Automatizirani sustav prikupljanja podataka o objektima održavanja primjenom IIoT senzora

Curman, Martin

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:485182

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Martin Curman

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Dragutin Lisjak

Student:

Martin Curman

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru, prof. dr. sc. Dragutinu Lisjaku na ukazanom povjerenju, stručnim savjetima i podršci. Također, zahvaljujem i komentoru Davoru Kolaru, dr. sc., na kontinuiranoj podršci, savjetima, pomoći i motiviranju tijekom izrade rada.

Veliku zahvalu posvećujem svojoj obitelji i prijateljima, posebno roditeljima koji su bili uz mene u svim trenucima, kontinuirano me podržavali i vjerovali u mene. Na kraju, posebnu zahvalu upućujem svojoj djevojci Ivi Oroz na bezuvjetnoj ljubavi i podršci.

Martin Curman

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602 - 04 / 21 - 6 / 1	
Ur. broj: 15 - 1703 - 21 -	

DIPLOMSKI ZADATAK

Student: **MARTIN CURMAN** Mat. br.: 0035206381

Naslov rada na hrvatskom jeziku: **Automatizirani sustav prikupljanja podataka o objektima održavanja primjenom IIoT senzora**

Naslov rada na engleskom jeziku: **Automated system for collecting maintenance facilities data using IIoT sensors**

Opis zadatka:

Industrija 4.0 je koncepcija koji podrazumijeva automatizaciju i razmjenu podataka u industrijskim postrojenjima primjenom tzv. "pametnih" senzora. Unutar koncepcije Industrije 4.0 javlja se i pojam Održavanje 4.0, a pod istim se podrazumijeva povezivanje objekata održavanja u informacijsko-komunikacijsku mrežu s ciljem prikupljanja i analize podataka te učinkovitijeg upravljanja procesom održavanja. Primjenom tehnologije Industrijskog interneta stvari (eng. Industrial Internet of Things - IIoT), moguće je automatizirano udaljeno prikupljanje podataka o objektima održavanja, a na temelju njihove analize i donošenje odluka o potrebnim aktivnostima održavanja. Primjena IIoT senzora omogućava održavateljima lakšu procjenu trenda tj. "zdrastvenog" stanja udaljenog objekata održavanja, a time i planiranje potrebnih preventivnih aktivnosti održavanja.

U skladu s prethodno navedenim, u radu je potrebno:

1. Opisati koncepciju Industrije 4.0 i Održavanja 4.0.
2. Predložiti rješenje automatiziranog sustava za prikupljanje podataka primjenom IIoT senzora.
3. Razviti korisničko sučelje za komunikaciju s IIoT senzorom, prijenos i pohranu podataka u bazu te vizualizaciju podataka.
4. Na konkretnom primjeru objekta održavanja prikazati funkcionalnosti i rad sustava.
5. Na temelju analize sprovedenog istraživanja izvesti zaključak.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
12. studenog 2020.	14. siječnja 2021.	18. siječnja do 22. siječnja 2021.

Zadatak zadao: prof. dr. sc. Dragutin Lisjak

Predsjednica Povjerenstva: prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS KRATICA

Kratica	Opis
IIoT	*Industrial Internet of Things* – Industrijski internet stvari
IoT	*Internet of Things* – Internet stvari
CPS	*Cyber-Physical System* – Kibernetsko-fizički sustav
IEEE	*Institute of Electrical and Electronics Engineers* – Institut električnih i elektroničkih inženjera
TCP	*Transmission Control Protocol* – Protokol nadzora prijenosa
UDP	*User Datagram Protocol* - Protokol izmjene korisničkih datograma
MQTT	*Message Queuing Telemetry Transport* – Protokol za telemetrijski prijenos podataka
CoAP	*Constrained Application Protocol* – Ograničeni aplikacijski protokol
IP	*Internet Protocol* - Internet protokol
CASE	*Computer Aided Software Engineering* – Računalno potpomognut razvoj programske podrške
SSADM	*Structured systems analysis and design methodology* – Metodologija strukturne analize i dizajna sustava
MIRIS	Metodika za Razvoj Informacijskih Sustava
RED	*Rapid Embedded Development* – Brzi integrirani razvoj
API	*Application Programming Interface* – Aplikativno programsko sučelje
UID	*Unique Identification* – Unikatna identifikacija
ASIC	*Application-specific integrated circuit* – Aplikacijski-specifičan integrirani krug
HTTP	*Hypertext Transfer Protocol* - Protokol za prijenos hiperteksta
UML	*Unified Modeling Language* - Jedinstveni jezik za modeliranje
SQL	*Structured Query Language* - Strukturirani jezik upita
LAN	*Local Area Network* – Lokalna mreža
DHCP	*Dynamic Host Configuration Protocol* - Protokol dinamičke konfiguracije domaćina
UTP	*Unshielded Twisted Pair* - Neoklopljena upletena parica
VPN	*Virtual Private Network* – Virtualna privatna mreža
HTML	*HyperText Markup Language* - Hipertekstualni označni jezik
CSS	*Cascading Style Sheets* - Kaskadni listovi stilova
AJAX	*Asynchronous JavaScript and XML* – Asinkroni Javascript i XML
IDE	*Integrated Development Environment* – Integrirana razvojna okolina
BBF	Oštećenje kotrljajućeg elementa kotrljajućeg ležaja
ORBF	Oštećenje vanjske staze kotrljanja kotrljajućeg ležaja
IRBF	Oštećenje unutarnje staze kotrljanja kotrljajućeg ležaja
CBF	Kombinirano oštećenje kotrljajućeg ležaja
ERF	Ekscentričnost rotora
CRF	Nagnutnost rotora
NS	Normalno stanje
IMRF	Neravnoteža rotora

SAŽETAK

Moderne strategije održavanja poput prediktivnog i preskriptivnog održavanja, koje su proizašle iz koncepta Industrije i Održavanja 4.0, podrazumijevaju primjenu tehnologije Industrijskog interneta stvari (IIoT) s ciljem povezivanje objekata održavanja u jedinstvenu cjelinu pomoću koje je olakšano prikupljanje podataka, analiza te donošenje odluka o aktivnostima održavanja. Prikupljanjem podataka je prvi korak i temelj svake moderne strategije održavanja jer se upravo njihovom analizom mogu dobiti korisne informacije o stanju objekata održavanja. Tema ovog diplomskog rada je razvoj automatiziranog sustava za prikupljanje podataka o objektima održavanja primjenom IIoT senzora. Nakon kratkog uvoda u rad, u početnom dijelu rada kratko je opisan koncept Industrije i Održavanja 4.0 te su opisane karakteristike Industrijskog interneta stvari. Odabrana je metodologija projektiranja procesa te je opisan trenutni sustav s njegovim ograničenjima na temelju čega su postavljeni ciljevi novog sustava. U središnjem dijelu rada opisana je arhitektura sustava te su odabrane tehničke varijante komponenti sustava, nakon čega su detaljno opisani i prikazani procesi sustava kao i modeli podataka koje sustav koristi kako bi izvršio procese. Također, opisana je i izrada sustava koja uključuje konfiguraciju sustava, programiranje korisničkog sučelja te testiranje sustava. U zadnjem dijelu prikazan je primjer simulacije sustava kao i moguća primjena u vodoopskrbnom sustavu te je na kraju izveden zaključak. Razvojem automatiziranog sustava za prikupljanje podataka postignuta je značajna ušteda inicijalnih troškova komponenti sustava, omogućeno je udaljeno i automatizirano prikupljanje podataka te je ostvarena podloga da daljnji razvoj.

Ključne riječi: Održavanje 4.0, Industrijski internet stvari, automatizirano prikupljanje podataka, informacijski sustav, Tinkerforge, web aplikacija

SUMMARY

Modern maintenance strategies, such as predictive and prescriptive maintenance, which derived from the concept of Industry and Maintenance 4.0, involve the application of the Industrial Internet of Things (IIoT) to connect maintenance facilities into a unique composition, enabling data collection and analysis which will help make better decisions on maintenance activities. Data collection is the first step and the foundation of any modern maintenance strategy because it collects data that can then be analyzed to provide useful information about the state of maintenance of facilities. The topic of this master thesis is the development of an automated system for collecting maintenance facilities data using IIOT sensors. After a brief introduction to the thesis, the initial part of the paper briefly describes the concept of Industry and Maintenance 4.0 alongside with characteristics of the Industrial Internet of Things. The methodology of the design process is selected, after which the current system and its limitations are described on the basis of which the goals of the new system are set. The central part of the paper describes the system architecture followed by the selection of technical variants of the system components, after which the system processes are described and presented in detail as well as the data models that the system uses to perform tasks. Also, system design is described, which includes system configuration, user interface programming and system testing. The last part of the thesis shows an example of system simulation as well as possible application in the water supply system after which the conclusion is drawn. The development of an automated data collection system has achieved significant savings in the initial costs of system components, enabled remote and automated data collection and provided a basis for further development.

Key words: Maintenance 4.0, Industrial Internet of Things, automated data collection, information system, Tinkerforge, web application

1 UVOD

Razvojem industrijskih i informacijskih tehnologija te modernizacijom proizvodnih procesa pojavila se potreba i za modernizacijom metoda održavanja. Održavanje 4.0 je koncept koji podrazumijeva primjenu IIoT (eng. *Industrial internet of things*) senzora kako bi se objekti održavanja povezali u informacijsko-komunikacijsku cjelinu s ciljem učinkovitijeg upravljanja procesima. Industrijski internet stvari (IIoT) odnosi se na senzore, instrumente i druge uređaje koji se koriste u industriji, umrežene zajedno putem interneta s računalima. Međusobna povezanost omogućuje prikupljanje podataka, razmjenu informacija i analizu, što potencijalno olakšava poboljšanje produktivnosti i učinkovitosti, kao i planiranje aktivnosti održavanja. Glavna prednost prikupljanja podataka primjenom IIoT senzora je što se na temelju udaljeno prikupljenih podataka može praktički u realnom vremenu raditi analiza "zdrastvenog" stanja objekata održavanja, a time ujedno i učinkovitije upravljanje procesom i aktivnostima održavanja.

Cilj ovog diplomskog rada je opisati i dokumentirati automatizirani sustav za prikupljanje podataka o objektima održavanja pomoću IIoT senzora, koji će biti u stanju prihvatiti ulazne parametre prikupljanja, prikupljati podatke o akceleraciji, ponoviti postupak prikupljanja više puta s više senzora, spremiti podatke u bazu podataka te isporučiti podatke korisniku u obliku grafa kako bi se dobila vizualna predodžba o podacima. Automatizirani sustav prikupljanja podataka koristit će TinkerForge IIoT senzore za prikupljanje podataka o akceleraciji te dodatne uređaje za upravljanje i komunikaciju. Podaci koji su spremljeni u bazu podataka služe kao sredstvo za daljnju analizu podataka (strojno učenje, analiza vremenskih serija...) pomoću kojih će se donositi procjene i odluke o daljnjim aktivnostima održavanja. U radu je opisan koncept Održavanja 4.0 i IIoT-a kao i detaljan opis, analiza rada, oblikovanje i primjena automatiziranog sustava te je na kraju izveden zaključak.

2 INDUSTRIJA 4.0

Industrija je grana ekonomije koja proizvodi materijalna dobra koja su nužna za stabilno funkcioniranje ekonomije i svijeta oko nas. Od prvih početaka industrijalizacije tehnološki napreci su doveli do promjene paradigme koje danas nazivamo „industrijske revolucije“ . U prvoj industrijskoj revoluciji javila se pojava mehanizacije poslova dok drugu industrijsku revoluciju karakterizira intenzivno primjena električne energije. Treća industrijska revolucija podrazumijeva rasprostranjena digitalizacija na temelju čega se gradi i četvrta paradigma industrijalizacije tzv. 4. industrijska revolucija (Slika 1.) [1]. Paradigma Industrije 4.0 podrazumijeva međusobno povezivanje fizičkih objekata kao što su senzori, uređaji i imovina poduzeća na Internet. Industrija 4.0 zahtjeva pretvorbu uobičajenih strojeva u samosvjesne i samouke strojeve kako bi se poboljšao njihov rad, omogućila održiva proizvodnja i smanjile potrebe za neplanirano održavanje. Cilj Industrije 4.0 je izgradnja pametne umrežene informacijske proizvodne platforme [2].

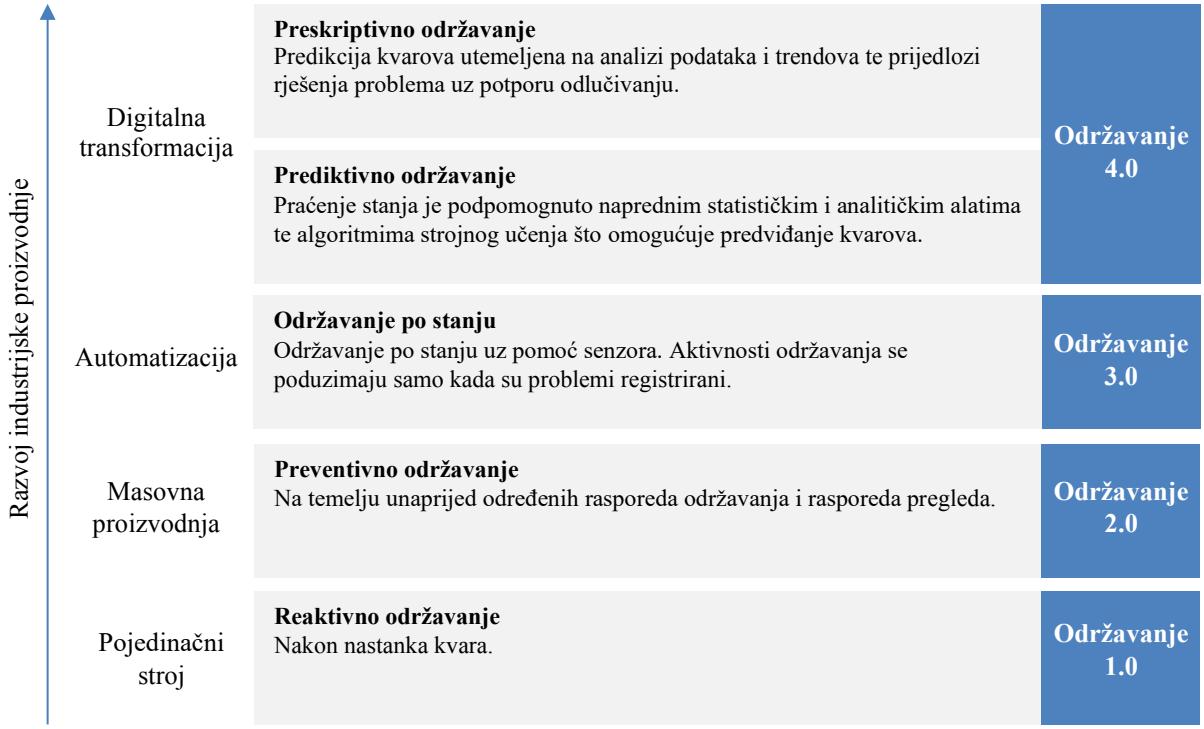
Slika 1. Četiri industrijske revolucije [2]

Prije nekoliko godina Industrija 4.0 se smatrala budućnošću, a danas je široko prihvaćena stvarnost koja mijenja način poslovanja poduzeća i koja utječe na gotovu svaku industriju širom svijeta. U kontekstu definicije Industrije 4.0 pristup održive proizvodnje može se povezati s proizvodnim sustavima razvijenim kako bi bili svjesni, transparentni, inteligentni, učinkoviti, fleksibilni, pokretljivi i odgovorni. U tom kontekstu uspostavljene su razne inicijative i pristupi koji pomažu poduzećima da usvoje načela četvrte industrijske revolucije po pitanju održivosti. Unutar takvih inicijativa, jedna od prevladavajućih tema pametne i održive proizvodnje je i

uporaba suvremenih pristupa održavanja poput Održavanja 4.0 (također se naziva i Pametno Održavanje) [3].

2.1 Održavanje 4.0

Evolucijom tehnologije tehnički sustavi postaju sve kompleksniji i kritičniji u pogledu pouzdanosti i raspoloživosti. Održavanje ima ključnu ulogu kako bi se smanjio rizik i posljedice neočekivanih zastoja u proizvodnji. Tijekom vremena održavanje je evoluiralo od reaktivnog (Održavanje 1.0) do preventivnog (Održavanje 2.0), a zatim do održavanja temeljenom na stanju (Održavanje 3.0) te trenutno do prediktivnog i preskriptivnog pristupa koji se naziva Održavanje 4.0. [3] Na slici 2. prikazan je razvoj pristupa održavanja.



Slika 2. Razvoj strategija održavanja [3]

Tijekom perioda reaktivnog održavanja strojevi su bili spori i jednostavni za uporabu, konstrukciju i održavanje. Radnici na strojevima bili su odgovorni za održavanje stroja na kojemu su radili, a aktivnosti održavanja provodile su se kada su se kvarovi dogodili kako bi se otklonili. Kako je kompleksnost strojeva rasla tako su se pojavili i odjeli za održavanje u poduzećima. Cilj odjela za održavanje bilo je smanjiti broj korektivnih aktivnosti održavanja tako što su se radili rasporedi pregleda strojeva i zamjene dijelova. Koncept sustava planskih

pregleda karakterizira preventivno održavanje koje podrazumijeva održavanje strojeva između unaprijed definiranih vremenskih intervala (dnevno, tjedno, mjesečno...). Početkom automatizacije i pojavom kompleksnih sustava javlja se i potreba za većom raspoloživosti i pouzdanosti strojeva kako bi se osigurala bolja kvaliteta proizvoda, dug vijek trajanja strojeva te isplativost. Održavanje temeljeno na stanju je omogućilo upravo navedeno tako što su se odluke donosile na temelju informacija prikupljenih iz nadzora stanja stroja. Pojavom Industrije 4.0 razvile su se nove paradigme i metode održavanja. Održavanje se prilagodilo zahtjevima Industrije 4.0 kako bi se postigao cilj pametne tvornice. [3]

Održavanje 4.0 je podskup pametnog proizvodnog sustava koje primjenjuje suvremene tehnologije i tehnike koje su u stanju predvidjeti kada će nastati kvar, predložiti najbolje rješenje te analizirati moguće odluke. Prediktivno održavanje podrazumijeva primjenu senzora za prikupljanje podataka koji se zatim analiziraju primjenom metoda nadziranog ili nenadziranog strojnog učenja kako bi se dobio opis stanja stroja i predvidjelo kada će nastati kvar. Ključne tehnologije koje sudjeluju u provođenju prediktivnog održavanja su Internet Stvari (eng. *Internet of Things – IoT)*, računarstvo u oblaku, prediktivna analitika (neizrazita logika, neuronske mreže, evolucijski algoritmi, strojno učenje...) te moderne tehnologije popravka stroja i zamjene dijelova. Uz prediktivno održavanje kod Održavanje 4.0 pojavljuje se i pojam preskriptivno održavanje. Preskriptivno održavanje koristi napredne tehnike analize podataka koje omogućuje ne samo da se kvarovi i zastoji predvide nego da se i preporuče radnje koje je potrebno poduzeti kako bi se kvarovi izbjegli te kako bi se optimirali rasporedi održavanja kao i potrebni resursi. Preskriptivno održavanje koristi kombinaciju naprednih analitičkih metoda, zbirke standardiziranih postupaka održavanja, povijesnih podataka o održavanju te trenutnih podataka koji se prikupljaju kako bi se predvidjeli zastoji i predložilo rješenje. [3]

Moderne strategije održavanja zahtijevaju velike količine podataka i informacija koje je potrebno obraditi što znači da je potrebna i velika računalna moć. Rast količine podataka te primjena istih korisna je kako bi se ostvarila proaktivna uloga održavanja u proizvodnji. Kako bi se podaci mogli koristiti u svrhu predviđanja potrebno ih je i obraditi. Analitika velike količine podataka je ono što razdvaja moderne metode održavanja (prediktivno i preskriptivno) od klasičnih. Usporedba podataka u različitim etapama razvoja održavanja prikazano je u tablici 1. [4]

Tablica 1. Usporedba podataka u različitim konceptima održavanja [3]

	Održavanje 1.0	**Održavanje 2.0**	**Održavanje 3.0**	**Održavanje 4.0**
Izvori podataka	Iskustvo radnika	Održavatelj i strojevi	Radnik, održavatelj, strojevi, informacijski sustavi	Radnik, održavatelj, informacijski sustav, proizvođač i dobavljač
Prikupljanje podataka	Ručno	Ručno	Polu automatizirano	Automatizirano, primjenom senzora i IoT
Pohrana podataka	Sjećanje radnika	Pisana dokumentacija	Baza podataka	Računarstvo u oblaku
Analiza podataka	Proizvoljno	Teorija pouzdanosti	Konvencionalne statističke metode	Neizrazira logika, strojno učenje, evolucijski algoritmi
Prijenos podataka	Verbalna komunikacija	Pisana dokumentacija	Digitalna dokumentacija	Digitalna dokumentacija
Upravljanje podacima	-	Ljudski radnici	Informacijski sustav	Umjetna inteligencija

Analitika velike količine podataka je proces prikupljanja, organiziranja i analiziranja velikih količina podataka često zvanih *Big data* koje karakterizira veliki volumen, velika brzina nastajanja i velika raznolikost. Prikupljanje podataka ostvaruje se primjenom pametnih senzora koji prevode fizičke pojave koje emitira stroj u digitalne signale koje predstavljaju parametre poput temperature i vibracija. Jednostavno prikupljanje podataka o strojevima nije dovoljno. Održavanje 4.0 zahtijeva infrastrukturu Interneta stvari (točnije Industrijskog interneta stvari) koja bežično povezuje objekte održavanja s podatkovnim centrima i omogućava prikupljanje i distribuciju podataka senzora. [4]

2.2 Industrijski internet stvari

Internet stvari (IoT) je koncept koji opisuje mrežu povezanih fizičkih objekata (stvari) koji sadrže senzore, programsku podršku i druge tehnologije kojima je cilj povezati i razmijeniti podatke s drugim fizičkim objektima i sustavima preko Interneta. Spajanje fizičkih objekata na Internet poboljšava održivost i sigurnost industrije i društva te omogućava učinkovitiju interakciju između fizičkog svijeta i kibernetsko-fizičkog sustava. Kao podvrsta Interneta stvari postoji i Industrijski internet stvari (IIoT) koji pokriva domenu međusobne strojne komunikacije i industrijske komunikacije s automatiziranom primjenom. Industrijski internet stvari omogućava bolje razumijevanje proizvodnog procesa što omogućuje bolju efikasnost i održivu proizvodnju. IoT, IIoT i Industrija 4.0 su usko povezani koncepti, ali ne mogu se koristiti kao sinonimi. Internet stvari se smatra umreženost uređaja, gdje se naglašava cilj razmjene podataka između uređaja, no područja primjene interneta stvari su toliko različita da neki zahtjevi (povezanost, fleksibilnost, brzina odziva...) mogu biti jako različiti ovisno o željenom cilju i korisnicima stoga se pojam Interneta stvari se kolokvijalno naziva i potrošački Internet stvari (eng. *consumer Internet of things*). [5]

Potrošački IoT je baziran na čovjeku, „stvari“ su pametni potrošački elektronički uređaji koji su međusobno povezani kako bi poboljšali korisnicima svijest o prostoru oko sebe što u konačnici omogućava uštedu vremena i novca. S druge strane kod Industrijskog IoT sve se svodi na povezivanje industrijskih objekata, uključujući strojeve i upravljačke sustave s informacijskim sustavima i poslovnim procesima. Kao posljedica toga velika količina prikupljenih podataka može služiti kao podloga analitičkim procesima analiza rada sustava. Primjena IIoT može se naći u raznim scenarijima poput aplikacija za nadzor sustava ili inovativnih pristupa za samo organizirajuću proizvodnju. [5]

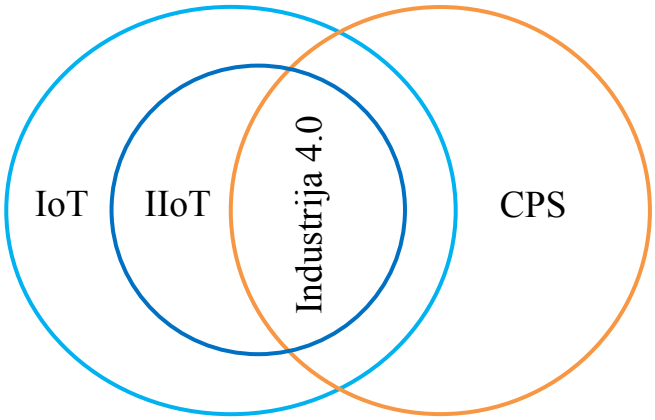
Razlike između IoT i IIoT proizlaze u nekoliko segmenata. IoT se više usredotočuje na dizajn novih komunikacijskih standarda koji mogu umrežiti nove uređaje na fleksibilan i korisnički nastrojen način. Suprotno tome, trenutni dizajn IIoT-a stavlja naglasak na moguću integraciju i međusobno povezivanje radnih ćelija i strojeva. Zbog tog razloga IIoT se smatra evolucijom IoT-a, a ne revolucijom. U pogledu povezanosti, IoT je više fleksibilniji te dopušta više načina povezivanja, te ima veće tolerancije na kašnjenja i pouzdanost veze dok s druge strane IIoT obično koristi fiksna i infrastrukturno utemeljena rješenja koja osiguravaju pouzdanost i brzo vrijeme odziva. [5] Uspoređujući količine podataka koji se razmjenjuju, kod IoT količina uvelike ovisi o području primjene, dok je IIoT baziran na analitici čija primjena npr. kod prediktivnog održavanja znači i ogromne količine prikupljenih podataka (više stotina

GB ili ponekad i TB podataka). U tablici 2. prikazane su osnovne razlike između Interneta stvari i Industrijskog interneta stvari. [5]

Tablica 2. Razlika između IoT i IIoT [5]

	Potrošački IoT	**Industrijski IoT**
Utjecaj	Revolucija	Evolucija
Orijentacija	Prema čovjeku	Prema stroju
Usmjerenost	Novi uređaji i standardi	Postojeći uređaji i standardi
Povezanost	Infrastruktura nije potrebna	Struktura je potrebna
Kritičnost	Nije na prvom mjestu	Jako je bitna točnost, pouzdanost, sigurnost i privatnost
Količina podataka	Srednja do visoka	Visoka i jako visoka

Koncept Industrije 4.0 je nastao kada se IoT paradigma spojila s idejom o kibernetsko-fizičkim sustavima (CPS) (slika 3.). Kibernetsko-fizički sustav sadrži skup interaktivnih fizičkih i digitalnih komponenti, koje mogu biti centralizirane ili distribuirane, a sustav pruža kombinaciju senzorskih, kontrolnih, računalnih i mrežnih funkcija kako bi pomoću fizičkih procesa utjecao na ishode u stvarnom svijetu [6]. U suštini, u proizvodnim sustavima susreću se IoT i Industrija 4.0 te tako nastaje IIoT.

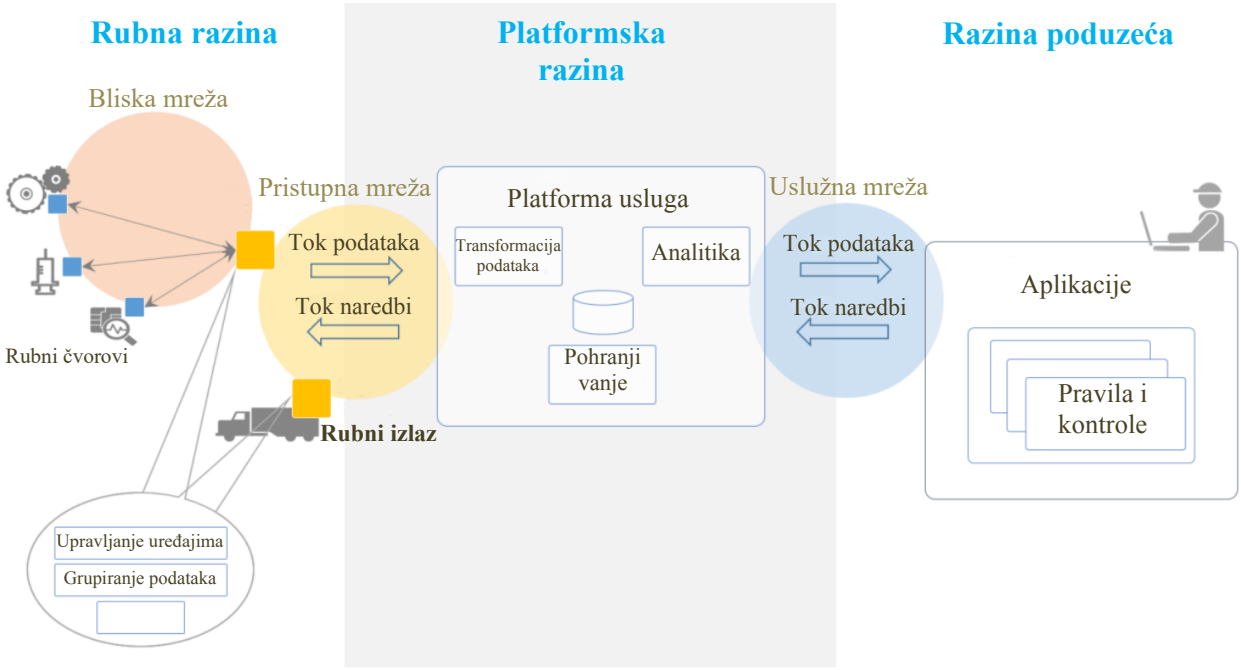


Slika 3. IoT, CPS, IIoT i Industrija 4.0 u Vennovom dijagramu [5]

Potrebno je napomenuti da IIoT nije namijenjen kako bi zamijenio tradicionalne automatske sustave, nego je stvoren kako bi povećao znanje o fizičkim objektima i sustavima koji su od interesa. Kao posljedica toga IIoT se primjenjuje za nadgledanje, optimizaciju i predikciju. Temelj automatiziranog sustava za prikupljanje podataka o objektima održavanja je upravo IIoT čija je arhitektura detaljnije opisana u nastavku. [5]

2.2.1 Arhitektura Industrijskog interneta stvari

Koherentni IIoT sustavi koriste arhitektonske obrasce koji predstavljaju pojednostavljeni i apstraktan prikaz IIoT sustava koji se pojavljuje u mnogo slučajeva primjene IIoT sustava. Arhitektonski obrasci su također fleksibilni i podložni promjenama ovisno o primjeni IIoT sustava. Jedan od takvih arhitektonskih obrazaca je i troslojni arhitektonski obrazac (Slika 4.) koji se sastoji od rubne razine, platformske razine i platforme poduzeća. Navedene razine imaju specifičnu ulogu u procesuiranju toka podataka i toka naredbi. [7]



Slika 4. Troslojna IIoT arhitektura [7]

Rubna razina definira domenu u kojoj IIoT komponente međusobno komuniciraju, sastoji se od rubnih čvorova (eng. *Edge nodes*) koje čine senzori, kontroleri i aktuatori koji su međusobno povezani neovisnim lokalnim mrežama na rubni izlaz (eng. *Edge gateway*). Rubna razina služi za prikupljanje podataka s rubnih čvorova primjenom lokalne mreže. Rubni izlaz možemo definirati kao „vrata“ koja daju prolaz za povezivanje s većim mrežama (pristupna mreža) razine platforme, pružajući širu pokrivenost. Rubni uređaj također može imati i moć procesuiranja podataka, ovisno o njegovoj karakteristici. Razina platforme se koristi na uslužnoj mreži kako bi se podaci procesuirali, konsolidirali i proslijedili na razinu poduzeća. Razina platforme pruža funkcionalnosti kao što su analitika i upravljanje uređajima. Razina poduzeća implementira aplikacije specifične za domenu primjena te pruža korisničko sučelje.

Važno je napomenuti kako obrada podataka može biti širok pojam koji uključuje sortiranje, spremanje, analitičke operacije i sl. [7]

Povezanost IIoT varira ovisno o infrastrukturi i arhitekturi rubnih čvorova, a koriste se bežična ili žičana tehnologija. Ključni cilj IIoT povezanosti je izbjeći izolirane sustave koji se temelje na zaštićenim rješenjima te omogućiti razmjenu podataka i interoperabilnosti između postojećih zatvorenih sustava i sustava koji će tek doći unutar industrije. Niti sedmeroslojni OSI model, kao ni peteroslojni Internet model nisu prikladni ako se u obzir uzima raspodijeljena priroda senzora, aktuatora i ostalih komponenti koje sudjeluju u IIoT komunikaciji. Inicijativa IIoT-a zahtjeva komunikacijske protokole koji mogu podržavati učinkovito, pravovremeno i sveprisutno prikupljanje i dostupnost informacija. Niže razine stoga moraju adekvatno odgovarati zahtjevima skalabilnosti i fleksibilnosti, a gornje razine moraju omogućiti „pametnim“ uređajima prijenos podataka. Na temelju navedenog mogu se definirati tri sloja IIoT komunikacije [5]:

1. Umrežavanje – razmjena paketa podataka (standardi IEEE 802.3, IEEE 802.11, IEEE 802.15...)
2. Povezivanje – prijenos poruka (standardi: TCP, UDP, MQTT, CoAP, WebSocket...).
3. Informacije – strukture podataka krajnjeg korisnika.

Najnovije tehnologije (npr. mnoge verzije Etherneta) izvorno usvajaju Ethernet i IP protokole olakšavajući tako tehničku interoperabilnost tj. mogućnost dijeljenja paketa u zajedničkom formatu. Ethernet mreža i Internet, povezivanjem industrijske opreme, omogućuju vremenski osjetljivu komunikaciju omogućujući tako proizvodnju u stvarnom vremenu. [5]

2.2.2 Podjela Industrijskog interneta stvari

Industrijski internet stvari može se podijeliti u više kategorije. Pomoću šest IIoT kategorija, koje se također djele na podkategorije, moguće je klasificirati IIoT uređaj. Kategorije su [6]:

- industrijski sektor,
- lokacija uređaja,
- povezanost,
- karakteristike uređaja,
- tehnologija uređaja,
- vrsta korisnika.

Podjela s obzirom na industrijski sektor poprima više mogućnosti s obzirom da postoji znatan broj sektora gdje se IIoT uređaj može koristiti. Industrijski sektor primjene IIoT uređaja može biti transport, proizvodnja, maloprodaja, energija, telekomunikacija, rudarstvo, agrokultura i mnogi drugi. [6]

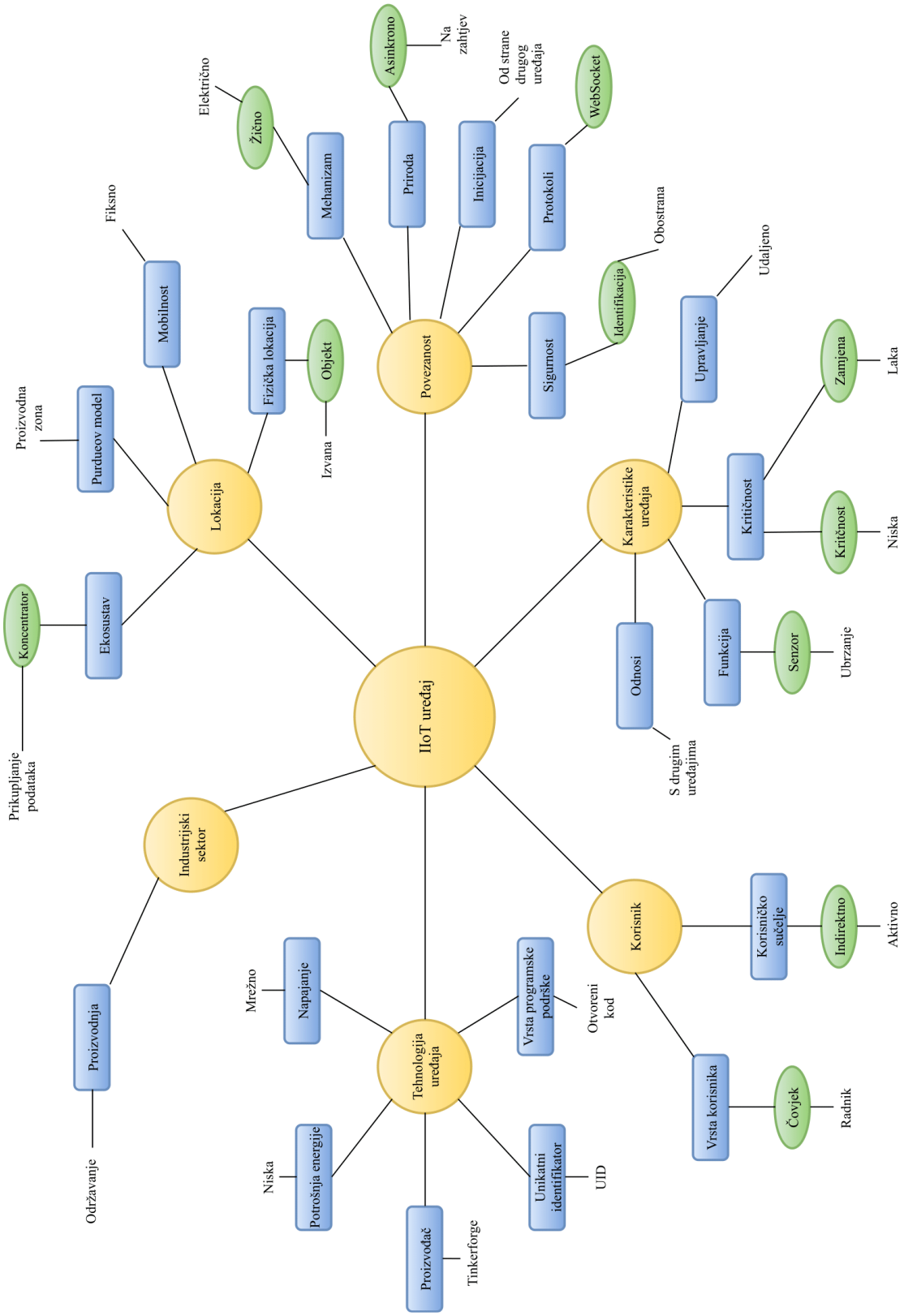
Podjela s obzirom na lokaciju može se vršiti dodatno s obzirom na ekosustav, Purdue-ov model, fizičku lokaciju te mobilnost. Pod pojmom ekosustav se podrazumijeva IIoT ekosustav koji se odnosi na IIoT arhitekturu navedenu u prethodnom poglavlju. Purdue-ov model hijerarhije upravljanja je model u proizvodnoj industriji koji dijeli uređaje s obzirom na hijerarhijsku funkciju uređaja. Podjela s obzirom na fizičku lokaciju uređaja podrazumijeva njegovu stvarnu fizičku lokaciju dok podjela mobilnosti govori je li uređaj fiksno smješten na nekoj lokaciji ili je mobilan. [6]

Cilj podjele s obzirom na povezanost je utvrditi bitne značajke umrežavanja i komunikacijske povezanosti između uređaja i IIoT sustava unutar kojega djeluje. Potkategorije povezanosti su mehanizam povezanosti, priroda povezanosti, inicijacija, protokoli i sigurnost. Mehanizam povezanosti se usredotočuje na fizički način spajanja uređaja kako bi se podaci prenosili na ili s uređaja. Priroda povezanosti odnosi se vremensku domenu prikupljanja podataka koja može biti u stvarnom vremenu (eng. *real-time*), gotovo u stvarnom vremenu (eng. *near real-time*) ili na zahtjev. Inicijacija povezanosti odnosi se na to kako započinje komunikacija, a podjela na protokole dijeli IIoT uređaje s obzirom na protokole s kojima se uspostavlja veza i pomoću kojih se prenose podaci. Podjela s obzirom na sigurnost se usredotočuje na razinu sigurnosti pri uspostavljanju veze i prijenosa podataka. [6]

Podjela s obzirom na karakteristike uređaja se usredotočuje na funkcionalnost uređaja, a posebno na važnost uloge uređaja u sustavu i način upravljanja uređaja. Potkategorije podjele s obzirom na karakteristike mogu biti kritičnost, funkcija, odnos i način upravljanja. Kritičnost se odnosi na kritičnost uređaja s obzirom na cjelokupni rad sustava te koliko je lako popraviti ili zamijeniti uređaj, što je veća kritičnost to je teže zamijeniti ili popraviti uređaj te sigurnost uređaja također mora biti veća. Funkcija opisuje glavne funkcije uređaja koji može biti senzor, aktuator ili procesni element. Potkategorija odnosa razvrstava uređaj s obzirom na odnos s drugim uređajima u IIoT sustavu. [6]

Podjela s obzirom na tehnologiju sustava razvrstava uređaj ovisno o tehničkim karakteristikama poput izvora napajanja, potrošnje energije, operativnog sustava, tipa programske podrške, sklopovlja i sl. dok zadnja podjela na korisnika razvrstava uređaj s obzirom na vrstu korisnika koji može biti stroj ili čovjek te na vrstu sučelja koje može biti nepostojeće, direktno ili indirektno. [6]

Podjela IIoT uređaja koji će se koristiti u ovom radu je sljedeća; industrijski sektor je kao što je već ranije spomenuto proizvodnja, a područje je održavanje. Uređaj će u ekosustavu održavanja imati ulogu prikupljanja podataka te će biti montiran fiksno na vanjskoj površini objekta održavanja. Povezanost uređaja je žična, prikupljanje podataka vrši se na zahtjev, a inicijacija se odvija od strane drugog uređaja te se koristi WebSocket protokol koji osigurava obostranu identifikaciju uređaja. Karakteristika uređaja u pogledu njegove funkcije je da prikuplja podatka s udaljenih lokacija što znači da se može klasificirati kao senzor čija je kritičnost niska, a zamjena laka. Tehnologija koju će IIoT uređaj koristiti je postavljena od strane proizvođača koji je u ovom slučaju TinkerForge, a uređaj karakterizira niska potrošnja električne energije te primjena otvorenog koda za razvoj programske podrške. Konačno, korisnik koji će koristiti IIoT uređaj je radnik (održavatelj), a interakcija je ostvarena preko aktivnog indirektnog pristupa (web aplikacija). Tehničke pojedinosti uređaja navedene su u sljedećim poglavljima, a grafički prikaz podjele IIoT uređaja može se vidjeti na slici 5.



Slika 5. Podjela IIoT uređaja

3 PLANIRANJE SUSTAVA

Automatizirani sustav za prikupljanje podataka, zbog svojih karakteristika, može se smatrati modulom informacijskog sustava. Informacijski sustav je sustav čija je svrha prikupljanje, pohrana, procesuiranje i isporučivanje informacija važnih za organizaciju i društvo na temelju kojih se mogu donositi poslovne odluke. Sastoji se od niza međusobno zavisnih komponenti odnosno tehnologija koje omogućuju njegovo uspješno izvršavanje zadataka, a to su [8]:

- programska podrška (software),
- računalna i komunikacijska infrastruktura (hardware),
- ljudi (korisnici).

Planiranje informacijskog sustava je temelj izrade informacijskog sustava pri kojem se identificiraju nedostaci postojećeg informacijskog sustava (ako postoji) te definira opseg novog sustava kao i njegovi ciljevi i plan razvijanja. Planiranje informacijskog sustava treba biti definiran, strukturiran i planiran proces kako bi se osigurala njegova funkcija, struktura i integritet. Kako bi se osiguralo navedeno koriste se metodologije projektiranja informacijskih sustava. [8]

3.1 Metodologija projektiranja sustava

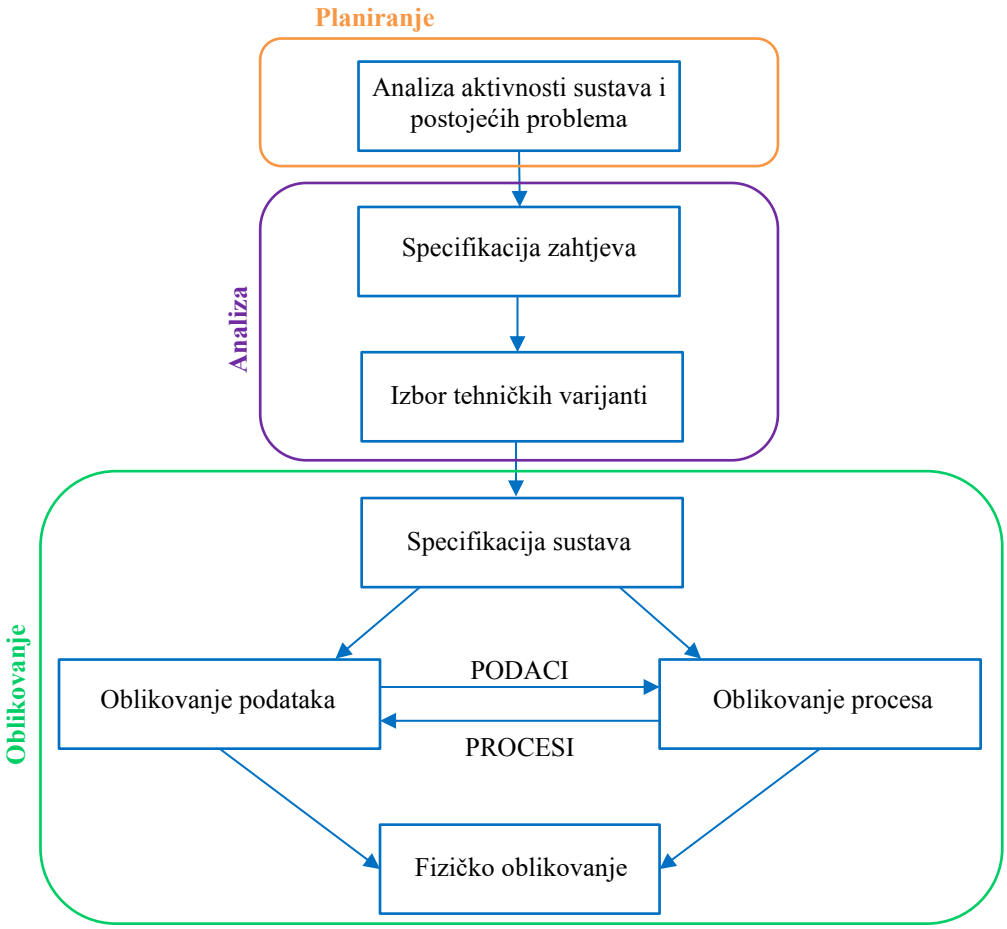
Metodologija projektiranja sustava podrazumijeva skup načela, pravila, metoda i tehnika koje se koriste za projektiranje, izgradnju i održavanje informacijskih sustava. Metode projektiranja sustava imaju ulogu definiranja aktivnosti razvoja sustava, određivanje redoslijeda aktivnosti sustava te definiranje polazišnog modela razvoja koje mogu biti temeljene na modelu procesa, modelu podataka ili modelu događaja. Neke od metoda temeljenih na modelu podataka su [8]:

- CASE (eng. *Computer Aided Software Engineering*) metoda,
- SSADM (eng. *Structured System Analysis and Design Method*),
- Yourdon/OO (*Yourdon/Object Oriented*) metoda,
- MIRIS (Metodika za razvoj informacijskih sustava).

Kao podloga za razvoj automatiziranog sustava za prikupljanje podatka koristi se metoda strukturne analize i dizajna sustava (SSADM) zbog njezinog radnog okvira koji jasno i logično slijedi korake razvoja. Metoda strukturne analize i dizajna sustava razdvaja logički i fizički dizajn. Programski/sklopovski logički dizajn se može prevesti u fizički dizajn što omogućava

razvojnim inženjerima da rješavaju jedan po jedan problem te sprječava nepotrebna ograničenja u ranom razvoju sustava. Odvajanje logičkog i fizičkog dizajna također omogućava korisnicima bez informatičkog znanja lakše razumijevanje načina rada sustava [9].

Faze razvoja su podijeljene u nekoliko koraka koji definiraju ulaz, izlaz i zadatke koji se trebaju izvršiti. Najprije se proučava trenutni sustav kako bi se steklo razumijevanje potrebe novog sustava i napravila specifikacija zahtjeva novog sustava. Specifikacija zahtjeva detaljna je do te mjere da se mogu formulirati detaljne tehničke mogućnosti. Detaljni dizajn dovršava se na logičkoj razini prije nego što se riješe pitanja implementacije. Konačno, logički dizajn pretvara u fizički dizajn [9]. Faze razvoja prema SSADM metodi prikazane su na slici 6.



Slika 6. SSADM faze razvoja [8] [9]

Faze razvoja prema SSADM metodi:

1. Analiza aktivnosti sustava i postojećih problema.

   Trenutni sustav se analizira zbog nekolika razloga. Jedan od njih je kako bi se naučila terminologija i proučilo stanje postojećeg sustava kao i struktura podataka koji su potrebni za rad sustava. [9] Metoda strukturne analize i dizajna sustava podrazumijeva pristup u kojem se struktura podataka u sustavu značajno ne mijenja tijekom vremena. Drugi razlog za analizu je kako bi se proučili i otkrili problemi prema kojima se definiraju i zahtjevi novog sustava. [8]

2. Specifikacija zahtjeva.

   Novi sustav ne smije biti ograničen s mogućnostima trenutnog sustava. Analiza aktivnosti sustava i postojećih problema osigurava razvojnim inženjerima da dobiju logičko razumijevanje sustava što im omogućava da se usredotoče na funkcije novog sustava neovisno o arhitekturi starog sustava. [9] U ovoj fazi se izrađuju detaljne specifikacije novog sustava na temelju korisničkih zahtjeva. Specifikacije sustava podrazumijevanju definiranje njegove arhitekture, logike i funkcije ovisno o korisničkim zahtjevima. [8]

3. Izbor tehničkih varijanti.

   U ovoj fazi se donosi odluka o izboru tehničkih varijanti koje omogućavaju rad sustava. Tehničke varijante mogu biti već postojeće ili nepostojeće (potrebna kupnja) te mogu biti u obliku programske podrške (software) ili sklopovlja (hardware). Tehničke varijante se odabiru u dogovoru s korisnikom ovisno o njihovoj cijeni i korisničkim zahtjevima. [9]

4. Oblikovanje procesa

   Cilj ove faze jest pružanje detaljnog objašnjenja logike procesa. Utvrđuju se logički upiti te logičke promjene podataka. Logika procesa se uspoređuje s logikom podataka kako bi se osigurala funkcionalnost. [9]

5. Oblikovanje podataka

   Ova faza je u simbiozi s prethodnom fazom, a s ciljem kreiranja logički dizajn podataka tako da uključi sve podatke u procesu informacijskog sustava. Primjenjuje se tehnika relacijske analize na skupine podataka koje su definirane u 2. fazi nakon koje se podaci logički oblikuju. Logika podataka se provjerava s logikom procesa kako bi se osigurala funkcionalnost sustava. [9]

6. Fizičko oblikovanje

Cijelo logičko oblikovanje procesa i podataka se pretvaraju u dizajn koji će se izvoditi na ciljanom okruženju. Opisuje se stvaran fizička lokacija baze podataka koja se nalazi na medijima za pohranjivanje. [8]

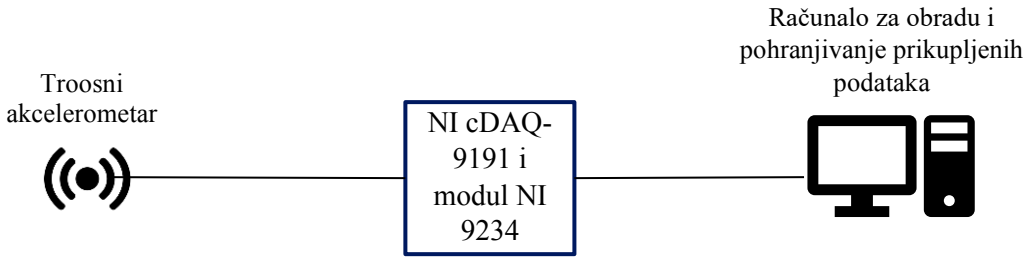
Nakon završetka zadnje faze SSADM metode kreće se u izradu sustava pri kojoj se programiraju, testiraju i dokumentiraju programski moduli te se nakon toga sustav implementira na za to predviđenom mjestu (zadnje faze CASE metode) [8]. Tijekom rada sustava javlja se i potreba za održavanjem i nadogradnjom sustava tijekom vremena ovisno o korisničkim zahtjevima.

3.2 Analiza aktivnosti postojećeg sustava

Postojeći sustav prikupljanja podataka nalazi se u *Laboratoriju za održavanje Fakulteta strojarstva i brodogradnje* te je primijenjen na simulatoru kvarova rotacijske opreme. Podaci koji se prikupljaju su vrijednosti akceleracije u x, y i z osi. Sustav za prikupljanje podataka sastoji se od [10]:

- troosnog senzora akceleracije proizvođača *PCB Piezotronics*,
- sustav za prikupljanje podataka *National Instruments cDAQ-9191* s modulom za akviziciju podataka sa senzora vibracija,
- računala za obradu i pohranu prikupljenih podataka.

Arhitektura trenutnog sustava prikazana je na slici 7.



Slika 7. Postojeći sustav prikupljanja podataka

Osnovne tehničke karakteristike trenutnog troosnog senzora akceleracije prikazane u tablici 3.

Tablica 3. Karakteristike PCB troosnog akcelerometra [10]

Karakteristika	Iznos	Mjerna jedinica
Frekvencijsko područje	2 - 7000	Hz
Mjerno područje akceleracije	± 500	g
Osjetljivost	10	mV/g
Napon pobude	18 - 30	VDC
Konstantna struja pobude	2 - 20	mA
Izlazni napon	7 - 12	VDC

Kako bi se podaci prikupili sa senzora koristi se sustav za prikupljanje i obradu podataka temeljen na *National Instruments CompactDAQ* platformi. Takav sustav sastoji se od osnovne jedinice spojene s računalom te modula za spajanje senzora. Modul za spajanje senzora je *National Instruments* 9234 za dinamičku akviziciju analognog signala sa senzora. [10]

Parametri prikupljanja i postavke senzora podešavaju se pomoću korisničkog sučelja na računalu. Postojeća *LabView* aplikacija ima mogućnost podešavanja nekoliko parametra poput [10]:

- načina prikupljanja podataka,
- frekvencije uzrokovanja,
- broja uzoraka u mjerenju,
- broju ponavljanja uzorkovanja,
- vrijeme između prikupljanja,
- podaci potrebni za generiranje i pohranu mjernih datoteka,
- vrsta simuliranog uzorka,
- učestalost vrtnje rotora,
- trenutni broj iteracija uzrokovanja.

Na početku mjerenja korisnik zadaje parametre prikupljanja te nakon što prikupljanje završi podaci se spremaju u *.csv* datoteke na tvrdi disk lokalnog računala te se mogu kasnije koristiti za analizu stanja promatranog objekta [10]. Izgled postojeće aplikacije za prikupljanje i obradu podataka nalazi se na slici 8.

Slika 8. Korisničko sučelje postojećeg sustava za prikupljanje podataka [10]

3.3 Ograničenja postojećeg sustava prikupljanja podataka i ciljevi novog sustava

Postojeći sustav prikupljanja podataka uspješno izvršava svoje zadatke, no dakako postoje određena ograničenja koja će se nastojati ispraviti razvojem novog sustava prikupljanja podataka. Jedno od glavnih ograničenja je nemogućnost prikupljanja podataka s ciljem distribucije na udaljene lokacije. Naime, cijeli sustav prikupljanja je lokaliziran na jednoj lokaciji što znači da prikupljanje, obrada i spremanje podataka nije moguće s neke druge (udaljene) lokacije. To znači da mjeritelj svaki puta mora doći na mjesto gdje se nalazi računalo s programom za prikupljanje podataka, prikupiti podatke te ih zatim prebacivati na drugu lokaciju gdje se nalazi druga aplikacija za analizu prikupljenih podataka. U slučaju da se podaci moraju prikupljati s više objekata održavanja, održavatelj bi morao fizički posjetiti svaku lokaciju te prikupiti podatke sa svake posebno. Takav način prikupljanja podataka je nepraktičan te može oduzimati dosta vremena koje bi se inače moglo produktivnije iskoristiti. Zbog toga se postavlja zahtjev za sustavom koji će omogućiti udaljeno prikupljanje podataka o objektima održavanja te omogućiti transformaciju lokalno održivih sustava za prikupljanje

podataka u sustav sa širokom dostupnosti, što je jedan od glavnih tehničko-tehnoloških izazova modernih poduzeća. Takav sustav moguće je realizirati primjenom IIoT senzora u kombinaciji s web aplikacijom koja se može nalaziti lokalno (unutar poduzeća) ili udaljeno (u „oblaku“). Konačno odredište prikupljenih i obrađenih podataka može biti centralizirana baza podataka. Baza podataka omogućuje cjelovit, konzistentan i skalabilan način pohrane podataka gdje se su podaci zatim koriste u različite svrhe koje mogu biti poslovna inteligencija, analiza vremenskih serija i sl. Još jedna prednost baze podataka jest neovisnost njene lokacije o lokaciji aplikacije. Baza podataka može se također može nalaziti lokalno ili udaljeno od same aplikacije. Još jedno ograničenje postojećeg sustava su troškovi komponenti. Okvirni troškovi komponenti postojećeg sustava za prikupljanje podataka su dani u tablici 4.

Tablica 4. Okvirni troškovi komponenti postojećeg sustava prikupljanja podataka

Komponenta	Okvirna cijena (kn)
PCB troosni akcelerometar	20 000
National Instruments cDAQ-9191	4 000
National Instruments 9234	20 000
Stolno računalo	7 000
LabView licenca	21 000
Ukupno:	**72 000 kn**

Ukupni okvirni trošak sustava od 72 000 kn je značajan ako se uzme u obzir to da se na svaku lokaciju treba staviti poseban sustav prikupljanja podataka. Potrebno je napomenuti da je cijena *LabView* licence jednokratna, no i bez nje trošak se umnožava s brojem lokacija te taj iznos može značajno narasti. Inicijalni trošak može se drastično smanjiti primjenom IIoT senzora koji su cjenovno značajno pristupačniji.

Zaključno, glavni cilj novog sustava prikupljanja podataka je omogućiti prikupljanje podataka s udaljenih lokacija, omogućiti prikupljanje podataka s više lokacija (objekata održavanja), omogućiti spremanje prikupljenih podataka u bazu podataka te smanjiti inicijalni trošak komponenti potrebnih za rad sustava, a za razvoj sustava koristit će se SSADM metoda u kombinaciji s zadnjim fazama CASE metode.

4 ANALIZA I OBLIKOVANJE SUSTAVA

Analiza sustava je metoda znanstvenog istraživanja temeljena na dekompoziciji problema na sastavne dijelove. Cilj je istražiti kako sustav postiže svoj cilj što znači da treba istražiti procese koje prihvaćaju ulaz u sustav, istražiti što se sve događa s ulazom dok se on ne transformira u izlaz i sustav ne postigne svoj cilj. Transformaciju ulaza u izlaz obavljaju procesi koji su skup povezanih aktivnosti na temelju kojih komponente sustava ostvaruju cilj, a da bi se procesi izvršili potrebni su određeni resursi (programska podrška, sklopovlje, ljudi...) i vrijeme. Osnovni cilj analize sustava je utvrđivanje logičkog toka podataka kroz sustav [8].

4.1 Specifikacija zahtjeva

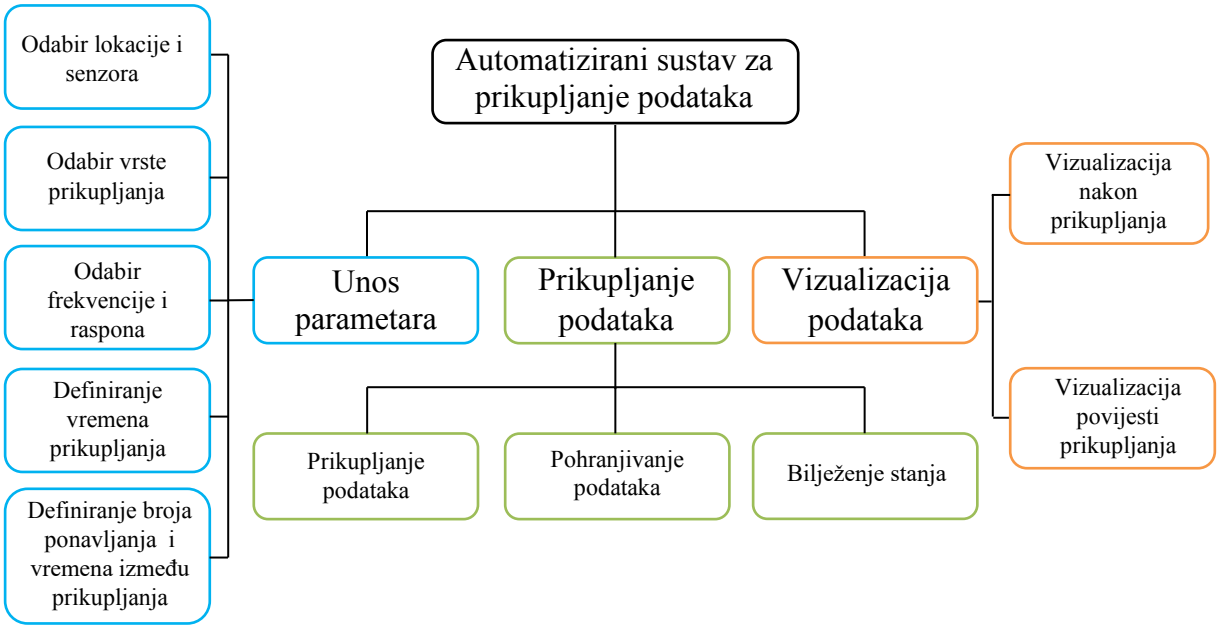
Zahtjevi koje se postavljaju na sustav mogu biti različitog tipa. Sustav se mora prilagoditi zahtjevima korisnika, ali uzimajući u obzir tehnička ograničenja. Osnovna funkcija automatiziranog sustav za prikupljanje podatka je prikupiti podatke o objektu održavanja, no ta funkcija nije samostalna nego zavisi o nizu drugih funkcija koje omogućavaju uspješan rad sustava. U nastavku su navedeni zahtjevi koji se postavljaju na automatizirani sustav prikupljanja podataka:

- prikupljanje podataka,
- pohranjivanje podataka u bazu podataka,
- odabir lokacije prikupljanja i senzora,
- odabir vrste prikupljanja, pojedinačno (s jednog senzora) ili skupno (s više senzora),
- odabir frekvencija prikupljanja i raspona vrijednosti,
- vizualizacija prikupljenih podataka,
- vizualizacija povijesti prikupljanja podataka,
- definiranje duljine trajanja prikupljanja,
- definiranje broja ponavljanja prikupljanja i vremena između ponavljanja,
- bilježenje stanja prikupljanja.

Svaki od navedenih zahtjeva u nekoj mjeri ovisi jedno o drugom zbog čega su procesi sustava međusobno povezani. Metodom dekompozicije moguće je probleme rastaviti na sastavne dijelove kako bi se dobio hijerarhijski opis sustava. Pojednostavljeno, zahtjevi sustava dijele se u tri kategorije:

1. Unos parametara – odabir lokacije i senzora, odabir vrste prikupljanja, odabir frekvencije i raspona, definiranje duljine prikupljanja, definiranje broja ponavljanja i vremena između ponavljanja.
2. Prikupljanje podataka – prikupljanje podataka, spremanje u bazu i bilježenje stanja prikupljanja.
3. Vizualizacija podataka – vizualizacija podataka nakon prikupljanja i vizualizacija povijesti podataka.

Prikaz dekompozicije aktivnosti sustava može se vidjeti na slici 9.

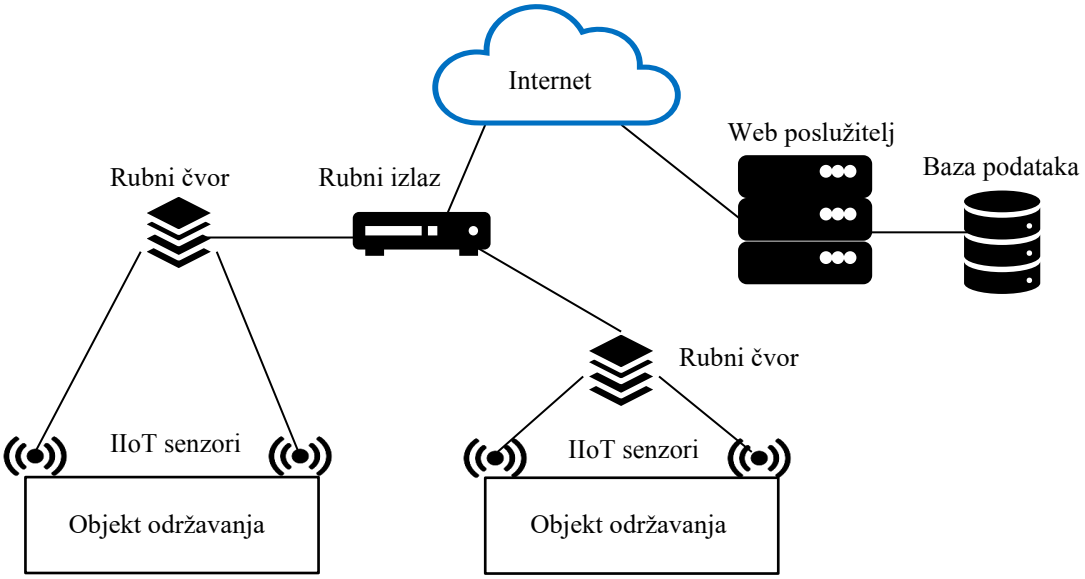


Slika 9. Hijerarhijski dijagram aktivnosti sustava

Kako bi se navedeni zahtjevi ispunili potrebno je najprije definirati komponente sustava koji će sudjelovati u izvršavanju procesa.

4.2 Izbor tehničkih varijanti

Izbor tehničkih varijanti nužan je kako bi se osigurao rad sustava i izvršavanje postavljenih zahtjeva. Tehničke varijante ponajviše ovise o proračunu informacijskog sustava i korisničkim zahtjevima. Arhitektura sustava je konceptualni model koji definira strukturu, ponašanje i pregled sustava. Arhitektura sustava može se sagledati kao opis sustava i njegovih komponenti organiziranih tako da opišu funkcioniranje sustava. [11] Arhitektura sustava za automatizirano prikupljanje podataka pomoću IIoT senzora prikazana je na slici 10.



Slika 10. Arhitektura automatiziranog sustava za prikupljanje podataka pomoću IIoT senzora

Arhitektura automatiziranog sustava za prikupljanje podataka pomoću IIoT senzora sastoji se od sljedećih komponenti:

1. IIoT senzora (akcelerometar).
2. Rubnog čvora (eng. *edge node*).
3. Rutera ili drugog rubnog izlaza.
4. Web poslužitelja na kojemu se nalazi web aplikacija.
5. Baze podataka.

Objašnjenje, odabrana varijanta i karakteristike komponenti sustava nalaze se u nastavku.

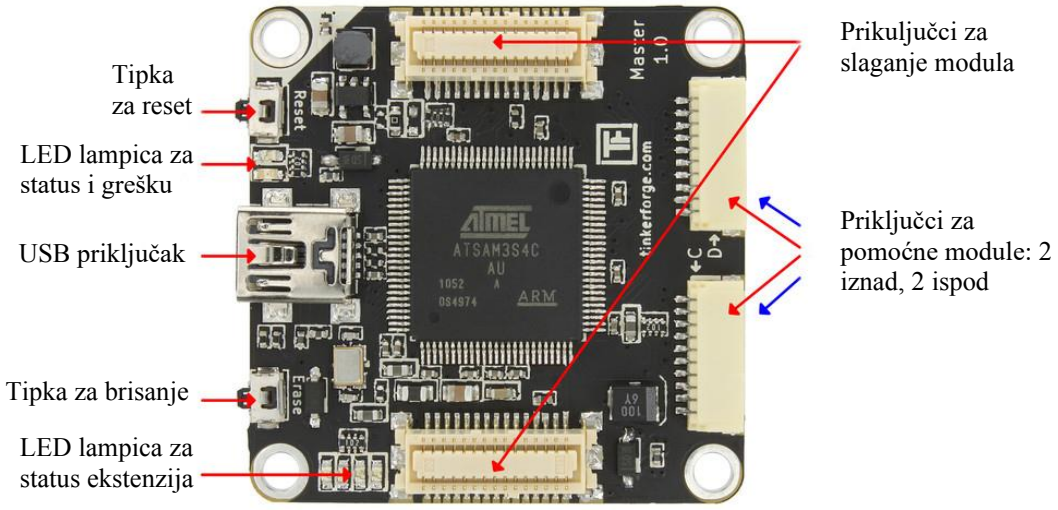
4.2.1 Rubni čvor

Rubni čvor omogućava prikupljanje, procesuiranje i slanje podataka s jedne mreže na drugu. Jedan od proizvođača rubnih čvorova je TinkerForge GmbH. Tinkerforge je sustav sastavnih modula za profesionalnu uporabu. Moduli imaju široku upotrebu te se koriste kako bi se projekti realizirali brzo i efikasno. Moduli se mogu slagati jedan na drugi te nisu ovisni o redoslijedu slaganja, a integracija s programskim sklopovljem se ostvaruje putem API-a. Tinkerforge podržava sve popularne programske jezike poput C/C++, C#, Delphi/Lazarus, Java, JavaScript, LabView, Mathematica, MATLAB/Octave, Perl, PHP, Python, Ruby, Shell i Visual Basic .NET. Tinkerforge moduli mogu se kontrolirati preko USB veze sa osobnim računalom ili Raspberry PI-em. Uz USB kontrola je moguća i preko Wi-Fi-a i Etherneta.

Primjenom RED (eng. *Rapid Embedded Development*) modula moguće je kontrolirati sustav bez vanjskog računala, a program koji se treba izvršavati može se staviti direktno na RED modul. [12] Tinkerforge komponente mogu se podijeliti u tri kategorije: [12]

1. Moduli (eng. *Bricks*) – su moduli veličine 40x40 mm koji se mogu kontrolirati preko Mini-USB veze. Svaki modul obavlja jedan zadatak poput kontrole motora, komunikacije i sl.
2. Pomoćni moduli (eng. *Bricklets*) – koriste se kako bi proširile funkcionalnost modula. Njihova primjena može biti u obliku mjerenja (senzor) ili kontrole. Svaki pomoćni modul povezuje se s ostalim modulima pomoću žične veze.
3. Glavne ekstenzije (eng. *Master Extensions*) – proširuju komunikacijske mogućnosti modula. Osim preko USB-a, moduli se mogu kontrolirati i preko Wi-Fi-a, Etherneta ili RS485 komunikacije.

Moduli imaju mogućnost slaganja jednog na drugi. Glavni modul (eng. *Master Brick*) je najniži modul koji je odgovoran za komunikaciju sa svim razinama konfiguracije. On usmjerava poruke između ostalih modula i upravljačkog uređaja. Zbog te funkcionalnosti, nije potrebna direktna komunikacija s ostalim modulima već samo s glavnim modulom. U slučaju da je potreban jači izvor napajanja moguće je u konfiguraciju dodati i modul koja služi kao izvor napajanja. Glavni modul ima 4 utora za povezivanje pomoćnih modula što je korisno u slučajevima gdje je potrebno koristiti više pomoćnih modula. Slaganjem više glavnih modula moguće je proširiti broj priključaka za pomoćne module. [13] Maksimalan broj naslaganih modula je 36. Izgled i dijelovi glavnog modula prikazani su na slici 11. Komunikacija i kontrola pomoćnih i ostalih modula moguća je jer svaki element ima svoj unikatni identifikacijski broj (UID). Svi moduli koriste programsku podršku u obliku programa *Brick Daemon* i *Brick Viewer*. *Brick Daemon* je program koji služi kao most između glavnih i pomoćnih modula i API veze za različite programske jezike dok *Brick Viewer* pruža grafičko sučelje za testiranje i konfiguraciju modula. Svaki uređaj ima svoj prikaz u *Brick Viewer*-u koji prikazuje glavne značajke i omogućava kontrolu. Također, *Brick Viewer* omogućava i kalibraciju analogno-digitalnih pretvornika kako bi se poboljšala kvaliteta mjerenja.



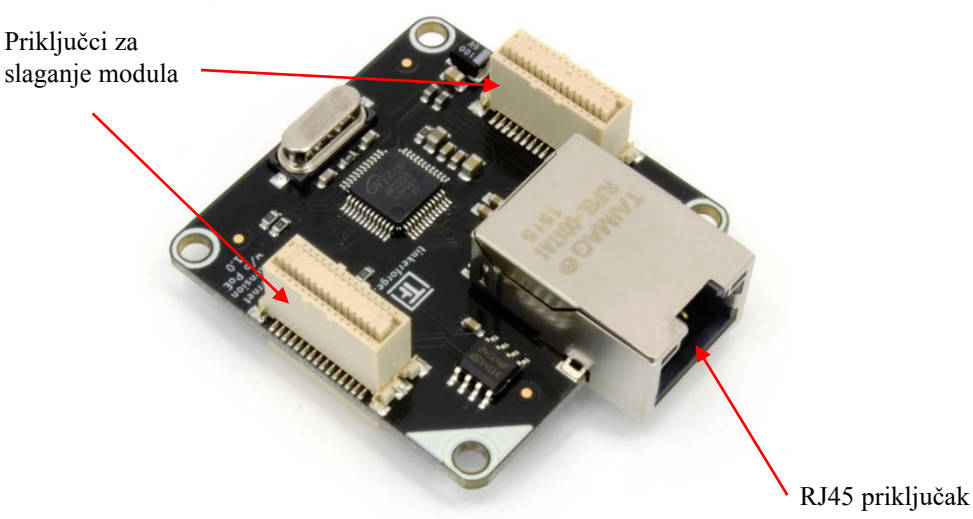
Slika 11. Glavni modul [13]

Tehničke karakteristike glavnog mtudodula dane su u tablici 5.

Tablica 5. Tehničke karakteristike glavnog modula [13]

Karakteristika	Vrijednost
Broj priključka za pomoćne module	4
Dimenzije	40x40x16 mm
Masa	12 g
Napon	5 V
Struja	82 mA

Kako bi se glavni modul mogao kontrolirati preko Interneta, potrebno je dodati jednu od glavnih ekstenzija. Odabire se Ethernet glavna ekstenzija zbog konstantne brzine prijenosa podataka i stabilnosti. Ethernet glavna ekstenzija omogućuje kontrolu glavnih i pomoćnih modula te integraciju u lokalnu mrežu (eng. *Local Area Network*). Ethernet ekstenzija podržava 10BaseT/100BaseTx, a modul je kompatibilan i sa 1000BaseTX standardom. [14] Za funkcioniranje ekstenzije nužno je imati glavni modul ili RED modul koji omogućuje komunikaciju između složenih modula i priključenih pomoćnih modula. Izgled i dijelovi Ethernet ekstenzije prikazani su na slici 12.



Slika 12. Ethernet ekstenzija [14]

Tehničke karakteristike Ethernet ekstenzije prikazane su u tablici 6.

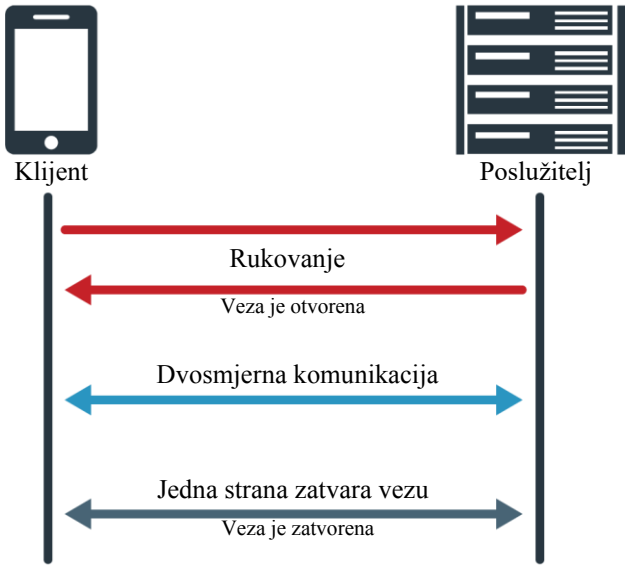
Tablica 6. Tehničke karakteristike Ethernet ekstenzije [14]

Karakteristika	Vrijednost
Maksimalan broj TCP/IP veza	7
Dimenzije	40x40x16 mm
Masa	22 g
Brzina	10/100 Mbit/s do 1000 Mbit/s
Struja	100 mA – 175 mA

Ethernet ekstenzija može se konfigurirati da koristi dinamičku ili statičku IP adresu. Dinamičku IP adresu dodjeljuje DHCP protokol dok ju u slučaju primjene statičke IP adrese potrebno ručno namjestiti IP adresu, Subnet masku i pristupnu adresu. Ethernet ekstenzija također podržava WebSockets protokol. [14]

WebSocket je komunikacijski protokol koji omogućuje *full-duplex* komunikaciju između poslužitelja i klijenta preko samo jedne TCP veze što je pogodno za „*real-time*“ prijenos podataka te će se koristiti u automatiziranom sustavu za prikupljanje podataka kao komunikacijski okvir rubnog čvora i web poslužitelja. Standardiziranjem WebSocket protokola omogućuje se konstantna međusobna razmjena podataka između poslužitelja i klijenata bez da klijent svaki puta traži zahtjev za slanje podataka. Protokol se sastoji od dva dijela; rukovanja i prijenosa podataka. Ako su oba rukovanja bila uspješna započinje prijenos podataka. Budući

da je komunikacija dvosmjerna, poruke se mogu slati sa poslužitelja do klijenta i obratno. [15] Princip rada WebSocket protokola prikazan je na slici 13.



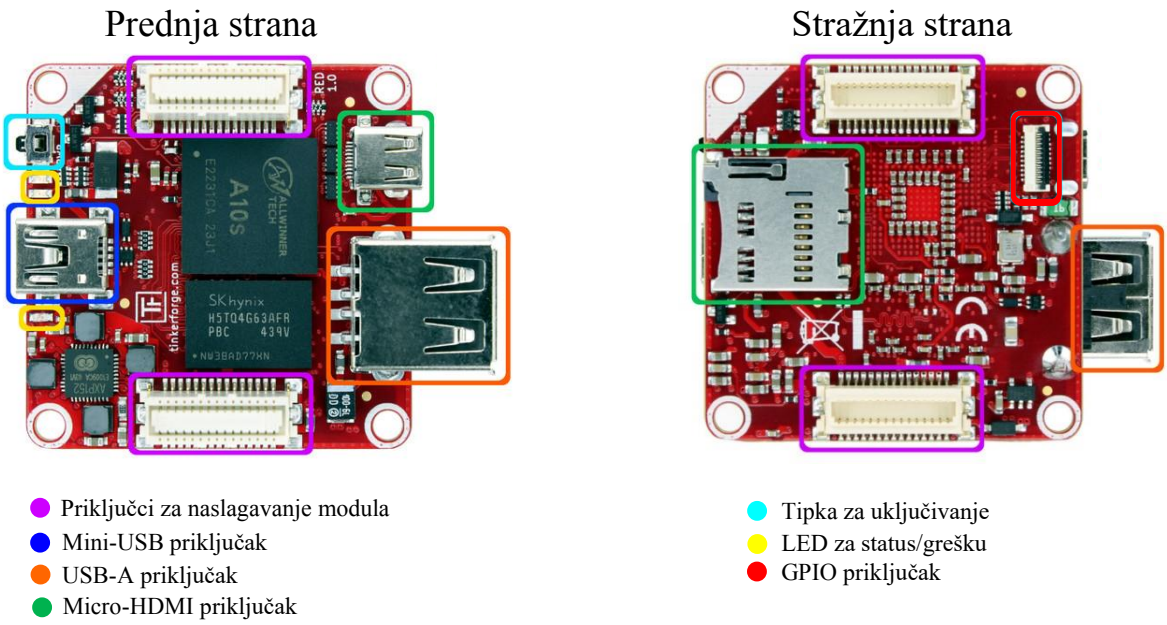
Slika 13. Princip rada WebSocket protokola [16]

Zadnja komponenta rubnog čvora je RED modul koji može kontrolirati ostale module. Programi, koji kontroliraju module, mogu se staviti direktno na RED modul koji pokreće Debian Linux operativni sustav, a on se nalazi na Micro-SD kartici na donjoj strani RED modula. Moguće je koristiti više programa istovremeno, namjestiti vrijeme njihovog izvršavanja kao i njihovo motrenje. RED modul podržava prethodno navedene programske jezike od kojih svaki koristi Tinkerforge API kao podlogu za komunikaciju s modulima, a taj pristup omogućuje brzo i lako ostvarivanje projekata. RED modul se mora koristiti u kombinaciji s glavnim modulom jer on sam nema utore za priključak pomoćnih modula. RED modul podržava cijeli niz usluga od kojih su neke [17]:

- GPU - omogućuje spajanje HDMI monitora za vizualni prikaz informacija.
- Okruženje radne površine – omogućuje prikaz grafičkog sučelja Debian Linux operativnog sustava.
- Web poslužitelj – RED modul može poslužiti i kao web poslužitelj te posluživati web stranice.
- Početni zaslon – početni zaslon omogućuje prikaz informacija pri pokretanju Linux operativnog sustava.
- Mobilna pristupna točka – omogućuje Wi-Fi pristupnu točku za spajanje drugih uređaja na mrežu.

- Mobilni internet – mogućnost povezivanja RED modula na mobilni internet ako se on spoji na GSM USB adapter.

Izgled i dijelovi RED modula prikazani su na slici 14.



Slika 14. RED modul [17]

Tehničke karakteristike RED modula prikazane su u tablici 7.

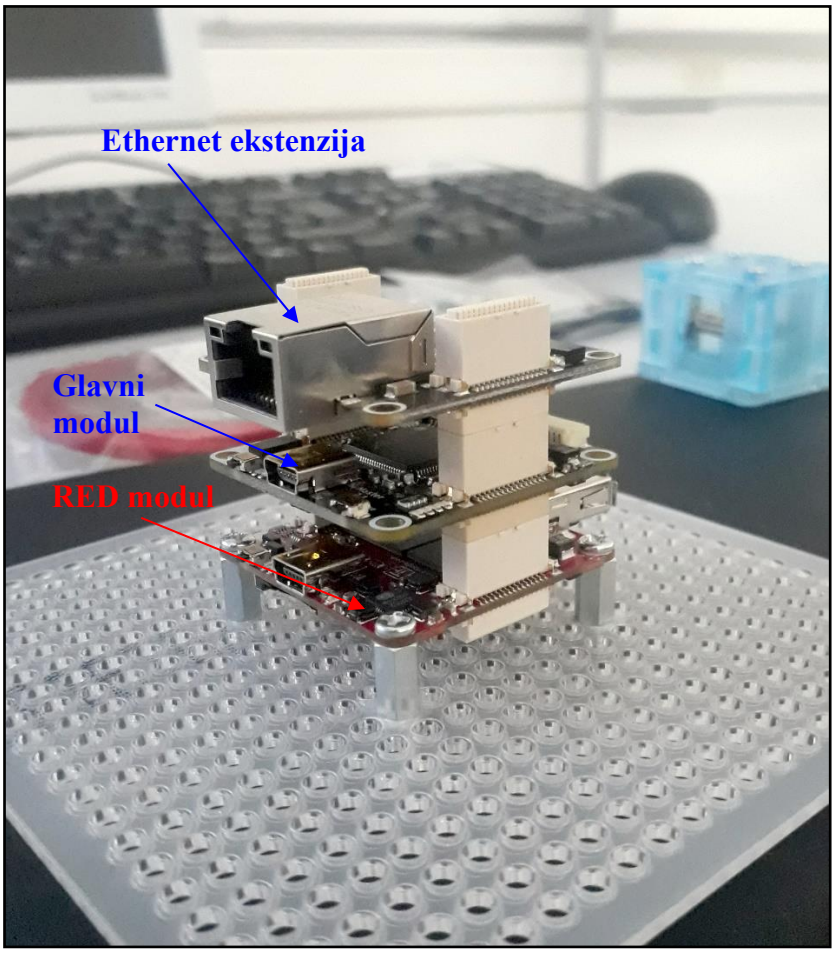
Tablica 7. Tehničke karakteristike RED modula [17]

Karakteristika	Vrijednost
Procesor	Allwinner A10s, Cortex A8 1 GHz, 3D Mali400 GPU, NEON
Memorija	512 MB DDR3 SDRAM, Micro-SD-Card kao Flash
Operativni sustav	Debian Linux
Priključci	USB 2.0 Host, Micro-HDMI (type D), Mini-USB, priključci za slaganje modula
Dimenzije	40 x 40 x 16 mm
Masa	14g
Napon	5V
Struja	150 mA – 220 mA

Nakon specifikacije pojedinih modula koji čine rubni čvor potrebno je module složiti u konfiguraciju. Redoslijed slaganje navedenih modula je sljedeći:

1. RED modul,
2. Glavni modul,
3. Ethernet ekstenzija.

RED modul daje računalnu snagu, glavni modul omogućava komunikaciju i povezivanje IIoT senzora dok Ethernet ekstenzija omogućava komunikaciju preko Interneta. Navedena tri modula su temelj automatiziranog sustava za prikupljanje podataka te su oni moraju biti uvijek prisutni (jedino se Ethernet ekstenzija može zamijeniti sa Wi-Fi ekstenzijom), no naravno moguće je dodati u konfiguraciju dodatne module koji omogućavaju neke druge funkcije (povezivanje više od 4 senzora, dodatno napajanje...). Izgled složene konfiguracije prikazan je na slici 15.



Slika 15. Prikaz modula u konfiguraciji

Moduli su dizajnirani tako da na sebi imaju rupe koje služe za međusobno spajanje modula u slučaju da se rubni čvor primjenjuje na zahtjevnim lokacijama. Konfiguraciju modula

je također moguće montirati na postolje što se može vidjeti na slici 15. Nakon odabira rubnog čvora potrebno je odabrati i senzore koji će prikupljati podatke.

4.2.2 IIoT senzor

IIoT senzor je najvažnija komponenta sustava za prikupljanje podataka jer bez njega nije moguće dobiti informacije (podatke) o objektu održavanja. Tinkerforge također nudi širok izbor senzora, a jedan od njih je i akcelerometar. Pomoćni modul *Accelerometer 2.0* je troosni akcelerometar koja je zamišljena kao dodatak za glavne module, a omogućuje mjerenje akceleracije u x, y i z smjeru frekvencije prikupljanja do 25,6 kHz što je pogodno za primjenu u prediktivnom održavanju. [18]

Akcelerometar ima ± 2 g, ± 4 g ili ± 8 g raspon prikupljanja, a osjetilni element je izrađen primjenom Kionixove plazmatske mikoobrade. Osjetilni element ubrzanja temelji se na principu diferencijalne kapacitivnosti koja proizlazi iz kretanja osjetilnog elementa uzrokovanog ubrzanjem. Osjetilni element zatim koristi proces smanjivanja šumova kako bi se smanjile greške zbog varijacije procesa, temperatura i ostalih atnosferskih utjecaja. Osjetilni element je hermetički zatvoren, a odvojeni ASIC uređaj omogućuje kondicioniranje signala i primjenu inteligentnih aplikacija za upravljanje koji se mogu programirati od strane korisnika. Akcelerometar dolazi s ugrađenim analogno-digitalnim pretvornikom što smanjuje potrebu za nabavom posebnog uređaja koji služi kao pretvornik što je slučaj u trenutnom sustavu za prikupljanje podataka. [19] Akcelerometar također karakteriziraju male dimenzije što mu omogućava primjenu na uskim i malim objektima održavanja. Izgled akcelerometra prikazan je na slici 16.

Slika 16. Tinkerforge akcelerometar [19]

Spajanje akcelerometra s glavnim modulom ostvaruje se kablom putem 7-polnog konektora. Također, za pravilnu primjenu akcelerometra preporučuje ga se montirati u kućište kako bi izmjerene vrijednosti bile točnije te kako bi se akcelerometar dodatno zaštitio od vanjskih utjecaja. Tehničke specifikacije akcelerometra nalaze se u tablici 8.

Tablica 8. Tehničke karakteristike Tinkerforge troosnog akcelerometra [19]

Karakteristika	Vrijednost
Frekvencija prikupljanja	0.781 Hz - 25.6 kHz (podesivo putem API)
Mjerno područje akceleracije	± 8 g
Osjetljivost	4096 - 16384 counts/g
Odstupanje	± 20 mg
Nelinearnost	0.6 %
Rezolucija	0.0001 g, 16-bit
Napon pobude	1.71 – 3.6 V
Konstantna struja pobude	145 mA
Izlazni napon	1.368 - 28.8 V

Najviša frekvencija prikupljanja ovisi o odabiru rezolucije. U tablici 9. prikazana je ovisnost broja osi prikupljanja i frekvencije.

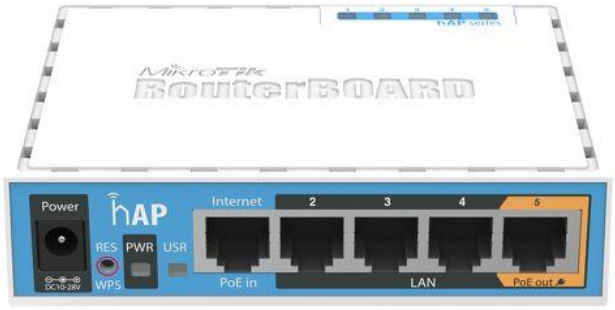
Tablica 9. Ovisnost broja osi prikupljanja o propusnosti Tinkerforge akcelerometra [20]

Broj osi prikupljanja	Propusnost (8-bit) [Hz]	Propusnost (16-bit) [Hz]
1	25 600	25 600
2	25 600	15 000
3	20 000	10 000

Broj osi prikupljanja moguće je odabrati putem Tinkerforge API-a za akcelerometar koji se može koristiti za sve programske jezike. Automatizirani sustav prikupljanja podataka koristi 3 osi prikupljanja i razlučivost od 16 bita, a najveća moguća frekvencija prikupljanja je 10 000 Hz.

4.2.3 Rubni izlaz

U automatiziranom sustavu prikupljanja podataka rubni izlaz je potreban kako bi povezao rubni čvor s web poslužiteljem preko Interneta. Kao rubni izlaz sustava koristi se MikroTik RB951-2nd hAP ruter koji je pogodan za manje sustave koji ne zahtijevaju puno korisnika i prometa. Zbog svojih malih dimenzija pogodan je za uporabu na skoro svakoj lokaciji. [21] Izgled MikroTik RB951-2nD hAP prikazan je na slici 17.



Slika 17. MikroTik RB951-2nD hAP ruter [21]

Tehničke karakteristike nalaze se u tablici 10.

Tablica 10. Tehničke karakteristike MikroTik RB951-2nD hAP rutera [21]

Karakteristika	Vrijednost
Operativni sustav	RouterOS
Procesor	Qualcomm QCA9531 650 MHz
Memorija	64 MB DDR3, 16 MB flash memorije
Bežični internet	802.11b/g/n 2.4 GHz
Broj Ethernet priključaka	5
Broj USB utora za GSM modeme	1
Napon	10 – 28 V
Struja	0.8 A
Maksimalna potrošnja energije	5 W

Ruter u sebi sadrži DHCP poslužitelj pomoću kojeg je moguće namjestiti konstantne vrijednosti IP adrese i Subnet maske za rubne čvorove.

4.2.4 Web poslužitelj

Pojam web poslužitelj može se odnositi na programsku podršku i/ili sklopovlje koji isporučuju sadržaj s weba klijentima koji to zatraže. Kako bi klijent pristupio web poslužitelju potrebno je imati web pretraživač koji uz pomoć HTTP ili nekog srodnog protokola komunicira s web poslužiteljem te korisniku isporučuje sadržaj. Web poslužitelj se često može nalaziti i unutar drugih uređaja poput printera, rutera ili rubnih čvorova te posluživati samo lokalnu mrežu. [22] Najpopularnija primjena web poslužitelja je „*hosting*" web stranica, no postoje i druge primjene kao što su igranje video igara, pohrana podataka ili vođenje aplikacija organizacije. Neki od najpopularnijih web poslužitelja su [23]:

- ISS – Internet Information Services,
- Apache Web Server,
- NGINX,
- Google Web Server.

Za izradu automatiziranog sustava za prikupljanje podatak koristi se *Microsoft Internet Information Services* zbog izvorne podrške unutar *Microsoft Visual Studio* razvojne okoline pomoću koje se razvija web aplikacija. Pri razvoju klijentska strana se nalazi na istom računalu kao i web poslužitelj, Lenovo ThinkPad T430.

4.2.5 Baza podataka

Baza podataka je digitalan način pohrane podataka koji se temelji na relacijskom modelu. Relacijske baze podataka koriste sustav upravljanja bazama podataka koji omogućava definiranje, kreiranje, održavanje i kontrolu nad bazom podataka. [24] Postoje različite vrste sustava upravljanja bazama podataka poput: [23]

- Microsoft SQL Servera,
- Oracle,
- MySQL,
- Microsoft Access.

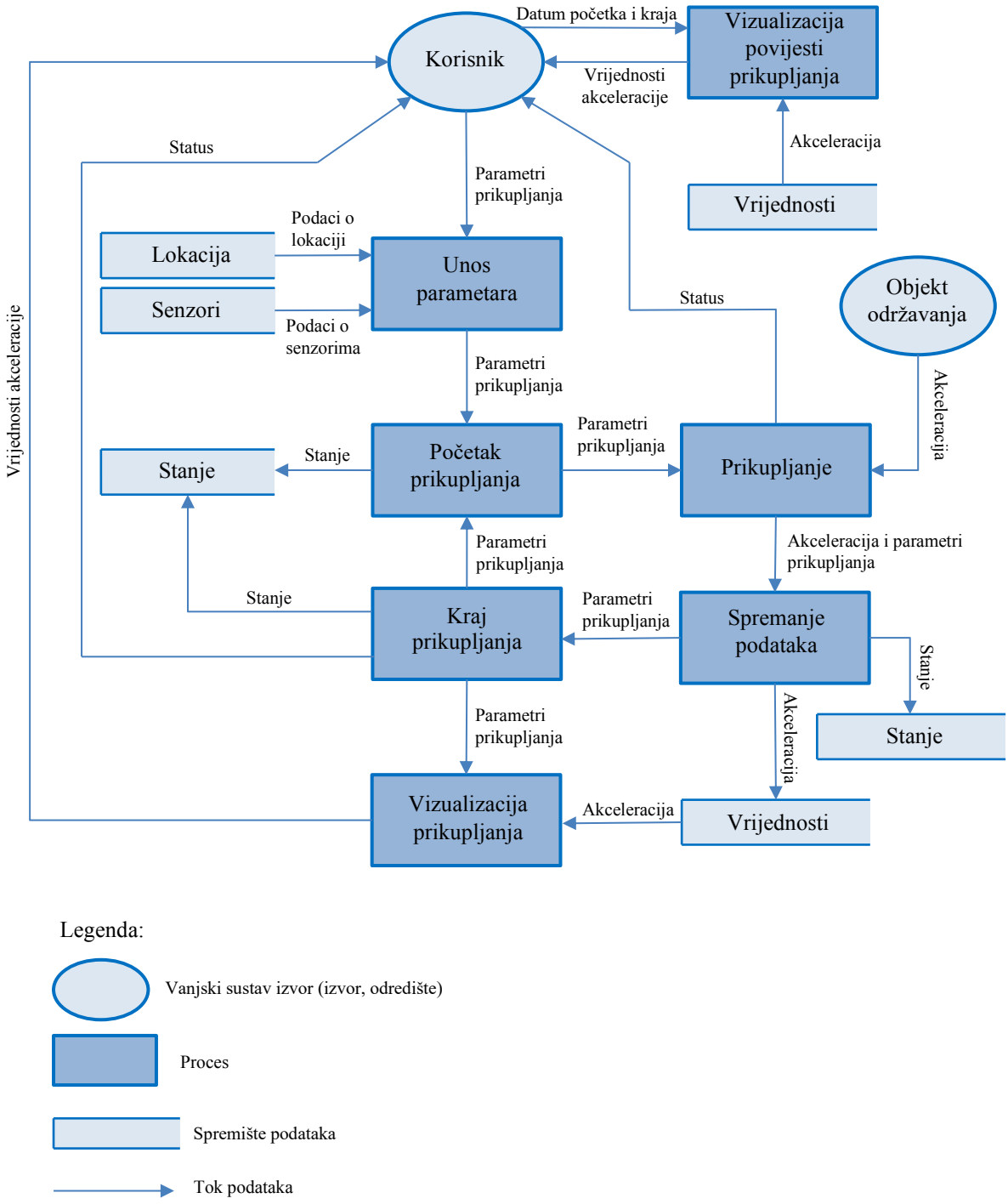
Automatizirani sustav prikupljanja podataka koristit će *Microsoft SQL Server Express* sustav za upravljanje bazom podataka.

4.3 Modeliranje logike procesa

U ranijim razinama metodologije projektiranja, gdje je bila riječ o zahtjevima na sustav, postojećem sustavu i tehničkim potrebama sustava, nije se spominjao logički dizajn sustava. Logički dizajn sustava predstavlja početak logičkog oblikovanja sustava unutar kojeg se detaljno definiraju procesi, tokovi podataka, struktura podataka i na kraju fizički dizajn podataka. Jedan od načina opisivanja procesa je dijagram toka podataka. Dijagram toka podataka je grafički način reprezentacije procesa sustava te specificira što će sustav raditi na logičkoj razini. Dijagram toka podataka sastoji se od ulazno – izlaznih tokova podataka (zaslona, senzora), vanjskih objekata koji šalju ili primaju tokove podataka od sustava (objekti održavanja), procesa sustava (programa) koji transformiraju ulazne tokove u izlazne tokove podataka i spremišta podataka (baza podataka) koje čuvaju podatke potrebne za izvršenje procesa ili sprema podatke koji su dobiveni kao rezultat rada procesa. [8] Detaljniji opis elemenata dijagrama toka podataka su sljedeće [25]

1. Proces – proces je transformacija podataka, uzima se tok podataka kao ulaz te se zatim odvijaju određene operacije nad podacima da se dobije drugačiji izlaz podataka. Procesi moraju imati barem jedan ulaz i izlaz podataka. Procesi se smatraju sve operacije osim onih koji ne mijenjaju tok podataka.
2. Tok podataka – tok podatak predstavlja tok informacija te spaja procese, spremišta podataka te vanjske tokove. Strelica toka pokazuje smjer kretanja informacija, a tokovi podataka mogu biti odvojeni ili objedinjeni.
3. Vanjski sustavi – predstavljaju nastanak ili destinaciju toka podataka, pasivan je i ima zadatak samo slati ili primati podatke. Vanjski tokovi su u vezi s promatranim sustavom no njihovi procesi se ne promatraju [8].
4. Spremište podataka – trajno čuva podatke, koriste ga procesi za pohranjivanje, čitanje, brisanje i osvježivanje podataka te svako skladište podataka treba odgovarati određenom entitetu u dijagramu entiteti-veze.

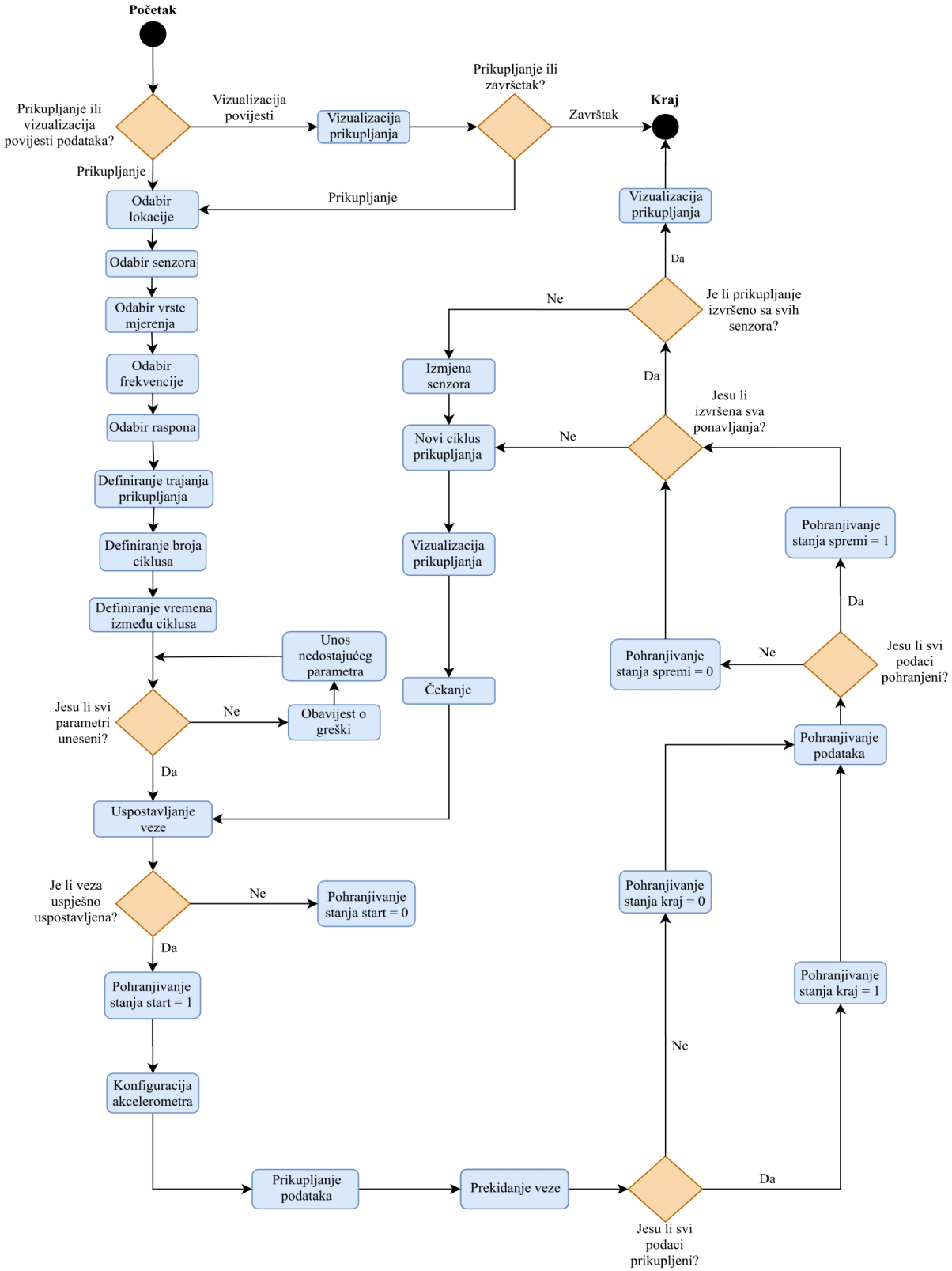
Nakon definiranja elemenata i njihovih uloga u dijagramu toka podataka slijedi konstrukcija dijagrama toka podataka. Dijagram toka procesa nalazi se na slici 18.



Slika 18. Dijagram toka podataka

Tok podataka započinje i završava s korisnikom. Korisnik unosi parametre u sustav koji mu na kraju isporučuje informacije u obliku obavijesti da su podaci prikupljeni te također i u obliku vizualizacije prikupljenih podataka. Kod procesa unosa parametara, sustav koristi podatke iz baze podataka (podaci o lokaciji modula i senzora te parametrima povezivanja) koje zatim korisnik može odabrati. Nakon unosa parametara slijedi proces početka prikupljanja u kojem se uspostavlja veza sa IIoT senzorom te se na temelju uspješne ili ne uspješne uspostave veze bilježi stanje u bazu. Bilježeno stanje može biti 1 što znači uspjeh ili 0 što znači da je došlo do greške prilikom povezivanja. Ako je veza uspješno uspostavljena prikupljanje započinje. Proces prikupljanja dohvaća podatke sa IIoT senzora koji je u interakciji s vanjskim elementom – objektom održavanja. Tijekom prikupljanja korisniku se vraćaju povratne informacije o statusu prikupljanja (ciklus, trenutno stanje, trenutni senzor...), a kada je prikupljanje završeno prikupljene vrijednosti odlaze na proces spremanja podataka. Spremanje podataka se izvršava tako da se prikupljeni podaci o akceleraciji pohranjuju u bazu podataka, a ovisno o uspješnosti pohrane bilježi se i stanje o procesu pohranjivanja (uspjeh ili neuspjeh). Po završetku spremanja podataka slijedi proces kraja prikupljanja gdje se procjenjuje, ovisno o parametrima prikupljanja, je li potrebno izvršiti dodatna mjerenja ili su sva mjerenja izvršena. Ako je potrebno izvršiti dodatna mjerenja onda se opet aktivira proces početka prikupljanja te se ponavlja postupak. Pri kraju svakog mjerenja prikupljeni podaci se vizualiziraju kako bi korisnik dobio osjećaj o prikupljenim vrijednostima akceleracije. Korisnik također ima mogućnost pregleda povijesti prikupljenih podataka pri čemu se aktivira proces vizualizacije povijesnih podataka koji dohvaća podatke iz baze podataka ovisno o korisnikovom unosu datuma početka i datuma kraja prikupljanja.

Dijagram toka podataka daje uvid u procese, tok podataka te interakciju sustava, no ne ulazi u detaljan opis rada procesa sustava. Kako bi se dobio dobar uvid u procesnu logiku koristi se neka od metoda modeliranja procesa. Objedinjeni jezik za modeliranje (eng. *Unified Modeling Language – UML*) je modelski jezik koji se koristi u polju razvoja programskog sučelja te pruža standardizirane načine vizualizacije procesa sustava.[18] Dijagram aktivnosti je jedan od UML metoda za grafički prikaz procesa sustava. Izvršenje koraka u dijagramu aktivnosti može biti istovremeno ili uzastopno. Aktivnost je prikazana kao okvir sa zaobljenim kutovima koji sadrži opis aktivnosti. Kontrolni tok prikazan je strelicom, a grane su prikazane kao dijamanti s višestrukim strelicama koje izlaze ili ulaze u njih. Na slici 19. prikazan je dijagram aktivnosti.



Slika 19. Dijagram aktivnosti sustava

4.4 Modeliranje podataka

Modeliranje podataka je proces koji započinje analiziranjem zahtjeva na automatizirani sustav prikupljanja, a završava izgradnjom baze podataka. Modeliranje podataka je proces koji se odvija paralelno i ovisno o modeliranju procesa. Cilj modeliranja podataka je izgradnja baze podataka koja ima minimalnu redundanciju, maksimalnu integriranost i konzistentnost podataka, odgovarajuću stabilnost i fleksibilnost strukture te pristup i iskoristivost. [8] Zahtjevi za definiranjem podataka iz konceptualnog pogleda su doveli do razvoja semantičkih tehnika modeliranja podataka. Stvarni svijet (u smislu resursa, događaja...) je simbolično definiran unutar fizičkih spremišta podataka. Semantički model podataka je apstrakcija koja definira kako se pohranjeni simboli odnose na stvarni svijet. Model mora biti istinski prikaz stvarnog svijeta. [25] Modeliranje podataka dijeli se u tri faze [8]:

1. Konceptualno modeliranje.
2. Logičko modeliranje.
3. Fizičko modeliranje.

U ovom poglavlju prikazan je konceptualni i logički model, dok je fizički model odvojen kao posebno poglavlje.

4.4.1 Konceptualno modeliranje

Konceptualni model je cjelovit, konzistentan i neredundantan opis podatak sustava. Izrađuju se kako bi se dobio razum o potrebama za informacijama koje koristi sustav, neovisan je o implementaciji, kako logičkoj (sustav za upravljanje podacima) tako i fizičkoj (baza podataka) te nastoji naći međusobni odnos između informacijskih objekata. [8] Jedan od konceptualnih modela je i model entiteti-veze (eng. *Entity-Relationship*). Naziv entiteti-veze se tako naziva jer prikazuje podatke u smislu entiteta i odnosa koji je opisan podacima. Model entiteti-veze se predstavlja dijagramom. Model entiteti-veze sadrži tri ključna koncepta: entiteta, atributa i veze. [25]

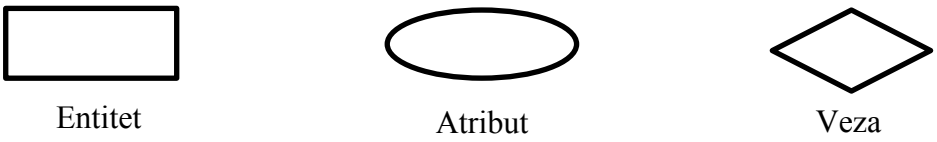
Entitet je stvaran ili apstraktan pojam o kojemu se prikupljaju i pohranjuju podaci. [8] Entiteti mogu biti ljudi, organizacije, materijalni objekti ili događaji. Atribut je opisno svojstvo ili karakteristika entiteta. Svaki atribut može poprimiti jednu vrijednost iz domene atributa. Atributi podrazumijevaju koncept ključa. Ključ je atribut ili grupa entiteta koja sadrži unikatnu vrijednost za svaku instancu entiteta. Primarni ključ je onaj koji će najčešće unikatno definirati instancu entiteta. Ostali ključevi koji nisu primarni nazivaju se alternativni ključevi [25]. Veza je prirodno udruživanje koje postoji između jednog ili više entiteta. Vezu može predstavljati

događaj koji povezuje entitete ili samo logički afinitet koji postoji između entiteta. Kardinalnost definira minimalni i maksimalni broj ponavljanja jednog entiteta koji može biti povezan s jednom pojavom drugog entiteta. Budući da su sve veze dvosmjerne za svaku je potrebno definirati kardinalnost u oba smjera. [25] Stupanj veze je broj entiteta koji sudjeluju u vezi, a postoje tri vrste odnosa [25]:

- jedan-prema-jednom (1-1),
- jedan-prema-mnogima (1-M),
- mnogi-prema-mnogima (M-M).

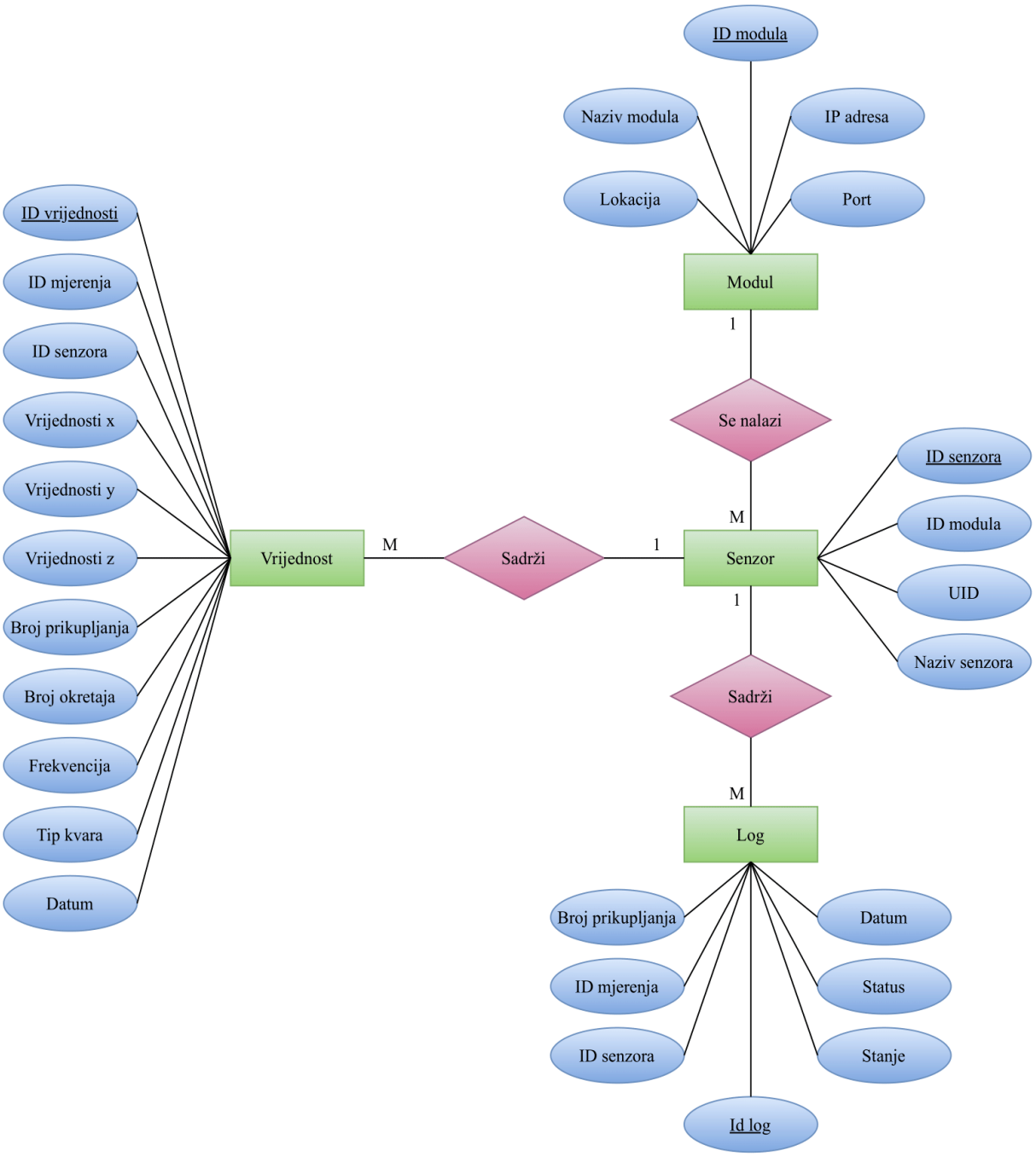
Odnos jedan-prema-jednom znači da jedna instanca entiteta pripada samo jednoj instanci drugog entiteta. Na primjer, broj osobne iskaznice može biti povezan samo s jednom osobnom i obrnuto. Odnos jedan-prema-mnogima znači da je jedna instanca entitet povezana s jednom ili više instanci drugog entiteta. Na primjer, otac može imati više sinova, no sin može imati samo jednog oca. Odnos mnogi-prema-mnogima zahtjeva izradu nove tablice kako bi se pohranila veza. Na primjer profesor može držati predavanja mnogim studentima, a studenti mogu slušati predavanja od više profesora. Zbog toga je potrebno napraviti novu tablicu predavanje koje sadrži strane ključeve iz obje tablice [25].

Dijagram modela entiteti-veze omogućava lakše razumijevanje modela podataka. Postoji nekoliko vrsta dijagrama entiteti-veze od kojih se najčešće primjenjuju Chenov i Martinov dijagram. Dijagram entiteti-veze mora sadržavati podatke potrebne za modeliranje sustava, entitete, atribute za svaki entitet, ključeve za svaki entitet, veze između entiteta te kardinalnost [8]. Elementi dijagrama su prikazani na slici 20.



Slika 20. Elementi dijagrama entiteti-veze [25]

Dijagram entiteti-veze za automatizirani sustav prikupljanja podataka nalazi se na slici 21.



Slika 21. Dijagram entitet-veza za automatizirani sustav prikupljanja podataka

Dijagram se tumači na sljedeći način; entitet Vrijednost sadrži atribute koji predstavljaju informacije o prikupljanju kao što su vrijednost x, y i z, senzor s kojega se prikupljalo, broj prikupljanja, broj okretaja, frekvencija, tip kvara i datum. Entitet Vrijednost je u vezi s entitetom Senzor, a odnos među njima je jedan-prema-mnogo što znači da jedna instanca vrijednosti može imati samo jedan senzor, a s druge strane više instanci senzora se može naći u jednoj instanci vrijednosti. Entitet Senzori, koji sadrži atribute poput identifikacije modula, vlastiti unikatni

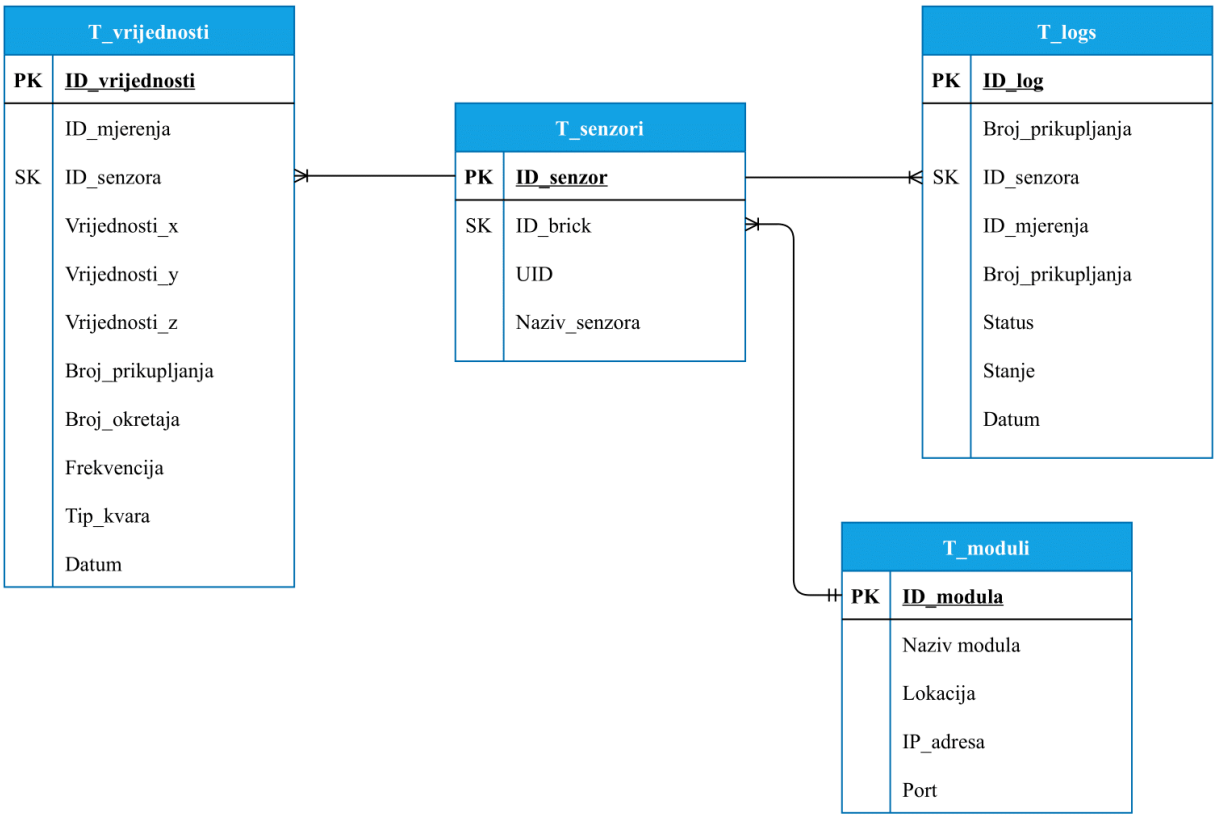
identifikator (UID) te svoj naziv, je u vezi s entitetom Modul koji sadrži atribute naziv modula, lokacija modula te IP i Port. Veza između entiteta Senzor i entiteta Modul je također jedan-prema-mnogo što znači da modul može sadržavati više senzora dok s druge strane senzor može pripadati samo jednom modulu. Entitet Senzori je također u vezi sa entitetom Logs. Entitet Logs sadrži atribute identifikatora mjerenja i senzora, broj prikupljanja, stanje, status i datum. Veza između entiteta Senzor i entiteta Logs je jedan-prema-mnogo što znači da instanca bilježenja stanja može sadržavati više senzora dok je samo jedan senzor u jednoj instanci bilježenja stanja.

4.4.2 Logičko modeliranje

Logičko modeliranje je pretvorba konceptualnog modela u logičku shemu baze podataka. Logički model ne rezultira razradom konačne fizičke strukture, a opisuje se pravilima određenog sustava za upravljanje bazom podataka. Najčešće se za opis logičkog modela koristi relacijski model koji se temelji na skupu matematičkih principa koji su proizašli iz teorije setova i predikatne logike. [8] Relacijskim modelom definirani su [8]:

- struktura baze podataka,
- integritet baze podataka,
- relacijska algebra.

Kako bi se riješio problem redundantnih, suvišnih i ponavljajućih podataka koriste se tehnike normalizacije baze podataka. Najčešće se koriste 3 normalne forme. Entitet u prvoj normalnoj formi (1NF) je onaj čiji atributi poprimaju samo jednu vrijednost za svaku instancu entiteta. Entitet je u drugoj normalnoj formi (2NF) ako je već u prvoj normalnoj formi i ako vrijednosti svih atributa neprimarnog ključa potpuno ovise o primarnom ključu, a ne samo o njegovom dijelu. Sve neključne atribute koji ovise samo o dijelu primarnog ključa treba premjestiti na mjesta gdje one potpuno ovise o primarnom ključu. Entitet je u trećoj normalnoj formi ako je ujedno i u drugoj normalnoj formi te ako posjeduje vrijednosti neprimarnih ključeva koje ne ovise ni o kojoj drugoj vrijednosti neprimarnih ključeva. [25] Struktura baze podataka može se vidjeti na slici 22.



Slika 22. Struktura baze podataka

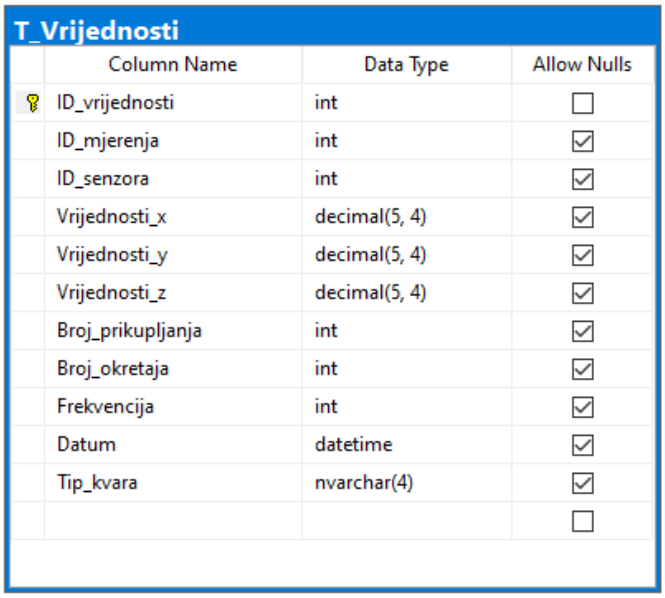
Integritet baze podataka se osigurava konceptom referencijalnog integriteta koji osigurava da podaci koji su u relacijskoj vezi između pojedinih tablica baze ostanu konzistentni. Svaka promjena relacijskog polja jedne tablice izaziva promjenu vrijednosti relacijskog polja druge tablice. Referencijalni integritet se ostvaruje vezama između tablica tako da se povežu tablice koje sadrže međusobno zavisne atribute (primarni ključ jedne tablice koji je strani ključ u drugoj tablici). [8] Na slici 22. je prikazana veza između tablica koje sadrže ključeve kao i kardinalnost veze. Oznakom PK su označeni atributi koji su primarni ključ dok su oznakom SK označeni strani ključevi. Integritet baze podataka se također postiže transakcijskim integritetom koji osigurava istovremene promjene u više tablica tako da podaci u svim tablicama ostanu konzistentni. U slučaju pojave neke greške prilikom operacija nad bazom podataka sadržaj svih tablica baze se vraća u početno stanje. Primjeri transakcija kao i relacijska algebra prikazani su u sljedećem poglavlju.

4.5 Fizičko modeliranje

Fizičko modeliranje je realizacija konceptualnog i logičkog modela podataka u obliku izrade baze podataka. Izrađuje se fizički dizajn podataka te kako je već napomenuto u poglavlju 4.2.5. za izradu i upravljanje bazom podataka koristi se *Microsoft SQL Server Management Studio 2018*. Izrada baze podataka započinje s izradom tablica. Tablice podataka se temelje na strukturi baze podataka (slika 17.) tako da baza podataka sadrži 4 tablice:

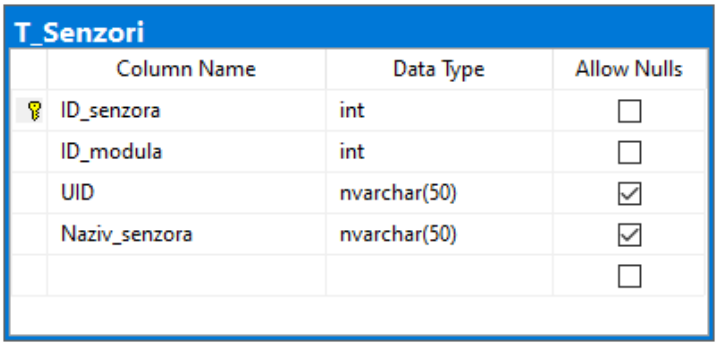
- tablica T_Vrijednosti,
- tablica T_Moduli,
- tablica T_Senzori,
- tablica T_Logs.

Struktura tablice T_Vrijednosti prikazana je na slici 23.



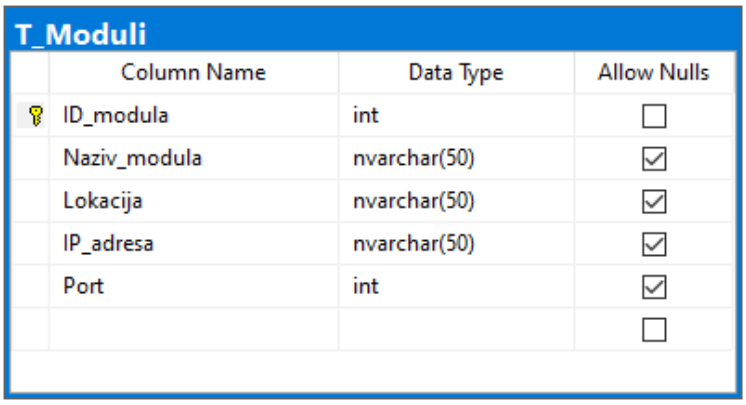
Slika 23. Struktura tablice T_Vrijednosti

Atributi ID_vrijednosti, ID_mjerenja, ID_senzora, Broj_prikupljanja, Broj_okretaja te Frekvencija su podatkovnog tipa *integer* što znači da poprimaju cjelobrojne vrijednosti. Atributi Vrijednosti_x, Vrijednosti_y te Vrijednosti_z su podatkovnog tipa *decimal(5,4)* što znači da se u njih pohranjuju decimalni brojevi koji imaju jedno decimalno mjesto ispred točke i 4 decimalna mjesta iza točke (npr. 1,2345). Atribut Datum je podatkovnog tipa *datetime*, a Tip_kvara *nvarchar(4)* što znači da može sadržavati najviše 4 znaka (npr. COMB). Struktura tablice T_senzori prikazana je na slici 24.



Slika 24. Struktura tablice T_Senzori

Atributi ID_senzora i ID_modula su podatkovnog tipa *integer* dok su atributi UID i Naziv_senzora podatkovnog tipa *nvarchar(50)* što predstavlja varijabilno tekstualno polje do 50 znakova. Struktura tablice T_Moduli prikazana je na slici 25.



Slika 25. Struktura tablice T_Moduli

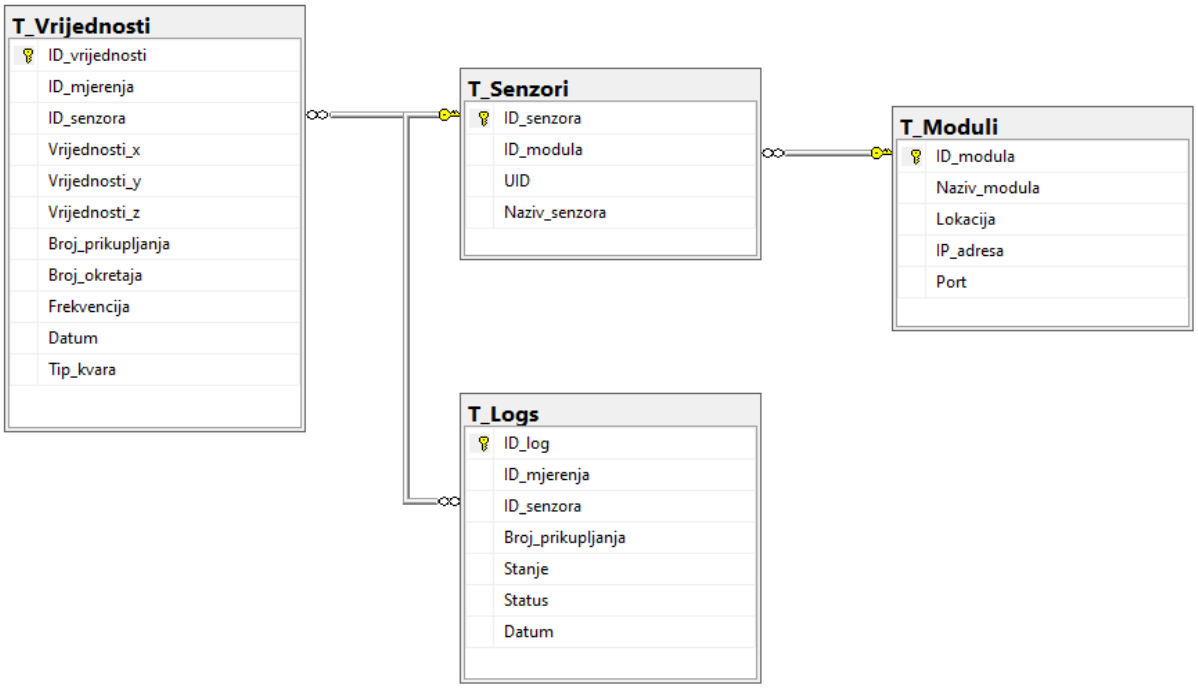
Atributi ID_modula i Port su podatkovnog tipa *integer* dok su atributi Naziv_modula, Lokacija i IP_adresa podatkovnog tipa *nvarchar(50)* što predstavlja varijabilno tekstualno polje do 50 znakova. Struktura tablice T_Logs prikazana je na slici 26.

T_Logs

	Column Name	Data Type	Allow Nulls
🔑	ID_log	int	☐
	ID_mjerenja	int	☑
	ID_senzora	int	☐
	Broj_prikupljanja	int	☑
	Stanje	nvarchar(50)	☑
	Status	int	☑
	Datum	datetime	☑
			☐

Slika 26. Struktura tablice T_Logs

Atributi ID_log, ID_mjerenja, ID_senzora, Broj_prikupjanja i Status su podatkovnog tipa *integer*, atribut Stanje je podatkovnog tipa *nvarchar (50)* te je Datum podatkovnog tipa *datetime*. Tablice u bazi podataka i njihove veze mogu se prikazati pomoću dijagrama baze podataka. Dijagram baze podataka prikazan je na slici 27.



Slika 27. Dijagram baze podataka

Baze podataka koriste zajednički deklarativni jezik SQL koji služi za stvaranje i definiranje strukture podataka, uređivanja prava korisnika, pristup i upravljanje sadržajem baze. *Microsoft SQL Server* koristi vlastitu inačicu SQL-a koja se naziva T-SQL [26]. Postoje različite vrste upita nad bazom podataka od kojih neki mogu biti upiti za prikazivanje podataka,

upiti za definiciju objekata, upiti za upravljanje objektima te upiti za upravljanje podacima. Blokovi SQL koda koji mogu izvršavati neke aktivnosti na poslužitelju nazivaju se uskladištene procedure (eng. *Stored procedures*). Procedura može sadržavati ulazne parametre, naredbe za kontrolu toka kao i naredbe za rukovanje iznimkama. Osnovni ciljevi uskladištenih procedura su povećati brzine prijenosa podataka unutar mreže, prevođenje upita te povezivanje tablica [26]. Kako bi sustav funkcionirao baza podataka treba sadržavati uskladištene procedure za selekciju i umetanje podataka. Procedure koje sadrži baza podatak su sjedeće:

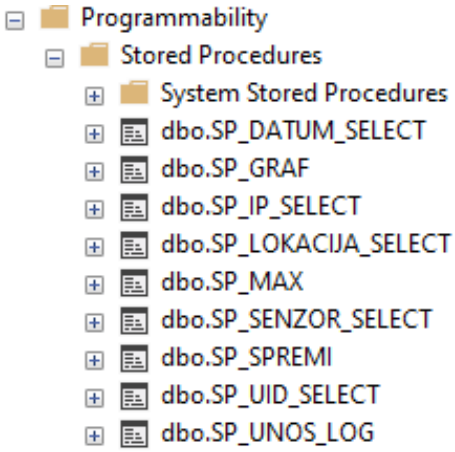
- procedura za selekciju podataka ovisno o datumima,
- procedura za selekciju podataka za graf,
- procedura za selekciju IP adrese i Porta,
- procedura za selekciju lokacije,
- procedura za selekciju maksimalne vrijednosti ID_prikupljanja,
- procedura za selekciju senzora,
- procedura za spremanje prikupljenih podataka,
- procedura za selektiranje UID senzora,
- procedura za spremanje stanja prikupljanja.

Kako bi se tijekom izvršavanja procedura zadržao integritet podataka koriste se transakcije. Transakcije su programski moduli koji mogu sadržavati više međusobno povezanih SQL naredbi. Transakcija predstavlja logičku cjelinu rada nad bazom podataka pri čemu se transakcija izvršava u cijelosti ili se transakcija poništava i sve njene posljedice nestaju [26]. Primjer transakcije i uskladištene procedure:

```
CREATE PROCEDURE [dbo].[SP_SPREMI]
--Definicija ulaznih parametara i njihovih tipova podataka
      @ID_mjerenja int,
      @x decimal(5,4),
      @y decimal(5,4),
      @z decimal(5,4),
      @broj_prikupljanja int,
      @broj_okretaja int,
      @frekvencija int,
      @tipkvara nvarchar(4),
      @id_senzora int,

      @GRESKA int OUTPUT
AS

BEGIN TRAN -- POCETAK TRANSAKCIJE

-- 3. Spremanje podataka u tablicu T_Vrijednosti
```

```
INSERT INTO
dbo.T_Vrijednosti(ID_mjerenja,Vrijednosti_x,Vrijednosti_y,Vrijednosti_z,Broj_priku
pljanja,Broj_okretaja,Frekvencija,Datum,Tip_kvara,ID_senzora)

VALUES
(@ID_mjerenja,@x,@y,@z,@broj_prikupljanja,@broj_okretaja,
@frekvencija,GETDATE(),@tipkvara,@id_senzora)

-- 5. Provjera je li je doslo do greske prilikom spremanja podataka
IF @@ERROR <> 0 -- Ako je broj breske <> 0 transakcija spremanja podataka nije
uspjela.
	BEGIN ROLLBACK TRAN-- Rollback -> Baza se vraca u prvotno stanje.
	SELECT @GRESKA = @@ERROR
	RETURN @GRESKA
END

COMMIT TRAN -- KRAJ TRANSAKCIJE

SELECT @GRESKA = @@ERROR
RETURN @GRESKA  -- Ako je transakcija spremanja podatak uspjela GRESKA=0.
```

Na isti način se izvode i ostale procedure koje su potrebne za rad sustava. Na slici 28. prikazan je popis svih procedura. Relacijske operacije koje se vode nad bazom su tipa naredbe *SELECT* i *INSERT* into što znači zahvaćanje podataka iz tablica i umetanje u tablice.



Slika 28. Pregled kreiranih uskadištenih procedura

Nakon izrade baze podataka i uskladištenih procedura slijedi popunjavanje tablica s podacima za rad sustav. Popunjavaju se tablice T_Senzori i T_Moduli dok će tablice T_Vrijednosti i T_Logs biti popunjavane automatski pri radu sustava.

Ovim poglavljem završen je teoretski dio opisivanja sustava u kojem su definirani zahtjevi, odabrane tehničke varijante komponenti sustava te dani detaljni opisi logike procesa i podataka kako bi se dobio uvid u funkcionlalnost sustava. Analiza i oblikovanje sustava ključni su koraci pri razvoju informacijskih sustava jer se njima postižu preduvjeti za izradu sustava koja je opisana u sljedećem poglavlju.

5 IZRADA SUSTAVA

Analizom i oblikovanjem sustava definirani su zahtjevi i arhitektura sustava, odabrane su tehničke varijante su detaljno opisani i objašnjeni procesi i podaci koje sustav koristi. Nakon analize i oblikovanja sustava na red dolazi i izrada sustava. Izrada automatiziranog sustava prikupljanja podataka podijeljena je u tri faze:

1. Konfiguracija rubnog čvora i IIoT senzora
2. Izrada i programiranje web aplikacije.
3. Testiranje sustava.

Nulta faza izrade sustava je izrada baze podataka i procedura što je obrađeno u prethodnom poglavlju. U prvoj fazi konfigurira se Tinkreforge rubni čvor kako bi se dobila funkcionalnost IIoT senzora i rubnog čvora, zatim se izrađuje web aplikacija koja će ostvarivati komunikaciju i prikupljati podataka s IIoT senzora te se po završetku izrade web aplikacije sustav testira. Računalo na kojem će se izraditi i testirati sustav je Lenovo ThinkPad T430 s karakteristikama koje su navedene u tablici 11.

Tablica 11. Tehničke karakteristike Thinkpada T430 računala [27]

Karakteristika	**Vrijednost**
Operativni sustav	Windows 10 Pro
Procesor	Intel Core i5-3320M @2,60 GHz
Radna memorija	12 GB DDR3
Tvrdi disk	256 GB SSD
Grafika	Intel HD Graphics
Zaslon	14.0” HD (1366 x 768) (200 NITS)
Broj Ethernet priključaka	1
Broj USB utora	3

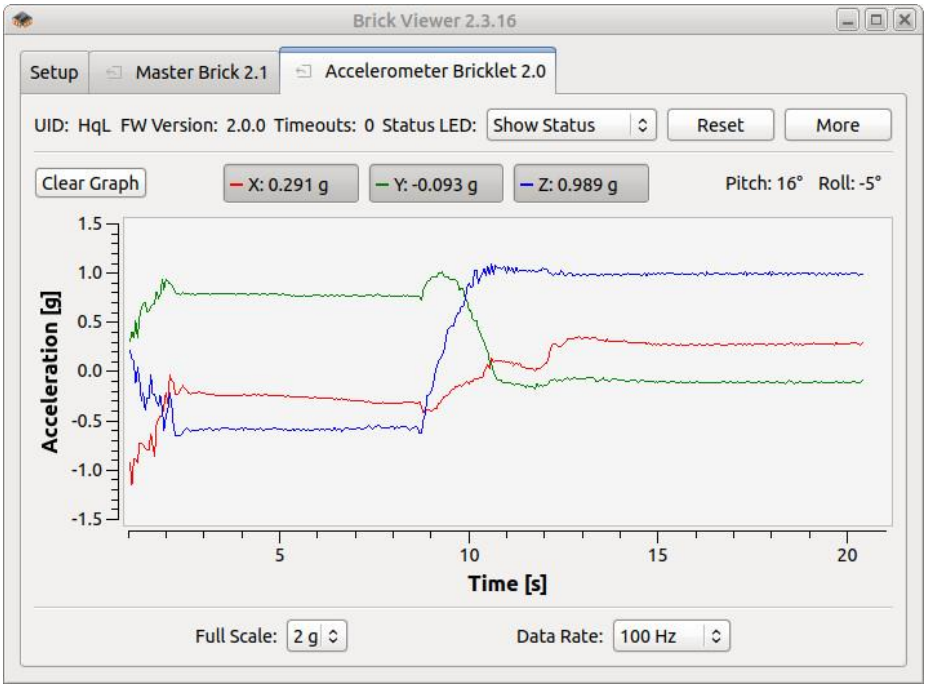
Izrada sustava započinje konfiguriranjem Tinkerforge rubnog čvora i IIoT senzora čime se postiže veza s okolinom te omogućuje prikupljanje podataka o objektima održavanja.

5.1 Tinkerforge konfiguracija

Prvi korak pri konfiguraciji i testiranju modula je instalacija potrebne programske podrške na prijenosno računalo. Programi koji su potrebni za uspostavljanje veze su *Brick Daemon* i *Brick Viewer* koji se preuzimaju sa Tinkerforge službene stranice. Moduli se testiraju zasebno kako bi se dobio uvid u njihovu ispravnost.

5.1.1 Konfiguracija i testiranje modula

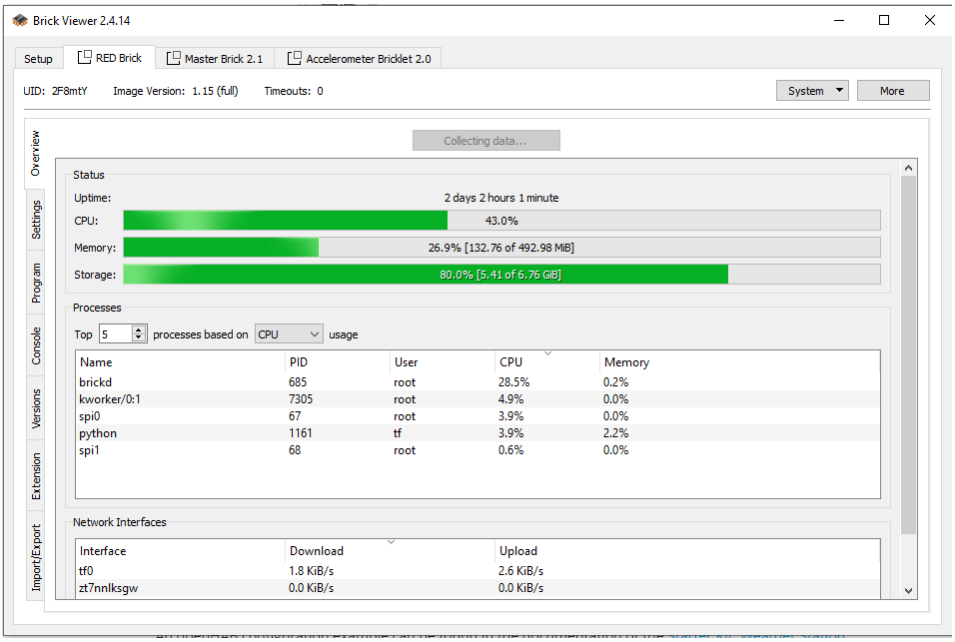
Nakon instalacije programa pokreće se *Brick Daemon* i *Brick Viewer* te se glavni modul spaja na računalo pomoću Mini-USB kabla. Nakon što je glavni modul spojen s računalom, pomoću *Brick Viewera* se ostvaruje veza s istim. Na *Brick Vieweru* se prikazuje novi prozor pod imenom „*Master Brick*“. Također, spajanjem akcelerometra na glavni modul pojavljuje se dodatni prozor „*Accelerometer“* kao što je prikazano na slici 29.



Slika 29. Izgled *Brick Viewer* sučelja [18]

Nakon uspješnog testa glavnog modula i akcelerometra prekida se veza sa glavnim modulom te se odspaja Micro-USB kabel kako bi se testirao RED modul. Prije spajanja RED modula na računalo na Micro-SD karticu se stavlja *RED Brick Image* koji je preuzet s Tinkerforge službene stranice. *RED Brick Image* sadrži operativni sustav koji se na Micro-SD karticu stavlja pomoću Micro-SD adaptera i programa *Etcher*. Po završetku procesa na Micro-SD karticu je stavljena posljednja dostupna verzija Debian operativnog sustava. Nakon što se Micro-SD kartica izvadi iz računala i adaptera, stavlja se u Micro-SD utor na RED modulu koji

se nalazi ispod Mini-USB priključka. Nakon umetanja SD kartice RED modul se spaja na računalo putem Mini-USB kabla te se pomoću *Brick Viewera* ostvaruje veza s RED modulom. Izgled sučelja *Brick Viewera* sa spojenim RED modulom prikazan je na slici 30.

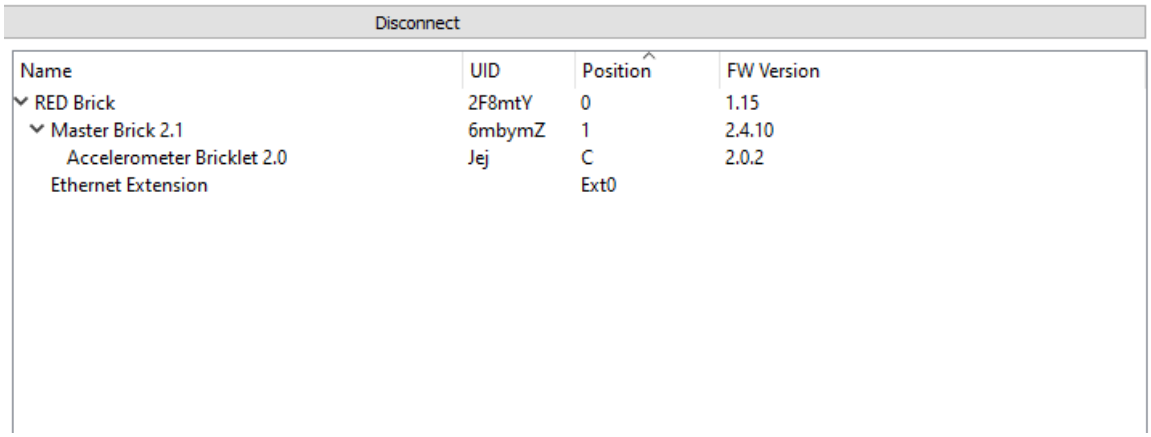


Slika 30. Izgled sučelja RED modula

Nakon testiranja, RED modul se isključuje iz rada te se odspaja s računala. Ethernet ekstenzija se testira tako da se priključi na glavni modul te se ponavlja postupak kao kod spajanja glavnog modula. U *Brick Vieweru* pod prozorom *„Master brick“* se pojavljuje novi blok pod nazivom *„Ethernet Extension“*. Nakon uspješnog testiranja pojedinačnih modula vrijeme je za spajanje modula. Spajanje u konfiguraciju postiže se tako da se na RED modul spoji glavni modul, a zatim se na glavni modul spaja Ethernet ekstenzija. Dalje se spaja:

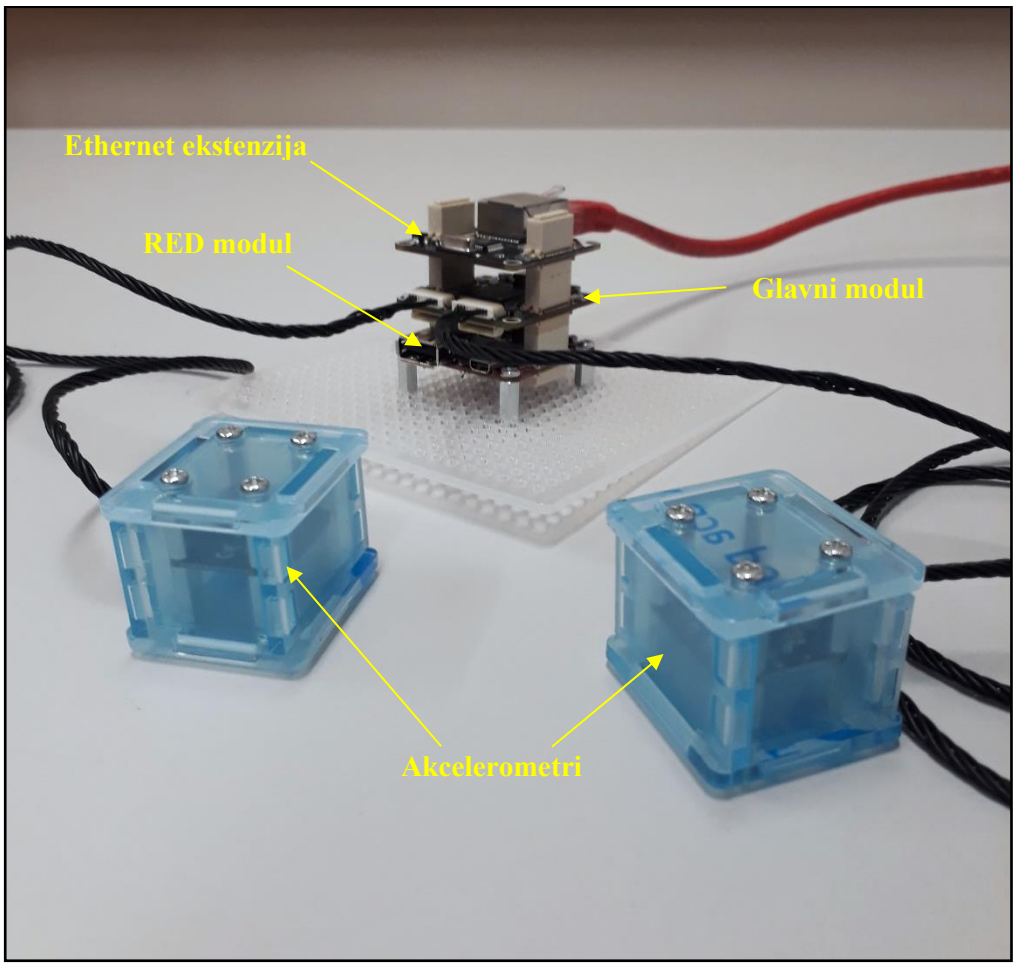
- pomoćni modul akcelerometar na glavni modul,
- ruter i Ethernet ekstenzija pomoću UTP kabla,
- RED modul i računalo pomoću Mini-USB kabla.

Nakon spajanja pokreće se *Brick Viewer*. Na zaslonu trebaju biti vidljivi svi dijelovi konfiguracije (Slika 31.).



Slika 31. Prikaz konfiguracije modula u Brick Viewer-u

Izgled Tinkerforge konfiguracije može se vidjeti na slici 32.

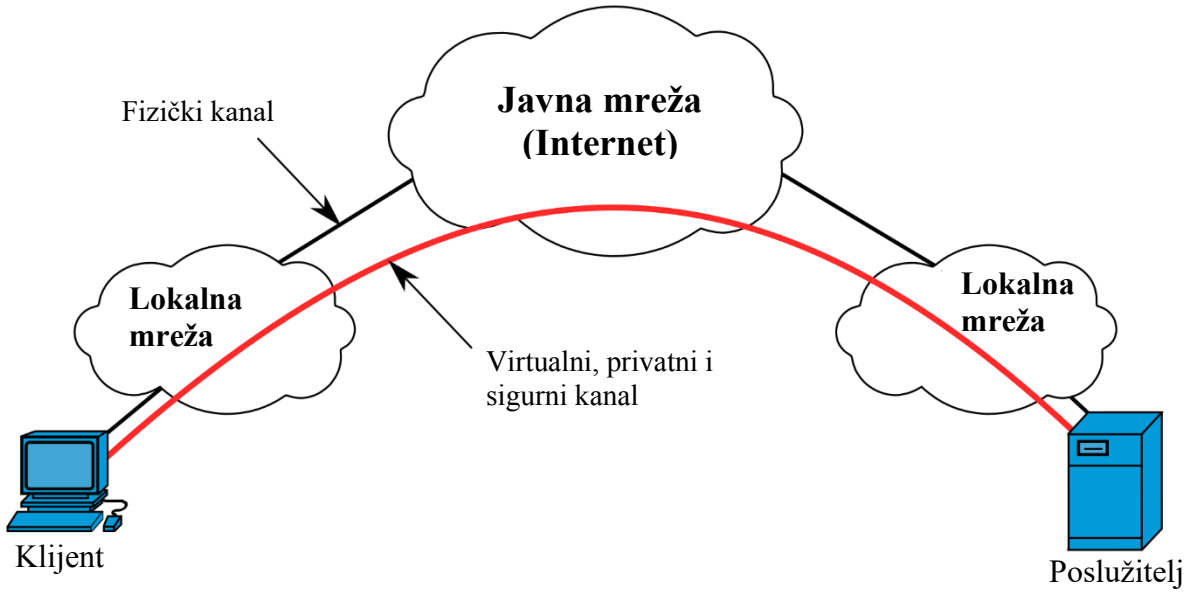


Slika 32. Tinkerforge konfiguracija

5.1.2 Konfiguracija LAN mreže

Nakon spajanja modula u konfiguraciju i uspostavljanja veze s računalom potrebno je na ruteru konfigurirati DHCP za Tinkerforge kako bi uvijek imao istu IP adresu. Prijavom na ruter, na koji je spojena Ethernet ekstenzija, u postavkama se Tinkerforgeu dodjeli DHCP IP adresa po želji. Računalo se također spaja na ruter pomoću UTP kabla što omogućuje povezivanje RED modula s računalom pomoću *Brick Viewera.* Kako bi se moglo pristupiti RED modulu i dohvaćati podatke s akcelerometra potrebno je dodijeliti WebSocket port u postavkama RED modula. Dodjeljuje se port 8081.

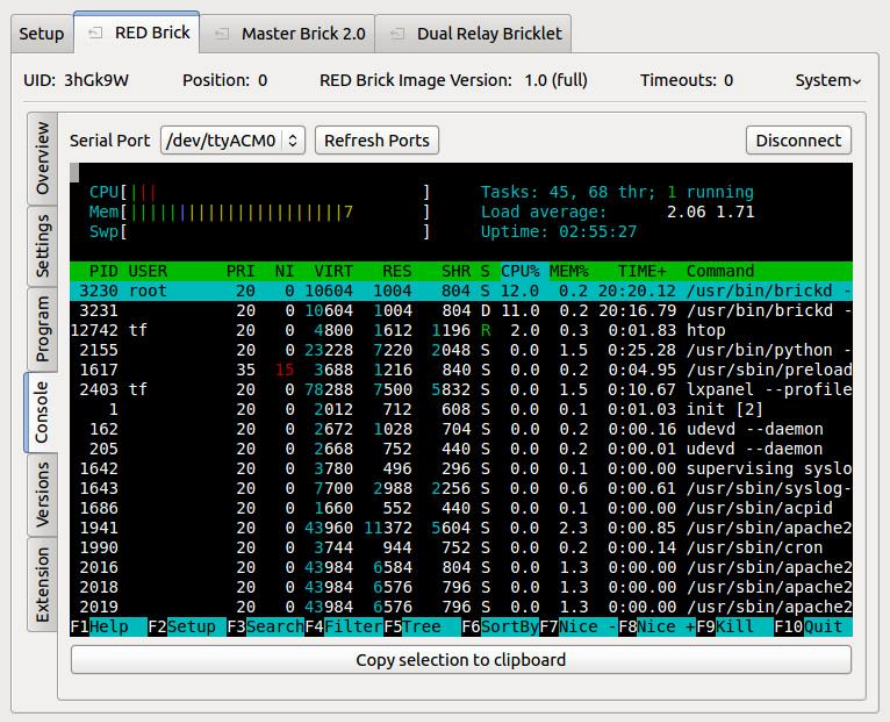
Kako bi se RED modulu moglo pristupiti s bilo koje lokacije potrebno je konfigurirati virtualnu privatnu mrežu. Virtualna privatna mreža (VPN) proširuje privatnu mrežu preko javne mreže tako što oponaša privatnu mrežu te time omogućuje slanje i primanje podataka putem javnih mreža. Virtualna privatna mreža se stvara uspostavljanjem *point-to-point* veze što omogućava korisniku s jedne strane mreže da pristupi resursima s druge strane mreže (Slika 33.) [28].



Slika 33. Shema virtualne privatne mrže [29]

Kako bi se dobila funkcionalnost virtualne privatne mreže koristi se *ZeroTier* programska okolina. Ona pruža napredne mogućnosti virtualizacije i upravljanja mrežom na lokalnim i širokim mrežama povezujući gotovo bilo koju vrstu aplikacije ili uređaja. [30] Prvi korak konfiguracije je kreiranje virtualne mreže pomoću *ZeroTier Central* korisničkog sučelja. Kreiranjem mreže stvara se 40-bitni mrežni identifikator koji služi za pridruživanje uređaja. Nakon kreiranja mreže, instalacijom *ZeroTier One* programa na računalo omogućuje se

pridruživanje uređaja mreži unošenjem mrežnog identifikatora. Drugi klijent za spajanje na mrežu je RED modul. Spajanjem RED modula na računalo pomoću Mini-USB kabla moguće je pristupiti operativnom sustavu preko *Brick Viewer*-a. Prozor koji sadrži podatke o RED modulu kao i njegove postavke također sadrži potprozor pod nazivom *„Console“*. Pomoću njega se može odabrati serijski port na kojem je spojen RED modul. Spajanjem na serijski port RED modula otvara se konzolna zapovjedna linija pomoću koje se može konfigurirati Debian Linux operativni sustav.



Slika 34. Konzolna zapovjedna linija RED modula [17]

Sljedeće što slijedi je upisivanje naredbi za instalaciju *ZeroTier One* klijenta. Postupak je sljedeći:

1) $ sudo apt update && sudo apt upgrade
2) $ sudo apt install curl
3) Verifikacija curl verzije:

   $ curl --version
4) Instalacija ZeroTier klijenta:

   $ curl -s https://install.zerotier.com | sudo bash
5) Provjera statusa, koji treba biti aktivan:

   $ sudo systemctl status zerotier-one
6) Pridružiti klijenta mreži (unos mrežnog identifikatora):

```
$ sudo zerotier-cli join 35c192ce9b27864d
```

Po završetku postupka provjerava se status pomoću naredbe *sudo systemctl status zerotier-one* te se status prikazuje kao aktivan što je prikazano na slici 35.

```
tf@red-brick:~$ sudo systemctl status zerotier-one
● zerotier-one.service - ZeroTier One
   Loaded: loaded (/lib/systemd/system/zerotier-one.service; enabled; vendor pre
   Active: active (running) since Tue 2020-10-13 13:39:07 CEST; 45min ago
 Main PID: 4482 (zerotier-one)
   CGroup: /system.slice/zerotier-one.service
           └─4482 /usr/sbin/zerotier-one

Oct 13 13:39:07 red-brick systemd[1]: Started ZeroTier One.
lines 1-8/8 (END)
```

Slika 35. Status ZeroTier One klijenta

Nakon toga uređaj se pomoću *ZeroTier Central* korisničkog sučelja odobrava u mrežu te se naredbom ping testira funkcionalnost mreže. Instalacijom i konfiguracijom virtualne mreže završava konfiguracija rubnog čvora te se prelazi na izradu web aplikacije.

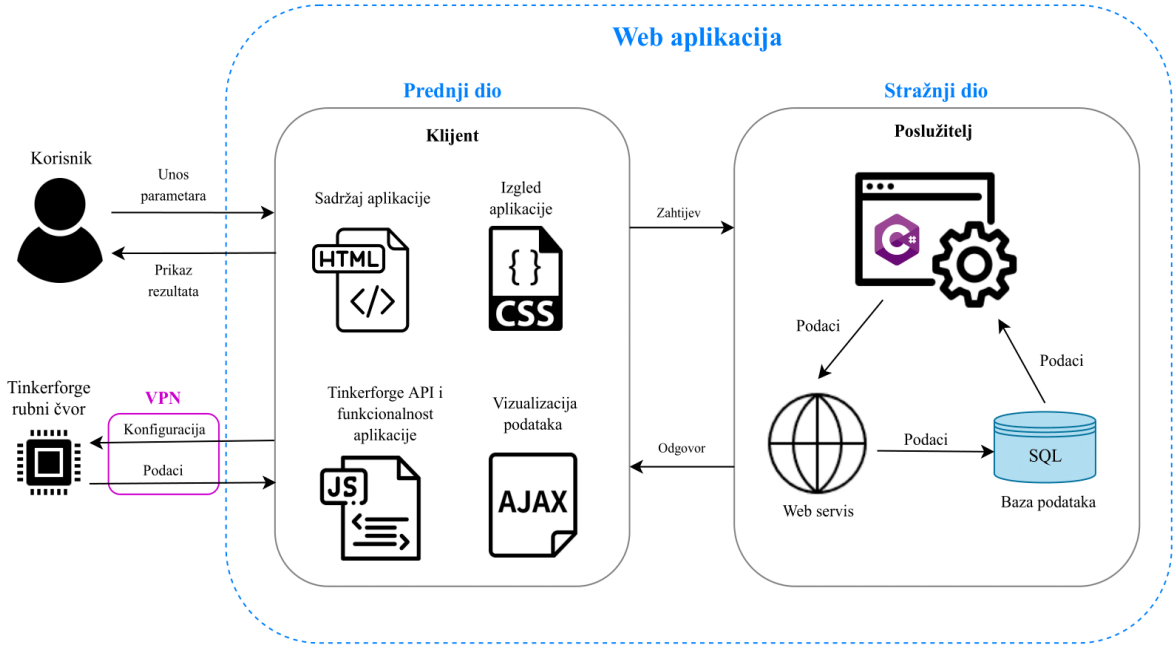
5.2 Web aplikacija

Web aplikacija je vrsta programske podrške (eng. *software*) koja se sastoji od prednjeg dijela (eng. *front-end*) preko kojega korisnik ostvaruje interakciju uz pomoć web preglednika te stražnjeg dijela (eng. *back-end*) koji pruže usluge i osigurava funkcionalnost. Postoje dvije vrste web aplikacija; statička i dinamička. Statička aplikacija koristi samo prednji dio kako bi prezentirala sadržaj korisniku dok dinamička web aplikacija koristi i prednji i stražnji dio te omogućuje izvršavanje funkcija kao i povezanost s bazom podataka. Razvoj web aplikacija zahtijeva razvojne procese koji uključuju analizu zahtijeva i programiranje kroz različite programske jezike. [31]

5.2.1 Arhitektura web aplikacije

Tinkerforge moduli mogu se kontrolirati putem aplikativnog programskog sučelja (engl. *Application Programming Interface - API*). Aplikacijsko programsko sučelje je način na koji dvije računalne aplikacije međusobno komuniciraju preko mreže (Interneta) koristeći zajednički jezik koji obje razumiju. [32] Tinkerforge API veze omogućavaju upravljanje modulima iz korisničko razvijenih programa i aplikacija. Tinkerforge API funkcionira tako da uspostavlja TCP/IP vezu između Ethernet ekstenzije i pozivatelja. Svaki poziv funkcije stvara TCP/IP paket koji se šalje preko Ethernet ekstenzije na modul. Dolazni paketi s modula se preusmjeravaju natrag pozivatelju. Tinkerforge API podržava spektar programskih jezika koji

su navedeni u poglavlju 4.2.1, a automatski sustav prikupljanja podataka će koristiti API za programski jezik JavaScript. JavaScript je interpretirani programski jezik koji omogućuje interaktivne web stranice i važan je dio web aplikacija. [33] Velika većina web stranica koristi JavaScript za upravljanje ponašanja klijentske prednje strane, a svi web preglednici sadrže JavaScript mehanizam kako bi omogućili njegovo izvršavanje. Uz HTML i CSS, JavaScript je temeljna tehnologija modernog *World Wide Web*-a. Budući da je JavaScript programski jezik koji se odvija na prednjoj strani, funkcija prikupljanja podataka odvijat će se na korisničkoj prednjoj strani dok će se na stražnjoj strani odvijati funkcija pohranjivanja podataka. Struktura web aplikacija za automatizirano prikupljanje podataka nalazi se na slici 36.



Slika 36. Struktura web aplikacije za automatizirano prikupljanje podataka

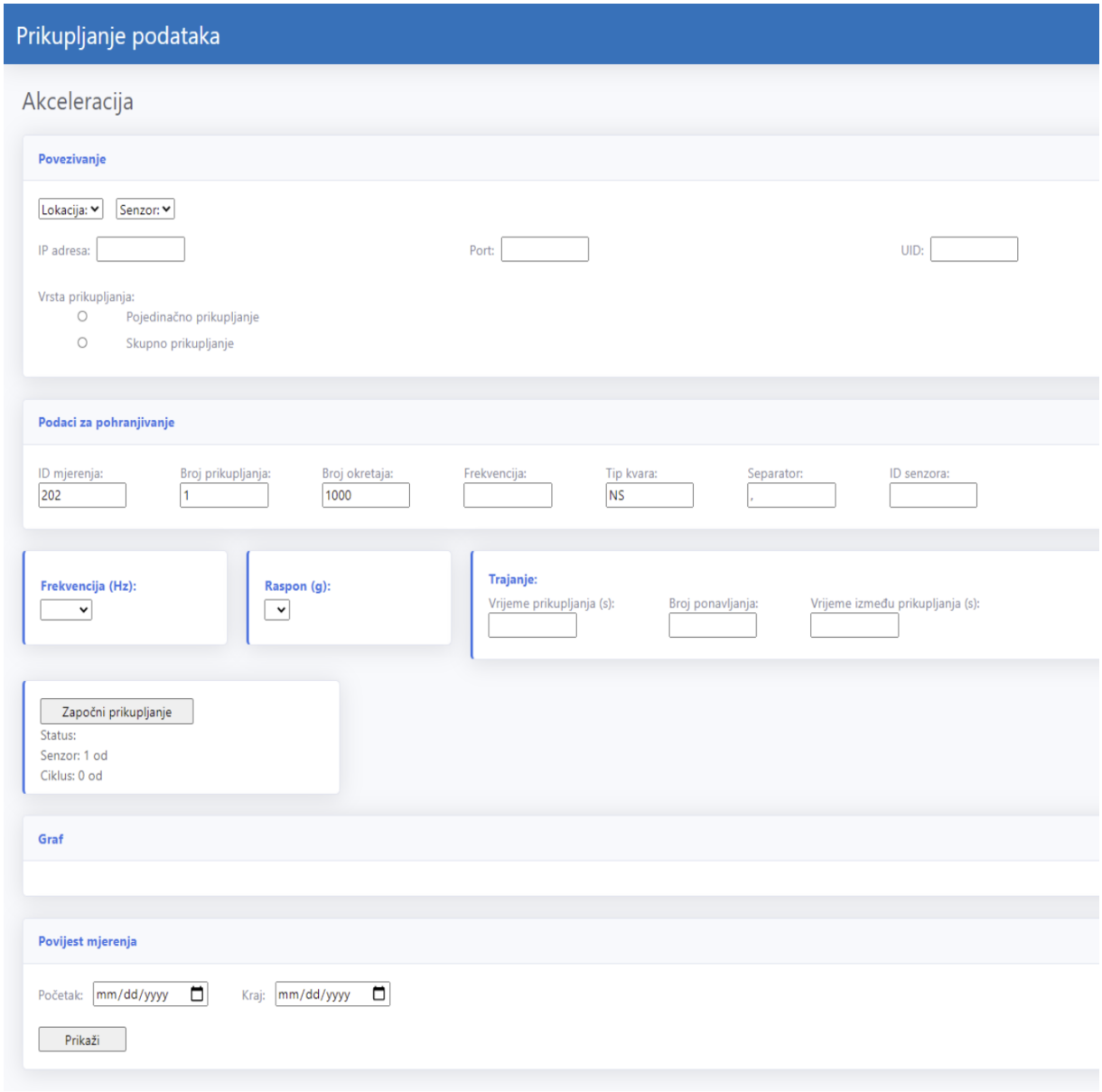
Kako je već spomenuto web aplikacija se sastoji od prednje i stražnje strane. Prednja (klijentska) strana se sastoji od HTML, CSS, JavaScripta i AJAX programskih jezika te svaki od njih služi određenoj funkciji. Funkcija HTML-a je da organizira i prikazuje sadržaj aplikacije. Sadržaj aplikacije uključuje elemente, skripte, stilove, redoslijed elementa, naziv stranice, vrsta dokumenta i sl. Kako bi se korisnicima uredno predočila funkcionalnost aplikacije koristi se CSS (eng. *Cascading Style Sheets – CSS)* jezik. Pomoću CSS-a definiraju se veličine elemenata, njihove boje, stil teksta te sve grafičke postavke korisničkog sučelja. Kako bi se web aplikacija mogla koristiti na različitim veličinama uređaja (računalu, tabletu, mobitelu...) koristi se *Bootstrap* CSS zbirka (eng. *library*) koja sadrži skup naredbi za lakšu manipulaciju veličinom elemenata. Tako se postiže estetično korisničko sučelje aplikacije

neovisno o uređaju pomoću kojeg korisnik pristupa aplikaciji. JavaScript osigurava funkcionalnost aplikacije na način da komunicira sa Tinkerforge rubnim čvorom preko kojeg se prikupljaju podaci, a također služi i za provjeru unosa parametara, manipulaciju HTML teksta i objekata, izvođenje aritmetičkih operacija te vizualizaciju podataka. Za vizualizaciju podataka se koristi *Plotly* JavaScript grafička zbirka koja omogućuje iscrtavanje podataka u obliku grafa. Koristeći AJAX (eng. *Asynchronous JavaScript and XML - AJAX*) web aplikacija može asinkrono (u pozadini) slati i dohvaćati podatke s poslužitelja bez ometanja prikaza i ponašanja web stranice. Pomoću AJAX-a postiže se funkcija vizualizacije podataka koja ne ometa normalan rad aplikacije (stranica se ne osvježava).

Stražnja strana (poslužitelj) web aplikacije služi za dohvaćanje podataka iz baze podataka, prosljeđivanje podataka na web servis, kao i praćenje stanja prikupljanja (broj prikupljanja). Web servis je poseban dio aplikacije koji se isključivo koristi za pohranjivanje vrijednosti akceleracije u bazu podataka. Stražnja strana web aplikacije koristi C# programski jezik koji je primijenjen u tu svrhu, a komunikacija s prednjom stranom odvija se po principu zahtjev-odgovor (eng. *request-response)*. Za razvoj i testiranje web aplikacije korištena je *Microsoft Visual Studio* integrirana razvojna okolina.

5.2.2 Programiranje aplikacije

Integrirana razvojna okolina (eng. *Integrated Development Enviroment – IDE*) je vrlo popularna među razvojnim inženjerima programske podrške jer pruža podršku za svakodnevne zadatke. Moderni programi za integrirani razvoj pružaju integrirane programe za otklanjanje grešaka, alate za pomoć dovršavanja koda, kontrolu verzija i mnoge druge. Jedan od najpopularnijih programa za integrirani razvoj je *Microsoft Visual Studio*. [34] Za razvoj automatiziranog sustava prikupljanja podataka instaliran je *Microsoft Visual Studio 2019* s ekstenzijom za pohranu i procesuiranje podataka uz ekstenziju za ASP.NET i web razvoj. Kreiranjem novog projekta unutar Visual Studio-a nudi se odabir vrste aplikacije kao i programski jezik. Odabire se ASP.NET Web Forms koja omogućava razvoj dinamičke web aplikacije koristeći model pokretan događajima (eng. *event driven model*) kao i jednostavnu integraciju za pohranjivanje, dozivanje i prikazivanje podataka. Izgled web aplikacije za automatizirano prikupljanje podataka nalazi se na slici 37.



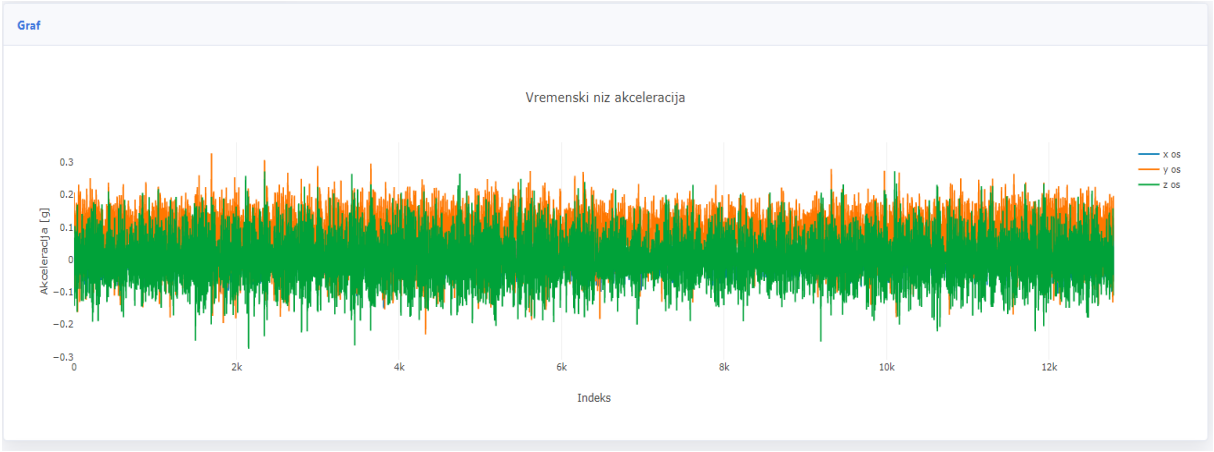
Slika 37. Izgled korisničkog sučelja web aplikacije

Aplikacija je podijeljena u nekoliko manjih dijelova koji služe za lakšu navigaciju. Prvi dio sadrži kontrole za povezivanje i odabir vrste prikupljanja. Korisnik ima na izbor odabir lokacije iz padajućeg izbornika na temelju kojega se popunjava drugi padajući izbornik koji sadrži popis senzora vezanih za tu lokaciju. Odabirom lokacije i senzora popunjavaju se polja IP adresa, Port, UID te ID senzora. Padajući izbornici podatke vezane za lokaciju i senzore dohvaćaju asinkrono (AJAX) iz tablica T_Senzori i T_Moduli koje se nalaze u bazi podataka. Uz odabir lokacije i senzora, korisnik ima i opciju odabira vrste prikupljanja koja može biti pojedinačna ili skupna. U slučaju odabira pojedinačnog prikupljanja podaci će biti prikupljeni

samo s odabranog senzora dok će u slučaju odabira skupnog prikupljanja podaci biti prikupljeni sa svih senzora koji su spojeni na rubni čvor.

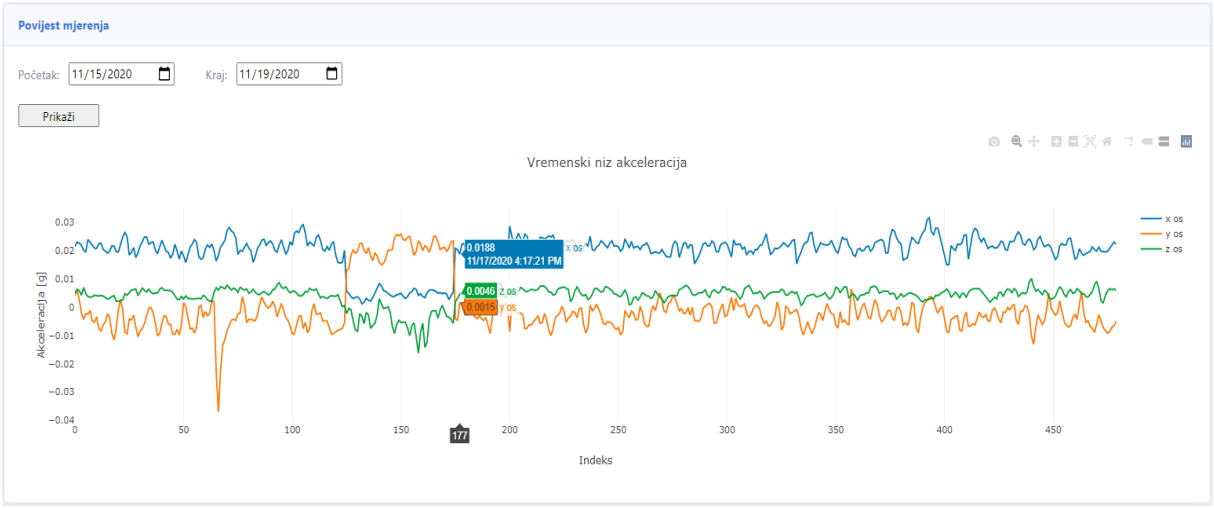
Drugi dio aplikacije sadrži podatke za pohranjivanje koji se pohranjuju u tablicu T_Vrijednosti kao što su broj okretaja, frekvencija, tip kvara...Podaci poput ID mjerenja, frekvencija, broj ponavljanja i ID senzora se automatski sami popunjavaju na temelju korisnikova izbora (frekvencija i ID senzora), tablica iz baze podataka (ID mjerenja) i trenutnog statusa prikupljanja (broj prikupljanja).

Središnji dio aplikacije sadrži padajući izbornik za odabir frekvencije gdje se može odabrati frekvencija od 25, 50, 100, 200, 400, 800, 1600, 3200, 6400 i 10 000 Hz. Za raspon je moguće odabrati 2, 4 ili 8 g, dok polja vrijeme prikupljanja, broj ponavljanja i vrijeme između ponavljanja dozvoljavaju unos proizvoljnih vrijednosti. Klikom na gumb „Započni prikupljanje“ uspostavlja se WebSocket veza sa IIoT senzorom te započinje proces prikupljanja te se obavještava korisnika o statusu prikupljanja koji može poprimiti tri stanja (prikupljam, čekam, izmjena senzora i prikupljanje je završeno) te se sukladno stanju prate i ciklusi te broj senzora s kojih se prikuplja. U program je ugrađena funkcija koja prije početka svakog prikupljanja provjerava jesu li sva potrebna polja popunjena te jesu li odabrani lokacija, senzor, vrsta prikupljanja, raspon te frekvencija. Ciklus prikupljanja prati se s obzirom na broj osvježavanja stranice. Nakon svakog prikupljanja i spremanja podataka stranica se osvježava te se taj broj zapisuje u skrivenom polju na koje ne utječe osvježivanje stranice, tako se postiže da svaki puta kada se podaci prikupe i pohrane u bazu broj osvježavanja se poveća za jedan što predstavlja i broj ciklusa prikupljanja. Na sličan način se prati i broj senzora s kojih se prikuplja tako da na kraju svih ciklusa mjerenja (npr. na kraju ciklusa 3 od 3) provjerava je li odabrana opcija pojedinačnog ili skupnog prikupljanja. Ako je odabrana opcija pojedinačnog prikupljanja prikupljanje se zaustavlja te se korisnika obavještava o završetku. Ako je odabrana opcija skupnog prikupljanja, automatski se vrši izmjena senzora te se ciklusi prikupljanja opet započinju. Pri završetku svakog ciklusa prikupljanja, prikupljeni podaci se vizualiziraju pomoću grafa. Na grafu su prikazane vremenske serije vrijednosti akceleracije osi x, y i z koje je moguće interaktivno manipulirati. Graf se može povećavati i pomicati, vrijednosti se mogu očitati lebdenjem pokazivača iznad njih te se također graf može i spremiti kao slika na uređaj s kojim se pristupa aplikaciji, a iscrtavanje grafa je asinkrono (AJAX). Prikaz vizualizacije prikupljenih podataka prikazan je na slici 38.



Slika 38. Vizualizacija vrijednosti akceleracije x, y i z osi

Zadnji dio aplikacije sadrži mogućnost vizualizacije povijesti prikupljanja. Korisnik ima mogućnost odabira datuma početka prikupljanja i datuma kraja prikupljanja na temelju kojih se dohvaćaju podaci iz baze podataka. Dohvaćanje podataka je asinkrono te se vizualizacija može koristiti prije početka, za vrijeme ili nakon prikupljanja. Prikaz vizualizacije povijesti prikupljenih podataka prikazan je na slici 39.



Slika 39. Vizualizacija povijesnih vrijednosti akceleracije x, y i z osi

Nakon programiranja aplikacije slijedi testiranje sustava, a detaljni uvid u kod klijentske, poslužiteljske strane i web servisa s komentarima nalazi se u prilogu.

5.3 Testiranje sustava

Testiranje sustava je zadnja faza izrade automatiziranog sustava za prikupljanje podataka. Testiranje sustava je nužno kako bi se dobio uvid u funkcionalnost sustava kao i moguće nedostatke. Testiranje se provodi tijekom i nakon izrade sustava. Tijekom izrade sustava testiranje je logičan korak koji se koristi kako bi se utvrdilo izvršavaju li pojedine komponente sustava svoju zadaću dok testiranje nakon izrade sustava služi da se sustav isproba u cijelosti prije puštanja u rad. Tijekom izrade sustava izvršeno je preko 200 testiranja sustava tijekom kojih se dobio uvid u neke od problema sustava poput:

1) Glavni modul je spojen na računalo, ali ne svijetli ni jedna LED lampica.

   Znači da modul ne dobiva napajanje, moguće da je pokvaren. Potrebno je iskopčati pa ukopčati napajanje ili promijeniti USB utor, u suprotnom se modul zamjenjuje.

2) Moduli se ne prikazuje na Brick Vieweru.

   Potrebno je provjeriti je li kup dobro spojen, pričekati par sekundi, probati prekinuti pa opet uspostaviti vezu.

3) Na RED modulu svijetli plava i crvena LED lampica.

   To znači da RED modul ne može pronaći operativni sustav koji se nalazi na Micro-SD kartici. Potrebno je probati:

   - odspojiti i opet spojiti napajanje,
   - odspojiti napajanje, izvaditi Micro-SD karticu i opet ju umetnuti,
   - napraviti reset RED modula tako da se drži gumb za uključivanje 5 sekundi dok se sve LED diode ne ugase, zatim ga pustiti i opet držati 3 sekunde.

4) RED modul se automatski isključuje

   Automatsko isključivanje događa se u slučajevima da RED modul ne izvršava zadatke dugo vremena.

Moguće je uočiti da su svi navedeni problemi vezani za sklopovlje te su lako otklonivi. Izradom i testiranjem sustava dobiveni su preduvjeti za simulaciju rada sustava koja je opisana u sljedećem poglavlju.

6 SIMULACIJA RADA SUSTAVA

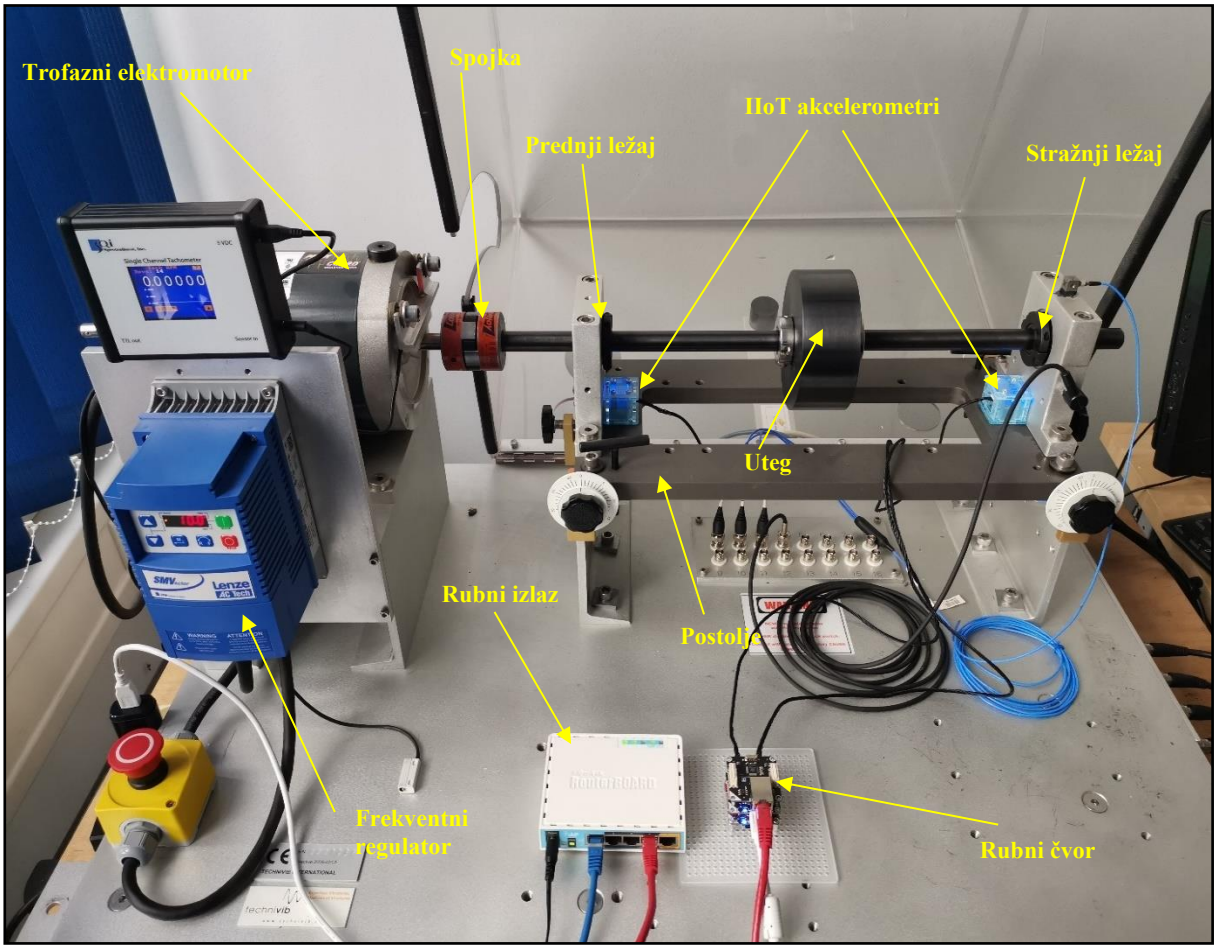
Simulacija rada automatiziranog sustava za prikupljanje podataka je u *Laboratoriju za održavanje Fakulteta strojarstva i brodogradnje* na simulatoru kvarova rotacijske opreme. Simulator kvarova rotacijske opreme dio je laboratorijskog postava te se koristi za istraživanje i razumijevanje različitih vibracijskih obrazaca, gdje se simulira rad pri normalnom (ispravom) stanju opreme i u različitim stanjima koji odgovaraju kvarovima pojedinih dijelova stroja. Podaci koji se prikupljaju pomoću simulatora kvarova služe za daljnju analizu i istraživanje. Simulator kvarova pokreće trofazni asinkroni motor čija se brzina regulira pomoću frekvencijskog pretvarača koji sadrži regulator brzine. Promjenom izlazne frekvencije napajanja motora omogućena je regulacija broja okretaja čime se mogu dosegnuti varijabilne učestalosti vrtnje u rasponu od 0 do 6000 o/min. Snaga motora se preko kandžaste spojke prenosi na vratilo na kojemu se nalazi uteg mase 5 kg koji se dobiva povećanje osnovnog opterećenja ležajeva. Vratilo je uležišteno s dva kuglična ležaja montirana koncentričnim sigurnosnim prstenima. [10] Promjenom vrste ležaja moguće je simulirati sljedeće kvarove [10]:

- oštećenje kotrljajućeg elemenata (BBF),
- oštećenje vanjske staze kotrljanja (ORBF),
- oštećenje unutarnje staze kotrljanja (IRBF),
- kombinirano oštećenje (CBF).

Također, uz simuliranje kvarova ležaja moguće je i simuliranje kvarova rotora i to [10]:

- neravnoteža rotora (IMRF),
- nagnutost rotora (CRF),
- ekscentričnost rotora (ERF).

Simulator kvarova nalazi se na portabilnom aluminijskom postolju koje ima mogućnost horizontalnog i vertikalnog poravnanja osi čime se omogućuje dodatno simuliranje različitih vrsta kvarova [10]. Na prednji i stražnji ležaj montiraju se IIoT senzori koji su žično povezani s rubnim čvorem. Rubni čvor je UTP kablom povezan na rubni izlaz koji povezuje rubni čvor s ostalim uređajima u mreži ili s Internetom. Izgled simulatora kvarova, IIoT senzora, rubnog čvora i rubnog izlaza nalazi se na slici 40.



Slika 40. Konfiguracija eksperimentalnog postava

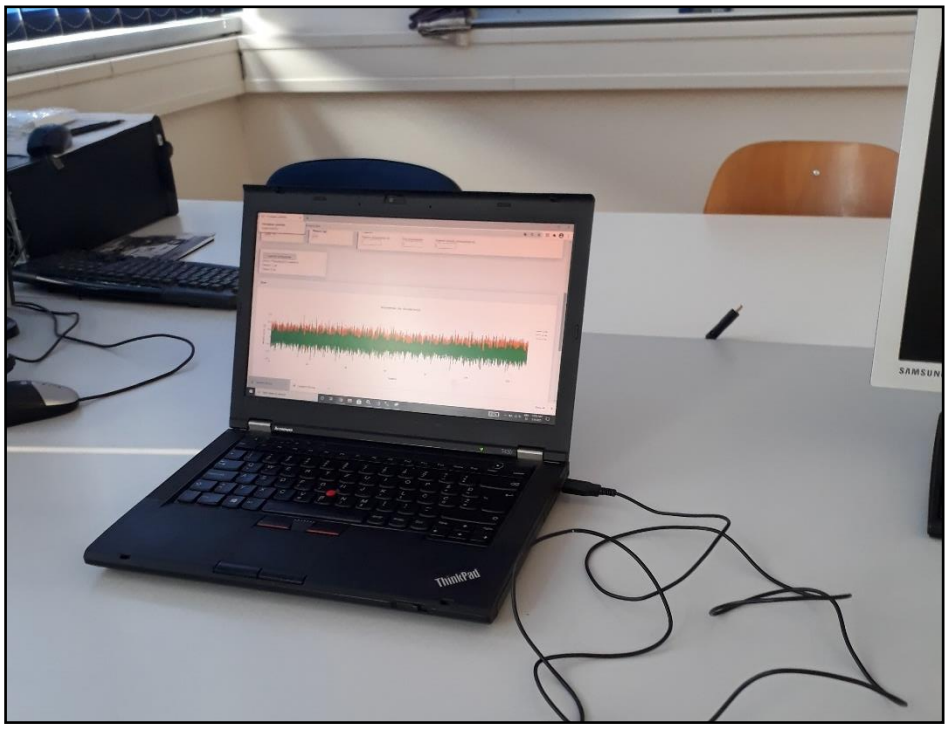
Za potrebe ovog rada, kako bi se prikazao rad sustava, provodi se eksperiment kvara s ležajem koji sadrži kombinirano oštećenje pojedinih elemenata (CBF), čiji se vibracijski obrasci uspoređuju s normalnim stanjem opreme (NS). Parametri prikupljanja nalaze se u tablici 12.

Tablica 12. Parametri prikupljanja

Učestalost vrtnje [min^{-1}]	600
Frekvencija uzrokovanja [Hz]	6400
Stanje opreme	NS, CBF
Broj prikupljanja po ležaju	30
Trajanje prikupljanja [s]	1
Vrijeme između prikupljanja [s]	1

Podaci se prikupljaju sa stražnjeg ležaja budući da se na njemu vrši izmjena. Prijenosno računalo preko kojeg se pristupa web aplikaciji pomoću koje se uspostavlja veza, zadaju

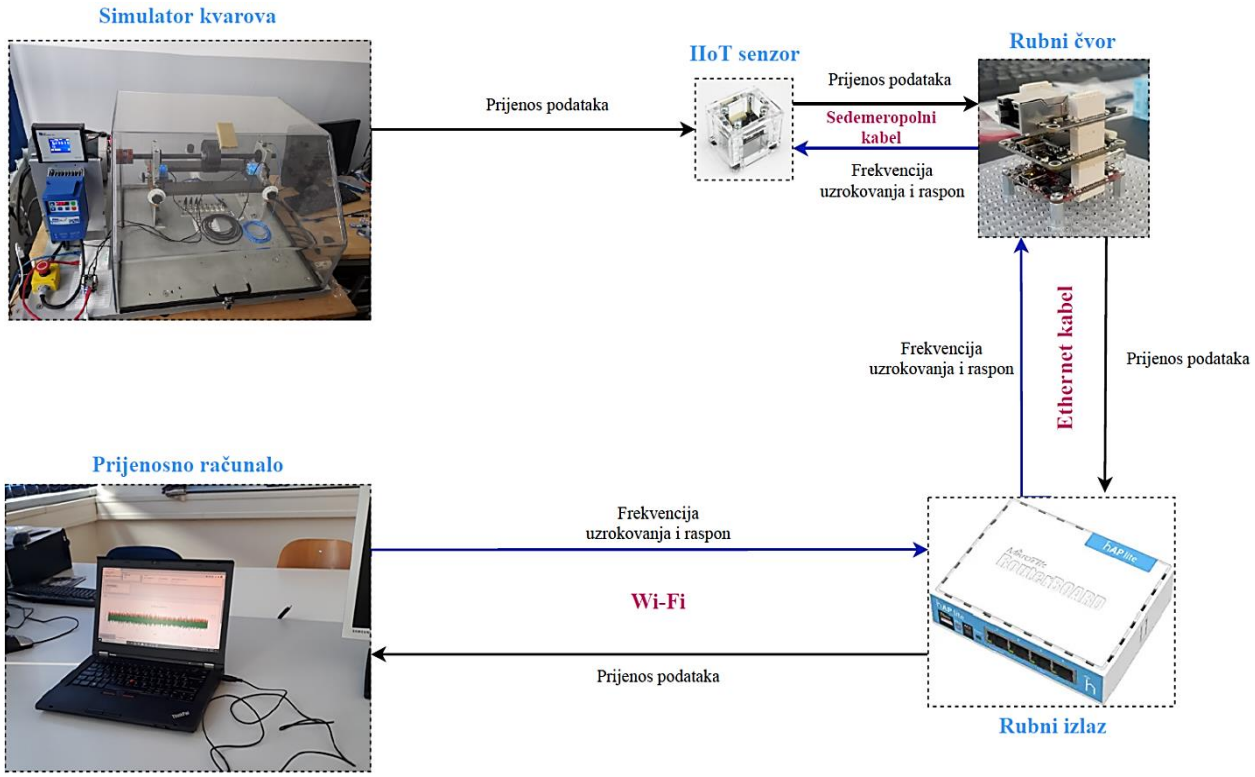
parametri prikupljanja te prikupljaju podaci nalazi se u drugoj prostoriji kako bi se demonstrirala mogućnost udaljenog prikupljanja (Slika 41.).



Slika 41. Prikaz prikupljenih podatak na zaslonu računala

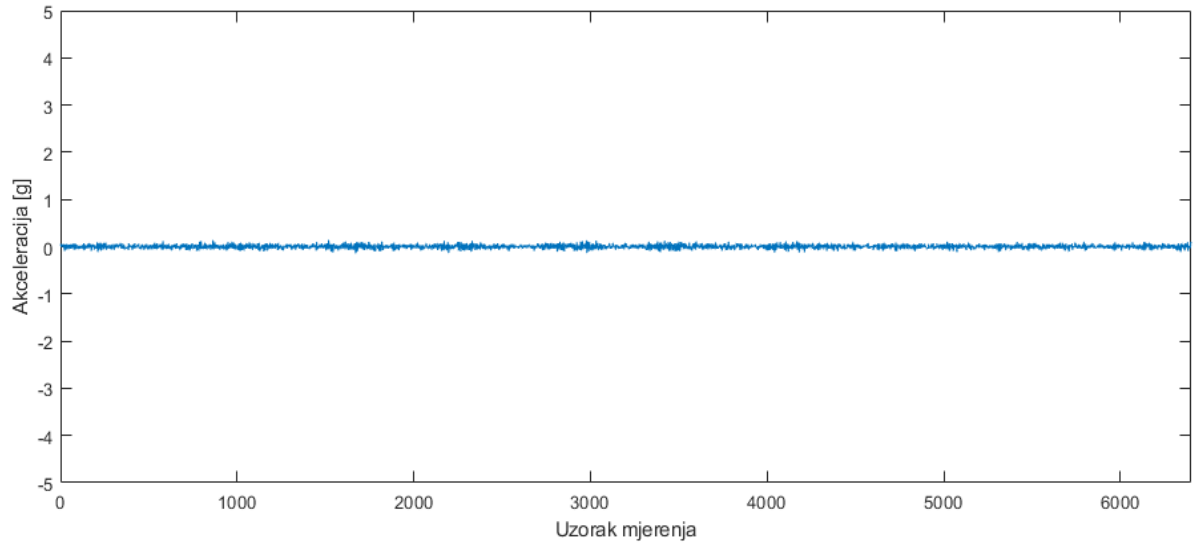
Pod pretpostavkom da je simulator kvarova ispravno podešen i u radu, prije početka prikupljanja web aplikaciji se pristupa pomoću web preglednika po želji. Za potrebe eksperimenta, unutar web aplikacije odabire se lokacija prikupljanja (u ovom slučaju Stroj 1) i senzor s kojeg se prikuplja. Budući da se podaci prikupljaju sa stražnjeg ležaja odabire se senzor broj 2 te opcija pojedinačnog prikupljanja, nakon čega se unose parametri u polja za pohranjivanje podataka u bazu. U slučaju prikupljanja kod normalnog stanja opreme u polje „Tip kvara“ upisuje se NS, a u slučaju kombiniranog kvara upisuje se parametar CBF. U oba slučaja prikupljanje se odvija pri 600 o/min tako da je to polje konstantno u oba slučaja. Pomoću padajućeg izbornika na web aplikaciji odabire se frekvencija uzrokovanja od 6400 Hz te raspon od 8 g. U polje „vrijeme prikupljanja“ i „vrijeme između prikupljanja“ unosi se 1 sekunda, dok se u polje broj prikupljanja unosi broj 30. Nakon klika na gumb započni prikupljanje uspostavlja se veza sa senzorom te mu se prenose zadani parameteri frekvencije uzrokovanja i raspona. Nakon uspostavljanja veze i prijenosa parametara započinje prijenos podataka od senzora, preko rubnog čvora i rubnog izlaza do korisničkog računala. Nakon proteklog vremena prikupljanja podaci se pohranjuju u bazu podataka te sustav čeka sljedeće prikupljanje ovisno o definiranom vremenu između prikupljanja (u ovom slučaju 1 sekunda) nakon čega ciklus opet

započinje i tako još 29 puta dok se svi ciklusi ne završe. Pri završetku prikupljanja podaci se vizualiziraju te se korisnika obavještava da je prikupljanje završeno te korisnik ima izbor ponovno prikupljati podatke, pregledati povijest prikupljanja ili izaći iz aplikacije. Shematski prikaz simulacije rada automatiziranog sustava prikupljanja podataka prikazan je na slici 42.



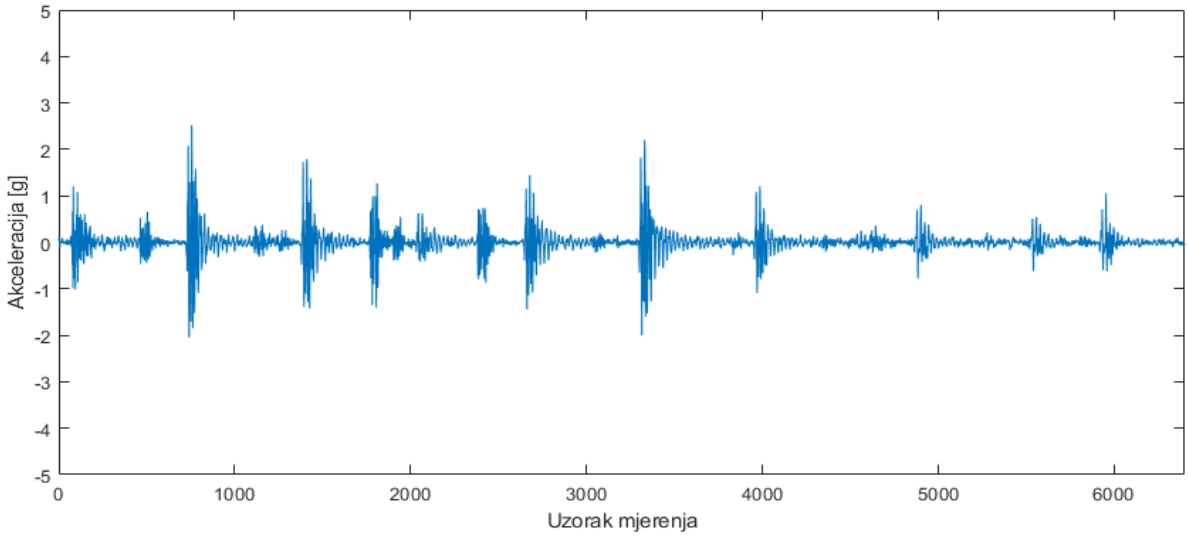
Slika 42. Shema simulacije rada sustava

Tijekom eksperimenta prikupljeno je 384 000 podatka od čega je 192 000 pri simuliranju normalnog stanja (NS) dok je drugih 192 000 prikupljeno pri simuliranju kombiniranog kvara ležaja (CBF). Kako bi se vizualizirala tolika količina podataka koristi se program *MATLAB* gdje se izdvaja jedan ciklus prikupljanja kako bi se dobio uvid u obrazac ponašanja. Na slici 43. nalazi se grafički prikaz jednog ciklusa prikupljanja pri normalnom stanju.



Slika 43. Grafički prikaz prikupljenih podataka pri normalnom stanju

Na slici se može uočiti kako pri normalnom stanju nema značajnih odstupanja vrijednosti akceleracije, kao niti obrazaca koji bi ukazali na prisustvo kvara. Usporedbe radi, na slici 44. se nalazi grafički prikaz prikupljenih podataka pri simuliranju kombiniranog kvara ležaja.



Slika 44. Grafički prikaz prikupljenih podataka pri kombiniranom kvaru ležaja

Sa slike se jasno može iščitati kako postoje značajna odstupanja pri simuliranju kombiniranog kvara ležaja vrijednosti akceleracije. Vrijednosti akceleracije poprimaju obrasce koji su karakteristični za takvo simulirano stanje. Usporedbom dva grafa može se zaključiti kako sustav uspješno prikuplja podatke te da su oni pogodni za daljnju analizu.

Trošak izrade automatiziranog sustava za prikupljanje podataka znatno je niži u odnosu na trenutni sustav prikupljanja podataka. U tablici 13. nalaze se okvirni troškovi pojedinih komponenti novog sustava.

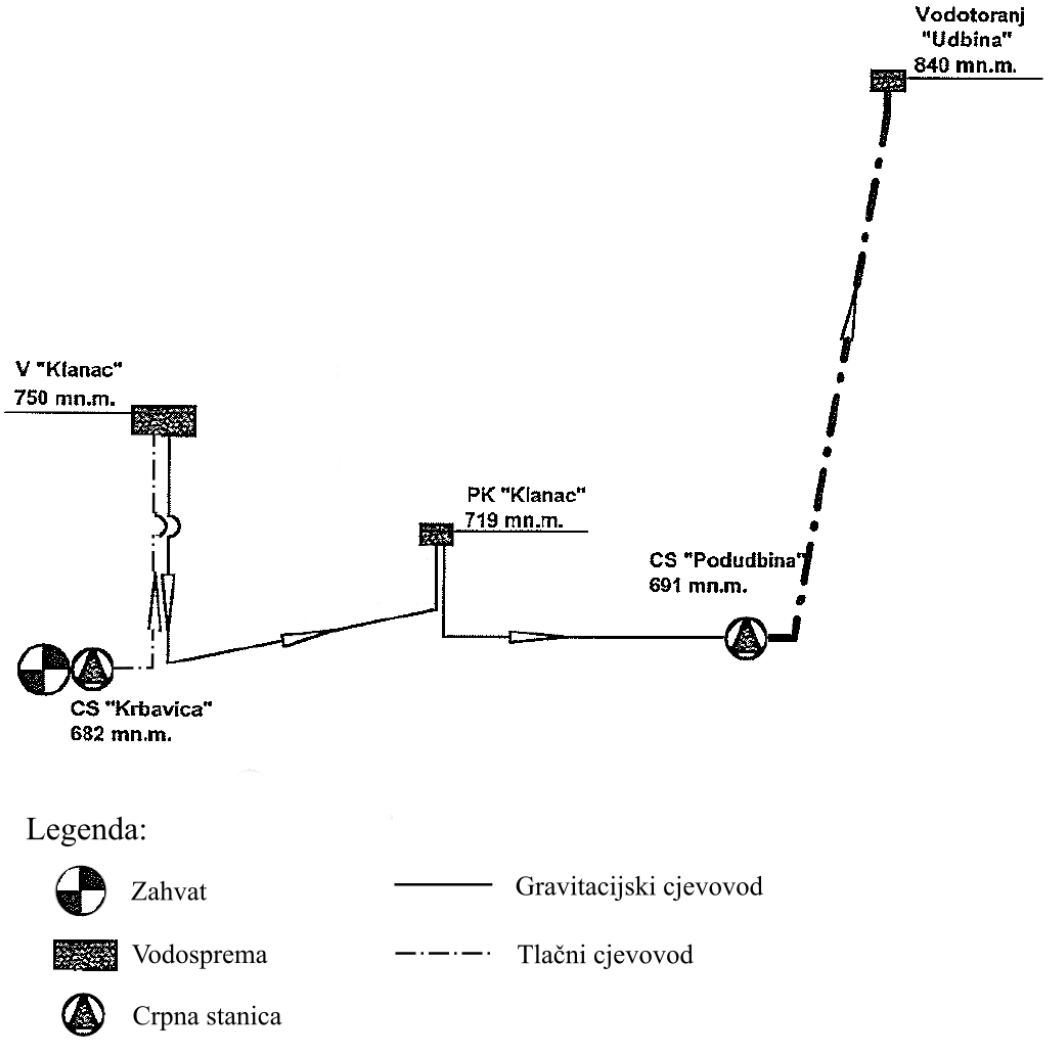
Tablica 13. Troškovi komponenata novog sustava

Komponenta	Okvirna cijena (kn)
TinkerForge akcelerometar	120
Tinkerforge Glavni modul	420
Tinkerforge RED modul	520
Tinkerforge Ethernet ekstenzija	300
MikroTik ruter	275
Prijenosno računalo	4 000
Ukupno:	**5 435 kn**

Ukupni inicijalni trošak automatiziranog sustava za prikupljanje podataka pomoću IIoT senzora iznosi otprilike 5.435 kn što je u usporedni s prethodnim sustavom koji je koštao 72.000 kn značajna ušteda od 92,45 %, a ako se izuzme *LabView* licenca ušteda je i dalje 89,5 %. Takva redukcija inicijalnog troška je značajna kada je u pitanju primjena sustava na više lokacija. S druge strane, jedan od glavnih ograničenja novog sustava je nemogućnost istodobnog prikupljanja s više senzora zbog same prirode prijenosa podataka koja ovisi o uspostavljanju IP veze između dva uređaja što znači da se ne može veza istodobno ostvariti s više uređaja. Zbog tog razloga sustav prikuplja podatke sekvencijalno sa svakog senzora.

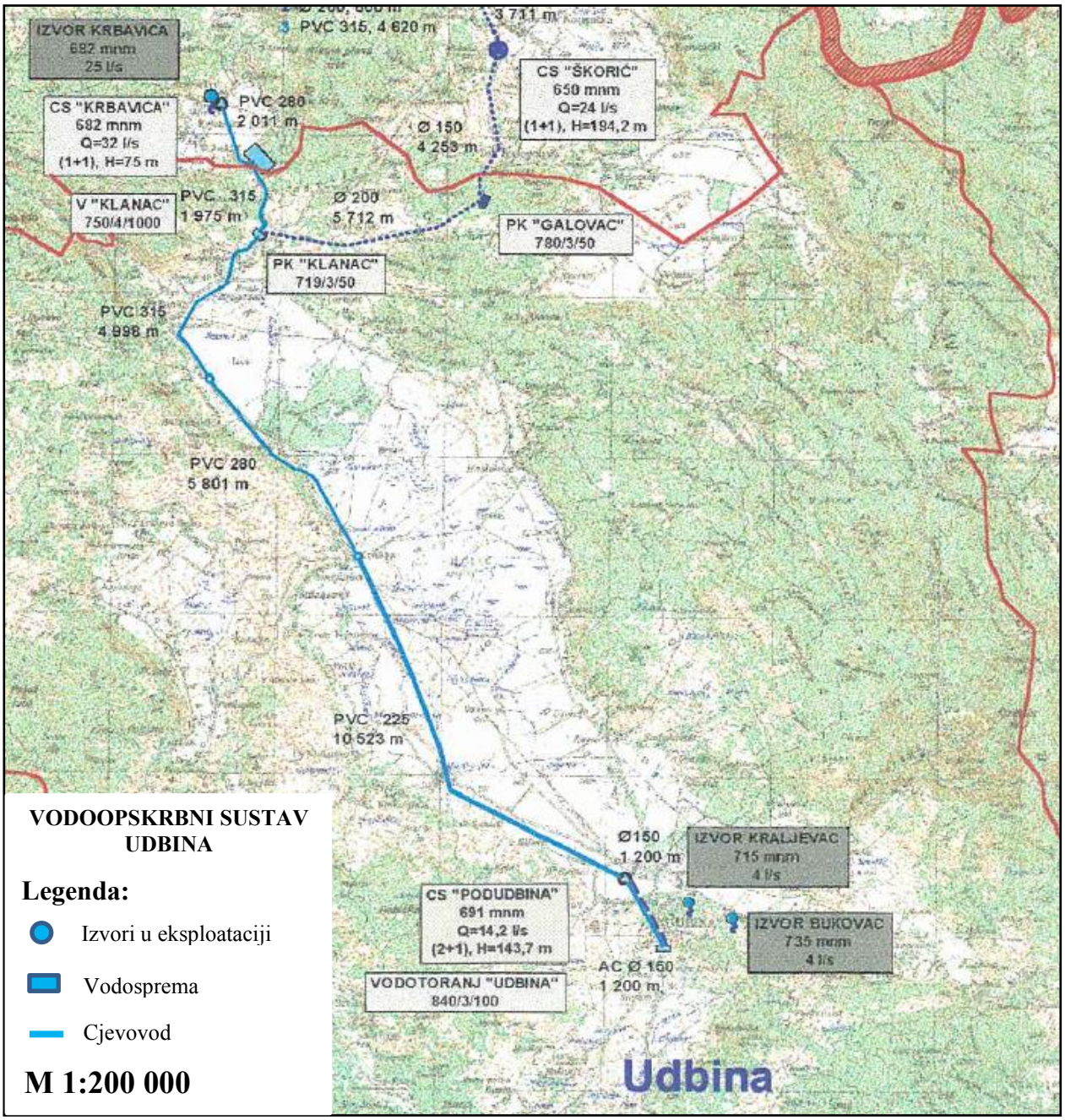
6.1 Mogućnost implementacije sustava prikupljanja u sustavu vodoopskrbe Udbine

Vodoopskrbni sustav Krbavica-Udbina je grupni vodovod u Ličko-senjskoj županiji koji služi za opskrbljivanje vodom naselja Krbavskog polja i Udbine. Zahvat vode je izveden na izboru Krbavica, gdje je izgrađena crpna stanica „Krbavica“ kapaciteta 35 l/s od kojih su 3 l/s za potrebe naselja Krbavica, a ostale 32 l/s za potrebe naselja na području Krbavskog polja i Udbine. Iz crpne stanice „Krbavica“ voda se tlači u vodospremu „Klanac“ iz koje se gravitacijski distribuira u prekidnu komoru. Iz prekidne komore voda gravitacijski ide u crpnu stanicu „Podudbina“ iz koje se tlači vodotoranj „Udbina“. [35] Dispozicijska shema grupnog vodovoda nalazi se na slici 45.



Slika 45. Dispozicijska shema grupnog vodovoda Krbavica-Udbina [35]

Udbina također sadrži lokalni vodovod koji ima zahvat vode na izvorištu „Kraljevac“ i „Bukovac“ ukupnog kapaciteta 4 l/s. Voda iz kaptaže dotječe u sabirni bazen odakle se putem crpne stanice „Podudbina“ tlači vodotoranj „Udbina“ volumena 100 m^3 koji se nalazi na 840 m nadmorske visine. [35] Lokacija vodoopskrbnog sustava Udbina može se vidjeti na na geografskoj karti koja je prikazana na slici 46.

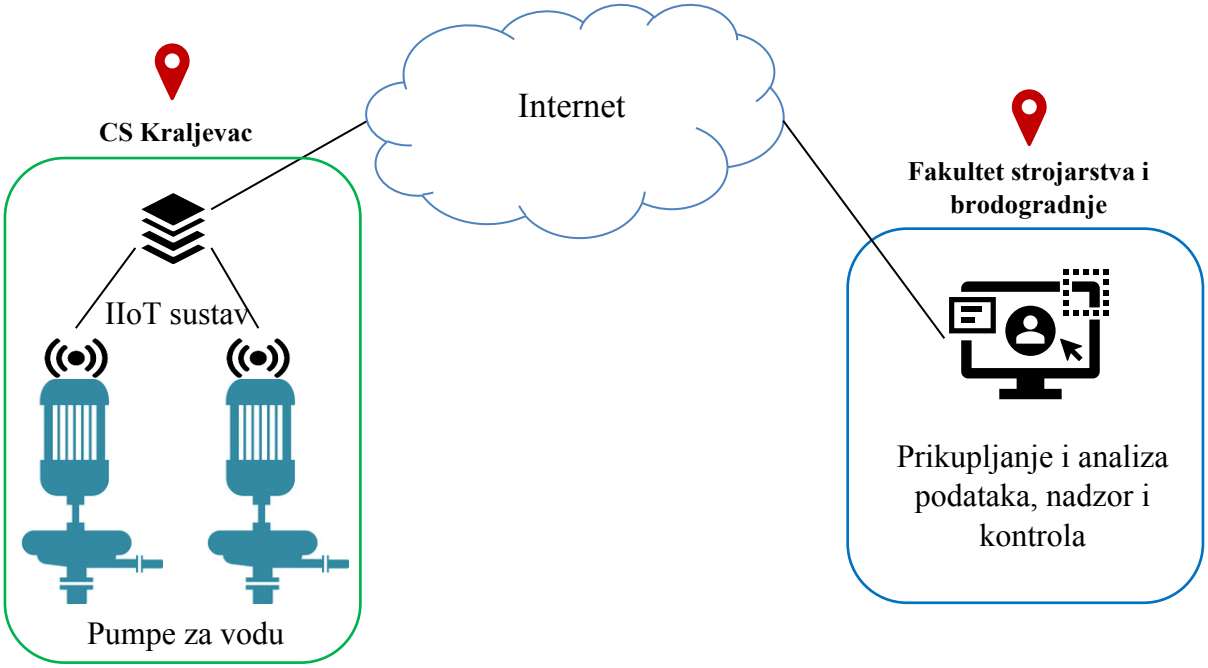


Slika 46. Lokacija vodovoda na geografskoj karti [35]

U sklopu unaprjeđenja vodoopskrbe, kako bi se smanjili gubici u mreži te povećala sigurnost i automatiziranost sustava, nedavno je završeno trogodišnje ulaganje u udbinski vodotoranj zajedničkim sredstvima Ministarstva regionalnog razvoja i fondova EU, Hrvatskih voda i općine, koje je uključivalo saniranje unutrašnjeg bazena vodotornja „Udbina“, zamjenu krova te rekonstrukciju pročelja kao i ugradnje nove stolarije i ograde objekta. Također, provedeno je i unaprjeđenje crpne stanice „Kraljevac“ kojemu je cilj prilagodba elektro dijela postrojenja trenutnom unaprijeđenom stanju, pripremu za buduću nadogradnju te osiguranje upravljanja agregata frekventnim regulatorima što uključuje daljinski nadzor i upravljanje. [36] Na slici 47. može se vidjeti izgled crpne stanice „Kraljevac“.

Slika 47. Crpna stanica Kraljevac

U crpnoj stanici „Kraljevac“ nalaze se dvije redundantne pumpe koje pogone dva trofazna asinkrona motora nazivne snage 18,5 kW s učestalosti vrtnje od 2850 o/min. Budući da je rad pumpi od izuzetne važnosti za opskrbljivanje vodotornja nameću se potrebe smanjenja vremena u kojem su pumpe izvan pogona. Upravo naglasak na daljinski nazor i upravljanje daje prostor primjene automatiziranog sustava za prikupljanja podataka pomoću IIoT sustava. Senzori, rubni čvorovi te rubni izlaz zbog svoje niske cijene, malih dimenzija i mogućnosti povezivanja na Internet mogu osigurati udaljeno prikupljanje podataka s druge lokacije. Idejno rješenje udaljenog prikupljanja podataka nalazi se na slici 48.



Slika 48. Idejno rješenje udaljenog prikupljanja podataka na susutavu vodoopskrbe

Senzori, koji bi se nalazili na crpnoj stanici „Kraljevac“, bi prikupljali podatke te ih slali na Fakultet strojarstva i brodogradnje preko Interneta. Inicijalni sustav za prikupljanje podataka koji je rezultat ovog rada samo je jedan (početni) dio većeg sustava čija je zamisao da ima mogućnosti analiziranja prikupljenih podataka pomoću algoritama strojnog učenja na temelju kojih bi se predviđali kvarovi te predlagale i donosile odluke o aktivnostima održavanja. Zamišljeni veći i bolji sustav imao bi karakteristike preskriptivnog održavanja što je trenutno najmodernija paradigma Održavanja 4.0, no to je tema za neke druge prilike.

Trenutno izrađeni sustav automatiziranog prikupljanja koji će se primjenjivati u *Laboratoriju za održavanje Fakulteta strojarstva i brodogradnje* i koji se može primjeniti u sustavu vodoopskrbe samo su neki od mogućih primjena. Automatizirani sustav za prikupljanje podataka pomoću IIoT senzora je zbog svoje fleksibilnosti, mogućnosti, skalabilnosti, relativno male veličine i niskih troškova moguće primijeniti u širokom spektru industrijskih djelatnosti gdje bi uz primjenu analitike uvelike pridonio smanjenju vremena zastoja i povećanju pouzdanosti objekata održavanja.

7 ZAKLJUČAK

Održavanje 4.0 je najmodernija paradigma održavanja u sklopu Industrije 4.0 koja zahtjeva primjenu suvremenih tehnologija za prikupljanje velike količine podataka koji bi se dalje koristili za analizu i predviđanje potencijalnih kvarova. Jedna od takvih suvremenih tehnologija su i IIoT uređaji koji nude širok spektar mogućnosti zbog svoje fleksibilnosti, skalabilnosti, niskih inicijalnih troškova te mogućnosti povezivanja na Internet što im omogućava raznovrsnu primjenu u industriji. Kako bi se udovoljila potreba za prikupljanjem podataka s udaljenih lokacija te unaprijedila ograničenja trenutnog sustava, javila se potreba za razvojem novog automatiziranog sustava prikupljanja podataka koji će koristiti IIoT tehnologiju za prikupljanje podataka o vrijednostima akceleracije s objekata održavanja.

Odabirom metodologije projektiranja sustava omogućen je organiziran i logičan slijed razvoja sustava koji započinje definiranjem zahtjeva koji proizlaze iz ograničenja trenutnog sustava i zahtjeva korisnika. Na temelju analize trenutnog sustava i zahtjeva korisnika oblikuje se novi sustav primjenom tehnika za modeliranje procesa i podataka koje za cilj imaju definiranje aktivnosti i funkcionalnosti sustava kao i opis podataka koje sustav koristi kako bi izvršavao svoje zadatke. Kako bi sustav uopće imao mogućnost prikupljanja podataka i izvršavanja ostalih procesa, odabiru se tehničke varijante komponenata sustava gdje se stavlja naglasak na ulogu Tinkerforge IIoT uređaja koji čine temelj automatiziranog sustava za prikupljanje podataka. Analizom i oblikovanjem sustava dobila se podloga za fazu izrade sustava gdje se sustav konfigurira, programira i testira kako bi konačno bio spreman za primjenu. Simulacija rada sustava ostvaruje se u *Laboratoriju za održavanje Fakulteta strojarstva i brodogradnje* gdje je postignuta ušteda inicijalnih troškova u iznosu od 92,45 %, a prikazana je i moguća primjena na vodoopskrbnom sustavu kao samo jedna od mnogih mjesta gdje bi se sustav mogao implementirati. Naravno, razvoj sustava ima dug i kontinuiran ciklus koji zahtijeva konstantna poboljšanja i otklanjanje potencijalnih problema. Automatizirani sustav za prikupljanje podataka dakako ima i svoja ograničenja, no on je početni korak u razvoju jednog većeg sustava koji će imati mogućnost analiziranja podataka, predviđanja kvarova, predlaganja aktivnosti održavanja te potpore odlučivanju što je i glavna karakteristika trenutnih trendova u održavanju, a sve s ciljem smanjenja troškova i povećavanja produktivnosti.

LITERATURA

[1] H. Lasi, P. Fettke, T. Feld i M. Hoffmann, »Industry 4.0.,« *Business & Information Systems Engineering,* svez. 6, pp. 239-242, 2014.

[2] V. Saurabh, P. Ambad i S. Bhosle, »Industry 4.0 – A Glimpse,« *Procedia Manufacturing,* svez. 20, pp. 233-238, 2018.

[3] M. Jasiulewicz-Kaczmarek, S. Legutko i P. Kluk, »Maintenance 4.0 Technologies – New Opportunities,« *Management and Production Engineering Review,* svez. 11, br. 2, pp. 74-87, 2020.

[4] M. Jasiulewicz - Kaczmarek i A. Gola, »Maintenance 4.0 Technologies for Sustainable Manufacturing – an Overview,« *IFAC PapersOnLine,* svez. 52, br. 10, pp. 91-96, 2019.

[5] E. Sisinni, A. Saifullah, S. Han, U. Jennehag i M. Gidlund, »Industrial Internet of Things: Challenges,« *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS,* svez. 14, br. 11, pp. 4724-4734, 2018.

[6] H. Boyes, B. Hallaq, J. Cunningham i T. Watson, »The industrial internet of things (IIoT): An analysis framework,« *Computers in Industry,* svez. 101, pp. 1-12, 2018.

[7] S.-W. Lin, B. Miller , J. Durand, G. Bleakley, A. Chigani, R. Martin, B. Murphy i M. Crawford , »Industrial Internet of Things Volume G1: Reference Architecture,« *Industrial Internet Consortium,* pp. 39-42, 1. Ožujak 2017.

[8] D. Lisjak i D. Kolar, »Osnove razvoja informacijskih sustava,« *Fakultet strojarstva i brodogradnje, Zagreb,* 2018.

[9] A. Caroline, »Structured systems analysis and design method (SSADM),« *Information and Software Technology,* svez. 30, br. 1, pp. 153-163, 1988.

[10] D. Kolar, »Model rane procjene kvarova rotacijske opreme primjenom dubokog strojnog učenja [doktorski rad],« *Fakultet strojarstva i brodogradnje, Zagreb,* 2019.

[11] H. Jaakkola i B. Thalheim, »Architecture-Driven Modelling Methodologies,« *Frontiers in Artificial Intelligence and Applications ,* svez. 225, pp. 97-116, 2010.

[12] TinkerForge, »Osnove rada Tinkerforge-a,« [Mrežno]. Dostupno na: https://www.tinkerforge.com/en/home/how-it-works/. [Pokušaj pristupa 16. Prosinac 2020].

[13] TinkerForge, »Tinkerforge Master Brick,« [Mrežno]. Dostupno na: https://www.tinkerforge.com/en/doc/Hardware/Bricks/Master_Brick.html#master-brick. [Pokušaj pristupa 14. Prosinac 2020].

[14] TinkerForge, »Tinkerforge Ethernet Extension,« [Mrežno]. Dostupno na: https://www.tinkerforge.com/en/doc/Hardware/Master_Extensions/Ethernet_Extension.html. [Pokušaj pristupa 15. Prosinac 2020].

[15] I. Fette i A. Melnikov, »The WebSocket Protocol,« u *RFC 6455 The WebSocket Protocol*, IETF, 2011, p. 1.7.

[16] T. A. Gamage, »HTTP and Websockets: Understanding the capabilities of today's web communication technologies,« Medium, 19. Studeni 2017. [Mrežno]. Dostupno na: https://medium.com/platform-engineer/web-api-design-35df8167460. [Pokušaj pristupa 16. Prosinac 2020].

[17] TinkerForge, »Tinkerforge RED Brick,« [Mrežno]. Dostupno na: https://www.tinkerforge.com/en/doc/Hardware/Bricks/RED_Brick.html. [Pokušaj pristupa 16. Prosinac 2020].

[18] TinkerForge, »Tinkerforge Accelerometer 2.0,« [Mrežno]. Dostupno na: https://www.tinkerforge.com/en/doc/Hardware/Bricklets/Accelerometer_V2.html#accelerometer-v2-bricklet. [Pokušaj pristupa 16. Prosinac 2020].

[19] Kionix, »KX122-1037: Akcelerometar«. SAD 2018.

[20] TinkerForge, »Tinkerforge Accelerometer 2.0 API,« [Mrežno]. Dostupno na: https://www.tinkerforge.com/en/doc/Software/Bricklets/AccelerometerV2_Bricklet_JavaScript.html#accelerometer-v2-bricklet-javascript-api. [Pokušaj pristupa 17. Prosinac 2020].

[21] MikroTik, »MikroTik Router,« [Mrežno]. Dostupno na: https://mikrotik.com/product/RB951Ui-2nD#fndtn-specifications. [Pokušaj pristupa 17. Prosinac 2020].

[22] P. Killelea, Web performance tuning (2nd ed.), Peking: O'Reilly, 2002, p. 264.

[23] H. P. Halvorsen, »ASP.NET and Web Programming,« Telemark University College, Porsgrunn, 2014.

[24] C. Begg i T. M. Connolly, Database Systems: A Practical Approach to Design, Implementation, and Management, Pearson, 2014.

[25] L. Quing i Y.-L. Chen, Modeling and Analysis of Enterprise and Information Systems, Berlin : Springer, 2009.

[26] D. Lisjak i D. Kolar, T-SQL, Zagreb: Fakultet strojarstva i brodogradnje, 2018..

[27] Lenovo, »ThinkPad T430 laptop,« [Mrežno]. Dostupno na: https://www.lenovo.com/us/en/laptops/thinkpad/t-series/t430/. [Pokušaj pristupa 22. Prosinac 2020].

[28] Microsoft Technet, »Virtual Private Networking: An Overview,« 9 12 2009. [Mrežno]. Dostupno na: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/bb742566(v=technet.10)?redirectedfrom=MSDN. [Pokušaj pristupa 23. prosinac 2020].

[29] Dulaney i Emmett, u *CompTIA Security+ Deluxe Study Guide*, Wiley Publishing, 2009, p. 124.

[30] ZeroTier, »ZeroTier Manual,« 2021. [Mrežno]. Dostupno na: https://www.zerotier.com/manual/. [Pokušaj pristupa 27. prosinac 2020].

[31] J. M. Rios i N. P. Souto, »Comparison of development methodologies in Web applications,« *Information and Software Technology,* svez. 119, 2020.

[32] D. Jacobson, G. Brail i D. Woods, APIs: A Strategy Guide, Sebastopol: O'Reilly, 2012.

[33] D. Flanagan, JavaScript - The definitive guide (6 ed.), Sebastopol: O'Reilly, 2011.

[34] S. Amann, S. Proksch, S. Nadi i M. Mezini, »A Study of Visual Studio Usage in Practice,« u *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Suita, 2016.

[35] Hidro consult d.o.o., »Vodoopskrbni plan Ličko-Senjske županije,« Hrvatske vode, Zagreb, 2001.

[36] Brkić, »Općina Udbina,« 28. srpnja 2020. [Mrežno]. Dostupno na: https://www.udbina.hr/index.php?limitstart=25. [Pokušaj pristupa 06. siječnja 2021].

PRILOZI

I. CD-R disk

II. Izlist koda klijentske strane

III. Izlist koda poslužiteljske strane

IV. Izlist koda web servisa

Prilog II.

Izlist koda klijentske strane

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebApp.aspx.cs"
Inherits="prikupljanje_podataka.WebApp" %>


<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Prikupljanje podataka</title>
      <!-- Custom styles for this template-->
  <link href="bootstrap.css" rel="stylesheet"/>
  <script src="plotly-latest.min.js"></script>
</head>
<body>
<form id="form1" runat="server">


  <!-- Page Wrapper -->
  <div id="wrapper">

    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

      <!-- Main Content -->
      <div id="content">

        <!-- Topbar -->
<nav class="navbar navbar-expand navbar-light topbar mb-4 static-top shadow"
style="background-color: #1671B9;">


<span class="h3 mb-0" style="color: white;">Prikupljanje podataka</span>


<div class="navbar-nav ml-auto">

    <button class="btn btn-light" type="button">Povratak</button>


</div>
</nav>

        <!-- End of Topbar -->

        <!-- Begin Page Content -->
<div class="container-fluid">

          <!-- Page Heading -->
          <div class="d-sm-flex align-items-center justify-content-between mb-4">
            <h1 class="h3 mb-0 text-gray-800">Akceleracija</h1>

          </div>


          <!-- Content Row -->
                <!-- Povezivanje -->
           <div class="row">
              <div class="col">
              <div class="card shadow mb-4">
                <!-- Card Header - Dropdown -->
                <div class="card-header py-3 d-flex flex-row align-items-center justify-
content-between">
                  <h6 class="m-0 font-weight-bold text-primary">Povezivanje</h6>
                </div>
                <!-- Card Body -->
                <div class="card-body">
                    <div class="row no-gutters align-items-center" style="margin-bottom:
20px;">
```

```
                        <asp:DropDownList ID="lokacija" runat="server"
CausesValidation="false" />     
                        <asp:DropDownList ID="senzor" runat="server" CausesValidation="false"
/>
                        <br />
                    </div>
                  <div class="row no-gutters align-items-center">
                    <div class="col">
                      <div class="text-s mb-1">IP adresa:  <input type="text"
runat="server" id="host"  size="12" name="ip" causesvalidation="False" /></div>
                    </div>
                    <div class="col">
                      <div class="text-s mb-1">Port:  <input type="text"
runat="server" id="port" size="4" width="50px" name="port" causesvalidation="False" /></div>
                    </div>
                    <div class="col">
                      <div class="text-s mb-1">UID:  <input type="text"
runat="server" id="uid" size="4" name="uid" causesvalidation="False" /></div>
                    </div>
                  </div>
                   <br />
                <div class="row no-gutters align-items-center">
                    <div class="col">
                    Vrsta prikupljanja:
                    <br />
                    <asp:RadioButton ID="pojedinacno" runat="server" GroupName="vrsta"
Text="Pojedinačno prikupljanje" />
                    <br />
                    <asp:RadioButton ID="skupno" runat="server" Text="Skupno prikupljanje"
GroupName="vrsta" />
                    </div>
                </div>
                </div>
              </div>
             </div>
            </div>

                <!-- Pohranjivanje -->
            <div class="row">
               <div class="col">
               <div class="card shadow mb-4">

                <div class="card-header py-3 d-flex flex-row align-items-center justify-
content-between">
                  <h6 class="m-0 font-weight-bold text-primary">Podaci za pohranjivanje</h6>
                </div>
                <!-- Card Body -->
                <div class="card-body">
                <div class="row no-gutters align-items-center">
                    <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-1">ID mjerenja: <input type="text" size="13"
runat="server" id="id_mj"/></div>
                    </div>
                            
                    <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-1">Broj prikupljanja: <input type="text"
size="13" runat="server" value="1" id="broj_prikupljanja"/></div>
                    </div>
                            
                    <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-1">Broj okretaja:  <input type="text"
size="13" value="1000" runat="server" id="broj_okretaja"/></div>
                    </div>
                            
                    <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-
1">Frekvencija:     <input type="text" size="13" runat="server"
id="frekvencija_mjerenja"/></div>
                    </div>
                            
                    <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-1">Tip
kvara:        <input type="text" size="13" value="NS"
runat="server" id="tip_kvara"/></div>
                    </div>
```

```
                            
                    <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-
1">Separator:       <input type="text" size="13"
runat="server" id="separator" value=","/></div>
                    </div>
                            
                    <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-1">ID
senzora:       <input type="text" size="13" runat="server"
id="id_sen"/></div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
           </div>


          <div class="row">
            <!-- Frekvencija -->
            <div class="col-xl-2 col-md-6 mb-4">
              <div class="card border-left-primary shadow py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-s font-weight-bold text-primary mb-1">Frekvencija
(Hz):</div>
                      <div class="h6 mb-0 font-weight-bold text-gray-800">
                        <select name="frek" id="frekvencija" runat="server">
                          <option disabled="disabled" hidden="hidden" selected="selected"
value=""></option>
                          <option value="5">25</option>
                          <option value="6">50</option>
                          <option value="7">100</option>
                          <option value="8">200</option>
                          <option value="9">400</option>
                          <option value="10">800</option>
                          <option value="11">1600</option>
                          <option value="12">3200</option>
                          <option value="13">6400</option>
                          <option value="14">10000</option>
                        </select>
                      </div>
                    </div>
                  </div>
                </div>
              </div>

              <!-- Raspon -->
              <div class="col-xl-2 col-md-6 mb-4">
                <div class="card border-left-primary shadow py-2">
                  <div class="card-body">
                    <div class="row no-gutters align-items-center">
                      <div class="col mr-2">
                        <div class="text-s font-weight-bold text-primary mb-1">Raspon (g):</div>
                        <div class="h6 mb-0 font-weight-bold text-gray-800">
                          <select name="frek2" id="raspon" runat="server">
                            <option disabled="disabled" hidden="hidden" selected="selected"
value=""></option>
                            <option value="0" label="2"></option>
                            <option value="1" label="4"></option>
                            <option value="2" label="8"></option>
                          </select>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>

              <!-- Trajanje -->
              <div class="col mb-4">
                <div class="card border-left-primary shadow">
```

```
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-s font-weight-bold text-primary mb-1">Trajanje:</div>
                      <div class="row no-gutters align-items-center">
                      <div class="col-md-auto" style="width:200px">
                      <div class="text-s mb-1">Vrijeme prikupljanja (s):  <input
type="text" runat="server" size="4" id="vrijeme_prikupljanja" name="time"/></div>
                    </div>
                            
                     <div class="col-md-auto" style="width:150px">
                      <div class="text-s mb-1">Broj ponavljanja:<input type="text" size="4"
runat="server" id="broj_ponavljanja" name="time"/></div>
                    </div>
                            
                     <div class="col-md-auto" style="width:280px">
                      <div class="text-s mb-1">Vrijeme između prikupljanja
(s):  <input type="text" id="vrijeme_izmedu" runat="server" size="4"
name="time"/></div>
                    </div>

                    </div>
                  </div>
                </div>
              </div>
            </div>

              <!-- Započni -->

          </div>

            <!-- Započni -->
          <div class="row">
            <div class="col-xl-3 col-md-3 mb-4">
                <div id="brdr" runat="server" class="card border-left-primary shadow">
                  <div class="card-body">
                    <div class="row no-gutters align-items-center">
                      <div class="col-auto mb-0 text-gray-800">
                        <input id="Button2" runat="server" type="button" onclick="start()"
value="Započni prikupljanje" style="width: 200px;"/>
                        <p style="margin-bottom: -10px;">
                          Status: <span id="status" runat="server"/>
                          <br />
                          Senzor: <span id="senzor_od"
runat="server">1</span> od <span runat="server" id="senzor_do"></span>
                          <br />
                          Ciklus: <span id="ciklus_od"
runat="server">0</span> od <span runat="server" id="ciklus_do"></span>
                          <br />

                        </p>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>

            <div class="row">

              <!-- Graf -->
              <div class="col">
                <div class="card shadow mb-4">
                  <!-- Card Header - Dropdown -->
                  <div class="card-header py-3 d-flex flex-row align-items-center justify-
content-between">
                    <h6 class="m-0 font-weight-bold text-primary">Graf</h6>
                  </div>
                  <!-- Card Body -->
                  <div class="card-body">
                      <div style="max-width: 100%;" id="graf"></div>
                  </div>
                </div>
              </div>
```

```
            </div>

             <div class="row">

             <!-- Graf -->
             <div class="col">
               <div class="card shadow mb-4">
                 <!-- Card Header - Dropdown -->
                 <div class="card-header py-3 d-flex flex-row align-items-center justify-
content-between">
                   <h6 class="m-0 font-weight-bold text-primary">Povijest mjerenja</h6>
                 </div>
                 <!-- Card Body -->
                 <div class="card-body">
                     <div>
                         Početak:   <asp:TextBox ID="datum_pocetak"
runat="server" TextMode="Date"
Width="150px"></asp:TextBox>         
                         Kraj:   <asp:TextBox ID="datum_kraj"  runat="server"
TextMode="Date" Width="150px"></asp:TextBox>
                         <br />
                         <br />
                         <input type="button" value="Prikaži" onclick="povijest()" />
                     </div>
                     <div style="max-width: 100%;" id="graf2">
                     </div>
                 </div>
               </div>
             </div>
            </div>


<asp:Button ID="Spremi" ClientIDMode="Static" runat="server" Text="Spremi u bazu"
OnClick="Spremi_Click" Width="150px" />
<asp:HiddenField id="xyz" runat="server"></asp:HiddenField>
<asp:HiddenField id="trenutniBrojPonavljanja" runat="server"></asp:HiddenField>
<asp:HiddenField id="brojSenzora" runat="server"></asp:HiddenField>
<asp:HiddenField id="trenutniSenzor" runat="server"></asp:HiddenField>
<asp:HiddenField id="ukupniBrojSenzora" runat="server"></asp:HiddenField>
<asp:HiddenField id="izborLokacije" runat="server"></asp:HiddenField>

<asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="true">
</asp:ScriptManager>

</div>
</div>
          <!-- Content Row -->

</div>
        <!-- /.container-fluid -->

</div>
      <!-- End of Main Content -->
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<script src="./Tinkerforge.js"></script>
<script>

 //Popunjavanje dropdown Senzora
    $("#lokacija").change(function (e)
    {
        $("#lokacija option[value='0']").hide();
        $.ajax({
            type: "POST",
            url: "WebApp.aspx/BindSenzor",
            contentType: "application/json; charset=utf-8",
            data: '{"ID_brick":"' + $(this).val() + '"}',
            dataType: "json",
            success: function (result, status, xhr) {
                $("#senzor").html(result.d);
            },
            error: function (xhr, status, error) {
                alert("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
            }
        });
```

```
        var odabirLokacije = document.getElementById('<%=lokacija.ClientID %>').selectedIndex;
        document.getElementById('<%=izborLokacije.ClientID %>').value = odabirLokacije;

    });

    //Popunjavanje polja IP adresa i Port
    $("#lokacija").change(function (e)
    {
        $.ajax({
            type: "POST",
            url: "WebApp.aspx/FillLokacija",
            contentType: "application/json; charset=utf-8",
            data: '{"ID_brick":"' + $(this).val() + '"}',
            dataType: "json",
            success: function (data) {
                x = data.d;
                var res = x.split(" ");
                document.getElementById('<%=host.ClientID %>').value = res[0];
                document.getElementById('<%=port.ClientID %>').value = res[1];
                },
                error: function (xhr, status, error) {
                    alert("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
                }
        });

    });

    //Popunjavanje polja UID i ID senzor
    $("#senzor").change(function (e)
    {
        $("#senzor option[value='0']").hide();
        $.ajax({
            type: "POST",
            url: "WebApp.aspx/FillSenzor",
            contentType: "application/json; charset=utf-8",
            data: '{"ID_senzora":"' + $(this).val() + '"}',
            dataType: "json",
            success: function (data) {
                uid = data.d;

                document.getElementById('<%=uid.ClientID %>').value = uid;
                console.log("Change:"+ uid);

           },
           error: function (xhr, status, error) {
               alert("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
            }
        });
        var input = document.getElementById('<%=id_sen.ClientID %>');
        var dropdown = document.getElementById('<%=senzor.ClientID %>');
        input.value = dropdown.options[dropdown.selectedIndex].value;

        var dulj = parseInt(document.getElementById('<%=senzor.ClientID %>').options.length) -
1;
        document.getElementById('<%=ukupniBrojSenzora.ClientID %>').value = dulj;
    });

    //Deklariranje globalnih varijabli
    var ipcon;
    var VrijednostiXYZ = [];
    var VrijednostiX = [];
    var VrijednostiY = [];
    var VrijednostiZ = [];
    ipcon = new Tinkerforge.IPConnection();
    document.getElementById('<%=Spremi.ClientID%>').style.display = "none";

    //Klik gumba za početak prikupljanja
    function start()
    {
        VrijednostiXYZ.length = 0;
        VrijednostiX.length = 0;
        VrijednostiY.length = 0;
        VrijednostiZ.length = 0;
```

```
        Frekvencija = document.getElementById('<%=frekvencija.ClientID%>').value;
        Raspon = document.getElementById('<%=raspon.ClientID%>').value;
        Host = document.getElementById('<%=host.ClientID%>').value;
        Uid = document.getElementById('<%=uid.ClientID%>').value;
        VrijemePrikupljanja =
document.getElementById('<%=vrijeme_prikupljanja.ClientID%>').value;
        BrojPonavljanja = document.getElementById('<%=broj_ponavljanja.ClientID%>').value;
        VrijemeIzmedu = document.getElementById('<%=vrijeme_izmedu.ClientID%>').value;
        Port = document.getElementById('<%=port.ClientID%>').value;
        var poruka;

        //Provjera unosa
        if (Frekvencija == 0 || Raspon == "" || isNaN(VrijemePrikupljanja) ||
VrijemePrikupljanja < 0.1 || BrojPonavljanja < 1 || BrojPonavljanja > 1 & VrijemeIzmedu < 1)
        {
            poruka = "Unesite parametre akcelerometra";
            document.getElementById('<%=status.ClientID%>').innerHTML = poruka;
            document.getElementById('<%=status.ClientID%>').style.color = "red";
            document.getElementById('<%=brdr.ClientID%>').className = "card border-left-danger
shadow";
        }
        else if (Host == "" || isNaN(Port) || Port < 0.1 || Uid == "")
        {
            poruka = "Unesite parametre povezivanja";
            document.getElementById('<%=status.ClientID%>').innerHTML = poruka;
            document.getElementById('<%=status.ClientID%>').style.color = "red";
            document.getElementById('<%=brdr.ClientID%>').className = "card border-left-danger
shadow";
        }
        else if (document.getElementById('<%=pojedinacno.ClientID %>').checked == false &&
document.getElementById('<%=skupno.ClientID %>').checked == false) {
            poruka = "Odaberite vrstu prikupljanja";
            document.getElementById('<%=status.ClientID%>').innerHTML = poruka;
            document.getElementById('<%=status.ClientID%>').style.color = "red";
            document.getElementById('<%=brdr.ClientID%>').className = "card border-left-danger
shadow";
        }
        else
        {
            prikupljanje();
            poruka = "Prikupljam";
            document.getElementById('<%=status.ClientID%>').innerHTML = poruka;
            document.getElementById('<%=status.ClientID%>').style.color = "DimGrey";
            document.getElementById('<%=ciklus_do.ClientID%>').innerHTML =
document.getElementById('<%=broj_ponavljanja.ClientID%>').value;
            var BrojCiklusa =
parseInt(document.getElementById('<%=ciklus_od.ClientID%>').innerHTML);
            BrojCiklusa++;
            document.getElementById('<%=ciklus_od.ClientID%>').innerHTML = BrojCiklusa;
        }

    }

    //Prikupljanje podataka
    function prikupljanje()
    {

        if (document.getElementById('<%=skupno.ClientID %>').checked == true)
        {
            var odabirLokacije = document.getElementById('<%=izborLokacije.ClientID
%>').value;
            $("#lokacija").prop('selectedIndex', odabirLokacije).change();
            var duljina = document.getElementById('<%=ukupniBrojSenzora.ClientID %>').value;
            document.getElementById('<%=senzor_do.ClientID %>').innerHTML = duljina;

            document.getElementById('<%=trenutniSenzor.ClientID %>').value =
parseInt(document.getElementById('<%=senzor_od.ClientID %>').innerHTML);
            var changeSenzor = parseInt(document.getElementById('<%=senzor_od.ClientID
%>').innerHTML);
            $("#senzor").prop('selectedIndex', changeSenzor).change();
            setTimeout(prikupi, 100);

        }
        else if (document.getElementById('<%=pojedinacno.ClientID %>').checked == true)
```

```
        {
            var senzorDo = document.getElementById('<%=brojSenzora.ClientID %>').value = 1;
            document.getElementById('<%=senzor_do.ClientID %>').innerHTML = senzorDo;
            prikupi();
        }

        function prikupi()
        {
            document.getElementById('<%=status.ClientID%>').innerHTML = "Prikupljam";
            document.getElementById('<%=brdr.ClientID%>').className = "card border-left-
success shadow";

            var Host = document.getElementById('<%=host.ClientID%>').value;
            var Port = parseInt(document.getElementById('<%=port.ClientID%>').value);
            var Uid = document.getElementById('<%=uid.ClientID%>').value;
            console.log("Za vezu: " + Uid);
            var Frekvencija =
parseInt(document.getElementById('<%=frekvencija.ClientID%>').value);
            var Raspon = parseInt(document.getElementById('<%=raspon.ClientID%>').value);
            var interval =
parseFloat(document.getElementById('<%=vrijeme_prikupljanja.ClientID%>').value) * 1000 + 1000;
            var expected = Date.now() + interval;

            if (ipcon !== undefined)
            {
                ipcon.disconnect();

            }

            var a = new Tinkerforge.BrickletAccelerometerV2(Uid, ipcon); // Kreiranje objekta
akcelerometar

            var state = 1;
            ipcon.connect(Host, Port, function (error) {
                console.log('Error: ' + error + '\n');
                state = error;
            }
            ); // Povezivanje na Brick


            //Log kada je događaj započeo;
            var connStatus;
            if (state == 1) {
                connStatus = "1";
                start_log(connStatus);
            }
            else {
                connStatus = "0";
                start_log(connStatus);
            }

            // Registriranje callback akcelerometra
            a.on(Tinkerforge.BrickletAccelerometerV2.CALLBACK_CONTINUOUS_ACCELERATION_16_BIT,
                // Callback funkcija za callback akcelerometra
                function (acceleration)
                {
                    var akc = acceleration;
                    var Raspon =
parseInt(document.getElementById('<%=raspon.ClientID%>').value);
                    if (Raspon == 0)
                    {
                        var multipl = 0.000061;
                    }
                    else if (Raspon == 1)
                    {
                        var multipl = 0.000122;
                    }
                    else if (Raspon == 2)
                    {
                        var multipl = 0.000244;
                    }

                    for (var i = 0, length = akc.length; i < length; i++)
                    {
                        akc[i] *= multipl
```

```
                    };

                    xos = [akc[0], akc[3], akc[6], akc[9], akc[12], akc[15], akc[18], akc[21],
akc[24], akc[27]];
                    yos = [akc[1], akc[4], akc[7], akc[10], akc[13], akc[16], akc[19],
akc[22], akc[25], akc[28]];
                    zos = [akc[2] - 1.03, akc[5] - 1.03, akc[8] - 1.03, akc[11] - 1.03,
akc[14] - 1.03, akc[17] - 1.03, akc[20] - 1.03, akc[23] - 1.03, akc[26] - 1.03, akc[29] -
1.03];


                    xos = xos.map(function (each_element)
                    {
                        return Number(each_element.toFixed(4));
                    });
                    yos = yos.map(function (each_element)
                    {
                        return Number(each_element.toFixed(4));
                    });
                    zos = zos.map(function (each_element)
                    {
                        return Number(each_element.toFixed(4));
                    });

                    VrijednostiX.push(...xos);
                    VrijednostiY.push(...yos);
                    VrijednostiZ.push(...zos);

                    akc1 = [xos[0], yos[0], zos[0]];
                    akc2 = [xos[1], yos[1], zos[1]];
                    akc3 = [xos[2], yos[2], zos[2]];
                    akc4 = [xos[3], yos[3], zos[3]];
                    akc5 = [xos[4], yos[4], zos[4]];
                    akc6 = [xos[5], yos[5], zos[5]];
                    akc7 = [xos[6], yos[6], zos[6]];
                    akc8 = [xos[7], yos[7], zos[7]];
                    akc9 = [xos[8], yos[8], zos[8]];
                    akc10 = [xos[9], yos[9], zos[9]];

                    VrijednostiXYZ.push(...akc1, ...akc2, ...akc3, ...akc4, ...akc5, ...akc6,
...akc7, ...akc8, ...akc9, ...akc10);

                }
            );

            //Postavljanje konfiguracije akcelerometra
            ipcon.on(Tinkerforge.IPConnection.CALLBACK_CONNECTED,
                function (connectReason)
                {
                    a.setConfiguration(Frekvencija, Raspon);
                    a.setContinuousAccelerationConfiguration(true, true, true, 1);

                }
            );

            //Završavanje prikupljanja
            ZavrsiPrikupljanje = setTimeout(step, interval);
            function step()
            {
                var dt = Date.now() - expected; // drift (pozitivan za prekoračenje)
                if (dt > interval)
                {
                    alert("Osvježite stranicu");
                }
                expected += interval;

                clearTimeout(ZavrsiPrikupljanje);
                ipcon.disconnect();

                //Rezanje vrijednosti
                var t =
parseFloat(document.getElementById('<%=vrijeme_prikupljanja.ClientID%>').value);
                var f = parseInt(document.getElementById('<%=frekvencija.ClientID%>').value);
                if (f == 5)
                {
```

```
    var frekv = 25;
}
else if (f == 6)
{
    var frekv = 50;
}
else if (f == 7)
{
    var frekv = 100;
}
else if (f == 8)
{
    var frekv = 200;
}
else if (f == 9)
{
    var frekv = 400;
}
else if (f == 10)
{
    var frekv = 800;
}
else if (f == 11)
{
    var frekv = 1600;
}
else if (f == 12)
{
    var frekv = 3200;
}
else if (f == 13)
{
    var frekv = 6400;
}
else if (f == 14)
{
    var frekv = 10000;
}
document.getElementById('<%=frekvencija_mjerenja.ClientID%>').value = frekv;
var n = frekv * t;
var n1 = n * 0.1; //Rezanje prvih 10%
var n2 = n + n1;

var cut = n1 * 3;
var cut1 = n2 * 3;
var XYZ = VrijednostiXYZ.slice(cut, cut1);

var cut2 = cut1 - cut;

//Provjera prikupljenih podataka i log završetka prikupljanja
if (XYZ.length < cut2 | XYZ.length > cut2)
{
    podaciStatus = "0";
    end_log(podaciStatus);
}

else
{
    var greska = [];
    for (i = 0; i < XYZ.length; i++)
    {

        // provjera postoji li NaN ili boolean
        if (Number.isNaN(XYZ[i]))
        {
            greska.push(1);
        }
        else
        {
            greska.push(0);
        }

    }

    if (greska.includes(1))
```

```
                    {
                        podaciStatus = "0";
                        end_log(podaciStatus);
                    }
                    else
                    {
                        podaciStatus = "1";
                        end_log(podaciStatus);
                    }
                }

                document.getElementById('<%=status.ClientID%>').innerHTML = "Prikupljeno";
                document.getElementById('<%=xyz.ClientID%>').value = XYZ.join(",");
                document.getElementById('<%=Spremi.ClientID%>').click(); //Poziv .cs metode
spremanja podataka

            }
        }

    }

    //Log početka prikupljanja
    function start_log(connStatus)
    {
        var idMjerenja = document.getElementById('<%=id_mj.ClientID %>').value;
        var idSenzora = document.getElementById('<%=id_sen.ClientID %>').value;
        var brojMjerenja = document.getElementById('<%=broj_prikupljanja.ClientID %>').value;
        var stanje = "Start";

        $.ajax({
            type: "POST",
            url: "WebApp.aspx/StartLog",
            contentType: "application/json; charset=utf-8",
            data: '{"ID_mjerenja":"' + idMjerenja + '","ID_senzora":"' + idSenzora +
'","Broj_prikupljanja":"' + brojMjerenja + '","Stanje":"' + stanje + '","Status":"' +
connStatus +'"}',
            dataType: "json",
            success: function (result, status, xhr) { },
            error: function (xhr, status, error) {
                alert("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
            }
        });
    }

    //Log završetka prikupljanja
    function end_log(podaciStatus) {
        var idMjerenja = document.getElementById('<%=id_mj.ClientID %>').value;
        var idSenzora = document.getElementById('<%=id_sen.ClientID %>').value;
        var brojMjerenja = document.getElementById('<%=broj_prikupljanja.ClientID %>').value;
        var stanje = "End";

        $.ajax({
            type: "POST",
            url: "WebApp.aspx/EndLog",
            contentType: "application/json; charset=utf-8",
            data: '{"ID_mjerenja":"' + idMjerenja + '","ID_senzora":"' + idSenzora +
'","Broj_prikupljanja":"' + brojMjerenja + '","Stanje":"' + stanje + '","Status":"' +
podaciStatus + '"}',
            dataType: "json",
            success: function (result, status, xhr) { },
            error: function (xhr, status, error) {
                alert("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
            }
        });
    }

   //Automatsko prikupljanje
   function autp()
   {
       var idMjerenja = document.getElementById('<%=id_mj.ClientID %>').value;
       var odabirLokacije = document.getElementById('<%=izborLokacije.ClientID %>').value;
       $('#lokacija').prop('selectedIndex', odabirLokacije).change();
```

```
        if (document.getElementById('<%=skupno.ClientID %>').checked == true)
        {
            var value1 = document.getElementById('<%=trenutniSenzor.ClientID %>').value;
            document.getElementById('<%=senzor_od.ClientID %>').innerHTML = value1;

            document.getElementById('<%=senzor_do.ClientID %>').innerHTML =
document.getElementById('<%=ukupniBrojSenzora.ClientID %>').value;
        }

        //Plot grafa nakon svakog mjerenja
        $.ajax({
            type: "POST",
            url: "WebApp.aspx/Graf",
            contentType: "application/json; charset=utf-8",
            data: '{"ID_mjerenja":"' + idMjerenja + '"}',
            dataType: "json",
            success: function (data) {
                var result = JSON.parse(data.d)
                var grafX = result.X;
                var grafY = result.Y;
                var grafZ = result.Z;
                graf(grafX, grafY, grafZ);


            },
            error: function (xhr, status, error) {
                console.log("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
            }
        });

        var vrijemeIzmedu =
parseInt(document.getElementById('<%=vrijeme_izmedu.ClientID%>').value) * 1000;
        setTimeout(function () { document.getElementById('<%=Button2.ClientID%>').click() },
vrijemeIzmedu);
        document.getElementById('<%=status.ClientID%>').innerHTML = "Čekam";
        document.getElementById('<%=brdr.ClientID%>').className = "card border-left-warning
shadow";

    }

     //Graf
     function graf(grafX, grafY, grafZ)
     {

         var trace1 =
         {
             y: grafX,
             type: 'scatter',
             name: 'x os',
         };

         var trace2 =
         {
             y: grafY,
             type: 'scatter',
             name: 'y os',
         };

         var trace3 =
         {
             y: grafZ,
             type: 'scatter',
             name: 'z os',
         };

         var data = [trace1, trace2, trace3];

         var layout =
         {
             title: 'Vremenski niz akceleracija',
             xaxis: {
                 title: 'Indeks',
                 showgrid: true,
                 zeroline: false
```

```
            },
            yaxis: {
                title: 'Akceleracija [g]',
                showgrid: false,
                zeroline: false
            },
        };

        Plotly.newPlot('graf', data, layout);
    }

    function end()
    {
        document.getElementById('<%=status.ClientID%>').innerHTML = "Prikupljanje je
završeno";
        document.getElementById('<%=brdr.ClientID%>').className = "card border-left-primary
shadow";
        document.getElementById('<%=ciklus_od.ClientID%>').innerHTML = 0;
        document.getElementById('<%=ciklus_do.ClientID%>').innerHTML = "";
        document.getElementById('<%=senzor_od.ClientID%>').innerHTML = 1;
        document.getElementById('<%=senzor_do.ClientID%>').innerHTML = "";
        document.getElementById('<%=trenutniSenzor.ClientID%>').value = 0;
        document.getElementById('<%=ukupniBrojSenzora.ClientID%>').value = 0;

        var idMjerenja = document.getElementById('<%=id_mj.ClientID %>').value;
        $.ajax({
            type: "POST",
            url: "WebApp.aspx/Graf",
            contentType: "application/json; charset=utf-8",
            data: '{"ID_mjerenja":"' + idMjerenja + '"}',
            dataType: "json",
            success: function (data) {
                var result = JSON.parse(data.d)
                var grafX = result.X;
                var grafY = result.Y;
                var grafZ = result.Z;
                graf(grafX, grafY, grafZ);

            },
            error: function (xhr, status, error) {
                console.log("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
            }
        });
    }

    //Završetak prikupljanja
    function checkSenzora() {

        if (document.getElementById('<%=pojedinacno.ClientID %>').checked == true)
        {
            end();
        }
        else if (document.getElementById('<%=skupno.ClientID %>').checked == true)
        {
            var odabirLokacije = document.getElementById('<%=izborLokacije.ClientID
%>').value;
            $('#lokacija').prop('selectedIndex', odabirLokacije).change();

            document.getElementById('<%=ciklus_od.ClientID%>').innerHTML = 0;
            document.getElementById('<%=ciklus_do.ClientID%>').innerHTML = "";

            var duljina = document.getElementById('<%=ukupniBrojSenzora.ClientID %>').value;
            document.getElementById('<%=senzor_do.ClientID %>').innerHTML = duljina;

            var count = parseInt(document.getElementById('<%=trenutniSenzor.ClientID
%>').value);

            if (count < duljina)
            {
                count++;
                var odabirLokacije = document.getElementById('<%=izborLokacije.ClientID
%>').value;
                $("#lokacija").prop('selectedIndex', odabirLokacije).change();
```

```
                //document.getElementById('<%=broj_prikupljanja.ClientID %>').value = 1;
                var vrijemeIzmedu =
parseInt(document.getElementById('<%=vrijeme_izmedu.ClientID%>').value) * 1000;
                setTimeout(function () {
document.getElementById('<%=Button2.ClientID%>').click() }, vrijemeIzmedu);
                document.getElementById('<%=status.ClientID%>').innerHTML = "Izmjena senzora";
                document.getElementById('<%=brdr.ClientID%>').className = "card border-left-
warning shadow";
                document.getElementById('<%=senzor_od.ClientID %>').innerHTML = count;
                document.getElementById('<%=trenutniSenzor.ClientID %>').value = count;

            }
            else
            {
                end();
            }
        }

    }

    //Povijest prikupljanja
    function povijest()
    {
        var datePocetak = document.getElementById('<%=datum_pocetak.ClientID %>').value;
        var dateKraj = document.getElementById('<%=datum_kraj.ClientID %>').value;
        var newdate = datePocetak.split("/").reverse().join("/");
        var newdate2 = dateKraj.split("/").reverse().join("/");

        var datum = [];
        var grafX = [];
        var grafY = [];
        var grafZ = [];

        $.ajax(
            {
                type: "POST",
                url: "WebApp.aspx/Povijest",
                contentType: "application/json; charset=utf-8",
                data: '{"datum_od":"' + newdate + '","datum_do":"' + newdate2 + '"}',
                dataType: "json",
                success: function (data) {
                    var result = JSON.parse(data.d)
                    datum = result.Datum;
                    grafX = result.X;
                    grafY = result.Y;
                    grafZ = result.Z;

                    var trace1 =
                    {
                        y: grafX,
                        type: 'scatter',
                        name: 'x os',
                        text: datum
                    };

                    var trace2 =
                    {
                        y: grafY,
                        type: 'scatter',
                        name: 'y os',

                    };

                    var trace3 =
                    {
                        y: grafZ,
                        type: 'scatter',
                        name: 'z os',

                    };

                    var data = [trace1, trace2, trace3];

                    var layout =
                    {
```

```
                        title: 'Vremenski niz akceleracija',
                        xaxis: {
                            title: 'Indeks',
                            showgrid: true,
                            zeroline: false
                        },
                        yaxis: {
                            title: 'Akceleracija [g]',
                            showgrid: false,
                            zeroline: false
                        },
                    };

                    Plotly.newPlot('graf2', data, layout);

                },
                error: function (xhr, status, error) {
                    console.log("Result: " + status + " " + error + " " + xhr.status + " " +
xhr.statusText)
                }
            });
    }

    //U slučaju nekog teškog zastoja ili greške
    function alert1()
    {
        if (alert('Došlo je do greške')) { }
        else window.location.reload();
    }

    //Refresh stranice
    if (window.history.replaceState) {
        window.history.replaceState(null, null, window.location.href);
    }
```

Prilog III.

Izlist koda poslužiteljske strane

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Web.Services;
using System.Configuration;
using System.Text;
using Newtonsoft.Json;

namespace prikupljanje_podataka
{
    public partial class WebApp : System.Web.UI.Page
    {
        public int max;

        protected void Page_Load(object sender, EventArgs e)
        {
            //Postavljanje ID mjerenja
            if (ViewState["VisitCount"] == null)
            {
                using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
4GOI8BQ\\SQLEXPRESS;Initial Catalog=AKC;Integrated Security=True"))
                {
                    SqlCommand cmd = new SqlCommand("dbo.SP_MAX", con);
                    con.Open();
                    max = (int)cmd.ExecuteScalar();
                    con.Close();
                    max++;
                    id_mj.Value = Convert.ToString(max);


                }
            }

            //Poziv za popunjavanje dropdown menija
            BindLokacija();

            //Count za broj ponavljanja
            int PostBackCount = 1;
            if (IsPostBack == true)
            {
                if (ViewState["VisitCount"] == null)
                {
                    ViewState["VisitCount"] = PostBackCount;// writing Values in ViewState

                }

                else
                {
                    ViewState["VisitCount"] = Convert.ToInt32(ViewState["VisitCount"]) + 1;//
writing values in viewstate

                }

                trenutniBrojPonavljanja.Value = ViewState["VisitCount"].ToString();
                int broj = Convert.ToInt32(broj_ponavljanja.Value);
                int count = Convert.ToInt32(trenutniBrojPonavljanja.Value);
                ciklus_do.InnerHtml = broj_ponavljanja.Value;
                ciklus_od.InnerText = Convert.ToString(count);


                if (count < broj)
                {
```

```
                    Page.ClientScript.RegisterStartupScript(this.GetType(), "CallMyFunction",
"autp()", true);
                }
                else
                {
                    ViewState["VisitCount"] = 0;
                    Page.ClientScript.RegisterStartupScript(this.GetType(), "CallMyFunction2",
"checkSenzora()", true);
                }
            }
        }

        //Popunajvanje dropdown Lokacija
        void BindLokacija()
        {
            SqlConnection conn = new SqlConnection();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;
            SqlCommand cmd = new SqlCommand();
            DataTable dataTable = new DataTable();

            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Connection = conn;
            cmd.CommandText = "SP_LOKACIJA_SELECT";

            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter();
            sqlDataAdapter.SelectCommand = cmd;
            sqlDataAdapter.Fill(dataTable);

            lokacija.DataSource = dataTable;
            lokacija.DataTextField = "Lokacija";
            lokacija.DataValueField = "ID_brick";
            lokacija.DataBind();
            lokacija.Items.Insert(0, new ListItem("Lokacija:", "0"));
            senzor.Items.Insert(0, new ListItem("Senzor:", "0"));
        }

        //Popunjavanje dropdown Senzor
        [WebMethod]
        public static string BindSenzor(string ID_brick)
        {
            int ID = Convert.ToInt32(ID_brick);
            SqlConnection conn = new SqlConnection();
            SqlCommand cmd = new SqlCommand();
            DataTable dataTable = new DataTable();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;

            cmd.Connection = conn;
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "SP_SENZOR_SELECT";
            cmd.Parameters.Add(new SqlParameter("@ID_brick", ID));

            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter();
            sqlDataAdapter.SelectCommand = cmd;
            sqlDataAdapter.Fill(dataTable);

            StringBuilder stringBuilder = new StringBuilder();
            stringBuilder.Append("<option value=\"0\">Senzor</option>");
            foreach (DataRow dr in dataTable.Rows)
                stringBuilder.Append("<option value=\"" + dr["ID senzora"] + "\">" +
dr["Naziv_senzora"] + "</option>");
            return stringBuilder.ToString();


        }

        //Dohvaćanje podataka za polja IP i Port
        [WebMethod]
        public static string FillLokacija(string ID_brick)
        {

            string vrijednosti;
            int ID = Convert.ToInt32(ID_brick);
            SqlConnection conn = new SqlConnection();
```

```
            SqlCommand cmd = new SqlCommand();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;

            cmd.Connection = conn;
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "SP_IP_SELECT";
            cmd.Parameters.Add(new SqlParameter("@ID_brick", ID));


            conn.Open();
            SqlDataReader rdr = cmd.ExecuteReader();
            rdr.Read();
            var IP = rdr[0].ToString();
            var Port = rdr[1].ToString();
            vrijednosti = IP + " " + Port;
            conn.Close();

            return vrijednosti;

        }

        //Dohvaćanje podataka za polja UID i ID senzora
        [WebMethod]
        public static string FillSenzor(string ID_senzora)
        {

            string vrijednosti;
            int ID = Convert.ToInt32(ID_senzora);
            SqlConnection conn = new SqlConnection();
            SqlCommand cmd = new SqlCommand();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;

            cmd.Connection = conn;
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "SP_UID_SELECT";
            cmd.Parameters.Add(new SqlParameter("@ID_senzora", ID));


            conn.Open();
            SqlDataReader rdr = cmd.ExecuteReader();
            rdr.Read();
            var IP = rdr[0].ToString();
            vrijednosti = IP;
            conn.Close();

            return vrijednosti;

        }

        //Spremanje log početka prikupljanja u bazu
        [WebMethod]
        public static void StartLog(string ID_mjerenja, string ID_senzora, string
Broj_prikupljanja, string Stanje, string Status)
        {

            int idMj = Convert.ToInt32(ID_mjerenja);
            int idSen = Convert.ToInt32(ID_senzora);
            int brMj = Convert.ToInt32(Broj_prikupljanja);
            int status = Convert.ToInt32(Status);

            SqlConnection conn = new SqlConnection();
            SqlCommand cmd = new SqlCommand();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;

            cmd.Connection = conn;
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "SP_UNOS_LOG";
            cmd.Parameters.Add(new SqlParameter("@ID_mjerenja", idMj));
            cmd.Parameters.Add(new SqlParameter("@ID_senzora", idSen));
            cmd.Parameters.Add(new SqlParameter("@Broj prikupljanja", brMj));
            cmd.Parameters.Add(new SqlParameter("@Stanje", Stanje));
            cmd.Parameters.Add(new SqlParameter("@Status", status));
```

```
            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }

        //Spremanje log završetka prikupljanja u bazu
        [WebMethod]
        public static void EndLog(string ID_mjerenja, string ID_senzora, string
Broj_prikupljanja, string Stanje, string Status)
        {
            int idMj = Convert.ToInt32(ID_mjerenja);
            int idSen = Convert.ToInt32(ID_senzora);
            int brMj = Convert.ToInt32(Broj_prikupljanja);
            int status = Convert.ToInt32(Status);

            SqlConnection conn = new SqlConnection();
            SqlCommand cmd = new SqlCommand();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;

            cmd.Connection = conn;
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "SP_UNOS_LOG";
            cmd.Parameters.Add(new SqlParameter("@ID_mjerenja", idMj));
            cmd.Parameters.Add(new SqlParameter("@ID_senzora", idSen));
            cmd.Parameters.Add(new SqlParameter("@Broj_prikupljanja", brMj));
            cmd.Parameters.Add(new SqlParameter("@Stanje", Stanje));
            cmd.Parameters.Add(new SqlParameter("@Status", status));

            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }

        //Dohvaćanje podataka za graf
        [WebMethod]
        public static string Graf(string ID_mjerenja)
        {
            int idMj = Convert.ToInt32(ID_mjerenja);
            SqlConnection conn = new SqlConnection();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;
            SqlCommand cmd = new SqlCommand();
            DataTable dataTable = new DataTable();

            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Connection = conn;
            cmd.CommandText = "SP_GRAF";
            cmd.Parameters.Add(new SqlParameter("@ID_mjerenja", idMj));

            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter();
            sqlDataAdapter.SelectCommand = cmd;
            sqlDataAdapter.Fill(dataTable);

            var x = dataTable.Rows.Cast<DataRow>().Select(row => row[0].ToString()).ToArray();
            var y = dataTable.Rows.Cast<DataRow>().Select(row => row[1].ToString()).ToArray();
            var z = dataTable.Rows.Cast<DataRow>().Select(row => row[2].ToString()).ToArray();

            var result = new
            {
                X = x,
                Y = y,
                Z = z,
            };

            var json = JsonConvert.SerializeObject(result);

            return json;
        }

        //Dohvaćanje podataka za povijest
        [WebMethod]
        public static string Povijest(string datum_od, string datum_do)
        {
```

```
            SqlConnection conn = new SqlConnection();
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["AKCConnectionString1"].ConnectionString;
            SqlCommand cmd = new SqlCommand();
            DataTable dataTable = new DataTable();

            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Connection = conn;
            cmd.CommandText = "SP_DATUM_SELECT";
            cmd.Parameters.Add(new SqlParameter("@DATUM_POCETAK", datum_od));
            cmd.Parameters.Add(new SqlParameter("@DATUM KRAJ", datum do));

            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter();
            sqlDataAdapter.SelectCommand = cmd;
            sqlDataAdapter.Fill(dataTable);

            var datum = dataTable.Rows.Cast<DataRow>().Select(row =>
row[0].ToString()).ToArray();
            var x = dataTable.Rows.Cast<DataRow>().Select(row => row[1].ToString()).ToArray();
            var y = dataTable.Rows.Cast<DataRow>().Select(row => row[2].ToString()).ToArray();
            var z = dataTable.Rows.Cast<DataRow>().Select(row => row[3].ToString()).ToArray();
            var result = new
            {
                Datum = datum,
                X = x,
                Y = y,
                Z = z,
            };

            var json = JsonConvert.SerializeObject(result);

            return json;
        }

        //Poziv webservisa za spremanje vrijednosti u bazu
        protected void Spremi_Click(object sender, EventArgs e)
        {
              char Separator = Convert.ToChar(separator.Value);
              decimal[] vrijednosti = Array.ConvertAll(xyz.Value.Split(Separator),
Decimal.Parse);
              int IdMj = Convert.ToInt32(id_mj.Value);
              int BrojPrikupljanja = Convert.ToInt32(broj_prikupljanja.Value);
              int BrojOkretaja = Convert.ToInt32(broj_okretaja.Value);
              int Frekvencija = Convert.ToInt32(frekvencija_mjerenja.Value);
              string TipKvara = tip_kvara.Value;
              int IDSen = Convert.ToInt32(id_sen.Value);

              localhost.BazaSpremi mys = new localhost.BazaSpremi();

              mys.Spremi(vrijednosti, IdMj, BrojPrikupljanja, BrojOkretaja, Frekvencija,
TipKvara, IDSen);
              int BrojP = Convert.ToInt32(broj_prikupljanja.Value);
              BrojP++;
              broj_prikupljanja.Value = Convert.ToString(BrojP);

        }



    }
}
```

Prilog IV.

Izlist koda web servisa

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;

namespace WebService_Spremanje
{
    /// <summary>
    /// Summary description for BazaSpremi
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the
following line.
    // [System.Web.Script.Services.ScriptService]
    public class BazaSpremi : System.Web.Services.WebService
    {
        SqlConnection con = new SqlConnection("Data Source=T430-SMST\\KOLAR_SQL;Initial
Catalog=AKC;User ID=martinc;Password=Martin123");

        [WebMethod]
        public void Spremi(decimal[] vrijednosti, int IdMj, int BrojPrikupljanja, int
BrojOkretaja, int Frekvencija, string TipKvara, int IDSen)
        {
            List<int> greske = new List<int>();
            int status;
            for (int i = 0; i < vrijednosti.Length; i += 3)
            {
                decimal x = vrijednosti[i];
                decimal y = vrijednosti[i + 1];
                decimal z = vrijednosti[i + 2];
                SqlCommand cmd = new SqlCommand("dbo.SP_SPREMI", con)
                {
                    CommandType = CommandType.StoredProcedure
                };
                cmd.Parameters.AddWithValue("@ID_mjerenja", IdMj);
                cmd.Parameters.AddWithValue("@x", x);
                cmd.Parameters.AddWithValue("@y", y);
                cmd.Parameters.AddWithValue("@z", z);
                cmd.Parameters.AddWithValue("@broj_prikupljanja", BrojPrikupljanja);
                cmd.Parameters.AddWithValue("@broj_okretaja", BrojOkretaja);
                cmd.Parameters.AddWithValue("@frekvencija", Frekvencija);
                cmd.Parameters.AddWithValue("@tipkvara", TipKvara);
                cmd.Parameters.AddWithValue("@id_senzora", IDSen);

                SqlParameter greska = new SqlParameter("@GRESKA", System.Data.SqlDbType.Int);
                greska.Direction = System.Data.ParameterDirection.Output;
                cmd.Parameters.Add(greska);

                con.Open();
                cmd.ExecuteNonQuery();
                con.Close();

                string Greska = greska.ToString();


                if (!String.IsNullOrEmpty(Greska))
                {
                    string poruka = "TRANSAKCIJA SPREMANJA JE USPJELA!";
                    System.Diagnostics.Debug.WriteLine(poruka);
                    int Greska1 = 0;
                    greske.Add(Greska1);
```

```
                }
                else
                {
                    string poruka = "TRANSAKCIJA SPREMANJA VRIJEDNOSTI NIJE USPJELA!";
                    System.Diagnostics.Debug.WriteLine(poruka);
                    int Greska1 = 1;
                    greske.Add(Greska1);
                }

            }

            if (greske.Distinct().Skip(1).Any())
            {
                status = 0;
            }
            else
            {
                status = 1;
            }
            SqlCommand cmd2 = new SqlCommand("dbo.SP_UNOS_LOG", con)
            {
                CommandType = CommandType.StoredProcedure
            };
            cmd2.Parameters.AddWithValue("@ID_mjerenja", IdMj);
            cmd2.Parameters.AddWithValue("@ID_senzora", IDSen);
            cmd2.Parameters.AddWithValue("@Broj_prikupljanja", BrojPrikupljanja);
            cmd2.Parameters.AddWithValue("@Stanje", "Spremi");
            cmd2.Parameters.AddWithValue("@Status", status);
            con.Open();
            cmd2.ExecuteNonQuery();
            con.Close();

        }


    }
}
```