења и практичну имплементацију у погону за обраду метала.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1 Дигитални сертификати

Дигитални сертификати су кључни механизам за заштиту комуникације и интегритет података у дигиталном окружењу. Њихова основна улога је аутентикација идентитета, обезбјеђивање интегритета путем дигиталних потписа и повјерљивости кроз енкрипцију. У индустријским системима користе се за аутентикацију уређаја, контролу приступа *SCADA* и човјек-машина интерфејсима (*Human-Machine Interface - HMI*), као и за потписивање софтвера. Најчешће се примјењује формат *X.509* [1], који дефинише кључне елементе сертификата [2].

2.2 Компоненте *PKI* система

Инфраструктура јавног кључа (*Public Key Infrastructure – PKI*) омогућава безбједну размјену информација коришћењем асиметричне криптографије, повезујући јавни кључ са идентитетом субјекта путем дигиталног сертификата који издаје поуздано сертификационо тијело (*Certificate Authority – CA*). *PKI* обезбјеђује аутентикацију, енкрипцију и дигитално потписивање, а *CA* има кључну улогу у верификацији идентитета, издавању сертификата и управљању њиховим животним циклусом. Како број сертификата расте, користи се хијерархијска структура са главним (*root*) и подређеним (*subordinate*) *CA*, уз механизме провјере валидности као што је протокол за провјеру статуса сертификата (*Online Certificate Status Protocol – OCSP*) или путем листе опозваних сертификата (*Certificate Revocation List – CRL*) [3].

2.3 Типови сертификата

Дигитални сертификати се класификују према намјени и нивоу повјерења, а најчешће категорије су сервер сертификати за безбједну комуникацију путем протокола *Transport Layer Security / Secure Sockets Layer (TLS/SSL)*, клијентски сертификати за аутентикацију корисника и уређаја, сертификати за потписивање кода (*code-signing*), те сертификати за потписивање и енкрипцију електронске поште (*e-mail*) [4]. Ове врсте сертификата обезбјеђују аутентикацију, интегритет и повјерљивост комуникације, као и заштиту од неауторизованих измјена софтвера и напада посредника („*man-in-the-middle*“).

2.4 Рок трајања, обнављање и опозив сертификата

Рок трајања дигиталног сертификата дефинише временски период у којем је сертификат важећи. Кратки рок повећава безбједност, али захтијева аутоматизацију процеса обнављања путем система за управљање животним циклусом сертификата (*Certificate Lifecycle Management – CLM*). У случају компромитације или промјене статуса, сертификати се опозивају коришћењем механизама *CRL* и *OCSP* за провјеру валидности у реалном времену [5].

2.5 Регулаторни захтјеви

Управљање дигиталним сертификатима у индустријским окружењима мора бити усклађено са регулаторним оквирима који дефинишу техничке стандарде и процесе безбједности. Кључне регулативе укључују директиву за критичну инфраструктуру *(Network and Information Security Directive 2 – NIS2*), за информациону безбједност (*International Organization for Standardization / International Electrotechnical Commission 27001 – ISO/IEC 27001*), за индустријску аутоматизацију (*International Electrotechnical Commission 62443 – IEC 62443*) и за безбједну обраду података (*General Data Protection Regulation – GDPR*).

2.6 Управљање великим бројем сертификата

У савременим индустријским системима број дигиталних сертификата може достићи стотине или хиљаде због раста IIoT уређаја, микросервиса и потребе за безбједном комуникацијом. Ручно управљање постаје неодрживо, па су неопходни скалабилни системи који омогућавају аутоматизовано издавање, обнављање и опозив сертификата, уз интеграцију са платформама за континуирану интеграцију и испоруку (*Continuous Integration / Continuous Deployment – CI/CD*) и *cloud* сервисима.

3. КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ

У развоју софтверског рјешења за управљање дигиталним сертификатима коришћене су модерне технологије које обезбјеђују сигурну имплементацију. *Backend* је развијен у *.NET Core* платформи, *frontend* у *React* библиотеци, док је *Azure Key Vault* коришћен за безбједно управљање сертификатима у *cloud* окружењу и *Microsoft Entra ID* за аутентикацију корисника. Ова комбинација омогућава високе перформансе, сигурну комуникацију и интеграцију са савременим безбједносним протоколима.

4. ТИПОВИ АРХИТЕКТУРЕ ЗА УПРАВЉАЊЕ СЕРТИФИКАТИМА

4.1 *On-premice* рјешење

On-premise рјешења за управљање дигиталним сертификатима подразумевају да се сви елементи *PKI* инфраструктуре, укључујући *CA*, базе података, политике и алате за мониторинг, налазе унутар локалне мреже организације, што омогућава потпуну контролу над безбједносним процесима. Овај приступ је погодан за индустријска окружења са високим захтјевима за приватност и усклађеност са стандардима, али захтијева значајне ресурсе за имплементацију, одржавање и обуку особља.

4.2 *Cloud* рјешење

Cloud рјешења за управљање дигиталним сертификатима ослањају се на сервисе попут *Azure*, *Amazon Web Services (AWS)* и *Google Cloud*. Интеграција са софтверским алатима који омогућавају аутоматизацију, интеграцију и надзор процеса развоја, тестирања и испоруке апликација (*DevOps*) и *CI/CD* процесима обезбјеђује континуитет безбједности, док сервиси као што су *Azure Key Vault* и *AWS* менаџер сертификата нуде централизовано управљање, ротацију кључева и аудит активности.

4.3 Хибридно рјешење

Хибридна архитектура управљања сертификатима комбинује локална и *cloud* рјешења како би организацијама омогућила контролу над критичним компонентама уз коришћење флексибилности и скалабилности које пружају *cloud* сервиси. Сертификати за интерне системе издају се и чувају локално, док се они за апликације које комуницирају са екстерним сервисима ефикасно управљају кроз *cloud* алате. Постојећа рјешења која се користе у хибридним моделима укључују *Microsoft Intune* у комбинацији са *ADCS* (*Active Directory Certificate Services*), *AWS* менаџер сертификата повезан са *Windows ADCS*, *HashiCorp Vault* интегрисан са *Azure Key Vault*-ом, *Enterprise Java Beans Certificate Authority* (*EJBCA*) *Hybrid PKI* за индустријске системе, као и *CertAccord Enterprise* који омогућава аутоматизовано управљање сертификатима. Ова рјешења потврђују да хибридни модел није теоријски концепт, већ практичан приступ који омогућава постепену миграцију ка *cloud*-у, оптимизацију ресурса, смањење трошкова и повећање отпорности система на безбједносне инциденте [6].

5. МЕХАНИЗМИ И АЛАТИ ЗА АУТОМАТСКО УПРАВЉАЊЕ СЕРТИФИКАТИМА

Аутоматизација процеса издавања, обнављања и опозива сертификата повећава безбједност, смањује ризик људске грешке и омогућава скалабилност система, уз коришћење специјализованих механизама и алата.

5.1 *ACME* протокол

ACME (*Automatic Certificate Management Environment*) је стандардизовани протокол за потпуно аутоматско управљање дигиталним сертификатима од генерисања захтјева (*Certificate Signing Request – CSR*), верификације, издавања до аутоматског обнављања. Развијен од стране *Internet Security Research Group (ISRG)* за *Let's Encrypt*, *ACME* смањује људске грешке, убрзава *TLS* имплементацију и омогућава скалабилност у динамичним окружењима. Валидација се врши преко *HyperText Transfer Protocol – 01 (HTTP-01)* или *Domain Name System – 01 (DNS-01)* механизама [7].

5.2 Аутоматизација издавања, обнављања и опозива

Аутоматско издавање сертификата обухвата генерисање *CSR*-а са јавним кључем, верификацију идентитета и издавање без мануелне интервенције, користећи алате као што су *HashiCorp Vault*, *Microsoft ADCS* и *ACME* протокол. Сертификати се обнављају помоћу заказаних задатака (*cron job*), механизама за аутоматско обавјештавање преко веба (*webhook*) или алата попут *Certbot*-а. У случају компромитације, аутоматизовани опозив ажурира *CRL* или *OCSP* у реалном времену, уз подршку алата као што су *Vault*, *ADCS* и *Security Information and Event Management (SIEM)* системи.

5.3 Интеграција са *CI/CD* алатима

У *CI/CD* окружењима сертификати обезбјеђују аутентикацију, потписивање артефаката и сигурну комуникацију током „*build*“ и „*deployment*“ процеса, уз аутоматизацију кроз алате попут *Jenkins* и *GitHub Actions*.

5.4 Аутоматизација у *cloud* окружењу

Cloud платформе као што су *Microsoft Azure, AWS* и *Google Cloud* нуде сервисе за управљање сертификатима без сопствене *PKI* инфраструктуре. *Azure Key Vault* омогућава генерисање и обнављање сертификата, *AWS* менаџер сертификата аутоматски издаје *SSL/TLS* сертификате, а *Google Cloud* менаџер сертификата подржава аутоматско управљање.

6. ЦЕНТРАЛИЗОВАНО УПРАВЉАЊЕ И МОНИТОРИНГ

6.1 Централна тачка за издавање и ревокацију

Централизована тачка за издавање и опозив сертификата обезбјеђује досљедну примјену правила, праћење и брзу реакцију на инциденте. Рјешења попут *Microsoft ADCS* и *HashiCorp Vault* омогућавају аутоматско управљање сертификатима уз интеграцију са *Active Directory*, *cloud* сервисима и *DevOps* алатима.

6.2 Мониторинг и аудитинг сертификата

Мониторинг прати рок важења, открива неауторизоване сертификате и валидира конфигурацију, док аудитинг биљежи и анализира све активности ради форензике и транспарентности. Интеграција са *SIEM* системима омогућава централизовано прикупљање логова и корелацију догађаја за већу безбједност.

6.3 Алати за управљање животним циклусом сертификата

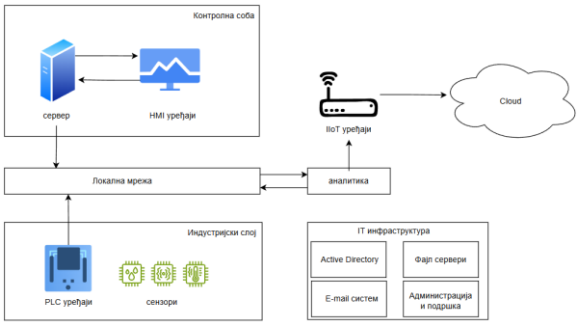
Алати за управљање сертификатима као што су *Venafi*, *Keyfactor* и *Certbot* аутоматизују издавање, обнављање, опозив и надзор. *Venafi TLS Protect* и *SSH Protect* нуде напредну ротацију и валидацију, *Keyfactor* управља *PKI* инфраструктуром уз подршку за *cloud* и *DevOps*, док је *Certbot* алат отвореног кода (*open-source*) за *ACME* протокол, идеалан за мање системе.

7. АРХИТЕКТУРА ХИБРИДНОГ РЈЕШЕЊА ЗА УПРАВЉАЊЕ СЕРТИФИКАТИМА ЗА ПОТРЕБЕ ПОГОНА ЗА ОБРАДУ МЕТАЛА

Избор хибридне архитектуре комбинује локални *CA* (*Microsoft ADCS*) за управљање сертификатима у затвореним мрежама оперативних технологија (*Operational Technology – OT*) и *Azure Key Vault* за *cloud* окружење. Овакав приступ омогућава централизовано, аутоматизовано и безбједно управљање сертификатима за *SCADA, HMI,* програмабилни логички контролер (*Programmable Logic Controller – PLC*) и *IIoT* уређаје, уз смањење трошкова и ризика. Систем подржава аутентикацију, енкрипцију комуникације, контролу приступа и аудит у складу са стандардима као што је *IEC 62443*.

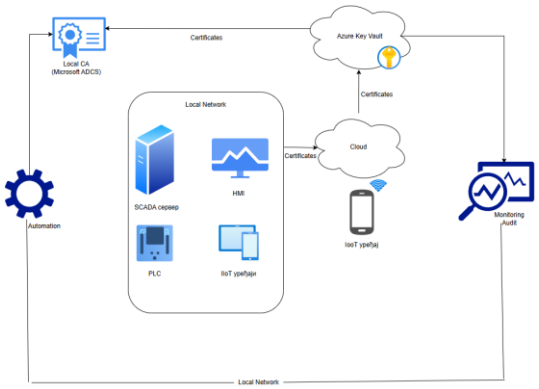
7.1 Опис индустријског система

Студија случаја приказује средње велику фабрику за обраду метала која користи *SCADA* систем за надзор и управљање производњом. Техничка архитектура система укључује *SCADA* сервер, *HMI* терминале, *PLC* уређаје, *IIoT* сензоре, локалну индустријску мрежу и *IT* инфраструктуру, што је приказано на Слика 1.



Слика 1. Техничка архитектура система

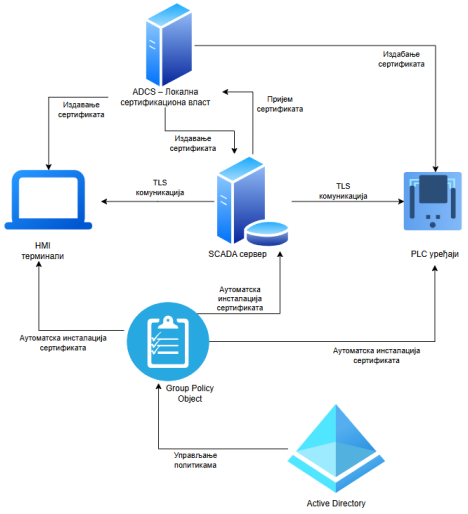
За управљање дигиталним сертификатима примјењена је хибридна архитектура која комбинује *Microsoft ADCS* за *SCADA*, *HMI* и *PLC* компоненте, и *Azure Key Vault* за *IIoT* уређаје и *cloud* сервисе. Токови сертификата, аутоматизација процеса и надзор у реалном времену приказани су на Слика 2.



Слика 2. Архитектура система

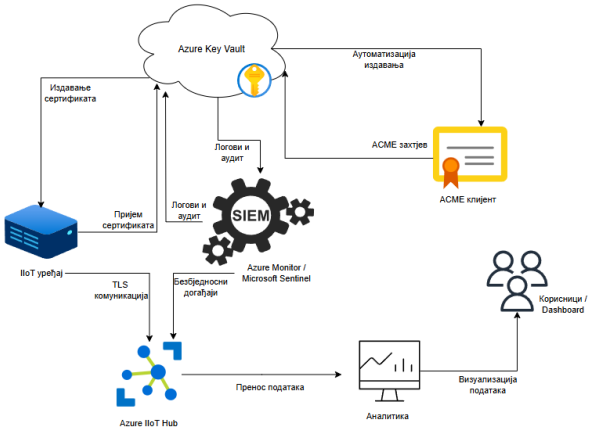
7.2 Архитектура хибридног рјешења за управљање сертификатима

У циљу обезбјеђивања безбједне комуникације и управљања идентитетима у сложеном индустријском окружењу, имплементирано је хибридно рјешење које комбинује локалну инфраструктуру (*Microsoft ADCS*) и *cloud* сервисе (*Azure Key Vault*), омогућавајући централизовано, аутоматизовано и скалабилно управљање сертификатима за *SCADA, HMI, PLC* и *IIoT* уређаје. Слика 3 приказује токове сертификата у локалној мрежи, укључујући издавање путем *ADCS*-а, дистрибуцију преко *Group Policy* објеката (*GPO*) и *PowerShell* скрипти, те аутентикацију корисника кроз *Active Directory*.



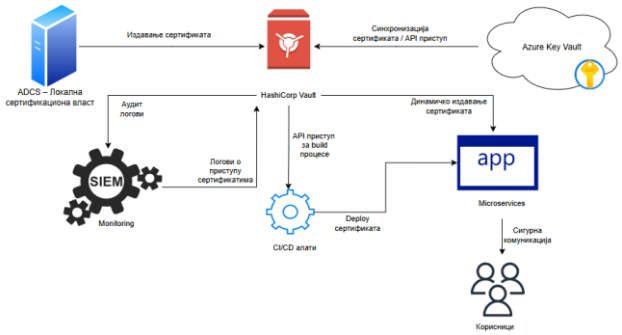
Слика 3. Архитектура локалне компоненте

За *cloud* компоненту, Слика 4 илуструје како *Azure Key Vault* управља сертификатима за *IIoT* уређаје, омогућава аутоматско издавање путем *ACME* протокола и интеграцију са *Azure IoT Hub*-ом, уз мониторинг кроз *Azure Monitor* и *Microsoft Sentinel*.



Слика 4. Архитектура *cloud* компоненте

Слика 5 приказује интеграциони слој са *HashiCorp Vault*-ом који повезује локални *CA* и *cloud* сервисе, омогућава динамичко издавање сертификата за микросервисе и *CI/CD* алате, те централизовану евиденцију и аудит. Аутоматизација се додатно подржава алатима као што је *Certbot*, скриптама и *webhook*-овима, док *SIEM* системи омогућавају надзор и детекцију инцидената.



Слика 5. Архитектура интеграционог слоја

Резултати имплементације укључују смањење броја инцидената, повећану ефикасност тимова, усклађеност са безбједносним стандардима и спремност система за будуће ширење.

8. ИМПЛЕМЕНТАЦИЈА ХИБРИДНОГ РЈЕШЕЊА ЗА УПРАВЉАЊЕ СЕРТИФИКАТИМА

У оквиру мастер рада развијено је софтверско рјешење засновано на *.NET Core* и *React* технологијама које омогућава хибридну архитектуру за управљање дигиталним сертификатима, комбинујући локалну инфраструктуру и *cloud* компоненту *Azure Key Vault*. Систем обезбјеђује генерисање, управљање, складиштење и дистрибуцију сертификата уз подршку за реалне индустријске сценарије, а архитектура је подијељена на *backend* који обрађује логику сертификата и *frontend* који пружа интерактивни интерфејс за управљање. На Слика 6 приказан је централни дио апликације са прегледом сертификата и акцијама као што су обнова, опозив, извоз и слање у *Azure Key Vault*.

Слика 6. Интерфејс апликације

Слика 7 показује *Azure* портал на коме је сертификат успјешно синхронизован, што потврђује исправност интеграције и безбједан пренос података.

Слика 7. Приказ сертификата на *Azure Key Vault*-у

Имплементација је довела до смањења инцидената везаних за истекле сертификате, повећања ефикасности *IT* и *OT* тимова, усклађености са безбједносним стандардима и спремности система за проширење на нове производне линије и *cloud* сервисе.

9. ПРЕДНОСТИ ХИБРИДНОГ РЈЕШЕЊА

Имплементација хибридног система за управљање дигиталним сертификатима комбинује локалну контролу преко *Microsoft ADCS*-а и *cloud* флексибилност *Azure Key Vault*-а. Рјешење засновано на *.NET* и *React* технологијама омогућава централизовано и аутоматизовано управљање сертификата, уз минималну ручну интервенцију. Систем доноси безбједносне предности као што су *TLS* енкрипција, јединствени сертификати и брза реакција на инциденте, као и оперативне предности попут скалабилности, интеграције са *DevOps* алатима и централизованог надзора. На Табела 1 приказана је компаративна анализа кључних параметара различитих рјешења за управљање дигиталним сертификатима, а плавом бојом је означено рјешење описано у овом раду.

Табела 1. Компаративна анализа хибридних рјешења

Критеријум	Описано рјешење	Microsoft Intune + ADCS	AWS ACM + ADCS	HashiCorp Vault + Azure KV	EJBCA Hybrid PKI	CertAccord Enterprise
Локални CA	ADCS	ADCS	ADCS	Опционо	EJBCA	ADCS
Cloud сервис	Azure Key Vault	Microsoft Cloud PKI	AWS ACM + HSM	Azure Key Vault	AWS/Azure/GCP	Опционо
Интеграциони слој	HashiCorp Vault	Не	Не	Vault као примарни	Опционо	Не
Аутоматизација	Висок - ACME, CI/CD	Средњи - Intune, GPO	Средњи - API	Висок - API, ACME	Средњи	Средњи
DevOps интеграција	Да	Ограничено	Ограничено	Да	Да	Не
OT/SCADA фокус	Да	Не	Не	Не	Ограничено	Не
SIEM/Мониторинг	Да - Sentinel	Основни	Основни	Основни	Основни	Основни
IoT подршка	Да	Ограничено	Да	Да	Да	Ограничено
Усклађеност	IEC 62443, NIS2	GDPR	GDPR	GDPR	IEC 62443	GDPR

Представљено рјешење се истиче интеграцијом *OT/SCADA* и *IIoT* компоненти, напредном аутоматизацијом кроз *ACME* и *CI/CD* процесе, мониторингом и *SIEM* интеграцијом, као и усклађеношћу са *IEC 62443* и *NIS2*.

10. ЗАКЉУЧАК

У овом раду обрађена је тема управљања дигиталним сертификатима, са фокусом на хибридна рјешења. Дигитални сертификати играју кључну улогу у аутентикацији, енкрипцији и контроли приступа у *SCADA* и *IIoT* системима. Аутоматизација управљања сертификатима, интеграција са *CI/CD* алатима и примјена *DevSecOps* приступа постају стандард, док се истовремено развијају рјешења за динамичко управљање у изворним *cloud* окружењима (*cloud-native*) и аутоматизовану усклађеност са регулативама. *DevSecOps* омогућава безбједност кроз све фазе развоја. Будућа истраживања усмјерена су ка отпорности на квантне пријетње, анализи ризика помоћу вјештачке интелигенције, стандардизацији протокола и проширењу функционалности апликација.

11. ЛИТЕРАТУРА

[1] https://www.sectigo.com/blog/x509-certificate-management (pristupljeno u aprilu 2022.)

[2] https://www.okta.com/identity-101/digital-certificate/ (pristupljeno u avgustu 2024.)

[3] A. Albarqi, E. Alzaid, F. Al Ghamdi, S. Asiri, J. Kar, "Public Key Infrastructure: A Survey", *J. Inf. Secur.*, vol. 6, pp. 31–37, January 2015.

[4] N. Ricchizzi, C. Schwinne, J. Pelzl, "Applied Post Quantum Cryptography: A Practical Approach for Generating Certificates in Industrial Environments", *arXiv preprint*, May 2025.

[5] N. Korzhitskii, N. Carlsson, "Revocation Statuses on the Internet", *Proc. of Passive and Active Measurement Conference (PAM 2021)*, pp. 175–191, April 2021.

[6] https://learn.microsoft.com/en-us/intune/intune-service/protect/microsoft-cloud-pki-deployment (pristupljeno u aprilu 2025.)

[7] D. A. Cordova Morales, A. S. Wazan, D. W. Chadwick, R. Laborde, A. R. R. Maramara, K. Cabral "Enhancing the ACME Protocol to Automate the Management of All X.509 Web Certificates", *IFIP Adv. Inf. Commun. Technol.*, vol. 679, pp. 265–278, April 2024.

Кратка биографија:

Јелена Унковић рођена је у Бањој Луци 2000. год. Мастер рад на Факултету техичких наука из области Електротехнике и рачунарства - Примењено софтверско инжењерство одбранила је 2026. год.
Контакт: jelenaad26@gmail.com