Razvoj sustava za praćenje u stvarnom vremenu energetskih vrijednosti industrijskih proizvodnih postrojenja baziran na industrijskom upravljačkom sustavu s integracijom aplikacije treće strane

Tisaj, Luka

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:450017

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 111

RAZVOJ SUSTAVA ZA PRAĆENJE U STVARNOM VREMENU ENERGETSKIH VRIJEDNOSTI INDUSTRIJSKIH PROIZVODNIH POSTROJENJA BAZIRAN NA INDUSTRIJSKOM UPRAVLJAČKOM SUSTAVU S INTEGRACIJOM APLIKACIJE TREĆE STRANE

Luka Tisaj

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 111

RAZVOJ SUSTAVA ZA PRAĆENJE U STVARNOM VREMENU ENERGETSKIH VRIJEDNOSTI INDUSTRIJSKIH PROIZVODNIH POSTROJENJA BAZIRAN NA INDUSTRIJSKOM UPRAVLJAČKOM SUSTAVU S INTEGRACIJOM APLIKACIJE TREĆE STRANE

Luka Tisaj

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 111

Pristupnik:	**Luka Tisaj (0036524900)**
Studij:	Elektrotehnika i informacijska tehnologija
Profil:	Elektrostrojarstvo i automatizacija
Mentor:	prof. dr. sc. Damir Sumina

Zadatak: **Razvoj sustava za praćenje u stvarnom vremenu energetskih vrijednosti industrijskih proizvodnih postrojenja baziran na industrijskom upravljačkom sustavu s integracijom aplikacije treće strane**

Opis zadatka:

Potrebno je razviti sustav za praćenje u stvarnom vremenu energetskih vrijednosti industrijskih proizvodnih postrojenja baziran na softverskom programibilnom logičkom kontroleru SIEMENS SIMATIC CPU1507S iz serije S7-1500 koji omogućuje povezivanje aplikacija treće strane preko programskog paketa SIMATIC ODK 1500S (Open Development Kit). Potrebno je razviti aplikaciju u okruženju Microsoft Visual Studio koja će primati podatke od PLC-a u stvarnom vremenu, pohranjivati i generirati izvještaj na bazi dana, tjedna i mjeseca te slati izvještaj na definirane e-mail adrese. Vizualizacija za navedenu aplikaciju će omogućiti prikaz stvarnih i arhiviranih podataka te mogućnost konfiguracije izvještaja prema zahtjevima korisnika. Sustav je potrebno eksperimentalno testirati na stvarnom postrojenju.

Rok za predaju rada: 28. lipnja 2024.

Nakon kratkih pet godina studiranja, vrijeme je da se stvar privede kraju.

Ovim putem htio bih istaknuti svoju obitelj i prijatelje, sve koji su bili uz mene te moje volje i nevolje u akademskom, ali i ukupnom obrazovanju. Moje prijatelje iz Ulice koji su trpili moja jadikovanja oko ispita, labosa, predavanja i svih ostalih problema koji dolaze u paketu sa studiranjem.

Također, mentoru prof. dr. sc. Damiru Sumini zahvaljujem na angažmanu, uloženom trudu i vremenu u moj, kako završni, tako i diplomski rad i sve ostale mentorski vođene radove na Fakultetu. Kolegama iz tvrtke Average Solutions koji su mi omogućili izradu ovog diplomskog rada, zahvaljujem na idejama, rješenjima problema oko rada i savjetima za bezbolnu izradu ovog projekta.

Najveću zahvalu dugujem svojim kolegama sa Zavoda za elektrostrojarstvo i automatizaciju te Ferovkama i Ferovcima uopće. Da nije bilo kolegijalnosti koja uvijek vlada među studentima ovog Fakulteta, vjerojatno bih još pokušavao položiti neki predmet s prve godine.

Hvala.

Sadržaj

Popis slika

Popis tablica

1. Uvod

U današnje doba ubrzanog razvoja automatizacijske i industrijske tehnologije, sve češća je potreba za praćenjem ključnih podataka i vrijednosti potrošnje nekog sustava. Osim praćenja potrošnje, sve veći je trend korištenje korisničkih, *license-free*, aplikacija kao rješenje za nadgledanje velike količine podataka u nekom manjem sustavu ili postrojenju.

Veliki industrijski proizvođači (*Siemens, Schneider Electric, ABB* i drugi) imaju svoja razrađena rješenja za nadziranje potrošnje i akviziciju velike količine podataka koji su spremni za korištenje i pregled na mjesečnoj/godišnjoj razini. Jedan od problema koji se pojavljuje je visoki cjenovni prag koji postavljaju velike kompanije na svoje proizvode i licence što nije financijska prepreka za velika postrojenja i proizvodne kompanije, ali za manje proizvodnje može biti trošak koji nije isplativ iako i dalje postoji potreba za nadziranjem potrošnje energije.

Cilj ovog diplomskog rada jest razvoj programskog rješenja prikupljanja podataka s industrijske opreme korištenjem neovisnog softvera. Ovakvo programsko rješenje omogućuje prilagodljivu i prihvatljiviju alternativu velikim proizvođačima, a zadržava sve osnovne funkcionalnosti *monitoring* sustava.

Glavna ideja rada je korištenje *InfluxDB* baze podataka za prikupljanje mjerenih veličina s perifernih uređaja te grafička obrada tih podataka u obliku izvještaja za krajnjeg korisnika. Generirani izvještaj prikazuje željene podatke u određenom vremenskom intervalu u obliku vremenskog dijagrama.

Kako bi se pokazala funkcionalnost ovog sustava, korišteno je nekoliko uređaja: SENTRON PAC za mjerenje električnih vrijednosti nekog pogona i SM 1281 modul koji uz četiri piezoelektrična senzora čini sustav za mjerenje vibracija. Za prikupljanje podataka sa

SM 1281 modula korišten je *Siemens* S7-1200 programirljivi logički kontroler (PLC) koji podatke šalje glavnom PLC-u koji se nalazi na industrijskom računalu IPC 277G. IPC posjeduje tzv. softverski kontroler 1507s koji je zapravo programska verzija standardnog PLC-a. Podaci dobiveni sa senzora vibracija i električne veličine sa SENTRON PAC-a prikupljaju se u softverskom PLC-u i korištenjem OPC UA protokola čitaju se pomoću *C#* skripte te zapisuju u *InfluxDB* bazu podataka.

Baza podataka kontinuirano prikuplja podatke o potrošnji i vibracijama koji su trajno dostupni korisniku. U ovom radu je također prikazan i grafički program *Grafana* koji posjeduje mogućnost slanja upita (engl. *query*) *InfluxDB* bazi te se na taj način grade *dashboard-ovi* koji ilustriraju trenutne ili prijašnje vrijednosti koje su zapisane u bazi.

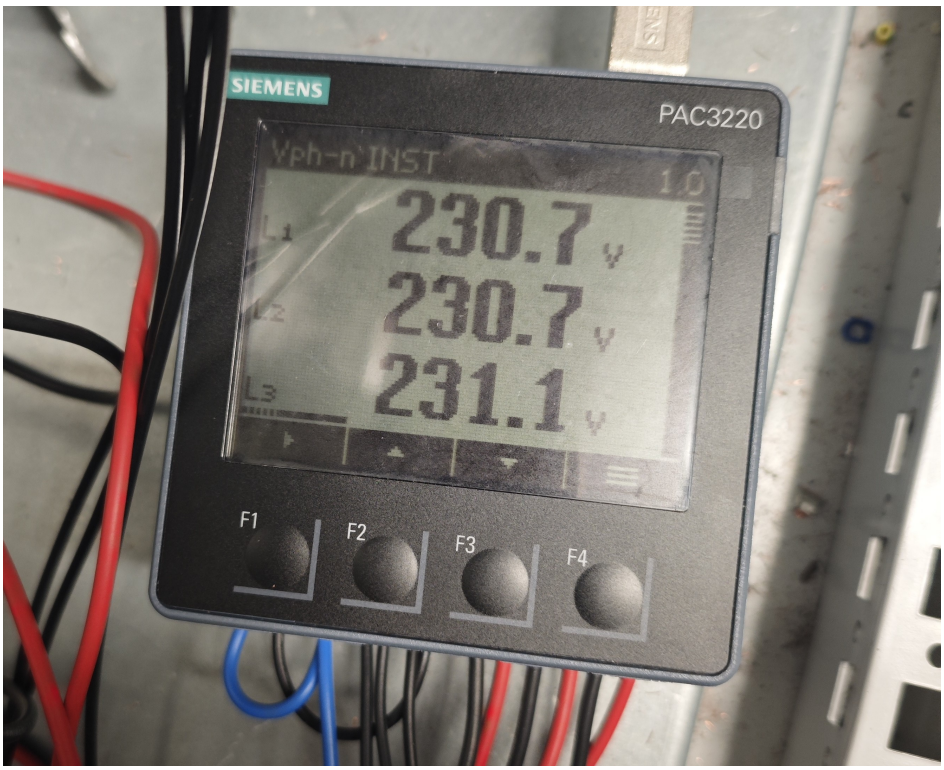
Osim *Grafane*, razvijen je i jednostavan program koji omogućava korisniku unos početnog i krajnjeg datuma te temeljem tog vremenskog intervala ispisuje vrijednosti pohranjene u bazi u obliku `.csv` datoteke i *MS Excel* izvještaja.

2. Opis zadatka i opreme

2.1. Mjerna oprema

2.1.1. SENTRON PAC3220 i elektromotorni pogon

Kako bi se izmjerile električne veličine nekog pogona, korišten je sofisticirani mjerni uređaj SENTRON PAC3220 tvrtke *Siemens*. PAC3220 je kompaktni mjerni uređaj električne energije prikladan za korištenje u industrijskim, administrativnim i komercijalnim okruženjima u kojima je potrebno osnovno nadziranje potrošnje energije i mjerenje stvarnih vrijednosti. Uređaj samostalno može nadzirati više od 100 parametara ili može biti primjenjivan u industrijskom nadzoru, automatizaciji građevina ili generalnom nadziranju el. energije[2]. Detaljni podaci uređaja navedeni su u tablici 2.1.1

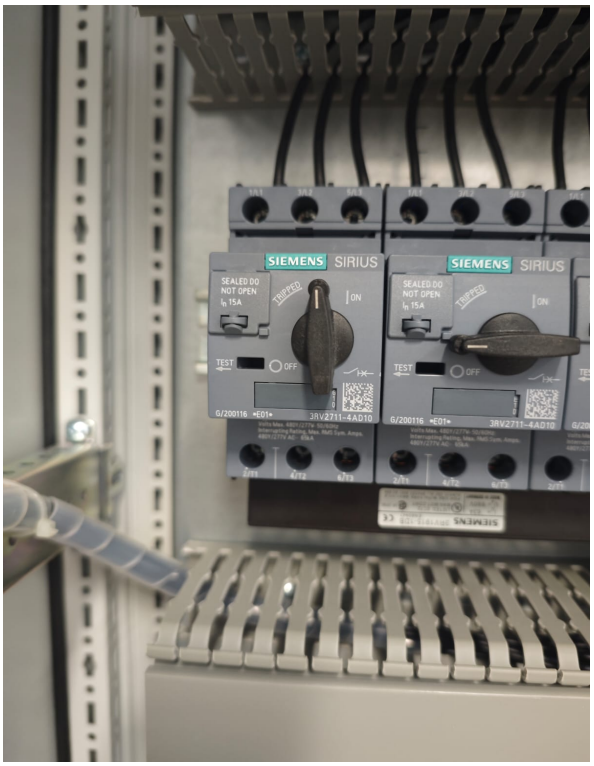


Slika 2.1.1 *Siemens* SENTRON PAC3220

Tablica 2.1.1 PAC3220 specifikacije

Pomoćni ulazi	100...250 V, 50/60 Hz, 8 VA
Ulazna struja	1/5 A
Ulazni napon	57.7/100...400/690 V

Naponi pogona se nadziru direktnim spojem na mjerni uređaj pomoću stezaljki u elektroormaru. Sve tri faze napona mjerene su na izlazu motorske zaštitne sklopke pripadajućeg motora(slika 2.1.2).



Slika 2.1.2 Motor-zaštitna sklopka

Elektromotor na kojem je izvršeno mjerenje snage je 37 kW i nazivne struje 66 A(tablica 2.1.2), upravljan je pretvaračem napona i frekvencije, a referentne brzine zadane su upravljačkim panelom na elektroormaru. Konfiguracija panela, pretvarača i popratne opreme nije obuhvaćeno ovim radom.

Budući da SENTRON PAC dozvoljava jakost struje do 5 A, potrebno je koristiti strujna kliješta ili strujne transformatore kako bi se mjerile vrijednosti struje pogona. Transformator u općenitom smislu principom elektromagnetske indukcije pretvara izmjenične naponske/strujne razine ovisno o broju zavoja na primarnom i sekundarnom namotu. Uzevši u obzir nazivnu struju motora te struju pokretanja/kratkog spoja asinkronog motora,koja može biti nekoliko puta veća od nazivne, korištena su tri strujna transformatora

prijenosnog omjera 200/5 A. Transformator je proizvod tvrtke *HOBUT*, a korišteni model je CTSCM40.

U slučaju strujnog transformatora u ovom projektu(slika 2.1.3), primarna strana transformatora omotana je oko energetskog kabela koji napaja motor i u kojemu se očekuju veliki iznosi struje, dok je sekundarna strana, s mnogo većim brojem namota, spojena na neki mjerni uređaj; u ovom slučaju, SENTRON PAC.

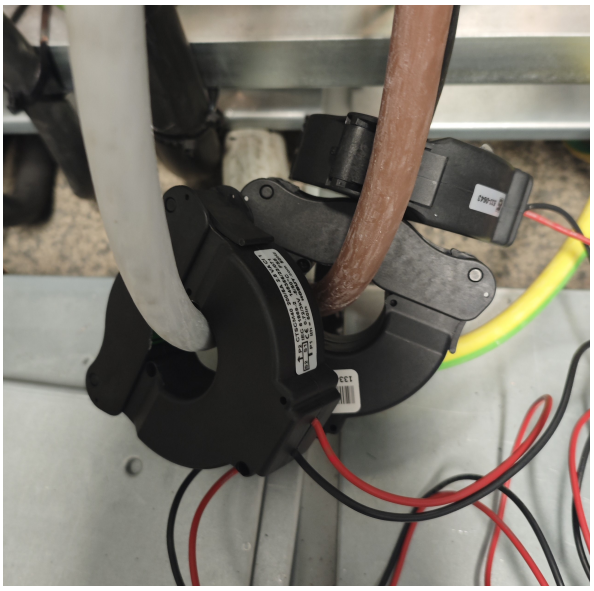
Tablica 2.1.2 Nazivni podaci elektromotora

Proizvođač	Končar
Nazivni napon	400 V
Nazivna struja	66 A
Nazivna snaga	37 kW
Nazivna frekvencija	50 Hz
Nazivna brzina	1480 o/min
Faktor snage	0.87

Ako se mjeri neka električna veličina mjernim transformatorom, nužno je uzeti u obzir faktor/omjer broja zavoja primara i sekundara kako bi se dobivene vrijednosti ispravno interpretirale. Kako je kod ovog strujnog transformatora omjer "200 na 5", zaključuje se da svaku struju dobivenu na sekundarnom namotu (ona koju mjeri SENTRON PAC) treba množiti s faktorom 40 kako bi se dobila stvarna vrijednost struje primara koja je zapravo od interesa. Ovo jednostavno skaliranje moguće je napraviti ili u mjernom uređaju, ili u nekom nadređenom sustavu pri obradi podataka. Ovdje je to napravljeno u postavkama SENTRON PAC-a tako da je unesen omjer transformatora na temelju kojeg uređaj samostalno skalira vrijednost struje.

Mjerenjem svih faza napona i struja, uređaj također računa vrijednosti prividne, radne i jalove snage; faktor snage, frekvenciju i druge veličine. Dodatno, PAC daje i prosječne vrijednosti struje te faznih i linijskih napona na mjernome mjestu.

SENTRON PAC pruža dva načina komunikacije s ostatkom sustava: *MODBUS TCP* ili *PROFINET* protokol. *MODBUS TCP* jedna je od inačica protokola iz *MODBUS* obitelji neovisna o proizvođaču opreme te je namijenjena nadzoru i upravljanju automatizacijske opreme.[3] Za *PROFINET* komunikaciju potreban je dodatan modul za mjerni uređaj koji nije bio dostupan u trenutku izvođenja ovog rada te nije korišten za komunikaciju.



Slika 2.1.3 Strujni transformatori

2.1.2. Mjerenje vibracija

Mehaničke vibracije vrsta su vibracija koje se mogu osjetiti i mjeriti na površini nekog objekta. U kontekstu strojne opreme, ovo se posebno odnosi na površine strojeva, dodatnih komponenata i temelja. Mehaničke vibracije sadrže veliku količinu informacija za nadzor pogona, a najbitnije su:

- pokazatelj stanja stroja,
- pokazatelj dinamičkih naprezanja pogona, postolja i srodnih komponenata,
- pokazatelj sigurnosti rada, vijeka trajanja i ekonomičnosti stroja,
- osnove dijagnostike i ublažavanja vibracija.[4]

Najčešći uzroci neželjenih vibracija kod rotacijskih strojeva su: neuravnoteženost stroja, oštećenje ležajeva, defekti na zupčanicima i dr. U današnje doba mjerenje vibracija pogona postaje od interesa zato što se pokazalo da vibracije značajno povećavaju dugoročne troškove servisa, remonta ili zamjene uređaja. Mjerenjem vibracija dobiva se okvirna informacija "zdravlja" stroja i mogućnost predikcije kvara koji se zatim mogu uklopiti u plan rada pogona kako bi pogon imao što kraće vrijeme izvan funkcije(engl. *downtime*). Dvije najbitnije veličine koje ukazuju na jačinu vibracija su brzina i akceleracija vibracije. Brzina vibracije mjeri se u mm/s te je propisana njezina dozvoljena/preporučena vrijed-

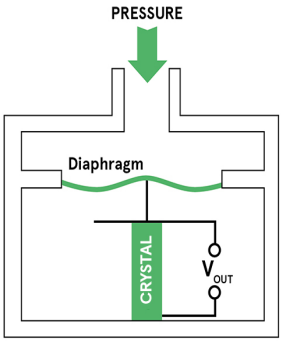
nost standardom ISO 10815, a u ovom projektu označena je s vRMS gdje je RMS(engl. *Root mean square*) efektivna vrijednost brzine. Akceleracije su dane u m/s^2 i nisu propisane standardom, ali postoje okvirne poželjne vrijednosti koje se očekuju. Ova veličina bit će označena s aRMS.

Česta metoda mjerenja je korištenje piezoelektričnih senzora. Ovi senzori iskorištavaju prirodnu reakciju nekih kristalnih struktura u materijalima koje, pri kompresiji uzrokovanoj vanjskom pritisnom silom, postaju izvor statičkoga naboja koji je mjerljiv uz korištenje pojačala signala. Prednost ovakvog senzora je robusnost, mogućnost rada u nepovoljnim uvjetima te manjak potrebe za vanjskim izvorom energije koji bi napajao senzor(npr. baterija ili razvod). Navedena svojstva čine piezo-senzore idealnima za primjenu u industrijskim uvjetima.[5]

SM 1281 koristi četiri piezoelektrične sonde oznake VIB1 do VIB4. Prema priručniku[4], tipični senzori akceleracije imaju frekvencijsko mjerno područje do 30 kHz nakon čega dolazi do rezonancije senzora, ali poremećaji mjereni u ovome slučaju ne ulaze u to područje stoga rezonancija nije uzeta u obzir.

Sonde prikazane na slici 2.1.4b osim piezo-senzora sadrže magnetnu ploču na svome kraju kako bi se jednostavno postavile na neki uređaj kojemu se mjere vibracije. Budući da je motor u pitanju nešto starije proizvodnje, izvedba njegovog kućišta je od čelika te je pogodan za postavljanje senzora na koju god lokaciju na kućištu je to potrebno. Kako bi se vibracije mjerile što je ispravnije moguće, dane su neke preporuke u priručniku modula.[4] Jedna od bitnih opaski kaže da se senzori ne smiju postaviti na "kapu" ventilatora na stražnjem dijelu motora. Kapa ventilatora trpi velike iznose vibracija zbog svoje elastičnosti i deformacija te nije dobar pokazatelj mehaničkog naprezanja ostatka stroja.

Na slici 2.1.5 vidljiv je ispitivani pogon i pozicije sva četiri senzora vibracija. Senzori VIB1 i VIB2 postavljeni su na stražnji, bočni dio motora i to u dva različita smjera kako bi se izmjerile vibracije u dvije osi. VIB3 i VIB4 nalaze se na prednjem dijelu jarma motora bliže ležajevima također u dvije osi. Svi senzori povezani su oklopljenim kabelom koji ublažava vanjske elektromagnetske smetnje koje su posebno izražene u laboratorijima električnih strojeva zbog prisutnosti energetskih pretvarača koji na visokim sklopnim frekvencijama proizvode veliku kolčinu intereferencije s ostalim signalima u svojoj oko-

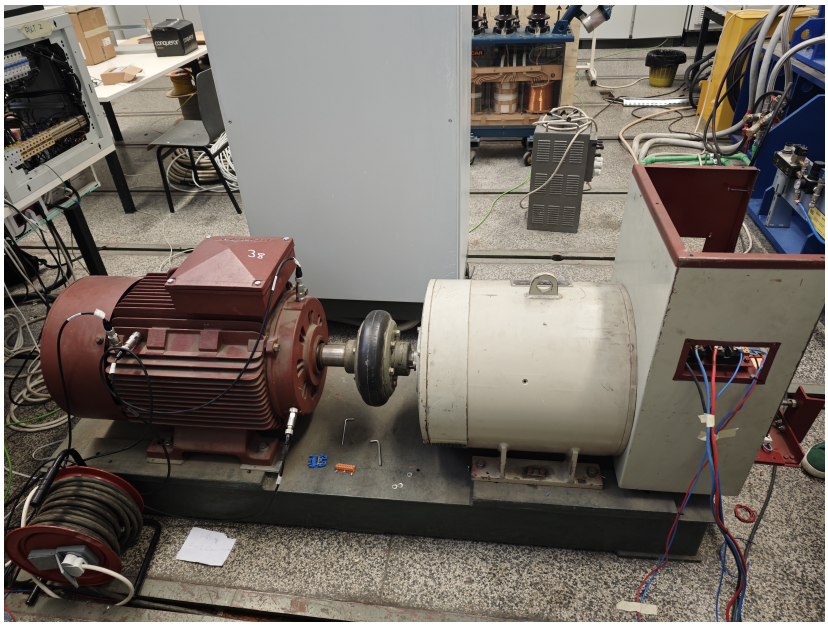


(a) Prikaz općenitog piezoeleketričnog senzora

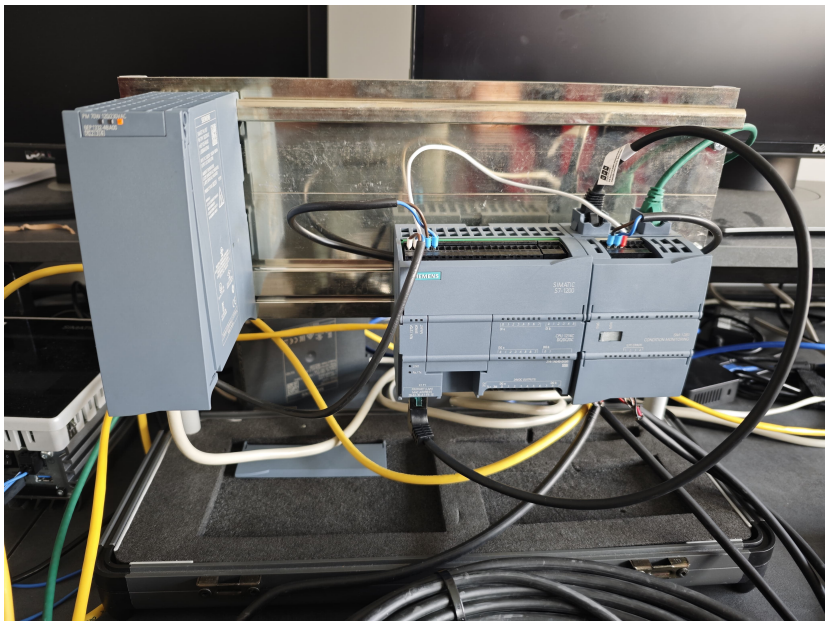
(b) VIB senzor korišten u radu

Slika 2.1.4 Konceptualni i stvarni prikaz piezo-senzora

lini. Kabeli su povezani konektorom s modulom SM 1281 koji je zatim povezan s S7-1200 programirljivim logičkim kontrolerom(PLC).



Slika 2.1.5 Ispitivani elektromotorni pogon s VIB senzorima



Slika 2.1.6 S7-1200 CPU i SM 1281 modul

2.2. Prikupljanje podataka

2.2.1. S7-1200

Kako bi modul SM 1281 obavljao traženu zadaću, potreban mu je S7-1200 PLC. Programiljivi logički kontroleri(PLC, u nastavku samo "kontroler") posebna su skupina robusnih i pouzdanih industrijskih računala koja imaju široku primjenu u svim područjima industrijske automatizacije i IoT-a(*Internet of things*). S7-1200 uređaj je proizvođača *Siemens* kao i pripadajući modul iz prošlog poglavlja. *Siemens* kontroleri u pravilu se programiraju u specijaliziranom programskom jeziku STEP7, a najčešće korišteno programsko okruženje jest *TIA Portal*.

Više o programiranju kontrolera i *TIA Portalu* je prikazano u poglavlju 3.1. U kontekstu ovog rada, svrha S7-1200 prikupljanje je podataka dobivenih sa senzora vibracije preko modula SM 1281. Glavni dio programa na ovom kontroleru je funkcijski blok(FB) *CMS* u kojemu su definirana 4 kanala, po jedan za svaki od 4 senzora vibracija. S7-1200 upravlja tokom podataka, nadzire stanje svakog senzora i obrađuje analogne signale sa SM 1281 modula. Kontroler sadrži jedan *Ethernet port*(slika 2.1.6) na donjoj strani kućišta i povezan je mrežnim kabelom sa SM 1281 modulom koji sadrži dva *porta* interno povezana u istu podmrežu.

2.2.2. Industrijsko računalo

Glavni dio *hardware-a* ovog projekta čini *Siemens-ov* IPC 277G(engl. *Industrial Personal Computer*). IPC-ovi su pouzdani, visoko performansni uređaji dizajnirani za primjenu u industrijskom okruženju. Njihova robusna konstrukcija, zajedno s naprednim značajkama, omogućuje im rad u teškim uvjetima, poput ekstremnih temperatura, vlage, prašine i vibracija. Ovi IPC-ovi često se koriste u automatizaciji proizvodnje, procesnom upravljanju, vođenju proizvodnih pogona i drugim industrijskim aplikacijama koje zahtijevaju pouzdan i kontinuiran rad.

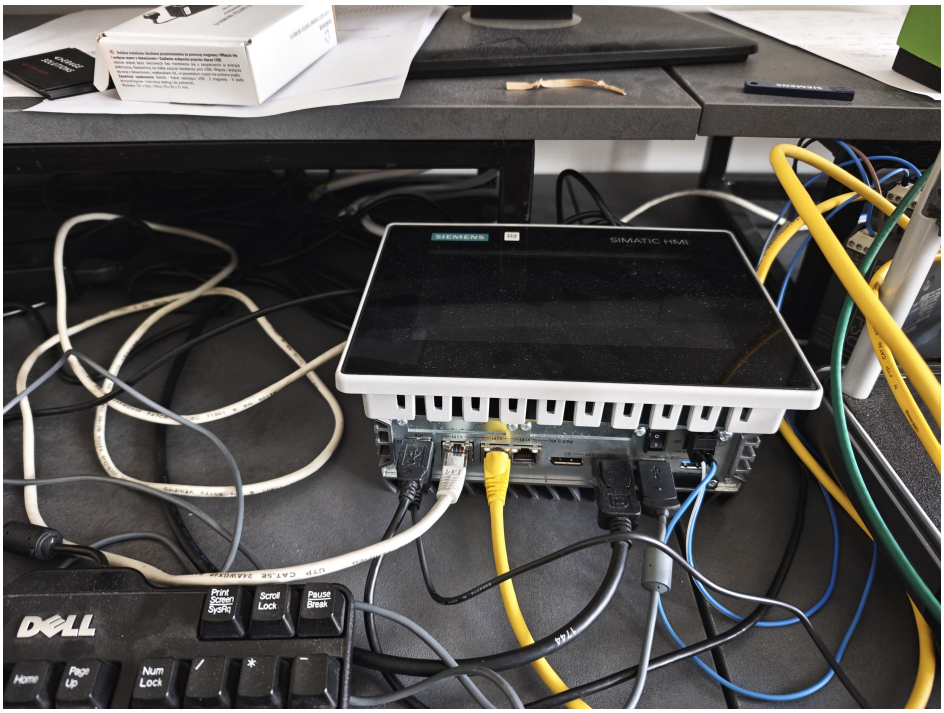
Siemensovi IPC-ovi dolaze u različitim formama, uključujući kompaktne PC-e, *rack-mount* PC-e i panel PC-e, kako bi zadovoljili različite potrebe industrijskih korisnika. Kompaktni PC-evi su dizajnirani za situacije gdje je prostor ograničen, ali je potrebna visoka računalna snaga. *Rack-mount* PC-evi prikladni su za ugradnju u standardne 19-inčne stalke, što omogućuje jednostavnu integraciju u postojeće infrastrukture. Panel PC-evi, s integriranim zaslonima osjetljivim na dodir, idealni su za interaktivne aplikacije i korisnička sučelja na proizvodnim linijama.

Ovi uređaji podržavaju širok spektar operativnih sustava, uključujući Windows i Linux, što omogućava fleksibilnost u izboru softverskih rješenja. Također postoje softverski alati i platforme za integraciju sa *Siemens* IPC-ovima, omogućujući korisnicima optimiziranje svojih procesa i povećanje učinkovitosti. Uz podršku za različite komunikacijske protokole i sučelja, Siemensovi IPC-ovi lako se povezuju s drugim uređajima i sustavima u industrijskom okruženju, čime omogućavaju jednostavnu integraciju u složene proizvodne mreže.

IPC 277G Panel PC jedan je od modela IPC-a u *Siemens-ovom* katalogu uređaja. Temelji se na IPC277G Box PC-u uz dodani zaslon za lakše rukovanje uređajem direktno u tvornici.

Specifikacije IPC-a navedene su u tablici 2.2.7:

Prema specifikacijama vidljivo je kako IPC ima nekoliko mogućnosti komunikacije putem *Ethrenet-a* na jednom od svoja tri ulaza. Čest je slučaj da se više uređaja povezuje na mrežni preklopnik(engl. *network switch*) koji posjeduje dodatne ulaze kojima se



Slika 2.2.7 IPC 277G Panel PC

Tablica 2.2.7 Specifikacije IPC277G [1]

Napajanje	24 VDC
Veličina zaslona	7"(17,78 cm)
Pohrana	256 GB (SSD)
Radna memorija	8 GB
Procesor	Atom X6414RE (4 jezgre)
Op. sustav	*Windows* 10 IoT *Enterprise*
Komunikacija	3x Gigabit *Ethernet* (IE/PN)

provodi komunikacija. Tako i ovom projektu, korišten je mrežni preklopnik s pet ulaza kako bi olakšao komunikaciju i ožičenje između uređaja.

Na *switch* su spojeni:

- IPC 277G,
- SM 1281 modul (preko njega priključen je i S7-1200),
- SENTRON PAC 3220,
- (za vrijeme programiranja) osobno računalo.

2.2.3. Softverski kontroler

Softverski kontroleri predstavljaju ključnu komponentu u modernim sustavima automatizacije. Oni omogućuju fleksibilnije, skalabilnije i često ekonomičnije rješenje u usporedbi s tradicionalnim hardverskim PLC uređajima. Softverski kontroleri su programi koji se izvršavaju na standardnim industrijskim računalima ili serverima, čime pružaju slične funkcionalnosti kao hardverski PLC, ali s dodatnim prednostima kao što su lakša nadogradnja, mogućnost integracije s drugim softverskim alatima, te veća računalna snaga i memorijski kapacitet.

Glavna prednost softverskih kontrolera je njihova sposobnost da se prilagode različitim industrijskim aplikacijama i brzo reprogramiraju prema potrebama specifičnog procesa. Softverski kontroleri također podržavaju razne komunikacijske protokole, omogućavajući integraciju s postojećim sustavima i uređajima. Osim toga, zbog svoje prirode kao softverska rješenja, mogu se implementirati na različitim hardverskim platformama, uključujući virtualne okoline, što dodatno povećava njihovu fleksibilnost i primjenjivost.

Jedan od glavnih modela Siemensovih softverskih kontrolera je SIMATIC S7-1500 *Software Controller*, koji omogućava visok stupanj fleksibilnosti i performansi. Ovaj softverski kontroler može se koristiti za kompleksne zadatke automatizacije, pružajući pouzdanu i brzu obradu podataka te mogućnost jednostavne integracije u postojeće sustave.

Specifičan model u ovoj seriji je SIMATIC S7-1507S. Ovaj model je posebno dizajniran za zahtjevne industrijske primjene gdje je potrebna visoka razina pouzdanosti i performansi. S7-1507S se izvodi na standardnim industrijskim računalima, što omogućava korištenje postojeće IT infrastrukture i smanjuje troškove.

SIMATIC S7-1507S podržava razne standarde i komunikacijske protokole, što omogućava lako povezivanje s drugim uređajima i sustavima unutar proizvodnog okruženja. Kontroler također nudi napredne funkcije za sigurnost i zaštitu podataka, što je ključna značajka u modernim industrijskim aplikacijama.

Ovaj softverski kontroler također omogućava jednostavnu integraciju s Siemensovim TIA Portalom (*Totally Integrated Automation*), što dodatno olakšava konfiguraciju, programiranje i održavanje. Ova integracija omogućava korisnicima da iskoriste sve

prednosti Siemensovih alata za automatizaciju u kombinaciji s fleksibilnošću softverskog kontrolera, čime se postiže optimalna učinkovitost i produktivnost u industrijskim procesima.[6]

Kontroler 1507s instaliran je na IPC 277G za potrebe ovog rada. Prilikom instalacije, računalo samo alocira jednu od 4 jezgara procesora koja će odrađivati zadatke vezane uz softverski kontroler. Također, jedan od 3 *Ethernet port-a* IPC-a se automatski zauzima te služi za komunikaciju sa softverskim kontrolerom, ali ne i s računalom na koje je instaliran. IPC 277G i 1507s softverski kontroler čine "mozak" cijelog ovog sustava i projekta. Podaci prikupljeni s SENTRON PAC-a o struji, naponu, snagama, prosječnim vrijednostima i sl. šalju se MODBUS TCP-om do softverskog kontrolera, a podaci prikupljeni sa senzora vibracija putem S7-1200 kontrolera i SM 1281 modula. Specifičan postupak razmjene podataka (tzv. *SEND - RECIEVE*) između 1507s i 1200 kontrolera opisan je u narednom poglavlju.

3. Programska rješenja

3.1. TIA Portal i konfiguracija PLC-a

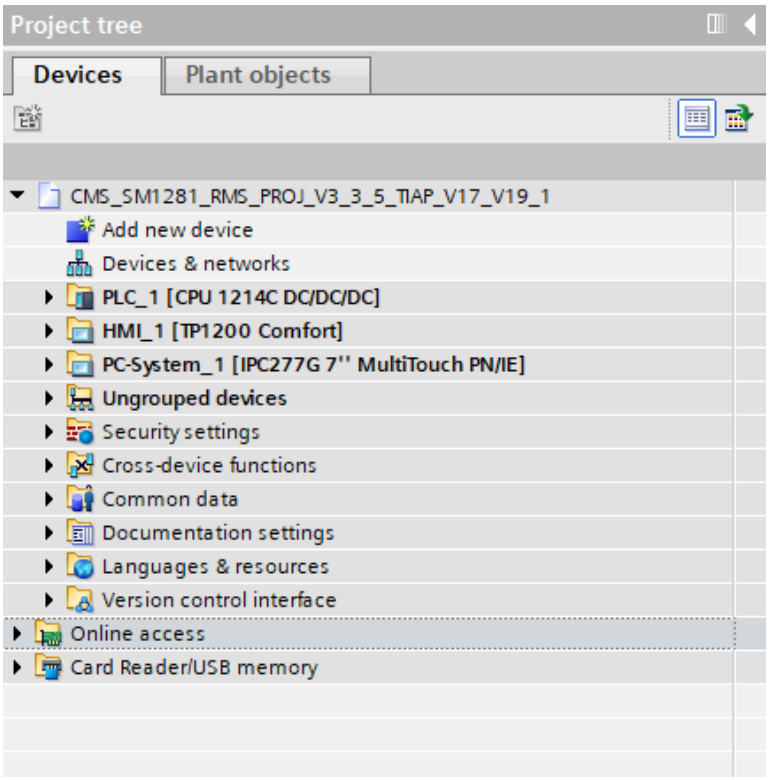
TIA Portal (Totally Integrated Automation Portal) je integrirana softverska platforma koju je razvio *Siemens* za upravljanje sustavima automatizacije. Ova platforma omogućuje inženjerima da dizajniraju, programiraju, konfiguriraju, dijagnosticiraju i održavaju sve elemente industrijskih automatizacijskih sustava unutar jedinstvenog okruženja. TIA Portal objedinjuje sve potrebne alate za projektiranje u jednu integriranu razvojnu sredinu, čime se znatno smanjuje vrijeme potrebno za razvoj i puštanje u rad automatizacijskih sustava.

Jedna od glavnih prednosti TIA Portala je njegova mogućnost da povezuje različite komponente automatizacije, uključujući PLC-ove, HMI (*Human Machine Interface*) uređaje, SCADA (*Supervisory Control and Data Acquisition*) sustave, te pogonske sustave. Ova integracija omogućava dosljedno programiranje i konfiguraciju svih komponenti iz jedne aplikacije, što pojednostavljuje rad i smanjuje mogućnost pogrešaka.

TIA Portal također nudi napredne dijagnostičke alate koji omogućuju brzo otkrivanje i rješavanje problema. To uključuje funkcije za *online* praćenje i simulaciju rada sustava, što pomaže u prepoznavanju i ispravljanju grešaka prije nego što dođe do stvarnih problema u proizvodnji. Ove mogućnosti znatno povećavaju učinkovitost i pouzdanost proizvodnih procesa.

Osnovna organizacijska jedinica u TIA Portalu jest projekt. Projekt sadrži najmanje jedan *Siemens-ov* uređaj kojemu se definira ponašanje pisanjem programskog koda. Kada se doda neki uređaj, on je vidljiv u projektnom "stablu" s lijeve strane sučelja TIA Portala (slika 3.1.1). Osim imena uređaja, programer ima pristup porgramiranju blokova u *"Program blocks"* prozoru, definiranje vlastitih tipova podataka(*"PLC data types"*), ta-

blice varijabli s mogućnošću praćenja vrijednosti varijabli u stvarnom vremenu(*"Tag table"*) i mnoge druge opcije.



Slika 3.1.1 Projektno stablo TIA Portala

Za razliku od "klasičnih" viših programskih jezika kao što su *C, C#, Java, Python* i dr., programiranje industrijske opreme izvodi se u posebnim jezicima od kojih su najčešći LAD(*Ladder Diagram*), FBD(*Function block diagram*) i SCL(*Structured control language*). Ovi jezici detaljno su definirani standardom *IEC 61131-3*. LAD i FBD grafički su jezici u kojima se programiranje svodi na slaganje slijednih blokova u veću cjelinu kako bi se dobio željeni algoritam. LAD jezik originalno je osmišljen kao jezik koji najviše podsjeća na relejnu logiku, preteču PLC programiranja, ona koristi "kontakte" kao gradivne jedinice kojima se postavljaju uvjeti izvođenja određene grane programa.

Jezik koji najviše nalikuje "klasičnim" programskim jezicima je SCL. Pogodan je u slučaju pisanja duljih matematičkih izraza/formula zbog svoje kompaktnosti i čitkosti. Također posjeduje mogućnosti korištenja petlji(FOR ili WHILE), grananja(IF) i SWITCH-CASE paradigme. Ove ključne riječi u programskom jeziku uvelike olakšavaju programiranje iterativnih algoritama koji izvode identičan kod ili dio koda nekoliko puta.

Organizacija programa za PLC u TIA Portalu ključna je za razvoj modularnog, čitlji-

vog i efikasnog koda. Svaki program sadrži: oragnizacijski blok(OB), funkcije(FC) i/ili funkcijske blokove(FB).

Organizacijski blokovi (OB)

Organizacijski blokovi predstavljaju osnovne blokove programa koji definiraju glavne operacije i sekvence rada PLC-a. Oni upravljaju izvršavanjem drugih programskih blokova i odgovoraju na različite događaje u sustavu. Najčešći OB blokovi su:

- **OB1 (Main Program Cycle)**: glavni ciklus programa koji se izvršava kontinuirano. Ovaj blok je srce PLC programa i sadrži glavnu logiku upravljanja (ekvivalent `Main` funkciji u drugim programskim jezicima),
- **Startup OB (npr. OB100)**: izvršava se prilikom pokretanja PLC-a. Koristi se za inicijalizaciju varijabli i sustava,
- **Time-Driven OB (npr. OB10, OB20)**: izvršavaju se u definiranim vremenskim intervalima. Koriste se za periodičke zadatke,
- **Interrupt OB (npr. OB40, OB80)**: aktiviraju se određenim događajima ili prekidima, kao što su alarmi ili greške.

Funkcije (FC)

Funkcije su blokovi koda koji izvršavaju specifične zadatke i mogu se pozivati iz drugih dijelova programa. One su slične funkcijama u tradicionalnim programskim jezicima i ne zadržavaju stanje između poziva.

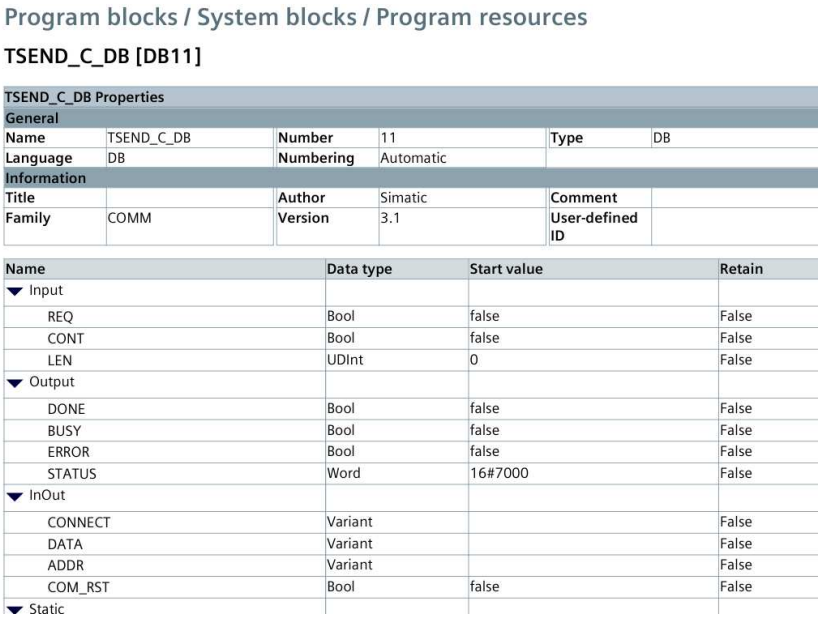
Funkcijski blokovi (FB)

Funkcijski blokovi su napredniji blokovi koda koji, za razliku od funkcija, zadržavaju stanje između poziva. Ovo je korisno za kreiranje složenijih kontrolnih struktura koje trebaju pratiti stanje kroz vrijeme. Primjeri uključuju:

- PID regulatori,
- brojači,

- tajmeri,
- komunikacijski blokovi i dr.

Kako bi FB mogao pamtiti stanja varijabli između poziva, TIA Portal koristi podatkovne blokove (DB, od *Data Block*). DB-ovi su posebno strukturirana grupa podataka koju može definirati programer, ali generiraju se i automatski prilikom svakog instanciranja funkcijskog bloka u nekoj drugoj programskoj jedinici (drugom FB-u ili OB-u).



Program blocks / System blocks / Program resources

TSEND_C_DB [DB11]

TSEND_C_DB Properties					
General					
Name	TSEND_C_DB	**Number**	11	**Type**	DB
Language	DB	**Numbering**	Automatic		
Information					
Title		**Author**	Simatic	**Comment**	
Family	COMM	**Version**	3.1	**User-defined ID**	

Name	Data type	Start value	Retain
Input			
REQ	Bool	false	False
CONT	Bool	false	False
LEN	UDInt	0	False
Output			
DONE	Bool	false	False
BUSY	Bool	false	False
ERROR	Bool	false	False
STATUS	Word	16#7000	False
InOut			
CONNECT	Variant		False
DATA	Variant		False
ADDR	Variant		False
COM_RST	Bool	false	False
Static			

Slika 3.1.2 Primjer instanciranog podatkovnog bloka

Prema slici 3.1.2, podatkovni blok sadrži sve varijable potrebne za predviđeni rad nekog funkcijskog bloka. U praksi, za vrijeme programiranja, podatkovni blok sadrži popis varijabli i njihovih pripadajućih tipova. Osim spremanja podataka, DB-ovi imaju mogućnost *online* praćenja trenutne vrijednosti neke varijable i postavljanje "bazne" vrijednosti.

3.1.1. S7-1200 kod

Projekt na S7-1200 PLC-u temelji se na već gotovom *Siemens-ovom* projektu za čitanje podataka dobivenih sa SM 1281 modula i senzora vibracija. Sastoji se od dva uređaja: S7-1214C upravljačke jedinice i TP1200 *Comfort* HMI panela. HMI (*Human - machine Interface*) je u ovom projektu simulacijske prirode i nije potrebno imati stvarni *Comfort* panel za rad programa. Ideja ovog baznog projekta je jednostavno simuliranje HMI-a kojim se uključuju svi senzori vibracija te se namještaju potrebni parametri. Nakon uklju-

čivanja senzora, sustav mjerenja vibracija je spreman za rad i pojavljuju se vrijednosti koje očitavaju senzori.

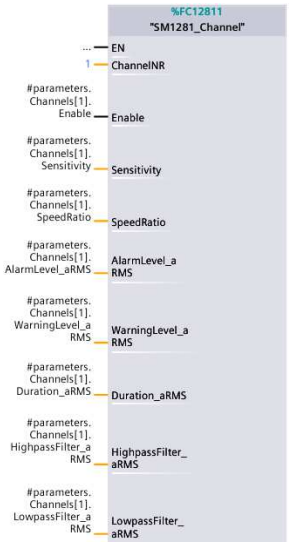
Osim HMI-a, projekt sadrži glavni funkcijski blok *CMS*(FB1) (slika 3.1.3). U FB1 obrađuju se signali dovedeni s 4 senzora vibracije odvojeni u četiri zasebna kanala. Na ulaze bloka dovedeni su podaci iz podatkovnog bloka "PlcHmi" koji povezuju namještanje parametara i prebacivanje između načina rada sustava mjerenja. Ulazi s nazivom "hwSubmodule" odnose se na konstantne vrijednosti koje predstavljaju fizičku lokaciju konekcije svakog od spojenih senzora kako bi program znao koji senzor asocirati s kojim rednim brojem od 1 do 4.

Slika 3.1.3 Poziv funkcijskog bloka *CMS*

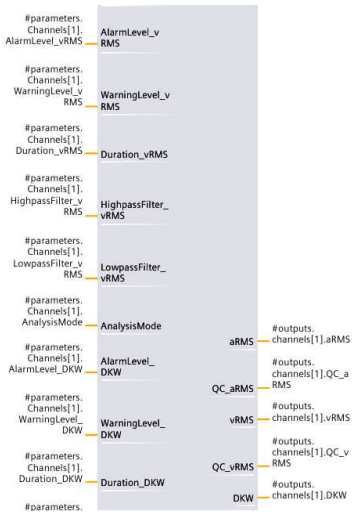
Unutar *CMS* bloka poziva se funkcijski blok *SM1281_Module*(FC12810) koji nadgleda brzinu vrtnje motora, međutim, u ovom radu nije korišten mjerni član brzine tako da ovaj blok nije od prevelike važnosti.

Svakom kanalu je pridružena jedna funkcija *SM1281_Channel*(FC12811) unutar *CMS* funkcijskog bloka. *SM1281_Channel* je zadužena za akviziciju i nadgledanje jednog od

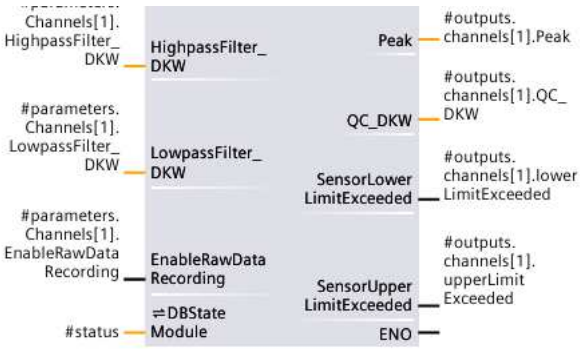
kanala/senzora. Tako se na izlazu ove funkcije mogu očitati vrijednosti brzine, akceleracije i DKW (engl. *diagnostic characteristic value*) od kojih je brzina vibracije od najvećeg interesa.[4]



(a) Prvi dio *Channel* funkcije



(b) Drugi dio *Channel* funkcije

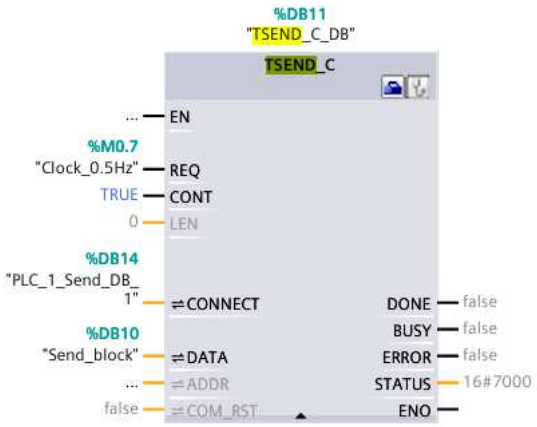


(c) Treći dio *Channel* funkcije

Slika 3.1.4 Poziv *Channel* funkcije u *CMS* bloku

Kako je ovo već gotov projekt od proizvođača, funkcijski blok *SM1281_Module* i funkcija *SM1281_Channel* nisu javno dostupne te njih nije moguće otvoriti u verziji TIA Portala V19 koja je korištena za izradu projekta. Uzimajući to u obzir, informacije navedene o ovim blokovima zaključene su iz testiranja programa, pretpostavkama i *Siemens-ovim* priručnicima.

Uz obradu podataka sa senzora, bitan dio programa je blok *TSEND_C*. Ovaj funkcijski blok pripada u skupinu sistemskih blokova korištenih za komunikaciju između uređaja *PROFINET* vezom. U ovome slučaju korišten je za slanje podataka o vibracijama prema softverskom 1507s kontroleru na IPC-u 277G. Blok se poziva odvojeno od *CMS* bloka u



Slika 3.1.5 *TSEND_C* blok

Main (OB1) organizacijskom bloku i izvršava se svake dvije sekunde. Interval od dvije sekunde postignut je dovođenjem memorijske lokacije `%M0.7` na `REQ` (*request*) ulaz bloka. Ovaj memorijski bit je sistemski i koristi se u situacijama u kojima je potrebno odraditi neku funkcionalnost u određenom intervalu, u ovom slučaju, 0.5 Hz (svake dvije sekunde).

Na ulaz `DATA` dovodi se podatkovni blok koji sadrži podatke za slanje na drugi uređaj (1507s PLC). Podatkovni blok "Send_block" i blok kojemu se šalju podaci moraju biti identične strukture kako bi se varijable ispravno prenijele i interpretirale. Također, oba podatkovna bloka u svojim postavkama moraju imati isključenu opciju "*optimized block*".

3.1.2. 1507s kod

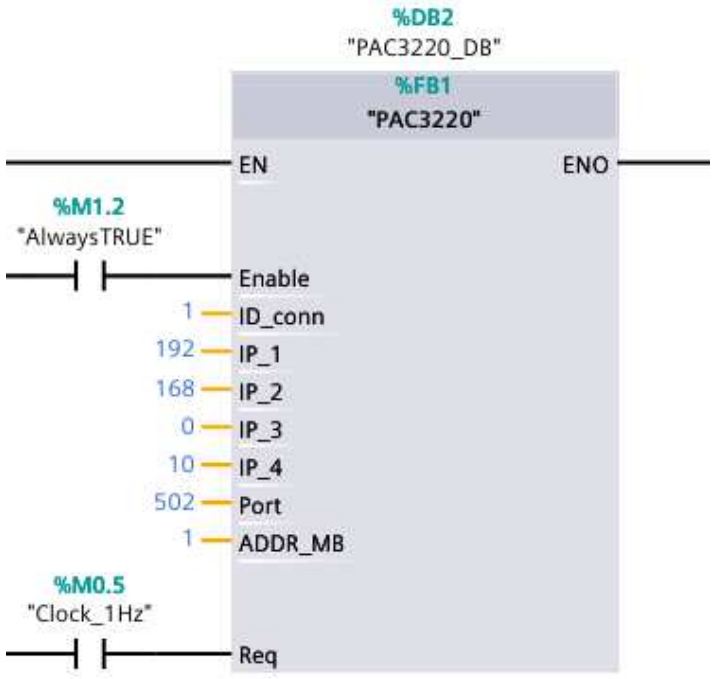
Na softverskom 1507s PLC-u nalazi se program s nekoliko jednostavnih blokova od kojih su bitni *PAC3220* i *TRCV_C* blokovi.

PAC3220 blok diktira komunikaciju sa SENTRON PAC-om koji prikuplja podatke o električnoj energiji. Unutar bloka nalaze se dva poziva `MB_CLIENT` sistemske funkcije. Ova sistemska funkcija dio je standardne biblioteke TIA Portala i koristi se za *MODBUS* komunikaciju s perifernim uređajima. Funkcija prima argument `CONNECT` kojemu se dovodi varijabla `#Connect`. Varijabla je tipa TCON_IP_v4 što je zapravo struktura podataka koja sadrži potrebne informacije o IP adresi i *portu* perifernog uređaja na kojeg se PLC povezuje. Također je povezan i podatkovni blok `#PAC3220_Data` u kojem se nalaze svi podaci dobiveni sa SENTRON PAC-a.

```
0001 REGION _name_
0002   #Connect.ID := #ID_conn;
0003   #Connect.RemoteAddress.ADDR[1] := INT_TO_BYTE(#IP_1);
0004   #Connect.RemoteAddress.ADDR[2] := INT_TO_BYTE(#IP_2);
0005   #Connect.RemoteAddress.ADDR[3] := INT_TO_BYTE(#IP_3);
0006   #Connect.RemoteAddress.ADDR[4] := INT_TO_BYTE(#IP_4);
0007   #Connect.RemotePort := #Port;
0008   #MB_CLIENT.MB_Unit_ID := INT_TO_BYTE(#ADDR_MB);
0009 END_REGION
0010
0011
0012 REGION _name_
0013   IF #Enable = 0 THEN
0014     #MB_CLIENT(REQ := 0,
0015            DISCONNECT := 0,
0016            MB_MODE := 0,
0017            MB_DATA_ADDR := 40002,
0018            MB_DATA_LEN := 70,
0019            MB_DATA_PTR := #PAC3220_Data,
0020            CONNECT := #Connect);
0021   END_IF;
0022
0023   IF #Enable = 1 THEN
```

Slika 3.1.6 Poziv `MB_CLIENT` funkcije

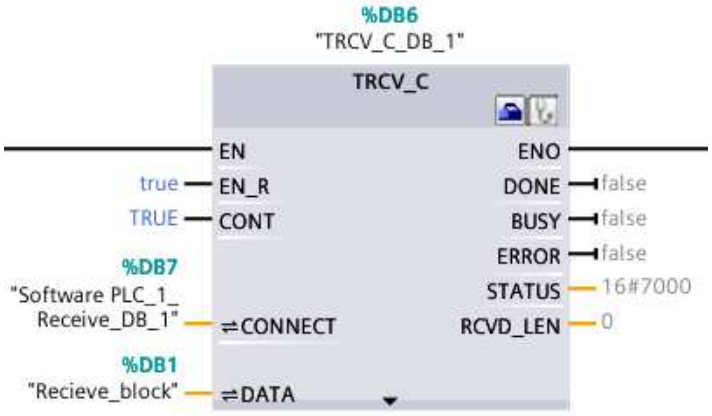
PAC3220 blok nalazi se u *Main* organizacijskom bloku, a na `Req` ulaz doveden mu je sistemski bit od 1 Hz (izvodi se svake sekunde).



Slika 3.1.7 Poziv funkcijskog bloka *PAC3220* u *Main*(OB1)

TRCV_C blok analogan je *TSEND_C* bloku iz prošlog poglavlja, a ovdje mu je funkcija uspješno primanje podataka sa S7-1200 PLC-a o vibracijama. Poziva se u glavnom organizacijskom bloku i nema vrijeme uzorkovanja kao *send* dio komunikacije. Sadrži

"Recieve_block" podatkovni blok koji je identičan po strukturi "Send_block" bloku.



Slika 3.1.8 Poziv *TRCV_C* bloka

3.1.3. OPC UA komunikacija

Nakon što se svi potrebni podaci mjerenja prikupe na 1507s PLC, potrebno ih je zapisati u neku bazu podataka. Jedan od načina komunikacije s aplikacijama treće strane je konfiguriranje OPC UA servera na PLC-u. OPC UA (*Open Platform Communications Unified Architecturen*) je komunikacijski protokol koji se često koristi u industrijskoj automatizaciji.

Razvijen od strane *OPC Foundation-a*, pruža sveobuhvatan okvir za razmjenu podataka i interoperabilnost između različitih sustava i platformi. OPC UA koristi model klijent-server za olakšavanje komunikacije između uređaja i sustava. OPC UA server prikuplja i obrađuje podatke iz raznih izvora, a ti podaci su dostupni OPC UA klijentima, koji ih koriste za nadzor, kontrolu i analizu. Arhitektura uključuje OPC UA čvorove koji predstavljaju pojedinačne entitete poput senzora, aktuatora ili softverskih aplikacija. Ovi čvorovi enkapsuliraju podatke i odnose, omogućujući nesmetanu interakciju unutar industrijskog ekosustava.[7]

U konfiguraciji PLC-a omogućena je opcija OPC UA servera. Pristupna točka PLC-a je jedna od pripadajućih IP adresa konfiguriranih u TIA projektu (`192.168.0.2` ili `192.168.73.1`). Klijent tada ima mogućnost jednostavnim zahtjevom zatražiti od servera npr. stanje neke varijable, a za autorizaciju mu je potrebna IP adresa servera i identifikacija čvora (*nodeID*).

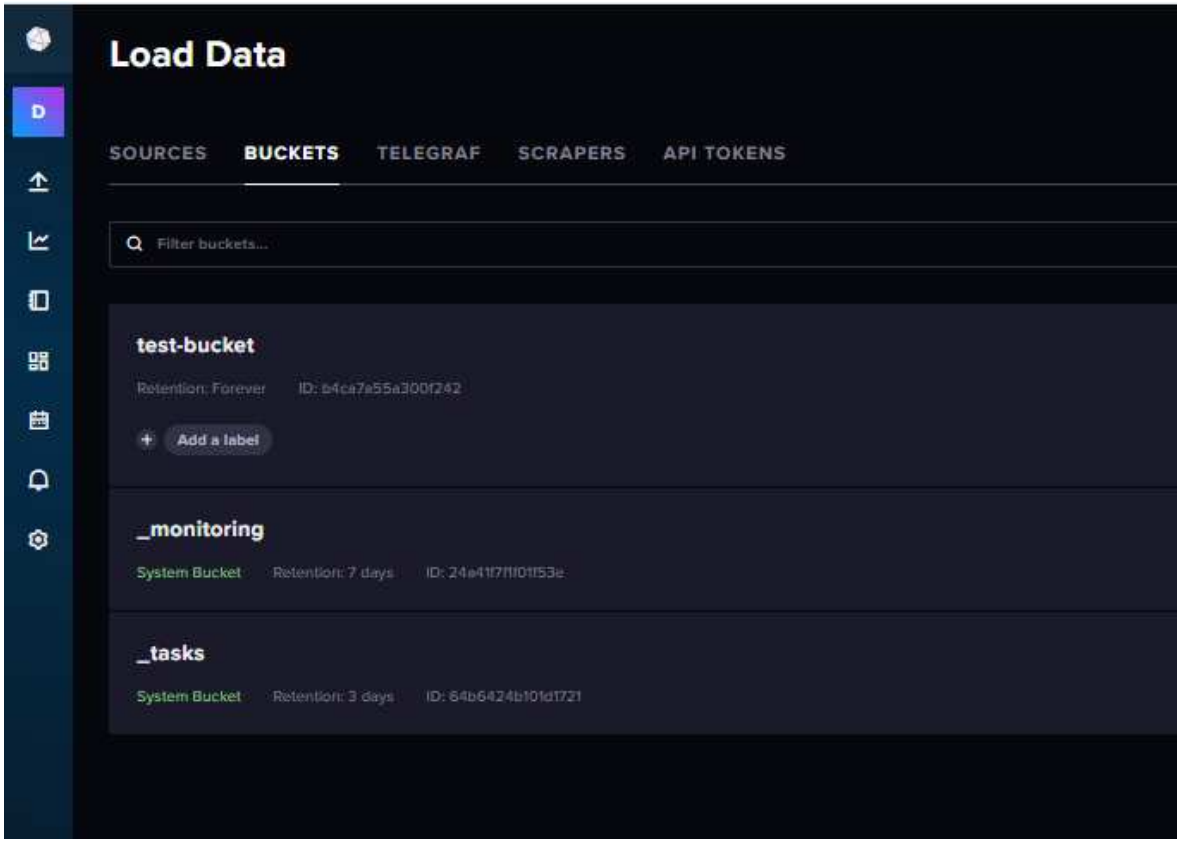
3.2. Baza podataka

3.2.1. O bazama podataka

U današnjem tehnološkom okruženju neizostavan dio svakog mjernog sustava je dugoročno i lako pristupačno pohranjivanje podataka. Rješenje koje se koristi u tim situacijama su baze podataka. Baze podataka u širem smislu su bilo koja struktura velike količine podataka označenih posebnim identifikacijskim ključevima prema kojima je lako dohvatiti, analizirati, grupirati ili sortirati podatke.

U ovom projektu odabrana je *InfluxDB* baza podataka koju karakterizira jednostavno manevriranje s vremenskim trenucima zapisa nekog podatka i integracija s drugim programskim rješenjima kao što su .NET aplikacije i *Grafana* vizualizacija.

Korištenje *InfluxDB* baze podataka svodi se na konfiguraciju lokalnog mrežnog poslužitelja (*web servera*) koji se stvara pri instalaciji *Influx-a*. Na slici 3.2.9 prikazano je početno korisničko sučelje *InfluxDB* poslužitelja u kojem se definiraju tzv. *bucketi*.



Slika 3.2.9 Korisničko sučelje mrežnog poslužitelja *InfluxDB-a*

3.2.2. Struktura, hijerarhija podataka i upiti u *InfluxDB-u*

Svaka baza podataka karakterizira se određenom hijerarhijskom strukturom koja olakšava dohvaćanje određenog podatka potrebnog za obradu u bilo kojem trenutku. Tako je u *InfluxDB-u* osnovna podjela podataka na *buckete* koji označavaju veliku skupinu podataka koja sadrži sve informacije o nekom većem dijelu mjernog područja (npr. lokacija jednog pogona).

Bucket sadržava proizvoljno ime kao što je *test-bucket* na slici 3.2.9 . Osim imena, svaki *bucket* pri kreiranju generira jedinstveni *API Token (Application Programming Interface)* koji se koristi za identifikaciju tog skupa podataka. *API Token* i *API* u općenitom smislu je naziv za bilo kakvu vrstu sučelja kojem je svrha komunikacija jednog sustava s vanjskim programom/aplikacijom/mrežnom stranicom i sl. Ovdje je *API Token* niz nasumično generiranih znakova potreban kako bi korisničke aplikacije opisane u poglavlju 3.3. mogle pristupiti i čitati/pisati nove podatke u bazu podataka.

Uz *bucket*, svi podaci pohranjeni u *InfluxDB* bazi mogu imati sljedeće odrednice:

- mjerenje (*measurement*),
- oznaka (*tag*),
- polje (*field*).[8]

Mjerenje označava veću skupinu mjerenih vrijednosti koja se odnosi na određeni npr. stroj/elektroormar/dio postrojenja te se imenuje tekstualnim podatkom tipa *String*. Oznaka ima oblik ključ - vrijednosti pri kojoj je ključ odrednica koja ima nekoliko varijacija, npr. ako postoji više identičnih senzora označenih brojevima, ključ bi bio naziv senzora, a vrijednost broj određenog senzora.

Zaključno, polje, slično kao i oznaka, sadrži dva podatka; prvi podatak je vrsta mjerenja (u slučaju spomenutih senzora, to može biti npr. temperatura), a drugi podatak je numerička vrijednost koju bilježi senzor za neko mjerenje.

Kako bi se pristupilo potrebnim podacima u bilo kojoj bazi podataka, potrebno je zatražiti podatak slanjem **upita** (engl. *query*). Upiti su posebno strukturirani nizovi znakova koje baza podataka interpretira i prepoznaje koji podaci su zatraženi.

U *InfluxDB-u* upiti imaju sintaksu kao u primjeru:

```
from(bucket: "example-bucket")
|> range(start: -1h)
|> filter(fn: (r) => r._measurement == "example-measurement"
and r.tag == "example-tag")
|> filter(fn: (r) => r._field == "example-field")
```

U naredbi `from()` navodi se ime *bucketa* iz kojeg se čitaju podaci, naredbom `range()` određuje se vremenski raspon podataka koji se uzimaju u obzir, raspon se može zadati točnim vremenskim trenutkom u UTC formatu ili kao u primjeru vremenskim odmakom od trenutnog vremena. Naredbom `filter()` definira se mjerenje/oznaka/polje od interesa traženog upita.

Budući da su upiti u *Influx-u stringovi*, jednostavno je manipulirati formatom upita u nekom višem programskom jeziku kao što su *C+, C#, Java* i dr. i na taj način ostvariti modularni upit koji se može mijenjati s obzirom na potrebe aplikacije/korisnika.

3.3. *Visual Studio Code*, *C#* i korisničke aplikacije

U ovom poglavlju opisane su dvije korisničke aplikacije koje služe čitanju/pisanju podataka u bazu podataka i komunikaciju sa softverskim PLC-om na IPC-u 277G. Obje aplikacije pisane su *C#* programskim jezikom (.NET platforma) i korišten je *Visual Studio Code* kao okruženje za programiranje. *Visual Studio Code (VSC)* jedan je od okruženja za programiranje često korišten za pisanje aplikacija koje koriste .NET platformu. Platforma .NET koristi se za izradu *Desktop* aplikacija na *Windows* operacijskom sustavu te sadrži veliki broj unaprijed pripremljenih baznih programa integriranih u programsko okruženje koji su spremni za nadogradnju i uvelike ubrzavaju izradu korisničkih aplikacija.

3.3.1. Čitanje OPC UA servera i upisivanje u *InfluxDB* bazu podataka

Prva od dvije korisničke aplikacije provjerava varijable PLC-a koristeći pristup OPC UA serveru te ih prenosi u *Influx* bazu i dodjeljuje im pripadajuće mjerenje/oznaku i polje.

Ova aplikacija je tzv. konzolna aplikacija te nema korisničko sučelje osim unosa osnovnih podataka o bazi podataka u konzoli (*Command Line*).

Program se sastoji od glavne klase `Program` koja sadrži nekoliko metoda; glavna metoda `Main()` sadrži funkcionalnost aplikacije, metoda `Console_CancelKeyPress()` obrađuje zatvaranje aplikacije, a `WriteInfluxDB()` opisuje način na koji će se upisivati podaci u bazu podataka.

```
class Program
{
//Loop exit condition
static bool exit = false;
static void Main(string[] args)
{
    string url = "http://localhost:8086";
    string token;
    string org;
    string bucket;

    //Input of InfluxDB data
    //Console.WriteLine("Enter InfluxDB server URL:");
    //url = Console.ReadLine();

    //Console.WriteLine("Enter InfluxDB token:");
    token = "_zDUuVU4k_lzGqT6LrrbYri5PdElFJpSfU0DDMAMxlfxQ_gsbDFf
    L9MMRtNHNp3AAU6n0SGWBdV9qWHsd_0Sww==";//Console.ReadLine();

    //Console.WriteLine("Enter InfluxDB organisation:");
    org = "DATA"; //Console.ReadLine();

    //Console.WriteLine("Enter InfluxDB bucket name:");
    bucket = "test-bucket";//Console.ReadLine();

    // OPC UA server settings
    string endpointUrl = "opc.tcp://192.168.73.1:4840";
```

Na početku metode definirane su potrebne varijable za identifikaciju mrežnog poslužitelja na kojem je baza podataka. U realnom korištenju ovakve aplikacije, identifikacijske komponente baze podataka se unose jednom te se ne bi trebale mijenjati osim ako se

ne definira potpuno nova struktura baze (npr. otvara se novi *bucket*). Za potrebe učestalog testiranja aplikacije postavljeni su korisnički unosi identifikacijskih parametara baze (linije 13 - 24, metoda `Console.ReadLine()`) kako bi se olakšalo prilagođavanje novim iteracijama baze prilikom testiranja, ali oni nisu potrebni kada je sustav spreman za dugoročni rad. Varijabla `endpointUrl` definira IP adresu OPC UA servera na PLC-u koja je potrebna za povezivanje aplikacije i uspostavljanje OPC klijenta.

Osim potrebnih varijabli za komunikaciju, definira se lista *stringova* u koju se unose *nodeID-evi* svake varijable koju treba pročitati na PLC-u. Struktura ID-a sadrži naziv podatkovnog bloka (DB), naziv strukture (VIB) i naziv točne varijable koja se promatra.

```
1    // List of nodeIds representing the variables to read
2        var nodeIds = new List<string>
3        {
4
5            "ns=3;s=\"Recieve_block\".\"VIB1\".\"V_RMS\"",
6            "ns=3;s=\"Recieve_block\".\"VIB1\".\"A_RMS\"",
7            "ns=3;s=\"Recieve_block\".\"VIB1\".\"DKW\"",
8
9            "ns=3;s=\"Recieve_block\".\"VIB2\".\"V_RMS\"",
10           "ns=3;s=\"Recieve_block\".\"VIB2\".\"A_RMS\"",
11           "ns=3;s=\"Recieve_block\".\"VIB2\".\"DKW\"",
12
13           "ns=3;s=\"Recieve_block\".\"VIB3\".\"V_RMS\"",
14           "ns=3;s=\"Recieve_block\".\"VIB3\".\"A_RMS\"",
15           "ns=3;s=\"Recieve_block\".\"VIB3\".\"DKW\"",
16           .
17           .
18           .
19           "ns=3;s=\"PAC3220_DB\".\"PAC3220_Data\".\"Total reactive
   power\"",
20           "ns=3;s=\"PAC3220_DB\".\"PAC3220_Data\".\"Total power
   factor\""
21
22       };
```

U gornjem dijelu koda nisu navedeni svi ID-evi te su podložni promjenama s obzirom na svrhu aplikacije, u ovom slučaju su navedeni samo neki kao primjer. Svrha ovakve

liste je mogućnost korištenje `foreach` petlje kako bi se pregledao svaki ID u pozivu programa. Zatim je potrebno uspostaviti vezu izmedju aplikacije i OPC UA servera i to na sljedeći način:

```
    var client = new EasyUAClient();
```

Konstruktor `EasyUAClient()` definira novu instancu te klase i pridružuje je varijabli `client`. Zatim se započinje `foreach` petlja ugnježđena u beskonačnu `while` petlju kojoj je jedini uvjet prekida varijabla `exit`. Varijabla `exit` kontrolirana je od korisnika i mijenja vrijednost pritiskom kombinacije *Ctrl + C* tipki na računalu što rezultira prekidom i isključenjem cijelog programa.

```
    while (!exit)
        {
            foreach (var nodeId in nodeIds)
            {
                // Read value from OPC UA server
                var opcValue = EasyUAClient.SharedInstance.ReadValue(
                $"{endpointUrl}",
                $"{nodeId}");
                decimal valueDec = Convert.ToDecimal(opcValue);
                Console.WriteLine(opcValue);
```

Varijabla `opcValue` poprima vrijednost koja se nalazi na IP-u `endpointUrl` i pod IDem `nodeId` te je odmah pretvorena u decimalnu vrijednost. Naredba `WriteLine()` ispisuje pročitanu vrijednost na prozor konzole za potrebe testiranja programa i nadziranja pravilnog čitanja vrijednosti.

Nakon što se vrijednost spremi u varijablu, potrebno je upisati ju u *bucket Influx-a*. Ovdje se koristi korisnička metoda `WriteInfluxDB()`, metoda kao parametre prima oznaku, broj oznake, naziv polja, vrijednost te potrebne informacije o *bucket-u* koji se koristi.

```
    private static void WriteInfluxDB(string tag, string tagNumber,
    string field, decimal value, string url, string token, string
    bucket, string org)
    {
        using (var influxDbClient = new InfluxDBClient(url, token))
        {
```

```
            // Define data point
            var point = PointData.Measurement("Measurements")
                                 .Tag(tag, tagNumber)
                                 .Field($"{field}", value)
                                 .Timestamp(DateTime.UtcNow,
    WritePrecision.S);
            // Write data point to database
            using (var writeApi = influxDbClient.GetWriteApi())
            {
                writeApi.WritePoint(point, bucket, org);
            }
        }
    }
```

Metoda inicijalizira `InfluxDBClient` varijablu kojom se uspostavlja komunikacija između baze podataka i aplikacije, osim toga, definira se i varijabla `point` u koju se upisuju sve odrednice nekog podatka i njegova numerička vrijednost. Budući da prilikom testiranja nije bilo mogućnosti za veliki broj različitih mjerenja na drugim lokacijama, svi podaci koji se upisuju u bazu imaju oznaku mjerenja `"Measurements"`, a oznaka, polje i vrijednost ovise o parametrima s kojima je metoda pozvana.

Svaki podatak također sadrži vremenski trenutak u kojem je zapisan, a to je ostvareno metodom `Timestamp()` koja kao parametar prima sistemsku konstantu `UtcNow` u UTC formatu koja predstavlja sadašnji trenutak s preciznošću u sekundama.

Nastavak programa poziva metodu `WriteInfluxDB()` za svaki slučaj različitog ID-a, logika se sastoji od `if` grananja koji provjeravaju uvjete metodom `Contains()`. Provjera osigurava da se podatak upisan u bazu unese pod točnom oznakom i nazivom polja.

U sljedećem primjeru program upisuje brzinu vibracije `V_RMS`(polje) pod oznakom `VIBX` gdje je `X` broj senzora vibracije od 1 do 4.

```
    Console.WriteLine(opcValue);
                try
                {
                    for (int i = 1; i < 5; i++)
                    {
                        if (nodeId.Contains($"VIB{i}"))
```

```
                        {
                            if (nodeId.Contains("V_RMS"))
                            {
                                WriteInfluxDB("VIB", i.ToString(), "
V_RMS", valueDec, url, token, bucket, org);
                                Console.WriteLine($"Value {opcValue}
from node {nodeId} written to InfluxDB successfully.");
```

Program radi kontinuirano te uzorkuje sve varijable navedene u listi ID-eva svakih 5 sekundi, vrijednost koja je lako promjenjiva s obzirom na potrebe nekog mjernog sustava.

```
System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(5000));
```

Iz sistemske biblioteke `Threading` koristi se metoda `Sleep` kojom se odgađa novo izvođenje beskonačne `while` petlje za onoliko milisekundi koliko je navedeno u argumentu metode.

Na ovaj način aplikacija prepisuje podatke koje očitavaju senzori vibracija i SENTRON PAC te ih prenosi u *InfluxDB* bazu podataka.

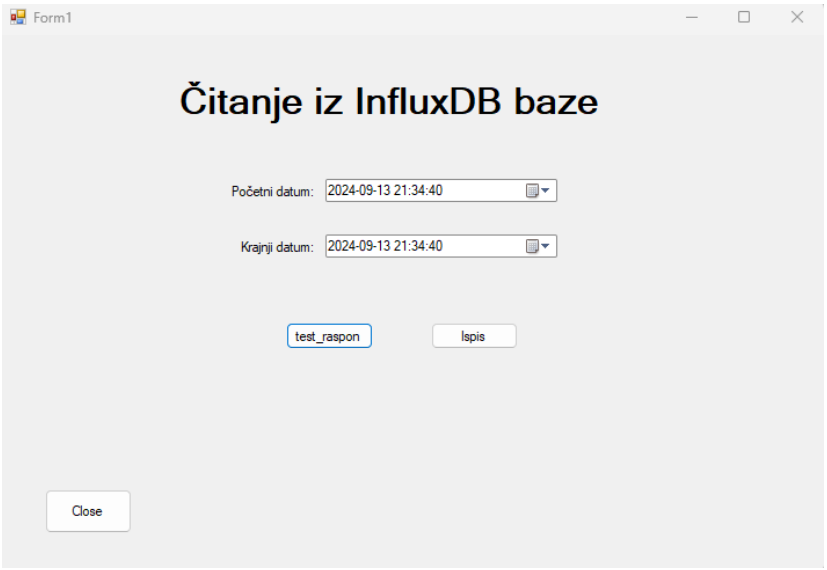
3.3.2. *Windows Form* i čitanje iz *InfluxDB* baze podataka

Čest je slučaj da podatke pohranjene u neku bazu podataka treba ispisati radi daljnje analize/pregleda/izvješća i dr. Za tu potrebu napravljena je aplikacija za čitanje podataka iz *InfluxDB* baze i to korištenjem *Windows Forme*. *Windows Form* je vrsta grafičkog korisničkog sučelja građena na .NET platformi koju koristi većina *Windows* aplikacija za vizualizaciju informacija prema korisniku.

U *VSC-u*, dizajniranje forme je jednostavan postupak *drag-and-dropom* potrebnih komponenata (tipke, polja za unos itd.), nakon dodavanja komponente, *VSC* samostalno dodaje definiciju komponente te programer ne mora brinuti o znanju sintakse dizajniranja forme.

Ova aplikacija korisniku pruža mogućnost zadavanja vremenskog intervala (datum i vrijeme početnog i krajnjeg trenutka) unutar kojega ga zanimaju vrijednosti upisane u *InfluxDB* bazu podataka. Nakon odabira intervala, pritiskom na tipku `Ispis` stvara se `.csv` datoteka u kojoj su pohranjene vrijednosti, vremenski trenuci i sve odrednice

svakog podatka koji se nalazi u željenom vremenskom intervalu. (Slika 3.3.10)



Slika 3.3.10 *Windows forma*

Svaka komponenta na vizualizaciji forme ima svoju programsku definiciju, npr:

```
 // IspisButton
            //
            this.IspisButton.Location = new System.Drawing.Point(372,
248);
            this.IspisButton.Name = "IspisButton";
            this.IspisButton.Size = new System.Drawing.Size(75, 23);
            this.IspisButton.TabIndex = 2;
            this.IspisButton.Text = "Ispis";
            this.IspisButton.UseVisualStyleBackColor = true;
            this.IspisButton.Click += new System.EventHandler(this.
button2_Click);
```

Tipka `Ispis` ima definiranu lokaciju na formi, ime, veličinu i druge atribute, a uz to postoji definiran događaj kada se pritisne tipka. Ovaj dio koda automatski je generiran od *VSC-a* i ažurira se svaki puta kada programer napravi promjenu u dizajnu forme (npr. ako se pormijeni lokacija tipke, `Location` atribut automatski ažurira nove numeričke koordinate). Definiranje izgleda forme nalazi se u datoteci `Form1.Designer.cs`, a algoritam aplikacije u `Form1.cs`.

Osim tipki i tekstova(`label`), na formi su dva polja za unos datuma koji se u .NET platformi nazivaju `DateTimePicker`. Oba polja za datum imaju, uz vizualizacijske atri-

bute, definiran događaj `ValueChanged` koji prati promjenu vrijednosti unesene u polje.

Kao i kod aplikacije za unos podataka u bazu, aplikacija za čitanje vrijednosti mora uspostaviti vezu s bazom podataka tako da se prvo definiraju varijable potrebne za pristup *bucket-u*:

```
    public partial class Form1 : Form
    {
        private readonly string url = "http://localhost:8086";
        private readonly string token = "
_zDUuVU4k_lzGqT6LrrbYri5PdElFJpSfU0DDMAMxlf
        xQ_gsbDFfL9MMRtNHNp3AAU6n0SGWBdV9qWHsd_0Sww==";
        private readonly string org = "DATA";
        private readonly string bucket = "test-bucket";
```

Potrebne su i početne i krajnje vrijednosti vremenskog intervala pohranjene u varijablama:

```
    private DateTime queryStartTime = DateTime.UtcNow;
    private DateTime queryEndTime = DateTime.UtcNow;
```

Varijable su incijalizirane na trenutno vrijeme, ali na svaku promjenu jednog od `DateTimePicker-a`, vrijednosti se ažuriraju:

```
    private void dateTimePicker1_ValueChanged(object sender, EventArgs
    e)
        {
            queryStartTime = startDatePicker.Value;
        }

        private void dateTimePicker2_ValueChanged(object sender,
    EventArgs e)
        {
            queryEndTime = endDatePicker.Value;
        }
```

Najveći dio algoritma odvija se na pritisak tipke `Ispis`, definira se opći oblik upita za bazu podataka:

```
    private async void button2_Click(object sender, EventArgs e)
        {
```

```
            try
            {
                // Create InfluxDB client
                using (var influxDbClient = new InfluxDBClient(url,
token))
                {
                    // Define query
                    var query = "from(bucket: \"" + bucket + "\") "
                    + $"|> range(start:
                    {queryStartTime:yyyy-MM-dd'T'HH:mm:ss'Z'},
                    stop:
                    {queryEndTime:yyyy-MM-dd'T'HH:mm:ss'Z'})"
                    + "|> filter(fn: (r) => r[\"_measurement\"] ==
                    \"Measurements\")";
```

Ovakav upit sadrži varijabilni početak i kraj vremenskog intervala koji se definira u `DateTimePicker-u`, a jedini filter koji je potrebno koristiti kako bi se dohvatili svi podaci iz baze podataka je onaj za mjerenje, stoga se provjerava `"Measurements"`. Ovdje je korištena posebna sintaksa u kojoj string započinje znakom $ što nalaže prevoditelju programa da sve riječi u stringu zamjeni s definiranim varijablama (`queryStartTime` i `queryEndTime`).

Također je potrebno odrediti naziv i lokaciju `.csv` datoteke koja će se generirati izvođenjem programa:

```
 string desktopPath = Environment.GetFolderPath(Environment.
SpecialFolder.Desktop);
 string csvFileName = "testna_tablica.csv";
 string csvFilePath = Path.Combine(desktopPath, csvFileName);
```

Potom se izvršava upit i svi podaci se spremaju u višedimenzionalno polje `fluxTables`:

```
 var fluxTables = await influxDbClient.GetQueryApi().QueryAsync(
query, org);
 using (var writer = new StreamWriter(csvFilePath, append: false))
 using (var csv = new CsvWriter(writer, System.Globalization.
CultureInfo.InvariantCulture))
```

Varijabla `csv` sadrži metode koje će se koristiti za upis vrijednosti u `.csv` datoteku.

Prvo se upisuju nazivi stupaca tako što se uzima zapis na poziciji `0` i `foreach` petljom se upisuje `label` svakog stupca u `.csv` datoteku metodom `WriteField()`, a metodom `NextRecord()` se prelazi u novi red i može započeti unos podataka:

```
    fluxTables.ElementAt(0).Columns.ForEach(fluxcolumn =>
    {
        csv.WriteField(fluxcolumn.Label);

    });
    csv.NextRecord();
```

Novom `foreach` petljom program prolazi kroz tablicu podataka i za svaki zapis (`fluxRecord`) unosi njegovu numeričku vrijednost.

```
    fluxTables.ForEach(fluxTable =>
        {
            var fluxColumns = fluxTable.Columns;
            var fluxRecords = fluxTable.Records;
            csv.NextRecord();
            fluxRecords.ForEach(fluxRecord =>
            {
                for (int i = 0; i < fluxColumns.Count; i++)
                    {
                        csv.WriteField($"{fluxRecord
                        .GetValueByIndex(i)}");
                    }
                    csv.NextRecord();
            });
        });
```

Nakon izvršavanja programa dobiva se izgled datoteke kao na slici 3.3.11 Konačna datoteka sadrži potrebne informacije o svakom podatku iz baze, u stupcu _value nalazi se numerička vrijednost, stupac _time pokazuje točni vremenski trenutak kada se podatak zabilježio u UTC formatu, prikazana je i oznaka (PAC3220) i vrijednost oznake (Average values), naziv polja (Average current) i naziv mjerenja.

	A	B	C	D	E	F	G	H	I	J	K	L
1	result	table	_start	_stop	_time	_value	PAC3220	_field	_measurement			
2												
3	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015553	Average va	Average cu	Measurements			
4	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015588	Average va	Average cu	Measurements			
5	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015481	Average va	Average cu	Measurements			
6	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015404	Average va	Average cu	Measurements			
7	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015472	Average va	Average cu	Measurements			
8	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015661	Average va	Average cu	Measurements			
9	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015412	Average va	Average cu	Measurements			
10	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.01534	Average va	Average cu	Measurements			
11	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.0152	Average va	Average cu	Measurements			
12	_result	0	2024-06-1(	2024-06-1:	2024-06-1:	0.015422	Average va	Average cu	Measurements			

Slika 3.3.11 Primjer `.csv` datoteke

Na ovaj način zapisani podaci iz baze podataka prenose se u vanjsku datoteku koja je pogodna za dodatnu obradu, prikazivanje u nekom grafičkom programu ili arhiviranje i generiranje izvještaja.

3.4. Prikaz podataka

Kako bi se podaci dobiveni mjerenjem prikazali krajnjem korisniku, u ovom radu opisane su dvije metode: izvješće u *Excelu-u* i vizualzacija *Grafana* programom. Generiranje izvješća je korisno rješenje kada postoji potreba za arhiviranjem podataka, analizom potrošnje energije na npr. godišnjoj bazi i sl. dok je vizualizacija korisna za estetski prihvatljiv prikaz trenutnih vrijednosti mjerenih podataka za nadzor u stvarnom vremenu.

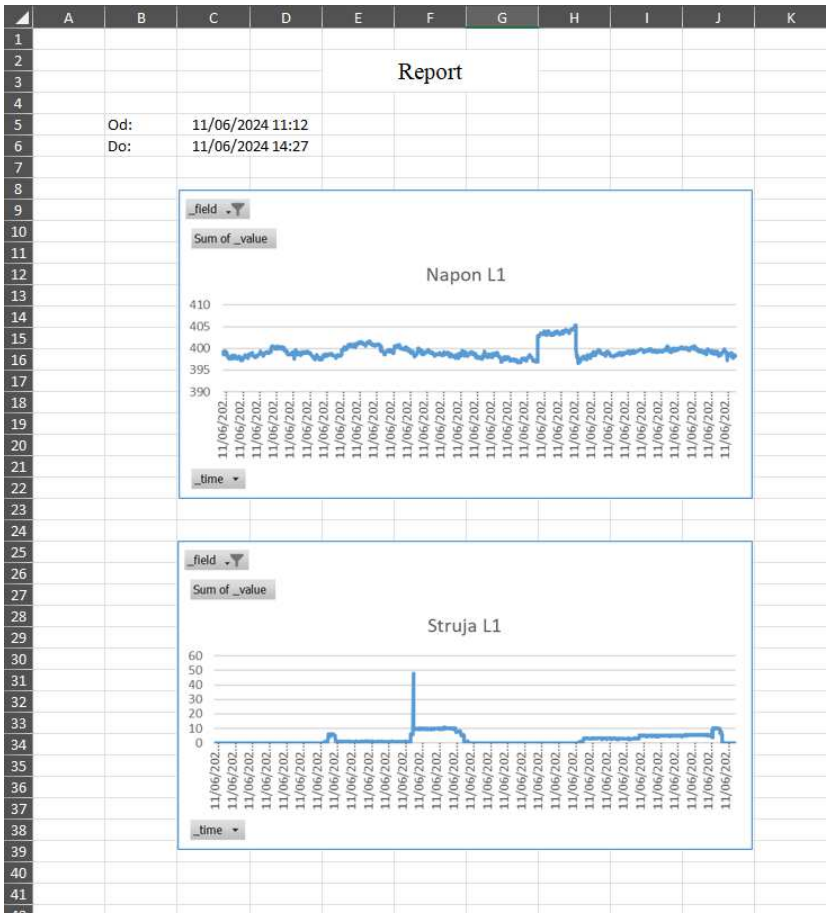
3.4.1. Generiranje izvješća u *MS Excel-u*

Microsoft Excel je jedan od pogodnih alata za obradu podataka zbog svoje jednostavnosti, ali i velikog broja mogućnosti prikaza podataka. Za potrebe ovog izvještaja korištene su tzv. *pivot* tablice; posebna vrsta tablica kojoj se definira izvor podataka na temelju kojega se *pivot* tablica sama automatski ažurira svaki put kada se izvorna datoteka promijeni.

	A	B	C	D	E	F	G
1	_start	_stop	_time	_value	VIB	_field	_measurement
2	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:12	0	1	A_RMS	Measurements
3	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:12	0	1	A_RMS	Measurements
4	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:12	0	1	A_RMS	Measurements
5	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:12	0	1	A_RMS	Measurements
6	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:12	0	1	A_RMS	Measurements
7	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:12	0	1	A_RMS	Measurements
8	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:12	0	1	A_RMS	Measurements
9	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
10	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
11	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
12	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
13	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
14	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
15	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
16	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
17	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:13	0	1	A_RMS	Measurements
18	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
19	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
20	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
21	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
22	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
23	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
24	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
25	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
26	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
27	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:14	0	1	A_RMS	Measurements
28	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:15	0	1	A_RMS	Measurements
29	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:15	0	1	A_RMS	Measurements
30	10/06/2024 13:48	12/06/2024 11:48	11/06/2024 11:15	0	1	A_RMS	Measurements

Slika 3.4.12 *Pivot* tablica u *MS Excelu*

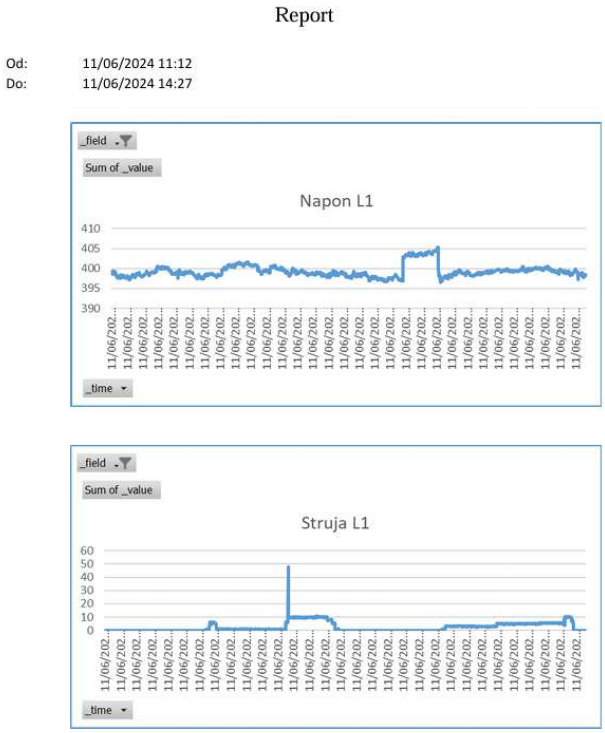
Svaka *pivot* tablica može imati pridružen grafički prikaz podataka unutar tablice koji se može izmjenjivati prema potrebama korisnika. Na ovaj način postavlja se nekoliko grafičkih prikaza podataka na isti list jedne *Excel* datoteke i dobiva se jednostavni prikaz podataka u nekom vremenskom intervalu.



Slika 3.4.13 Struktura izvještaja u *Excel-u*

Ovakav izvještaj može se izvesti u PDF datoteku pogodnu za arhiviranje ili ispis za potrebe administracije kao na slici 3.4.14 .

Izvještaj se automatski ažurira svaki put kada korisnik zatraži ispis podataka iz baze u nekom novom vremenskom intervalu, `.csv` datoteka s podacima se mijenja, što automatski rezultira i promjenom u *pivot* tablicama koji onda definiraju novi grafički prikaz tih podataka koji se nalaze u izvještaju.



Slika 3.4.14 Konačni izgled izvještaja u PDF formatu

3.4.2. Prikaz trenutnih vrijednosti i *Grafana*

Drugi način prikazivanja podataka je program *Grafana*. *Grafana* je mrežna aplikacija koja, slično kao i *InfluxDB*, prilikom instalacije i pokretanja stvara *web server* putem kojeg se konfigurira vizualizacija podataka, tzv. *dashboard*. Ova aplikacija je odabrana zbog već postojeće mogućnosti integriranja baze iz *InfluxDB-a* samo s potrebnim identifikacijskim parametrima kao i kod korisničkih aplikacija opisanih u ovom poglavlju.

Jednom kada se uspostavi veza s bazom podatka, potrebno je dizajnirati *dashboard* koji će pregledno prikazivati vrijednosti koje se mjere u danom trenutku. *Grafana* nudi nekoliko opcija kako prikazivati podatke od uobičajenih linijskih grafikona do "potenciometarskog" prikaza i raznih oblika dijagrama (stupčasti, tortni itd.).

Dodatni benefit *Grafane* je mogućnost dizajniranja *dashboard-a* i dijeljenje linka kojim se pristupa vizualizaciji, ali bez ikakve mogućnosti mijenjanja, brisanja podataka, komponenata i dr. kako ne bi došlo do neželjenih manipulacija nad mjernim sustavom

Slika 3.4.15 Primjer *dashboard-a*

od strane npr. osoblja pogona koje nije upućeno u rad sustava.

Slika 3.4.16 Prikaz akceleracija vibracije

4. Provođenje mjerenja i sustav u radu

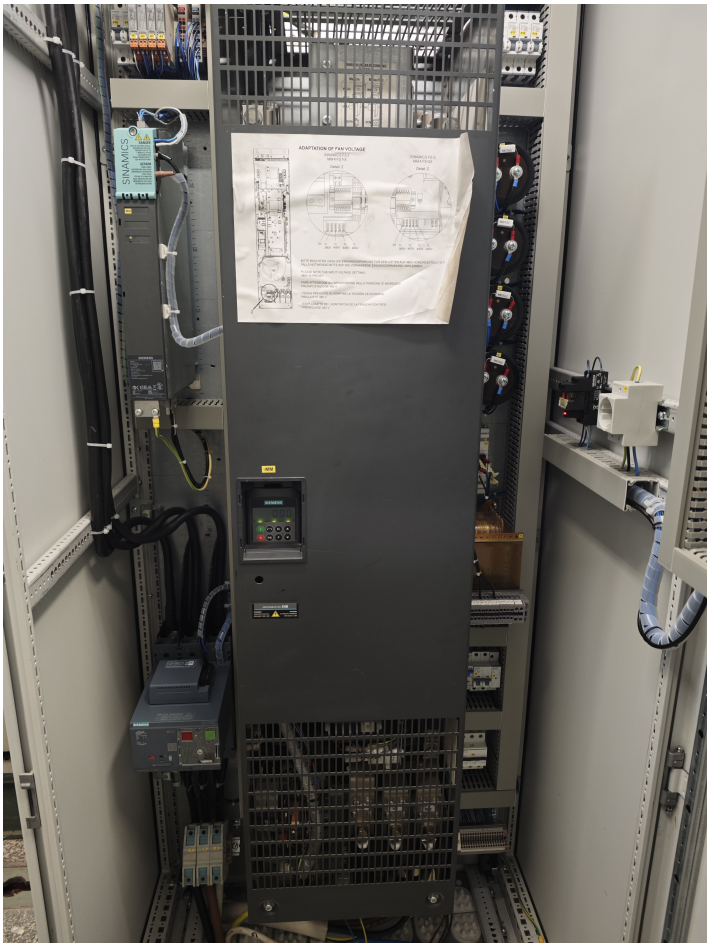
Kada su izrađene sve komponenete sustava, programiranje aplikacija, definiranje konfiugracija PLC-a, mjernih uređaja, postavljanje baze podataka, *Grafane* i *Excel-a*, potrebno je testirati rad sustava u cijelosti. U ovom poglavlju dan je kratki pregled jednostavnog testiranja rada gotovog sustava na elektromotornom pogonu opisanom u poglavlju 2.

4.1. Postupak mjerenja sekvence motora i rezultati

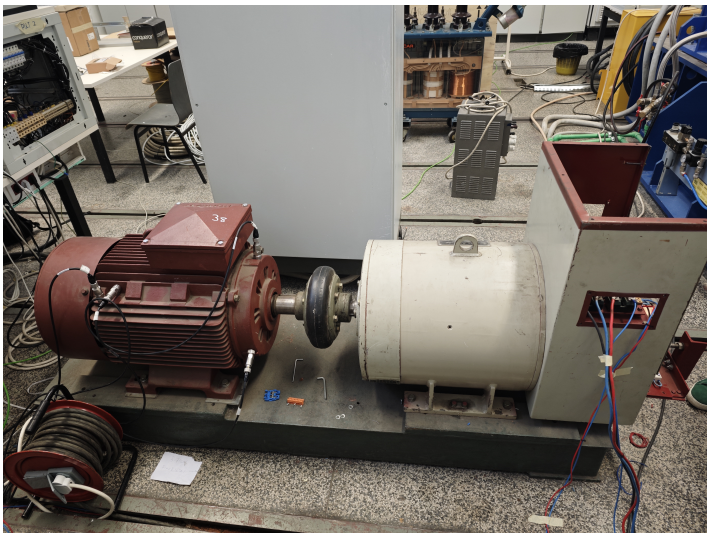
Elektromotorni pogon napajan je iz pretvarača napona i frekvencije u elektroormaru. Za potrebe testiranja sustava iskorišten je već postojeći sustav upravljanja pretvaračem i HMI panel na vratima ormara kojim se može upravljati motorom. Motor se može upravljati zadavanjem reference brzine u [o/min] ili zadavanjem momenta u [%] (postotak u odnosu na nazivni moment.

Korišteno je upravljanje po brzini i to u 3 radne točke: prazni hod na 500, 1000 i 1500 *o/min*. Uz to, na osovinu ispitivanog motora povezan je sinkroni generator koji je bilo moguće uklopiti na elektroenergetsku mrežu te tako vidjeti utjecaje uklopa generatora na električne vrijednosti i vibracije motora.

Nakon postavljanja svih VIB senzora na kućište motora, povezani su strujni transformatori opisani u poglavlju 2. i SENTRON PAC 3220 za mjerenje električnih vrijednosti. Senzori se priključuju na SM1281 modul koji je povezan s S7-1200 PLC-om, a SENTRON PAC komunicira sa softverskim PLC-om putem mrežnog kabela i mrežnog preklopnika(*switch*). U ovom testiranju cijeli sustav se nadzirao korištenjem *HP* osobnog računala i nadziranjem varijabli putem *TIA Portala* i njegovih mogućnosti praćenja vrijednosti u realnom vremenu.

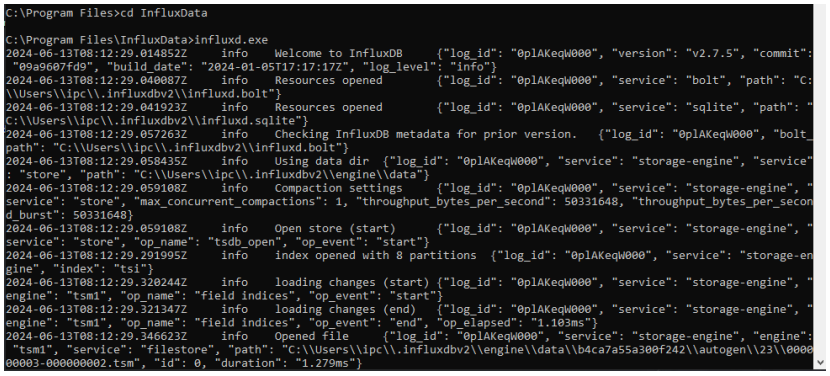


Slika 4.1.1 Pretvarač u elektroormaru



Slika 4.1.2 Elektromotorni pogon (asinkroni motor - lijevo, sinkroni generator - desno)

Na IPC-u je pokrenut rad *InfluxDB* baze (slika 4.1.3) i aplikacija za čitanje vrijednosti s PLC-a putem OPC UA poslužitelja. Budući da aplikacija, osim što upisuje vrijednosti u bazu, na konzolu ispisuje upisane vrijednosti iz PLC-a, ispravan rad sustava prati se i na IPC-u.



Slika 4.1.3 Pokretanje baze podataka

Kada se sve navedeno postavi, sustav je spreman za praćenje mjerenih podataka i testiranje rada. Na zaslonu (slika 4.1.4) se pokreće uklapanje pretvarača i zadaje se referenca od 500 o/min. Nakon 15 minuta, referenca je podignuta na 1000 o/min, a nakon još 15 minuta na 1500 o/min. U zadnjem dijelu je uklopljen generator pri brzini od 1500 o/min kao dodatni teret.

Slika 4.1.4 Panel na elektroormaru

Vibracije za vrijeme testiranja dane su sljedećim grafičkim prikazima:

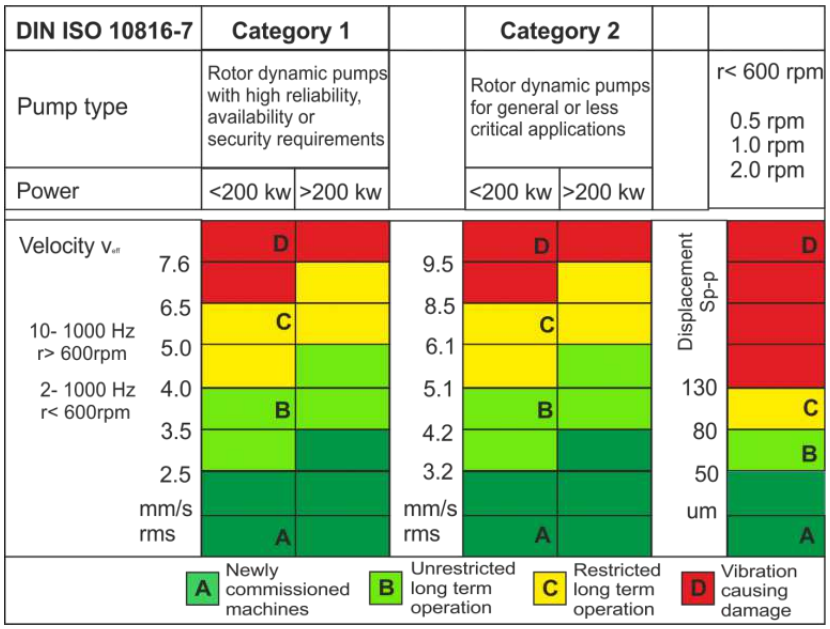
(a) Brzine vibracija

(b) Akceleracija vibracija

Slika 4.1.5 Prikaz vibracija za vrijeme testiranja

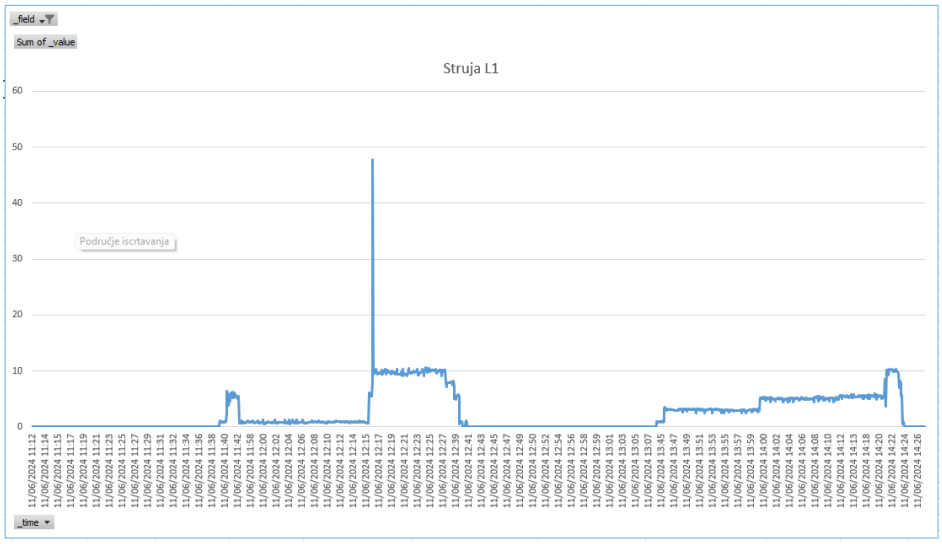
Može se primijetiti kako se na početku sekvence (niže brzine; 500 i 1000 o/min) vibracije i njihova akceleracija ne ističu dovoljno u stacionarnom stanju da bi se uopće primijetile. U posljednjem dijelu sekvence, pri brzini 1500 o/min, brzine i ubrzanja vibracija se osjetno povećavaju te se najviše mogu očitati na senzoru VIB1 postavljenom na sredinu kućišta na kojemu vibracija prelazi vrijednosti od 3 mm/s, a akceleracija 10 m/s^2. Prema ISO standardu 10816, koji opisuje granične vrijednosti mehaničkih vibracija, dozvoljene granice brzine vibracija u regularnom radu dolaze do 5.1 mm/s. Iznad te vrijednosti je propisom dozvoljen isključivo kratkotrajan rad. Prema gore vidljivim grafičkim prikazima, ispitivani motor ulazi u dozvoljeni raspon brzina vibracije za kategoriju motora s manje od 200 kW snage (4.1.6).

Praćene su i vrijednosti struje za vrijeme sekvence. Struja je ovisna o potrebnom mo-



Slika 4.1.6 Tablica dozvoljenih vibracija prema ISO 10816

mentu stroja za ubrzanje do željene brzine te se vrijednosti struje povećavaju u trenucima ubrzanja/usporavanja i za vrijeme većeg tereta na osovini (rad s uklopljenim generatorom). Napon pretvarača mjeren je na ulaznoj strani frekvencijskog pretvarača budući da SENTRON PAC mjeri sinusne veličine, a na ulazu u motor oblik napona je dobiven pulsno - širinskom modulacijom (PWM).



Slika 4.1.7 Struja jedne faze pretvarača

Nakon izvršavanja cijele sekvence, podaci su spremljeni u bazu podataka te su obrađeni drugom aplikacijom koja je napravila `.csv` datoteku sa svim zabilježenim vrijednostima u testu. Podaci su uvezeni u *Excel* tablicu iz koje je napravljen primjer izvještaja

i grafički prikazi koji se koriste u kroz ovaj rad. Osim slika iz *Excel-a*, napravljeni su i grafički prikazi u *Grafani*.

5. Zaključak

Glavni cilj ovog diplomskog rada bio je razvoj sustava za akviziciju podataka o potrošnji električne energije i mehaničkim vibracijama. Korištena je sofisticirana mjerna oprema zajedno s industrijskim programirljivim logičkim kontrolerima S7-1200 i softverski 1507s. Programi za unos vrijednosti u bazu podataka te čitanje vrijednosti u zadanom vremenskom intervalu su uspješno i zadovoljavajuće izvedeni s očekivanom funkcionalnošću. U zadatku rada spominje se i ODK (*Open Development Kit*) kao korištena komponenta, ali pokazala se neefikasnom u primjeni za problematiku ovog rada zbog kompliciranog mijenjanja programa u slučaju potrebe za proširenjem sustava, stoga nije korištena u radu.

Jedan od glavnih problema koji se nalazio na putu ovog projekta je velika količina različitih uređaja, proizvođača i drugih komponenata koje je trebalo integrirati na inovativan način kako bi se ostvarila prihvatiljiva funkcionalnost sustava; što je konačno i uspješno učinjeno.

U procesu osmišljanja strukture ovog sustava, glavna misao vodilja bila je modularnost i neovisnost ovog programskog rješenja o opremi ili mjerenim veličinama koje korisnik želi mjeriti i pohranjivati u bazu podataka. Tako je jedina potrebna izmjena u programskom kodu navođenje novih ID-eva čvorova na OPC UA serveru i promjena naziva polja, oznake i vrijednosti u *InfluxDB* bazi.

Ovakav rad je moguće usavršiti uvođenjem dodatnih analiza mjerenih podataka kao što su spektralne analize vibracija ili struje elektromotora koje mogu dati informaciju o zdravlju ležajeva i mogućim potrebnim servisnim radovima u budućnosti. Također, složenost i efikasnost napravljenih korisničkih programa može se uvelike poboljšati kako bi programi radili s većom brzinom i preciznošću.

Literatura

[1] Siemens, “Simatic panel ipc”, mall.industry.siemens.com/mall/en/WW/Catalog/Products/10424792?activeTab=productinformation&regionUrl=WW, pristupljeno 18.6.2024.

[2] I. Siemens Industry, *PAC3220 power meter data sheet*, 2021. [Mrežno]. Adresa: cache.industry.siemens.com/dl/files/507/109806507/att_1094488/v1/SIE_SS_PAC3220.pdf

[3] “An introduction to modbus TCP/IP”, rtautomation.com/technologies/modbus-tcpip/, pristupljeno: 16.6.2024.

[4] I. Siemens Industry, *SIMATIC SIPLUS CMS 1200 SM 1281 Condition Monitoring - operating instructions*, 2023.

[5] A. ABACUS, “Pressure sensors: The design engineer’s guide”, my.avnet.com/abacus/solutions/technologies/sensors/pressure-sensors/core-technologies/piezoelectric/, pristupljeno 17.6.2024.

[6] Siemens, “Simatic s7-1500 software controller”, siemens.com/global/en/products/automation/systems/industrial/plc/simatic-software-controller.html, pristupljeno 19.6.2024.

[7] Ferry, “What is opc-ua: A comprehensive guide to understanding the protocol”, deployferry.io/explain/opc-ua, pristupljeno 20.6.2024.

[8] I. Documentation, “Influxdb schema design and data layout”, docs.influxdata.com/influxdb/v1/concepts/schema_and_data_layout/, pristupljeno 12.9.2024.

Sažetak

Razvoj sustava za praćenje u stvarnom vremenu energetskih vrijednosti industrijskih proizvodnih postrojenja baziran na industrijskom upravljačkom sustavu s integracijom aplikacije treće strane

Luka Tisaj

Korištenjem *C#* aplikacija i *InfluxDB* baze podataka izveden je sustav dugoročne pohrane podataka o potrošnji električne energije i mjerenju vibracija elektromotornog pogona. Softverski kontroler 1507s i S7-1200 PLC prikupljaju podatke koji se potom zapisuju u bazu podataka te se *Winows form-om* mogu zatražiti zabilježeni podaci u određenom vremenskom intervalu u obliku `.csv` datoteke. Iz `.csv` datoteke generira se izvještaj ili se prikazuje grafički korištenjem programa *Grafana*.

Ključne riječi: vibracije; s7-1200; softverski PLC; InfluxDB; Grafana; C#; OPC UA; SENTRON PAC3220

Abstract

Development of a system for real-time monitoring of energy values of industrial production facilities based on an industrial control system with the integration of a third-party application

Luka Tisaj

Using *C#* applications and *InfluxDB* database, a long-term datalogging system of power and vibration monitoring was developed. The system was tested on an electrical drive configuration. The 1507s software controller and S7-1200 PLC collect necessary data which is then written into a database. The database can be accessed using a *Windows Form* with the user being able to define a desired time interval. A `.csv` file is generated as the output which can then be turned into a report or visualised using *Grafana* software.

Keywords: vibrations; s7-1200; software PLC; InfluxDB; Grafana; C#; OPC UA; SENTRON PAC3220