UPRAVLJANJE DIZALOM POMOĆU PROGRAMIRLJIVOG LOGIČKOG KONTROLERA

Šimundić, Valentin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

UPRAVLJANJE DIZALOM POMOĆU PROGRAMIRLJIVOG LOGIČKOG KONTROLERA

Završni rad

Valentin Šimundić

Osijek, 2019.

SADRŽAJ

1. UVOD

U ovom radu opisan je postupak automatizacije dizala pomoću programirljivog logičkog kontrolera. Dizalo je prijevozno sredstvo koje ima mogućnost kretanja gore i dolje, u svrhu prijevoza ljudi i tereta. Kako bi dizalo radilo na željeni način, potrebna je programska podrška koja će mu davati naredbe. Također, potreban je i programirljivi logički kontroler, PLC (engl. programmable logic controller) koji prima zahtjeve, obrađuje ih prema napisanom programu i daje naredbe dizalu. Dizalo će raditi u tri različita načina rada: ručni, poluautomatski i automatski. Osim upravljanjem iz samog dizala, dizalom će se moći i upravljati preko grafičkog sučelja (HMI, engl. Human-Machine interface) koje će prikazivati trenutnu poziciju i rad dizala. Zadatak također uključuje pisanje LANDEF tablice i električne sheme prema standardu tvrtke Danieli Systec i prikazivanje određenih informacija o radu dizala na web aplikaciji preko MODBUS protokola. Simulacija upravljanja i automatizacije stvarnog sustava dizala bit će ostvarena kroz korištenje makete dizala smještene u zgradi fakulteta.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je napraviti program za dizalo koje će raditi u ručnom, poluautomatskom i automatskom načinu rada u programskom okruženju TIA Portal. Uz to, potrebno je napraviti simulaciju na maketi dizala, vizualizaciju dizala (HMI), LANDEF tablicu, elektrosheme i prikazati rad dizala u web aplikaciji koristeći MODBUS protokol.

2. TEORIJSKA PODLOGA

2.1. Upravljanje procesima

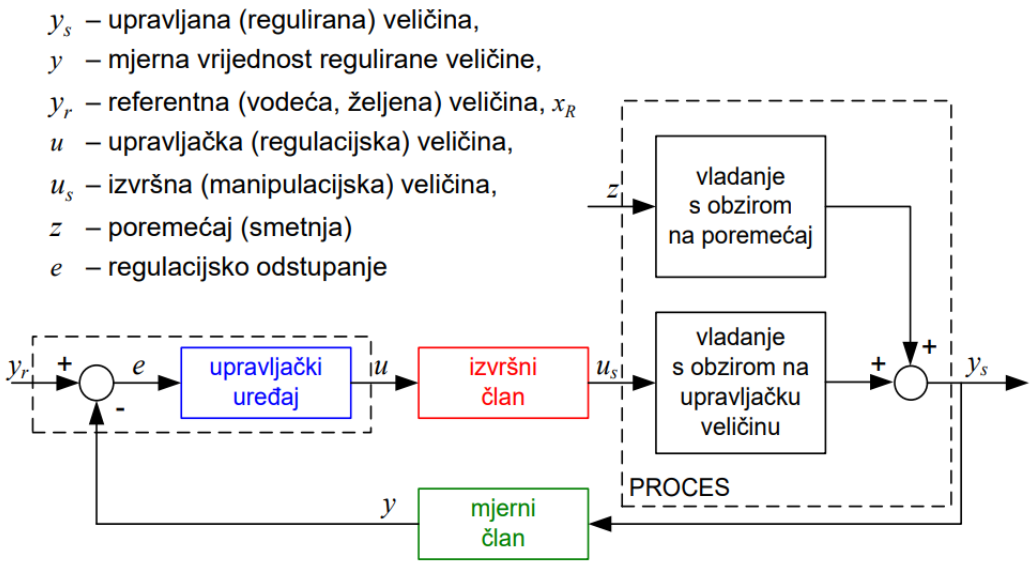
Još od prvih civilizacija pojavljuju se uređaji koji služe kao zamjena za čovjekov fizički rad. No, ti uređaji i strojevi mogu zamijeniti samo rad koji je, u svojoj osnovi, rutinski slijed operacija. Masovnu primjenu tzv. energetskih strojeva donijela je industrijska revolucija. Time je započelo doba mehanizacije koju obilježava zamjena ljudskog rada kako bi se proizvodnja poboljšala, povećala i pojeftinila, a rad olakšao. Iako je ljudska uključenost u proces proizvodnje znatno smanjena, čovjek je i dalje morao ostati u funkciji upravljanja strojevima. U 20. stoljeću su procesi proizvodnje postali sve složeniji, a čovjek nije mogao pratiti i povezivati sve promjenjive veličine. Iz tog razloga, bilo je potrebno da strojevi zamijene i umni rad čovjeka. Pojavom tzv. informacijskih strojeva koji su bili vezani uz elektroničku tehnologiju i elektronička računala, strojevi su počeli upravljati strojevima, a čovjekova uključenost u proces proizvodnje je znatno smanjena. Time započinje doba automatizacije. Prema [23], eru automatizacije može se nazvati „etapom proizvodnje koju obilježava oslobađanje čovjeka funkcije upravljanja proizvodnim procesom."

Dakle, cilj automatizacije je omogućiti stroju da samostalno nadzire ispravnost i upravlja procesom. Upravo se ovdje nameće automatsko upravljanje kao jedan od najvažnijih zadataka današnjeg razvitka. Automatsko upravljanje je jedna od najbitnijih disciplina za mnogo inženjerskih znanosti. Određena načela i pravila automatskog upravljanja često se koriste i u područjima koja nisu tehnička. Zadatak automatskog upravljanja je održavanje zadanog ili željenog stanja procesa protiv svih poremećaja, bilo unutarnjih ili vanjskih, pomoću odgovarajućih uređaja.

Upravo iz razloga što se automatsko upravljanje koristi u tehničkim i netehničkim dinamičkim sustavima, dinamički sustav treba promatrati globalno. Prema [24], „dinamički sustav predstavlja funkcijsku cjelinu za obradbu i prijenos energije, materije, informacije i kapitala, gdje se ulazne veličine sustava promatraju uzrokom, a izlazne veličine sustava njegovom vremenskom posljedicom."

2.1.1. Upravljanje s povratnom vezom

Prije objašnjavanja upravljanja s povratnom i bez povratne veze, bitno je prvo objasniti što je blokovski prikaz sustava. Zbog boljeg razumijevanja, koristi se standardizirana struktura sustava upravljanja gdje se svaki član može prikazati blokom, smjer djelovanja strelicom i matematička funkcija svojim simbolom.



Slika 2.1. Osnovna struktura upravljanja s povratnom vezom u blokovskom prikazu [25]

Slika 2.1. prikazuje osnovnu strukturu upravljanja u blokovskom prikazu. Ovdje je vidljivo da se regulacijski krug sastoji od upravljačkog uređaja (regulator), izvršnog člana, procesa (vladanje s obzirom na poremećaj i vladanje s obzirom na upravljačku veličinu) i mjernog člana. Upravljački član ili regulator je uređaj koji na temelju podataka o trenutnoj vrijednosti regulirane veličine izračunava upravljačku veličinu koja se šalje izvršnom članu s ciljem izjednačavanja stvarnih vrijednosti regulirane veličine sa zadanom. Izvršni član je uređaj koji omogućuje promjenu regulirane veličine. To su uglavnom crpke, elektromotori, ventili ili frekvencijski pretvornici. Proces je objekt upravljanja koji podrazumijeva promjenu regulirane veličine pod utjecajem izvršnog člana i raznih poremećaja. Mjerni član je uređaj koji mjeri reguliranu veličinu i daje regulatoru podatak o njezinoj trenutnoj vrijednosti.

Iz slike 2.1. se jasno vidi zašto se ovo upravljanje naziva upravljanje s povratnom vezom. Ulazna ili referentna veličina y_r je pobuda sustavu, odnosno željeno stanje u koje želimo da izlazna veličina dođe. Stvarna izlazna veličina y_s je vremenska posljedica referentne veličine. Pomoću mjernog člana možemo dobiti mjernu vrijednost regulirane veličine (y) koja zajedno s ulaznom veličinom daje regulacijsko odstupanje e po formuli:

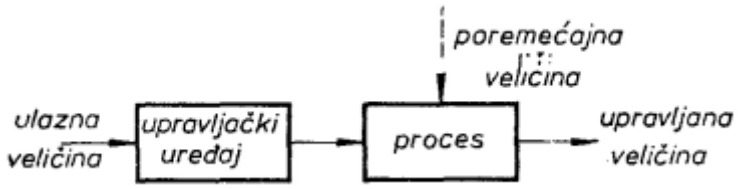
$$y_r(t) - y(t) = e(t) \tag{2-1}$$

Regulacijsko odstupanje je ulaz u regulator i čini najbitniji dio upravljanja procesom s povratnom vezom. Upravo ono označava koliko je proces blizu željenom stanju. Ako e iznosi 0, to znači da je izlazna veličina jednaka ulaznoj, odnosno sustav se ponaša kako želimo. U suprotnom, regulator obrađuje regulacijsko odstupanje i šalje upravljačku veličinu izvršnom članu koji će izvršiti

određenu radnju. Ta radnja direktno utječe na proces kojemu se onda mijenja izlazna veličina i proces se tako ponavlja u krug. Upravo zbog toga se i naziva upravljanje s povratnom vezom.

2.1.2. Upravljanje bez povratne veze

Upravljanje bez povratne veze, za razliku od upravljanja s povratnom vezom, ne sadrži povratnu vezu koja omogućava usporedbu referentne veličine i izmjerene izlazne veličine. Ulazna veličina stalno djeluje na izlaznu veličinu, ali bez pomoći izmjerene izlazne veličine. Bez obzira na to, regulator još uvijek postoji u sustavu, ali radi na drukčiji način. Koristi drukčije algoritme koji se ne oslanjaju na razliku između referentne veličine i izmjerene izlazne veličine. Izgled upravljanja bez povratne veze može se vidjeti na slici 2.2.



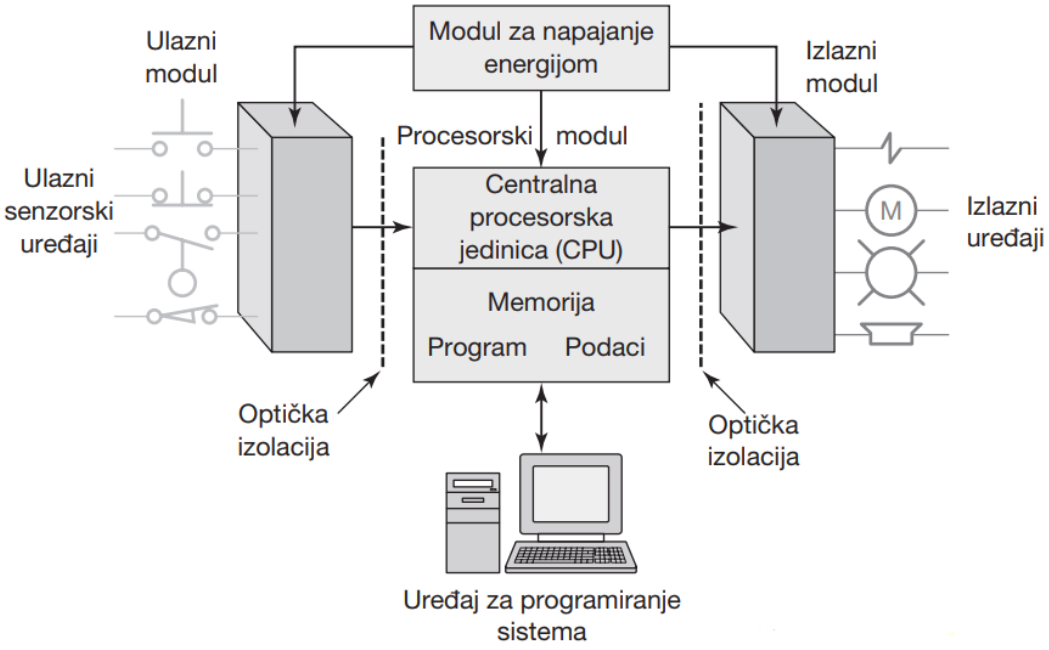
Slika 2.2. Upravljanje bez povratne veze u blokovskom prikazu [23]

U ovom završnom radu, koristit će se upravljanje bez povratne veze. Ulaznu veličinu predstavljat će unos željene pozicije pomoću tipke na dizalu ili iz vizualizacije. Regulator predstavlja algoritam koji će biti objašnjen u kasnijim poglavljima. Mjerni član inače predstavlja senzor na svakom katu, ali u ovom slučaju će to biti izračunata pozicija uz pomoć senzora, što će također biti objašnjeno u kasnijim poglavljima. Izvršni član predstavlja motor za kretanje gore i dolje. Pritiskom na tipku za određeni kat, algoritam (regulator) uspoređuje trenutnu poziciju (izlazna veličina) sa željenom (referentna veličina). U slučaju da pozicije nisu jednake, nadalje se provjerava je li trenutna pozicija veća ili manja od željene. Nakon utvrđivanja, motor (izvršni član) pokreće dizalo u jednom od smjerova sve dok dizalo ne dođe na željeni kat i mijenja se trenutna pozicija dizala. Dolaskom na željeni kat, pozicija se izjednačava sa željenom. Algoritam tada gasi motor i dizalo staje.

2.2. Programirljivi logički kontroler (PLC)

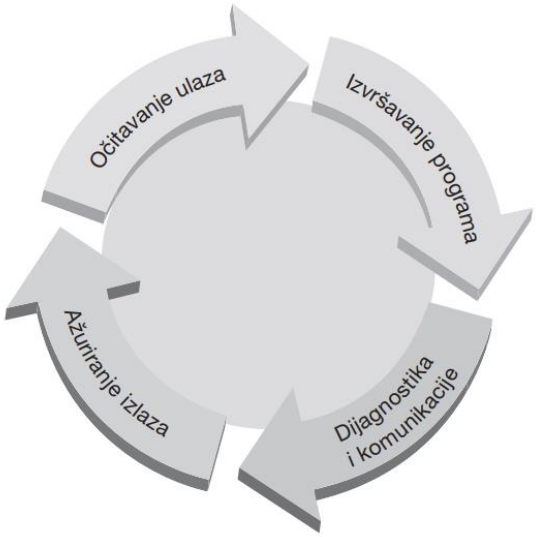
PLC ili Programirljivi logički kontroleri su uređaji zbog kojih je došlo do revolucije u području industrijske automatike. Danas, PLC-ovi se mogu naći kao dio gotovo svakog proizvodnog procesa, no do 1968. godine programirljivi kontroleri nisu postojali. Prije pojave PLC-a, upravljanje strojevima vršilo se pomoću releja. Releji rade na način da, kada se zavojnici dovede energija, stvara se magnetska sila koja povuče prekidač u upaljeno ili ugašeno stanje. Odvođenjem energije, magnetska sila nestaje i prekidač se vraća u početno stanje. Korištenje releja često je dovodilo do komplikacija i problema u strojevima. Ožičavanje je bilo previše komplicirano, nije bilo mogućnosti za nadogradnju i releji su se često istrošili. Tvrtka GM Hydramatic zatražila je promjenu relejnog sustava. Bedford Associates su za njih dizajnirali prvi PLC, među kojima je bio i Dick Morley, čovjek smatran ocem PLC-a. Prvi PLC, Modicon 084, dizajniran je kako bi se u njemu moglo programirati u ljestvičastom dijagramu koji je sličio shemama relejne logike koju je trebao zamijeniti. To je omogućilo lakši prijelaz inženjera i tehničara na logiku PLC-a. S godinama su se razvijali mnogi protokoli poput MODBUS-a, doneseni su standardi (IEC 61131-3), razvijeni novi programski jezici PLC-a, a danas su PLC-ovi sposobni upravljati vrlo složenim sustavima i dosegli su procesorsku snagu koja je usporediva sa snagom osobnog računala.

Programirljivi logički kontroler (PLC, engl. Programmable Logic Controller) je poseban oblik kontrolera koji je baziran na mikroprocesoru. Sadrži programirljivu memoriju za spremanje naredbi i implementiranje različitih funkcija s ciljem upravljanja strojevima i procesima. Za razliku od osobnih računala, PLC-ovi se najčešće koriste u industrijskim postrojenjima zbog njihove otpornosti na vlagu, visoku i nisku temperaturu, prašinu i ostalo. Ostale prednosti PLC-a su: jednostavno programiranje, velika brzina odziva, mrežna kompatibilnost, lako testiranje i otklanjanje grešaka i visoka pouzdanost. PLC se najčešće sastoji od središnje procesorske jedinice (CPU, engl. Central Processing Unit), napajanja, ulazno/izlaznih modula, komunikacijskog sučelja, programske i podatkovne memorije. Za razliku od osobnog računala, PLC je dizajniran da radi u industrijskom okruženju, stoga je opremljen posebnim ulaznim i izlaznim sučeljima i programira se pomoću posebnog jezika za upravljanje procesima. PLC je puno moćniji od relejne tehnologije koju zamjenjuje. Osim što može obavljati iste poslove kao i sustav temeljen na relejnoj logici, u mogućnosti je raditi i mnoge druge poslove, poput mjerenja vremena, prebrojavanja, izračunavanja i obrade analognih signala.



Slika 2.3. Prikaz dijelova PLC-a [4]

Centralna procesorska jedinica (engl. Central Processing Unit, CPU) može se smatrati „mozgom“ PLC-a. Obično CPU modul PLC-a sadrži mikroprocesor koji izvršava program i upravlja komunikacijom između modula. Rezultati logičkih operacija koje izvršava procesor se moraju spremiti na neko mjesto, stoga je procesoru neophodna memorija (EPROM za program, RAM, ROM). PLC izvršava napisani program kao naredbe koje se mogu ponavljati. Jedno izvođenje napisanog programa zove se ciklus (engl. cycle). Ciklus započinje tako što CPU čita ulaze PLC-a, odnosno stanja na ulazima i izvršava program. Nakon izvršenja programa, CPU obavlja poslove dijagnostike i komunikacija, nakon čega ažurira stanje na izlazima PLC-a. Dok je PLC u radnom režimu, postupak se neprestano ponavlja. Vrlo je bitno da vrijeme čitanja ulaza bude manje od vremena promjene ulaza. U protivnom, PLC neće moći odgovoriti dovoljno brzo.



Slika 2.4. Prikaz ciklusa izvršavanja PLC programa [4]

PLC radi u realnom vremenu jer svaki događaj koji se dogodi na ulaznom uređaju, rezultira određenom radnjom ili promjenom stanja na izlazu kontrolera.

Memorijska jedinica sprema program, informacije o stanjima i informacije o sustavskim operacijama kontrolera. Postoji nekoliko memorijskih elemenata u sustavu PLC-a:

- ROM (engl. Read-Only Memory) koja omogućava trajnu pohranu ugrađenog programa i nepromjenjivih podataka CPU-a.
- RAM (engl. Random Access Memory) za korisnički program.
- RAM za podatke. Ovdje se spremaju podaci i informacije sa stanja ulaza i izlaza, vrijednosti *timera* i ostalih unutrašnjih uređaja.
- EPROM (engl. Erasable and Programmable Read-Only Memory) je ROM koji se može programirati i u kojem se program može spremiti trajno.

Program i podatke u RAM-u može mijenjati korisnik. Svi PLC-ovi imaju određenu količinu RAM-a za spremanje korisničkih programa i podataka. Međutim, kako bi se izbjegao gubitak programa kada se PLC isključi, u PLC-u postoji baterija koja održava podatke u RAM-u određeno vrijeme. Nakon što se program spremi u RAM, moguće ga je spremiti i u EPROM memorijski čip kako bi ga se spremilo trajno. Također, postoje i međuspremnici za spremanje podataka ulaznih i izlaznih kanala.

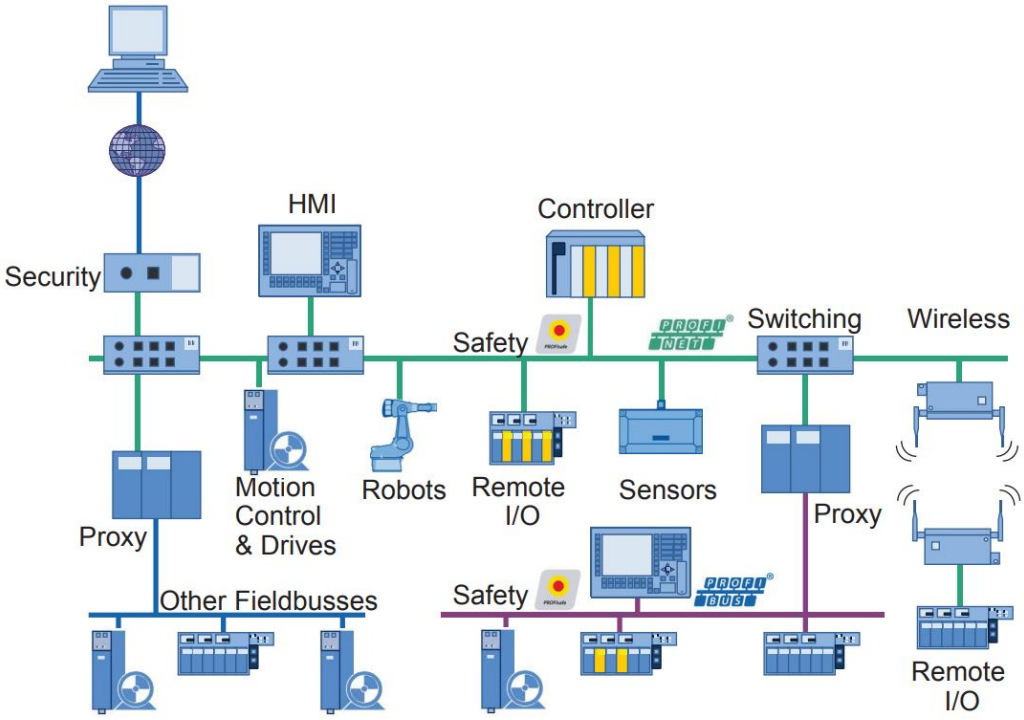
Ulazno/izlazna jedinica pruža sučelje između sustava i vanjskog svijeta. Omogućava stvaranje veze kroz ulazno/izlazne kanale do ulaznih uređaja (npr. senzora) i izlaznih uređaja (npr. motora). Svaki ulaz i izlaz ima svoju jedinstvenu adresu u PLC-u koju CPU može koristiti. Ulazno/izlazni

kanali pružaju zaštitu i procesiranje analognog signala kako bi se senzori i izvršni uređaji mogli direktno spojiti na PLC bez nekog dodatnog ožičavanja. Digitalni signal koji je općenito kompatibilan s mikroprocesorom PLC-a je istosmjerni napon vrijednosti 5V. Dakako, procesiranje signala na ulazima omogućava širok raspon ulaznih signala koji se mogu upotrijebiti. Također, digitalni izlazni signal će iznositi 5V i nakon procesiranja signala pomoću releja, tranzistora ili dvosmjernog tiristora, izlazni signal može imati istosmjernu i izmjeničnu vrijednost od 24V do 240V.

Programiranje PLC-a može se napraviti na više načina. Najčešće su to ručni uređaji, prijenosni uređaji ili stolna računala. Tek nakon što je program napisan za programirljivi uređaj, moguće ga je prenijeti na memorijsku jedinicu PLC-a. Ručni uređaji često imaju dovoljno memorije za pohranu programa kako bi se mogao prenijeti s jednog mjesta na drugo. Prijenosni uređaji su osobna računala koja imaju ekran za prikaz uz potpunu tipkovnicu, a često imaju i mogućnost dodira zaslona. Stolna računala su najčešće konfigurirana kao radne stanice za programiranje PLC-a. Neki PLC-ovi zahtijevaju da računalo ima samo odgovarajuću programsku podršku, dok ostali zahtijevaju posebne kartice za komunikaciju kako bi se PLC mogao povezati s računalom. Proizvođači PLC-ova često imaju svoja razvojna okruženja za pisanje programa PLC-a i vizualizaciju. Na primjer, tvrtka Siemens ima svoje okruženje pod nazivom SIMATIC STEP 7 koji u potpunosti ispunjava standard IEC 61131-3 za programske jezike PLC-a. U STEP 7 razvojnom programu, programeri mogu birati različite programske jezike PLC-a. Uz ljestvičasti dijagram (LAD) i funkcijski blok dijagram (FBD), moguće je programirati i u programskim jezicima: instrukcijska lista (STL), strukturirani upravljački jezik (SCL) i sekvencijalni funkcijski dijagram (SFC). Programski jezici korišteni u ovom radu bit će opisani u kasnijem poglavlju.

2.2.1. PROFINET sučelje

PROFINET je Ethernet standard za automatizaciju koji omogućava velike brzine prijenosa, siguran prijenos podataka između uređaja različitih proizvođača. Na taj način PROFINET omogućava dizajniranje inovativnih uređaja, strojeva i sustava. Ugrađena dijagnostika i funkcije alarmiranja osiguravaju brzo vrijeme provjere i puštanja strojeva u pogon. Uz to, izravan pristup TCP/IP protokolu omogućava prijenos velikog broja informacija. Uređaj koji podržava PROFINET slijedi iste principe kao i uređaj koji podržava PROFIBUS model, što znači da omogućava brzu i jednostavnu zamjenu dvaju komunikacijskih sučelja. Također, daje mogućnost pristupa uređaju preko internetskog sučelja koristeći običan internetski preglednik.



Slika 2.5. Prikaz mogućnosti spajanja PROFINET-a [5]

U ovom završnom radu, PLC i računalo će biti spojeno PROFINET kabelom, gdje jedan kraj ide u Ethernet utor računala, a drugi u utor PLC-a. PROFINET kabel nije ništa drugo nego običan Ethernet CAT 5e kabel koji je zaštićen od vlage, prašine i vibracija.

Slika 2.6. PROFINET kabel

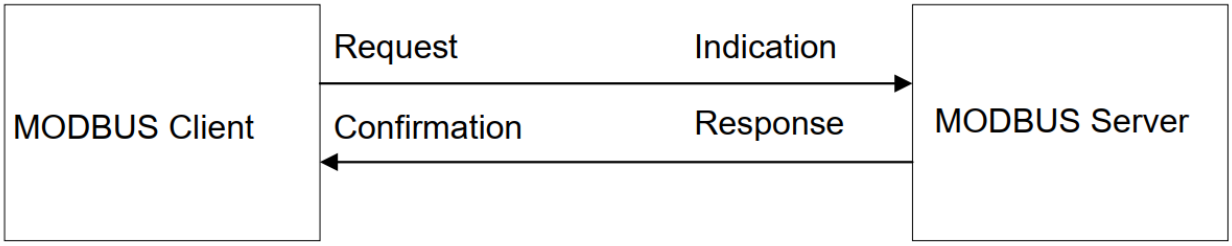
2.2.2. MODBUS protokol

MODBUS je industrijski protokol razvijen 1979. godine i danas je najraširenija komunikacijski protokol korišten u industriji. Protokol predstavlja vezu s višom razinom upravljanja. Smješten je na aplikacijskom sloju koji omogućava master/slave komunikaciju uređaja na različitim mrežama. *Slave*, u tom slučaju, ne može inicirati nikakvu komunikaciju, već mora čekati *master* uređaj kako bi se razmijenili podaci. *Master* uređaj može pisati u registre i čitati iz registara *slave* uređaja. Podržava paradigmu zahtjev/odgovor gdje svaka funkcija zahtjeva i odgovora ima svoju brojčanu oznaku (tablica 2.1.). Pomoću te oznake može se lako otkloniti bilo kakav problem u komunikaciji. Dva su najraširenija tipa MODBUS protokola: MODBUS RTU i MODBUS TCP. U ovom radu, koristit će se MODBUS TCP za prikupljanje podataka s PLC-a, slanje podataka u PLC i prikazivanje na web aplikaciji. Tipovi registara koji se koriste u MODBUS protokolu su: zavojnica (engl. coil), diskretni ulaz, ulazni registar i izlazni registar (engl. Holding register).

Tablica 2.1. MODBUS funkcije i pripadajuće šifre

Brojčana oznaka funkcije	Tip registra
1	*Read Coil*
2	*Read Discrete Input*
3	*Read Holding Registers*
4	*Read Input Registers*
5	*Write Single Coil*
6	*Write Single Holding Register*
15	*Write Multiple Coils*
16	*Write Multiple Holding Registers*

MODBUS TCP protokol omotava pakete zahtjeva i odgovora MODBUS RTU protokola u TCP pakete koji se prenose preko Ethernet mreže. Ovdje se mijenjaju definicije *master* i *slave* uređaja jer se u Ethernet mrežama češće koriste klijent i server. *Slave* uređaj postaje server, a *master* uređaj postaje klijent. U ovom slučaju, ne mora postojati samo jedan server, već ih može biti više. Isto tako, može biti i više klijenata.



Slika 2.7. Četiri tipa poruka između servera i klijenta MODBUS TCP-a [7]

Sa slike 2.7. može se vidjeti kako postoje četiri tipa poruka između servera i klijenta: upit, naznaka, odgovor i potvrda. Upit je poruka kojom klijent inicira prijenos podataka. Server prima taj upit koji njemu predstavlja poruku naznake. Nakon obrade, server šalje odgovor u kojemu se nalaze podaci, a klijent prima i potvrđuje primitak poruke. U ovom završnom radu, PLC će predstavljati server koji će na upite odgovarati traženim podacima, a Python aplikacija će predstavljati klijent.

3. KORIŠTENE TEHNOLOGIJE, ALATI I MAKETA DIZALA

Kako bi se zadaci mogli uspješno izvršiti, potrebni su određeni alati, kako alati sklopovlja, tako i alati programske podrške.

3.1. SIMATIC S7-1214 DC/DC/DC

PLC tvrtke Siemens korišten u ovom radu je iz Siemensove serije S7-1200. PLC-ovi iz serije S7-1200 koriste se za manje i srednje sustave i procese, u kojima njihova fleksibilnost i učinkovitost može doći do izražaja. Zajedno s CPU-om dolaze i ugrađeno napajanje, ulazno/izlazni modul s analognim ulazima i PROFINET sučelje.

Točan model PLC-a korišten u radu je SIMATIC S7-1214 DC/DC/DC. Sam CPU raspolaže s 14 digitalnih ulaza, 10 digitalnih izlaza, 2 analogna ulaza, Ethernet sučeljem za komunikaciju i može imati do 3 HMI panela. Zadnja dostupna verzija programske podrške za CPU je 4.2, a programsko okruženje koje podržava je STEP 7 v14 i sve verzije iza v14. Što se tiče memorije, sadrži 100kB ugrađene radne memorije koju nije moguće proširiti i 4MB memorije za pohranu programa s mogućnošću ubacivanja i dodatne memorijske kartice. CPU ima nizak odziv, gdje mu je potrebno 0.08μs po instrukciji za operacije s bitova, 1.7μs po instrukciji za operacije s tipom podatka WORD i 2.3μs po instrukciji za realne brojeve. Blokovi koje podržava su podatkovni blokovi, funkcije, funkcijski blokovi, brojači i *timeri* koji mogu imati adrese od 1 do 65535. Konfiguracija sklopovlja omogućava tri komunikacijska uređaja, jednu ploču za signale i osam signalnih modula.

Slika 3.1. SIMATIC CPU S7-1214 DC/DC/DC

3.2. SM 1223 DC/RLY

SM 1223 DC/RLY je signalni modul kojim se može proširiti CPU. Njime se dodaje još 16 digitalnih ulaza i 16 digitalnih izlaza na koje će se spojiti ulazi i izlazi dizala, poput tipki, senzora ili izlaza na motore. Na signalni modul su spojeni svi ulazi i izlazi makete dizala korištene za simulaciju završnog rada.

Slika 3.2. Ulazno/izlazna jedinica SM 1223 DC/RLY

3.3. Napajanje PM 1207

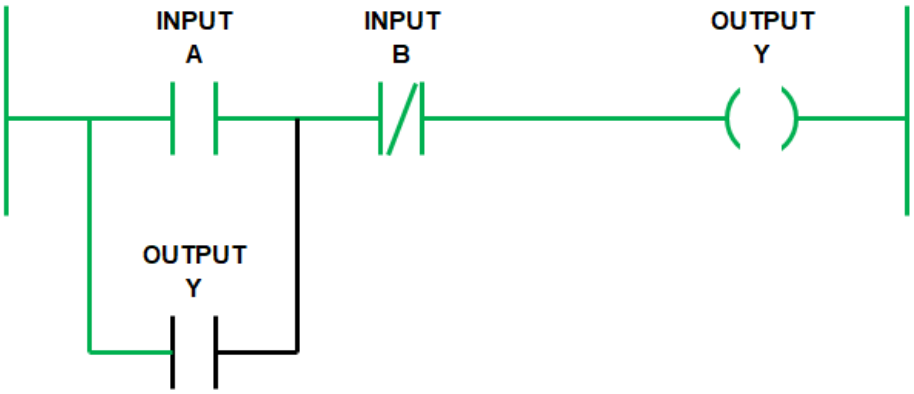
Napajanje je vrlo bitan dio sustava PLC-a bez kojeg CPU i ostale komponente ne mogu raditi. Napajanje prima do 264V izmjeničnog napona kojeg pretvara u 24V istosmjernog napona prema CPU i ima raspon od 0 do 2.5A električne struje.

Slika 3.3. Napajanje PM 1207

3.4.Korišteni programski jezici

Kako bi PLC mogao primiti podatke s ulaza, obraditi ih i poslati naredbe na željene izlaze, potreban je program. Najčešći programski jezici za pisanje PLC programa su ljestvičasti dijagram (LAD, engl. Ladder Diagram), funkcijski blok dijagram (FBD, Function Block Diagram) i SCL (engl. Structured Control Language). U ovom radu, najviše je korišten ljestvičasti dijagram uz malo SCL-a na mjestima gdje ga je jednostavnije upotrijebiti u odnosu na ljestvičasti dijagram.

Ljestvičasti dijagram standardizirani je PLC programski jezik koji je naziv dobio zbog svog izgleda, odnosno linija koje podsjećaju na ljestve. Vizualni je programski jezik, što znači da se za programiranje ne koristi tekst, već simboli. Ti simboli su napravljeni na način da izgledaju kao električni simboli (npr. kontakti i releji). Tri osnovna i najčešće korištena simbola ljestvičastog dijagrama su NO (engl. Normally open contact), NC (engl. Normally closed contact) i *coil* kao izlaz.



Slika 3.4. Prikaz osnovnih simbola ljestvičastog dijagrama

Slika 3.4. prikazuje osnovne simbole ljestvičastog dijagrama i način pisanja programa. Ulaz A predstavlja NO simbol, ulaz B predstavlja NC simbol, a izlaz Y predstavlja *coil*. Simboli NO i NC mogu se shvatiti kao *if* i *if not* naredbe u C programskom jeziku. Ako se dva simbola postave u istu liniju kao ulazi A i B, oni se mogu shvatiti kao „I“ sklop, a ako se postave jedan ispod drugog kao ulaz A i izlaz Y, mogu se shvatiti kao „ILI“ sklop.

SCL je standardizirani programski jezik visoke razine baziran na PASCAL-u. SCL upotpunjuje i proširuje programsku podršku i programske jezike STEP 7 programskog okruženja PLC-a. Može se koristiti zajedno s ljestvičastim dijagramom i ostalim programskim jezicima PLC-a. Za razliku od ljestvičastog programa, u SCL-u je jednostavno implementirati petlje i grananja, poput *for*, *while* i *case* naredbi. Primjer jednostavnog SCL koda dan je slikom 3.5.

Slika 3.5. Primjer SCL koda

3.5. TIA Portal

Totally Integrated Automation Portal (TIA Portal) razvojno je okruženje koje objedinjuje različite SIMATIC proizvode u jednu aplikaciju. Budući da su svi podaci spremljeni u jednom projektu, moguće je konfigurirati PLC, pisati kod i napraviti vizualizaciju u istom sustavu. Rješenja nekog automatizacijskog problema zahtijevaju:

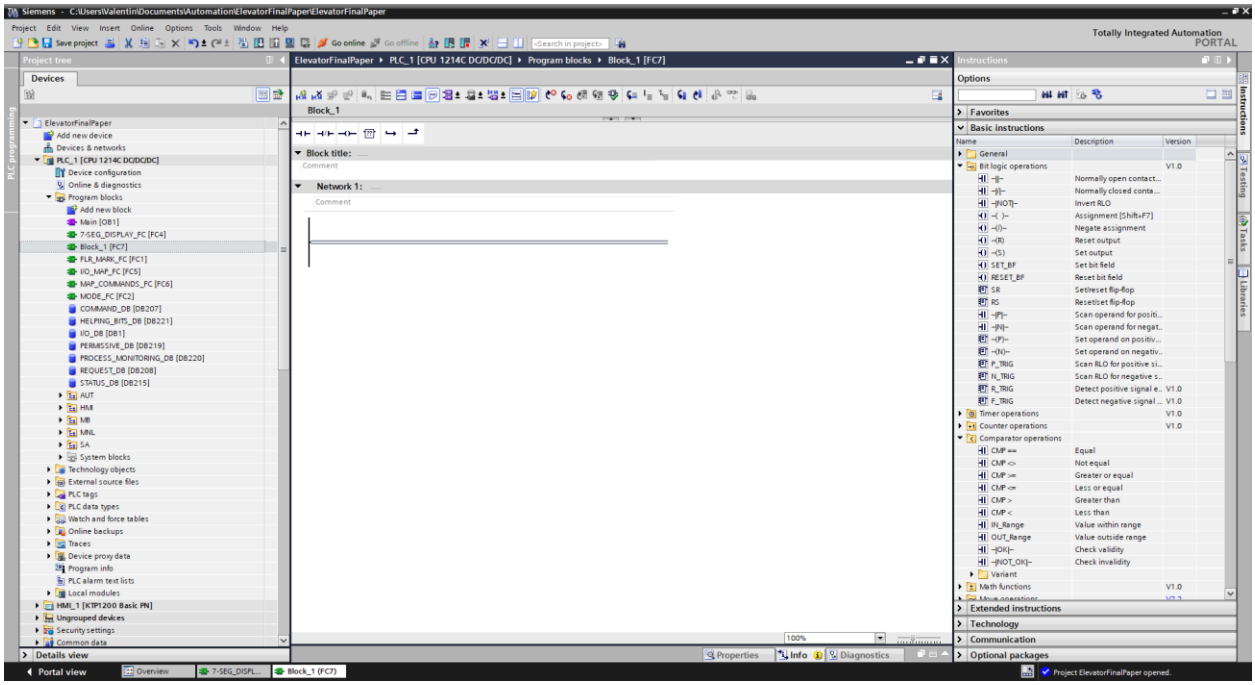
- PLC koji upravlja procesom uz pomoć programa.

- HMI uređaj koji pomaže upravljati procesom i vizualizirati isti.

Najčešći konfiguracijski koraci kod rješavanja problema automatizacije u TIA Portalu su:

- Kreiranje projekta
- Konfiguracija potrebnog sklopovlja
- Povezivanje uređaja
- Programiranje PLC-a
- Konfiguracija vizualizacije
- Učitavanje konfiguracijskih podataka
- Korištenje „online“ i dijagnostičkih funkcija

Dva najčešće korištena SIMATIC proizvoda su STEP7 Basic i WinCC Basic. STEP 7 Basic je paket programske podrške koji omogućuje konfiguraciju PLC-a i pisanje programa za PLC. WinCC Basic je program koji omogućuje izradu vizualizacije u kojoj se prikazuje trenutno stanje sustava, što sustav radi i mogu se dati određene naredbe sustavu.

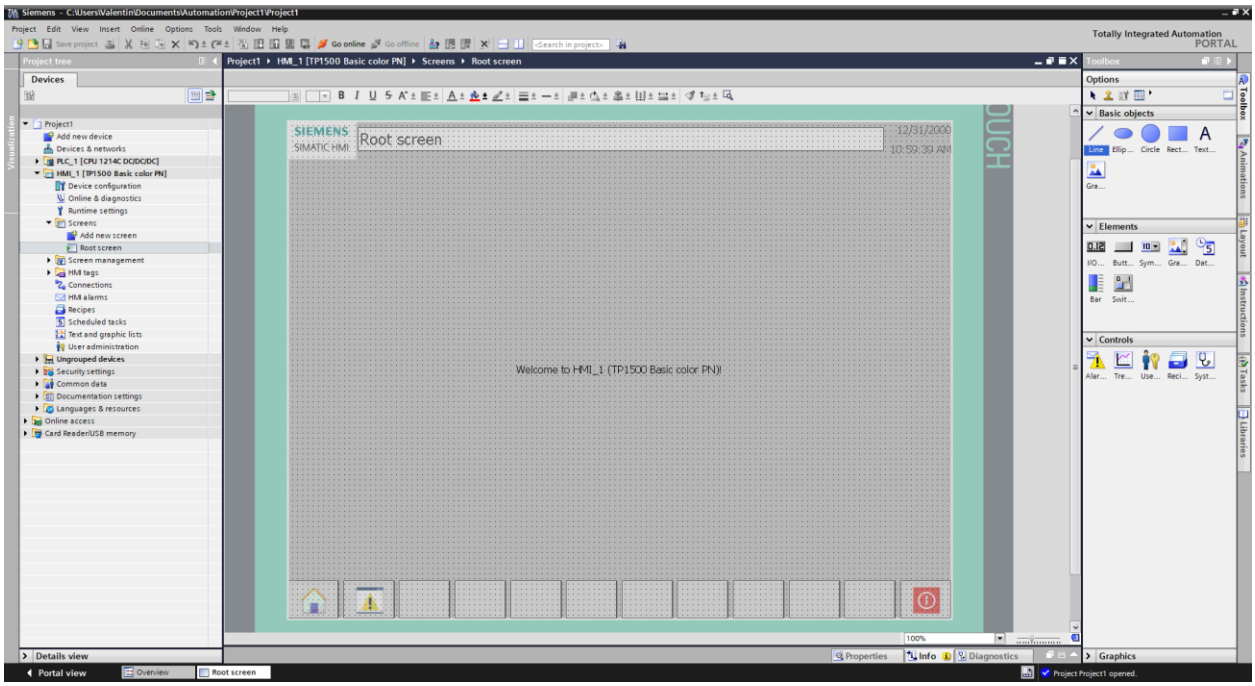


Slika 3.6. Izgled sučelja STEP 7

Postoje dva načina prikaza u TIA Portalu: „Portal view“ i „Project view“. Slika 3.6. prikazuje projektni način rada u kojemu se odvija sav razvoj programa. Projektno stablo TIA Portala omogućava prikaz svih blokova koji su napravljeni i daje mogućnost stvaranja novih blokova za rad. Postoji nekoliko blokova koji se koriste kod programiranja PLC-a:

- Organizacijski blok (OB) – blok koji služi za strukturu programa. Služi kao sučelje između operacijskog sustava i korisničkog programa. Mogu se izvoditi periodički, ovisno o veličini napisanog programa ili pod određenim uvjetima.
- Funkcija (FC) – blok koji najčešće izvodi određene operacije, ali ne sprema podatke u memoriju, odnosno nema vezan podatkovni blok na sebe.
- Funkcijski blok (FB) – blok koji koristi instancirani podatkovni blok za parametre i statičke podatke i izvodi određene operacije.
- Podatkovni blok (DB) – blok koji sprema podatke za blokove koda. Globalnom podatkovnom bloku mogu pristupiti svi blokovi, a instancirani blokovi spremaju podatke za određene funkcijske blokove.

Uz podatkovne blokove, korisno je i koristiti tablicu oznaka. Tablica oznaka se može napraviti ili koristiti već postojeća, a tablice se nalaze u projektnom stablu pod nazivom *PLC tags*.



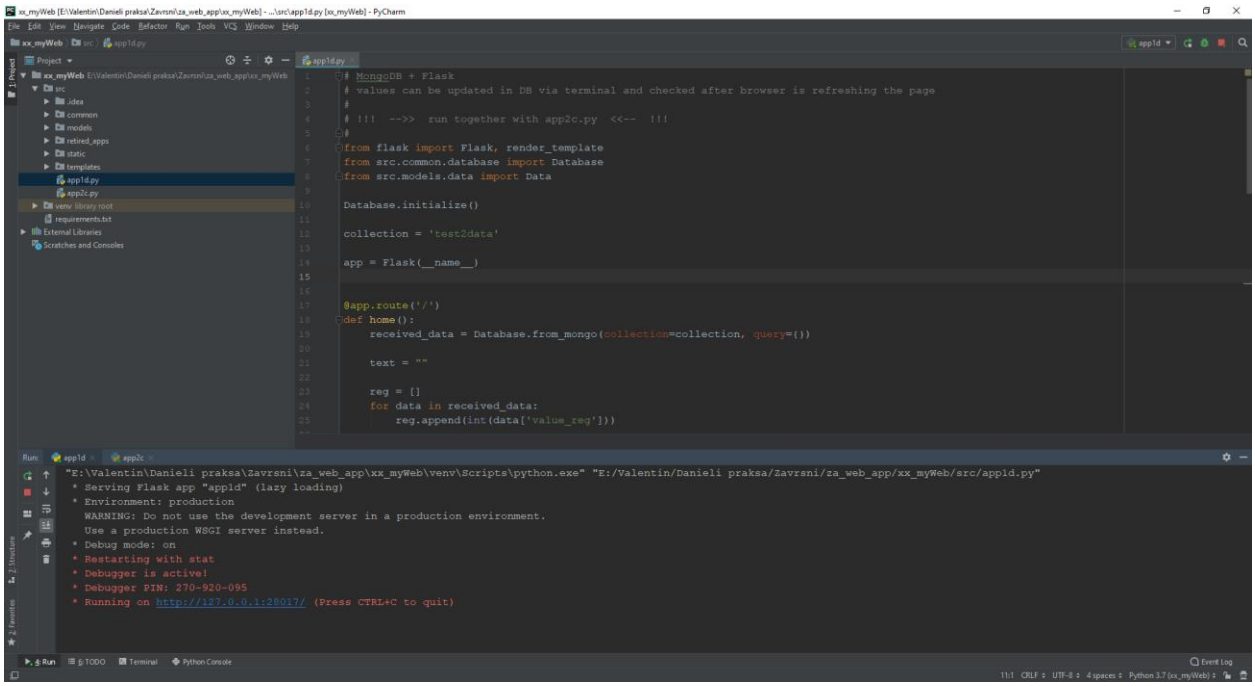
Slika 3.7. Izgled sučelja WinCC-a

Dodavanje elemenata u WinCC-u radi na principu „Drag and drop“, a elementi se mogu dodavati iz kartice *Toolbox*. Nakon postavljanja elementa, odlaskom u njegove postavke, mogu se namještati razne opcije. Također, mogu se dodavati animacije i događaji koji se izvršavaju ovisno o oznakama koje su im priložene. Te oznake se mogu dodavati u projektnom stablu, pod *HMI tags* kreiranjem novih ili korištenjem već postojećih tablica oznaka. Primjer tek kreiranog HMI-ja nalazi se na slici 3.7.

3.6.PyCharm

PyCharm tvrtke Jetbrains je razvojno okruženje za programski jezik Python koji pruža velik raspon alata za programiranje u programskom jeziku Python. Također, služi i kao okruženje za programiranje internetskih stranica. Dolazi u tri izdanja: *Professional, Community i Educational. Community* i *Educational* izdanja su projekti otvorenog koda i besplatni su, ali imaju manje značajki. U ovom završnom radu, koristit će se *Community* izdanje za povezivanje PLC-a, baze podataka MongoDB i internetskog sučelja koje će prikazivati određene podatke s PLC-a i slati podatke u PLC.

Pokretanje napisanog koda, moguće je pritiskom na zelenu strelicu u gornjem desnom kutu, nakon što se odabere aplikacija koja se želi pokrenuti. Nakon pokretanja, otvara se terminal u kojem se može komunicirati s napisanom aplikacijom. Zaustavljanje pokrenute aplikacije se jednostavno može učiniti pritiskom na crveni kvadrat u gornjem desnom kutu prozora.



Slika 3.8. Izgled sučelja i terminala u PyCharmu

3.7.MongoDB

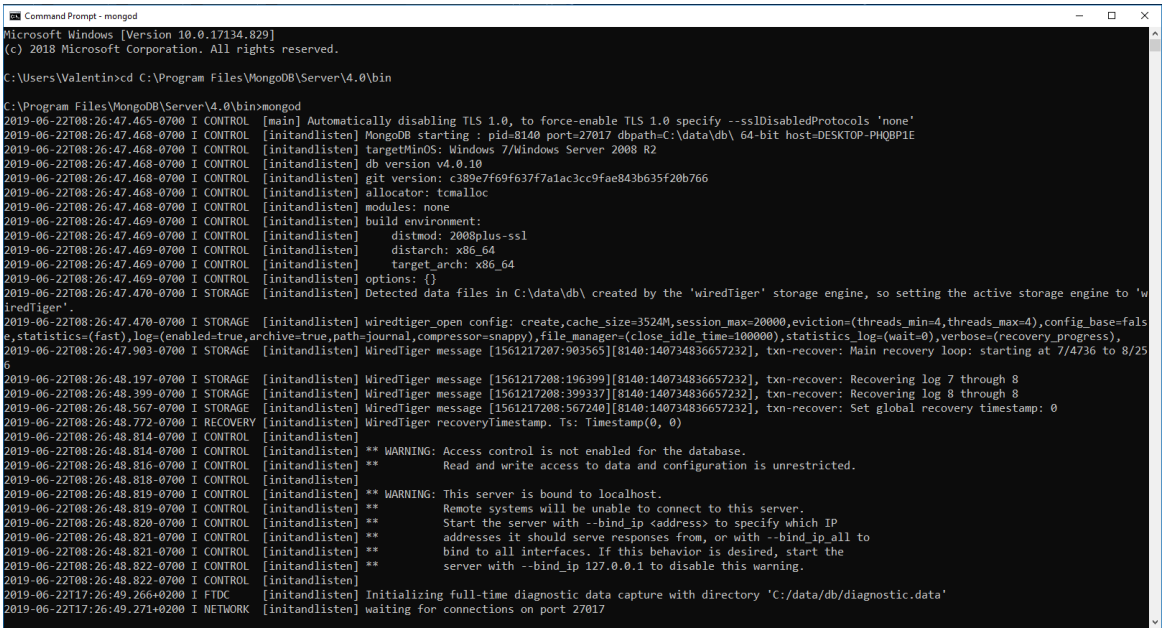
MongoDB je NoSQL baza podataka koja podatke sprema u obliku sličnom JSON objektu. Postoje dvije verzije baza podataka koju MongoDB nudi. To su:

- *Community* – besplatna verzija baze podataka,
- *Enterprise* – plaćena verzija koja nudi više značajki i opcija.

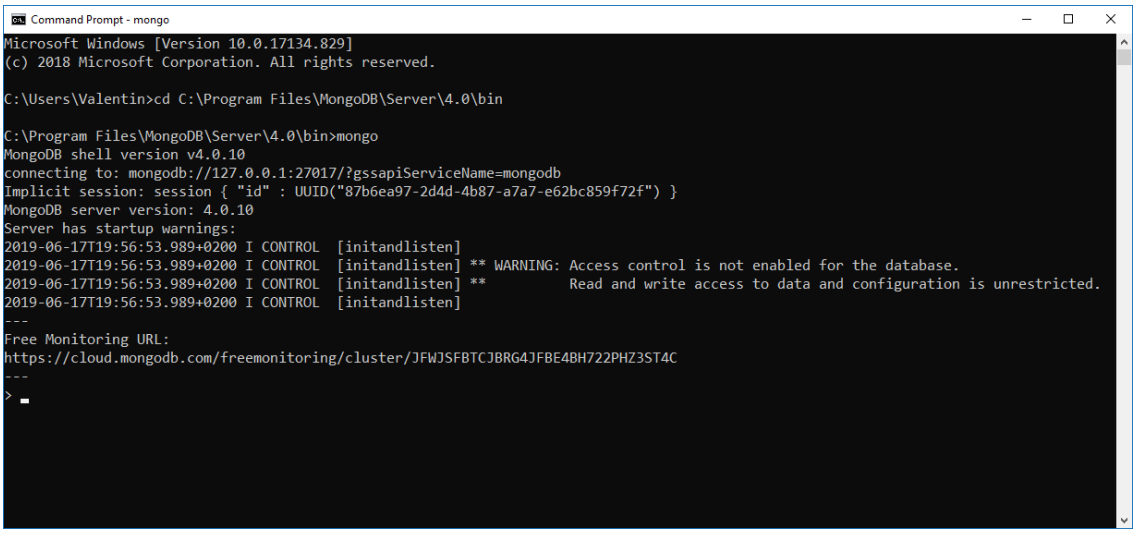
Zapis u bazu podataka je „dokument“, a on predstavlja strukturu podataka koja se sastoji od polja i vrijednosti tog polja. Primjer podatka spremljenog u MongoDB bazu podataka prikazan je slikom 3.9.

Slika 3.9. Struktura podatka spremljenog u MongoDB bazu podataka

MongoDB je potrebno pokrenuti iz dvije instance naredbenog retka. U prvu instancu naredbenog retka upisuje se „mongod“, što pokreće instancu MongoDB-a. U drugu instancu naredbenog retka upisuje se „mongo“, što pokreće MongoDB ljusku, odnosno tekstualno sučelje u kojemu je moguće izvršavati razne naredbe i upite. Pokrenuta instanca je prikazana na slici 3.10., a ljuska MongoDB-a na slici 3.11. Pokretanjem izvršne datoteke „mongo“, baza podataka je spremna za upisivanje i čitanje podataka koje će se odvijati preko aplikacije napisane u programskom jeziku Python.



Slika 3.10. Instanca MongoDB-a pokrenuta u Naredbenom retku



Slika 3.11. Ljuska MongoDB-a u Naredbenom retku

3.8. Maketa dizala

Ovaj završni rad izrađen je na maketi dizala koja se nalazi u Kampus zgradi fakulteta. Maketa se sastoji od postolja na kojemu se nalaze vertikalni stup i upravljačka ploča koja simulira upravljačku ploču u pravom dizalu. Postolje je drveno, a unutra se nalaze napajanje i sva potrebna elektronika kako bi maketa mogla pravilno raditi. Električne sheme makete dizala nalaze se u prilogu.

Slika 3.12. Maketa dizala

Slika 3.13. Vertikalni stup makete dizala

Vertikalni stup izrađen je dijelom od prozirne akrilne ploče, a dijelom od bijele plastike. Unutar akrilne ploče nalazi se kabina dizala. Maketa ima četiri kata, a svaki kat označavaju induktivni senzori (S1, S2, S3, S4). Također, na svakom katu se nalaze i vrata koje pokreću mikro servomotori (V1, V2, V3, V4). Vrata se mogu otvoriti na 5s, nakon čega se automatski zatvaraju. Svaka vrata imaju mikro prekidač koji označava kad su otvorena (otvara ih logička jedinica). Svi senzori vrata su serijski spojeni na jedan digitalni ulaz koji daje logičku jedinicu ako su sva vrata zatvorena. Pokraj vrata se nalaze i tipkala (T1, T2, T3, T4) s pripadajućom svjetlosnom diodom (D1, D2, D3, D4). Na vrhu vertikalnog stupa se nalazi 7-segmentni pokaznik koji može prikazivati brojeve od 0 do 9. Pokaznik je već kodiran pa je samo potrebno predati željeni broj u binarnom obliku na odgovarajuće digitalne izlaze (DP1, DP2, DP3, DP4). Izlaz DP1, u ovom slučaju predstavlja najmanje značajan bit, a izlaz DP4 predstavlja najznačajniji bit. Pokretanje je omogućeno pomoću jednog motora kojim je, pomoću izlaza MG i MD, moguće pokretati kabinu gore ili dolje.

Upravljačka ploča se također nalazi na postolju i sadrži tipkala (T11, T22, T33, T44) s pripadajućim svjetlosnim diodama za svaki kat i za alarm (TA, DA). Tipka i svjetlosna dioda za alarm, za razliku od ostalih koje su zelene, su crvene boje. Treba napomenuti kako su svjetlosne diode na stupu i upravljačkoj spojene zajedno. Stoga, slanjem logičke jedinice na D1, D2, D3 ili D4 pali svjetlosne diode na vertikalnom stupu i na upravljačkoj ploči za odgovarajući kat.

Slika 3.14. Upravljačka ploča makete dizala

4. IZRADA PROGRAMA I VIZUALIZACIJA

4.1. Programsko rješenje

Programsko rješenje zahtijeva tri načina rada dizala. To su: ručni, poluautomatski i automatski način rada. Zbog toga je kod podijeljen u više dijelova: svaki način rada zasebno, praćenje pozicije dizala, MODBUS, alarmi i prilagođavanje vrijednosti za vizualizaciju. Struktura koda je podijeljena na upite, dopuštenja, stanja i naredbe, što se može vidjeti po napravljenim DB-ovima i funkcijama. DB-ovi korišteni u kodu nalaze se u tablici 4.1.

Tablica 4.1. Prikaz svih podatkovnih blokova korištenih u kodu

Naziv podatkovnog bloka	Opis
I/O_DB	Podatkovni blok u kojemu se nalaze varijable ulaza i izlaza
REQUEST_DB	Podatkovni blok u kojemu se nalaze varijable upita
PERMISSIVE_DB	Podatkovni blok u kojemu se nalaze varijable dopuštenja
STATUS_DB	Podatkovni blok u kojemu se nalaze varijable stanja tipa *Bool*
PROCESS_MONITORING_DB	Podatkovni blok u kojemu se nalaze varijable stanja *Int, Word, LReal,* i *Time*
COMMAND_DB	Podatkovni blok u kojemu se nalaze varijable naredbi
HELPING_BITS_DB	Podatkovni blok u kojemu se nalaze pomoćne varijable
ALARM_DB	Podatkovni blok u kojemu se nalaze varijable stanja korištene isključivo za alarme
HMI_PROCESS_MONITORING_DB	Podatkovni blok u kojemu se nalaze varijable stanja tipa *Int* korištene za HMI
HMI_REQUEST_DB	Podatkovni blok u kojemu se nalaze varijable upita iz HMI-ja

Upiti (engl. Requests) su varijable u koje se pišu određeni ulazi, poput tipke na maketi dizala ili u HMI-ju. Dopuštenja (engl. Permissives) su varijable koje označavaju sve uvjete koji se moraju

ispuniti kako bi se određeno stanje moglo izvršiti. Stanje (engl. Status) je varijabla koja opisuje što će dizalo raditi ako se ispune svi uvjeti i ako je pritom došlo do upita. Stanja se vežu na varijable naredbi (engl. Commands) koje onda direktno idu na izlaze, kako bi dizalo moglo nešto učiniti. Također, napravljen je DB ulaza i izlaza kako bi se olakšalo pisanje koda, odnosno kako se ne bi moralo raditi s adresama PLC-a. Također, na adrese ulaza i izlaza su postavljene oznake zbog preglednosti koda.

I/O_TAG

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	COM_DI_T1	Bool	%I8.0	☐	☑	☑	☑	Push button floor 1
2	COM_DI_T2	Bool	%I8.1	☐	☑	☑	☑	Push button floor 2
3	COM_DI_T3	Bool	%I8.2	☐	☑	☑	☑	Push button floor 3
4	COM_DI_T4	Bool	%I8.3	☐	☑	☑	☑	Push button floor 4
5	COM_DI_T11	Bool	%I8.4	☐	☑	☑	☑	Push button floor 1 in elevator
6	COM_DI_T22	Bool	%I8.5	☐	☑	☑	☑	Push button floor 2 in elevator
7	COM_DI_T33	Bool	%I8.6	☐	☑	☑	☑	Push button floor 3 in elevator
8	COM_DI_T44	Bool	%I8.7	☐	☑	☑	☑	Push button floor 4 in elevator
9	COM_DI_TA	Bool	%I9.0	☐	☑	☑	☑	Push button alarm
10	COM_DI_SV	Bool	%I9.1	☐	☑	☑	☑	Sensor doors
11	COM_DI_S1	Bool	%I9.2	☐	☑	☑	☑	Sensor door on floor 1
12	COM_DI_S2	Bool	%I9.3	☐	☑	☑	☑	Sensor door on floor 2
13	COM_DI_S3	Bool	%I9.4	☐	☑	☑	☑	Sensor door on floor 3
14	COM_DI_S4	Bool	%I9.5	☐	☑	☑	☑	Sensor door on floor 4
15	COM_DO_MG	Bool	%Q8.0	☐	☑	☑	☑	Motor lift
16	COM_DO_MD	Bool	%Q8.1	☐	☑	☑	☑	Motor lower
17	COM_DO_DP1	Bool	%Q8.2	☐	☑	☑	☑	7-segment display bit 1
18	COM_DO_DP2	Bool	%Q8.3	☐	☑	☑	☑	7-segment display bit 2
19	COM_DO_DP3	Bool	%Q8.4	☐	☑	☑	☑	7-segment display bit 3
20	COM_DO_DP4	Bool	%Q8.5	☐	☑	☑	☑	7-segment display bit 4
21	COM_DO_DA	Bool	%Q8.6	☐	☑	☑	☑	LED alarm
22	COM_DO_D1	Bool	%Q9.0	☐	☑	☑	☑	LED floor 1
23	COM_DO_D2	Bool	%Q9.1	☐	☑	☑	☑	LED floor 2
24	COM_DO_D3	Bool	%Q9.2	☐	☑	☑	☑	LED floor 3
25	COM_DO_D4	Bool	%Q9.3	☐	☑	☑	☑	LED floor 4
26	COM_DO_V1	Bool	%Q9.4	☐	☑	☑	☑	Motor door floor 1
27	COM_DO_V2	Bool	%Q9.5	☐	☑	☑	☑	Motor door floor 2
28	COM_DO_V3	Bool	%Q9.6	☐	☑	☑	☑	Motor door floor 3
29	COM_DO_V4	Bool	%Q9.7	☐	☑	☑	☑	Motor door floor 4

Slika 4.1. PLC oznake za ulaze i izlaze

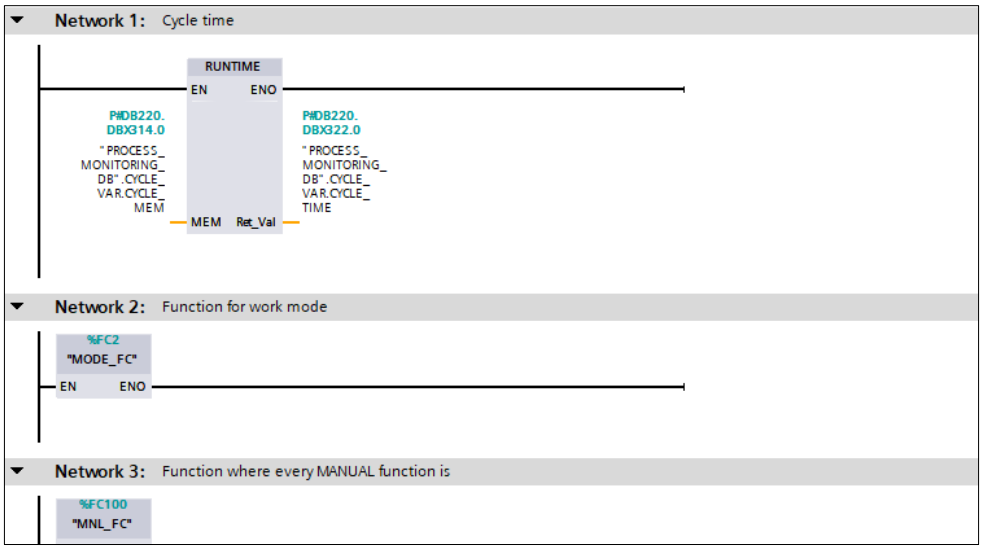
Budući da nije moguće vidjeti točno koja su vrata otvorena, već samo jesu li bilo koja vrata otvorena, svi *timeri* na varijablama stanja u kodu su postavljeni na 5s (ili 6s kad postoji odgoda otvaranja vrata u trajanju od 1s). U suprotnom, na HMI-ju ne bi bilo moguće pokazati kad su vrata otvorena. Na sam izlaz za dizanje dizala (MG) postavljen je uvjet da se dizalo ne može podizati ako dođe do četvrtog kata, odnosno do senzora S4. Slična stvar je napravljena i na izlazu za spuštanje dizala (MD) gdje se dizalo ne može više spuštati ako dođe do 1. kata.

4.1.1. Glavni blok

Glavni blok (engl. Main block) je organizacijski blok koji je korišten za pozivanje svih ostalih funkcija kako bi se one mogle izvršiti. U njemu su pozvane one funkcije koje predstavljaju veliku cjelinu koda. Funkcije pozvane u glavnom bloku mogu se vidjeti u tablici 4.2. Također, u glavnom bloku se mjeri vrijeme ciklusa koje služi za praćenje pozicije dizala što je i prikazano slikom 4.2.

Tablica 4.2. Prikaz svih funkcija koje se pozivaju u glavnom bloku

Ime funkcije	Opis
MODE_FC	Funkcija u kojoj se nalazi kod za upravljanje načinom rada dizala
MNL_FC	Funkcija koja sadrži sve funkcije vezane za ručni način rada
AUT_FC	Funkcija koja sadrži sve funkcije vezane za automatski način rada
SA_FC	Funkcija koja sadrži sve funkcije vezane za poluautomatski način rada
CMD_TO_OUT_MAP_FC	Funkcija koja povezuje naredbe svih triju načina rada s pripadajućim izlazom
I/O_MAP_FC	Funkcija u kojoj se izlazne i ulazne adrese pridružuju varijablama iz podatkovnog bloka „I/O_DB“
7-SEG_DISPLAY_FC	Funkcija u kojoj se upravlja 7-segmentnim pokaznikom
HMI_FC	Funkcija u kojoj se nalaze funkcije vezane za prilagođavanje varijabli za prikaz na HMI-ju
MB_FC	Funkcija koja sadrži sve funkcije vezane za MODBUS
ALARMS_FC	Funkcija u kojoj se nalaze svi alarmi
SEQ_CALIBRATION_FC	Funkcija u kojoj se nalazi kod za slijed naredbi kalibracije
FLR_MARK_FC	Funkcija u kojoj se nalaze sve funkcije potrebne za praćenje pozicije dizala

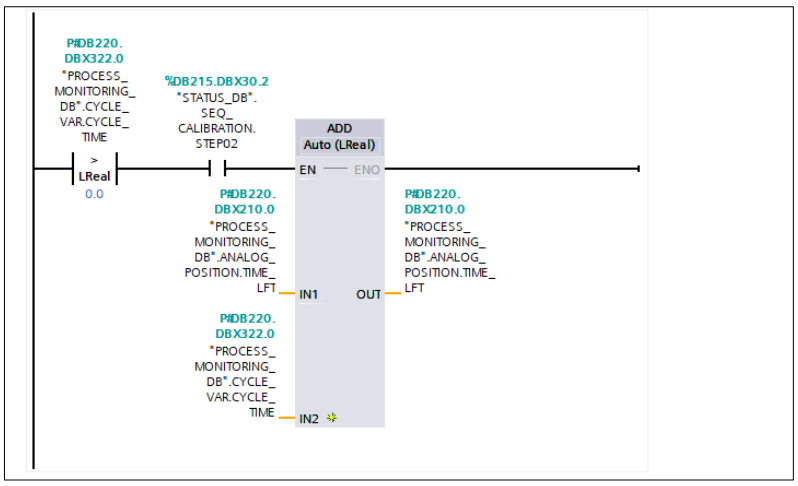


Slika 4.2. Dio glavnog organizacijskog bloka

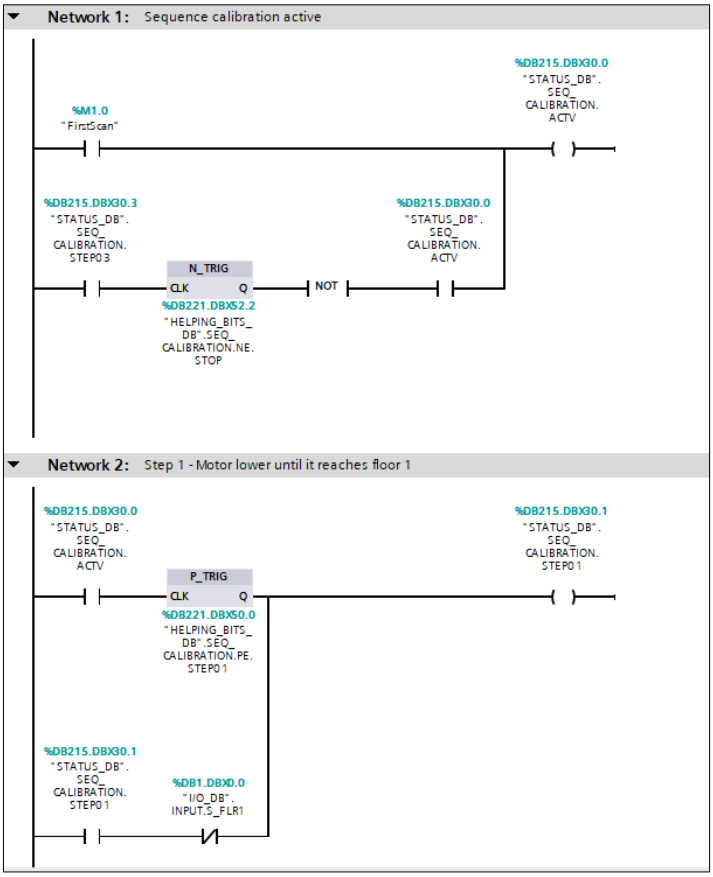
4.1.2. Praćenje pozicije dizala

Budući da su označavanje katova i pozicija dizala vrlo bitni dijelovi kod rada i razvijanja koda dizala, pozicija dizala je analogna, odnosno može biti bilo koja između 0 i izračunatog vremena u sekvenci kalibracije. Analognim praćenjem pozicije dobije se učinkovitije praćenje dizala jer je u svakom trenutku poznato gdje se nalazi dizalo. Također je moguće, u slučaju kvara senzora kata, stati na katu i otvoriti vrata jer se zna točna pozicija kata i točna trenutna pozicija dizala.

Sekvenca kalibracije je slijed naredbi u kojemu se dizalo u prvom koraku kreće prema prvom katu (slika 4.4.). Nakon što dotakne senzor na prvom katu, dizalo kreće prema četvrtom katu i svaki ciklus varijabli *TIME_LFT* dodaje varijablu *CYCLE_TIME* (vrijeme prošlog ciklusa) iz glavnog bloka (slika 4.3.). Krajnji rezultat u varijabli *TIME_LFT* je vrijeme potrebno dizalu da dođe od prvog do četvrtog kata. Budući da brzine dizanja i spuštanja nisu jednake, u zadnjem koraku je potrebno spustiti dizalo s četvrtog na prvi kat i izmjeriti vrijeme, odnosno opet dodavati varijablu *CYCLE_TIME* svaki ciklus u varijablu *TIME_LWR*. Vrijeme dizanja iznosi oko 11,5s, a spuštanja oko 10s. Upravo će izračunato vrijeme dizanja označavati i maksimalnu poziciju koju dizalo može postići.



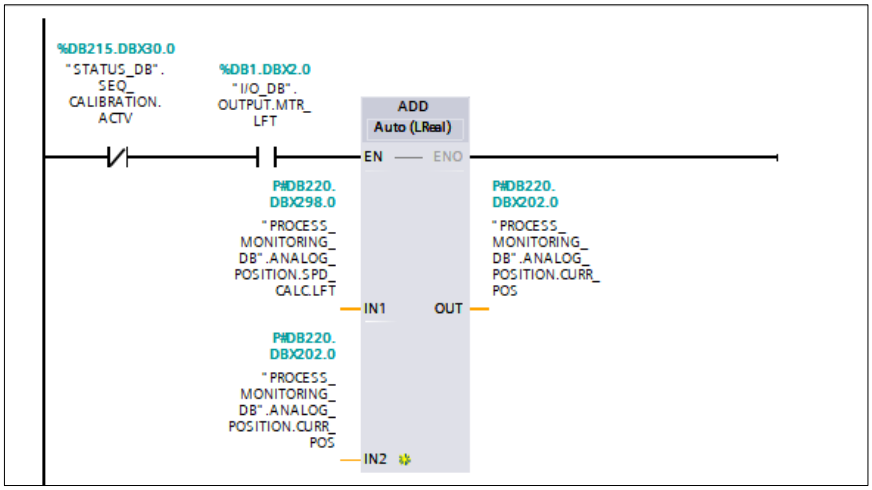
Slika 4.3. Mjerenje vremena koje je potrebno dizalu od prvog do četvrtog kata



Slika 4.4. Paljenje sekvence kalibracije i prvi korak

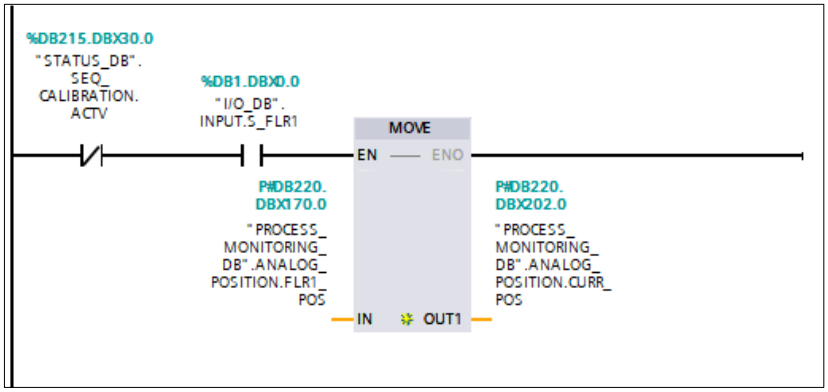
Katovi dizala su međusobno jednako udaljeni, stoga se i pozicije katova u kodu računaju tako da se za prvi kat uzima 0% maksimalne pozicije, za drugi kat 33.33%, za treći kat 66.67%, a za četvrti kat 100% maksimalne pozicije. Također, za svaki kat se izračunava i raspon pozicija za koje se uzima da je dizalo na tom katu. Taj raspon iznosi oko 0.1% iznad i 0.1% ispod izračunate pozicije za dani kat (osim za prvi kat, gdje je postavljena fiksna vrijednost). 0.2% razlike između maksimalne i minimalne pozicije kata je dovoljno da dizalo svaki put uđe u zadani raspon, a da pritom ne promaši puno senzor jer vrijeme ciklusa prosječno iznosi oko 0.0045s.

Što se tiče brzine dizala, ona nije jednaka za spuštanje i za dizanje jer ni izračunata vremena nisu jednaka. Brzina dizanja jednaka je vremenu ciklusa, odnosno varijabli *CYCLE_TIME*, a brzina spuštanja se izračunava na način da se najprije podijeli vrijeme dizanja s vremenom spuštanja kako bi se dobio koeficijent razlike vremena, a zatim se taj koeficijent množi s varijablom *CYCLE_TIME*. Kad se dizalo spušta, svaki ciklus se od trenutne brzine oduzima brzina spuštanja, a dodaje se brzina dizanja kad se kreće prema gore (slika 4.5.).



Slika 4.5. Dodavanje brzine dizanja na trenutnu poziciju pri kretanju dizala prema gore

Budući da ciklus nije uvijek isti pa može doći do greške, kao pomoć se koriste senzori katova. Kad se senzor kata okine, u trenutnu poziciju se sprema izračunata pozicija tog kata kao što je prikazano na slici 4.6. Dizalo će uvijek pokušati premašiti senzor za mali iznos kako bi se izbjegla situacija u kojoj, ako jedan senzor ne radi, dizalo može doseći prošli senzor na kojemu je bio. Na primjer, ako dizalo krene s drugog kata na treći, ono će premašiti senzor trećeg kata za mali iznos (ako senzor ne radi). Ako se opet pozove na drugi kat, dizalo će se vratiti točno na senzor drugog kata, a ako se pozove na četvrti kat, poziciju će ispraviti senzor četvrtog kata. Ovo je napravljeno iz razloga što brzine dizanja i spuštanja nisu jednake, a dizalo na maketi se ponekad sporije pokrene pa se pozicija ne izračuna dobro.



Slika 4.6. Spremanje izračunate pozicije kata u trenutnu poziciju pri okidanju senzora

4.1.3. Odabir načina rada

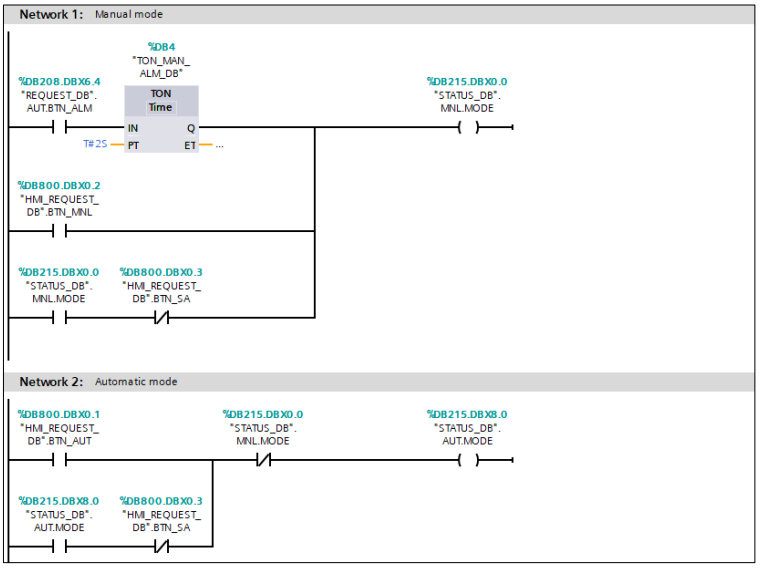
U funkciji „MODE_FC“ nalazi se kod za odabir načina dizala (ručni, poluautomatski i automatski način rada).

Poluautomatski način rada je način rada u kojemu dizalo radi većinu vremena, odnosno u kojemu se dizalo odaziva na upite s tipkala. Pri paljenju PLC-a, dizalo se već nalazi u poluautomatskom

načinu rada, a ako je u nekom drugom načinu rada, može ga se ponovno upaliti tipkom „SEMIAUTO“ u glavnom prozoru HMI-ja, što je prikazano na gornjem dijelu slike 4.7.

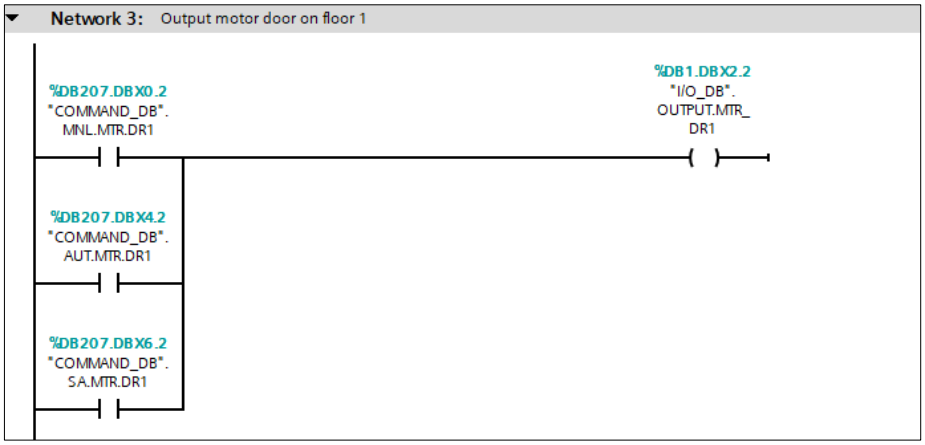
S druge strane, do ručnog načina rada se može doći na dva načina. Prvi način je držanjem tipke za alarm (TA) na maketi dizala 2s, a drugi način je pritiskom tipke „MANUAL“ na glavnom prozoru HMI-ja. Nakon toga, u idućem ciklusu, dolazi do samoodržanja koje ga sprječava da se ugasi. Isključivanje nije moguće tipkom alarma u dizalo, već samo tipkom „SEMIAUTO“ iz HMI-ja.

Automatski način rada se može upaliti pritiskom tipke „AUTOMATIC“ na glavnom prozoru HMI-ja nakon čega ulazi u samoodržanje u idućem ciklusu. Paljenje je moguće jedino iz poluautomatskog načina rada. Izlazak iz automatskog načina rada je moguć pritiskom tipke „SEMIAUTO“ na HMI-ju (donji dio slike 4.7).

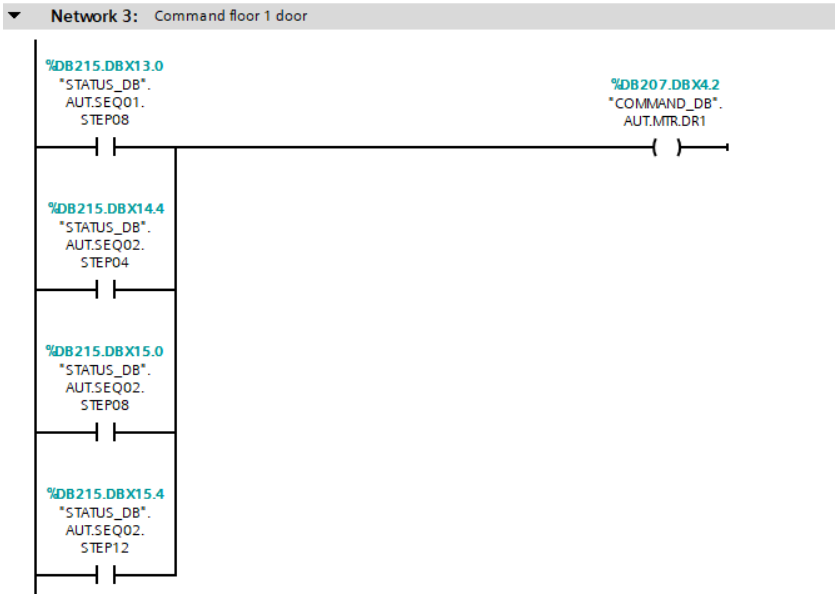


Slika 4.7. Prikaz paljenja ručnog i automatskog načina rada

Kako bi se kod mogao lakše pratiti, svaki način rada ima svoje varijable naredbi. Sve te naredbe na kraju završavaju na istom izlazu, a to je napravljeno na način da se pomoću više NO (engl. Normally open) kontakta radi „ILI“ sklop. Pomoću tog „ILI“ sklopa može se lako pratiti dolazi li neka naredba na određeni izlaz i iz kojeg dijela koda ona dolazi. To je prikazano na slici 4.8. Na sličan način su napravljene i varijable stanja koje završavaju na varijablama izlaza. Primjer u automatskom načinu rada, gdje određena stanja završavaju na varijabli naredbe za vrata prvog kata, dan je slikom 4.9.



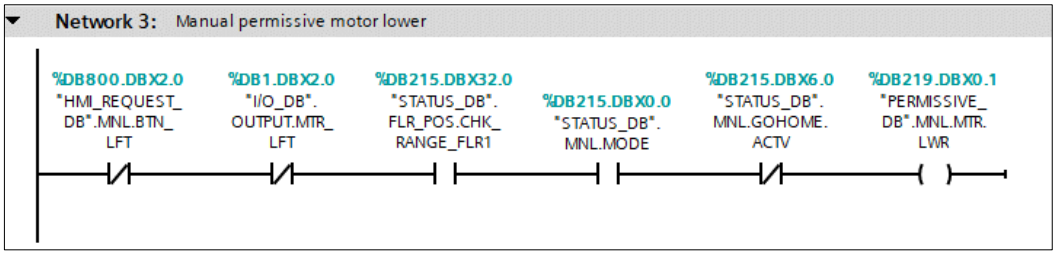
Slika 4.8. Prikaz spajanja varijabli naredbi s izlazom



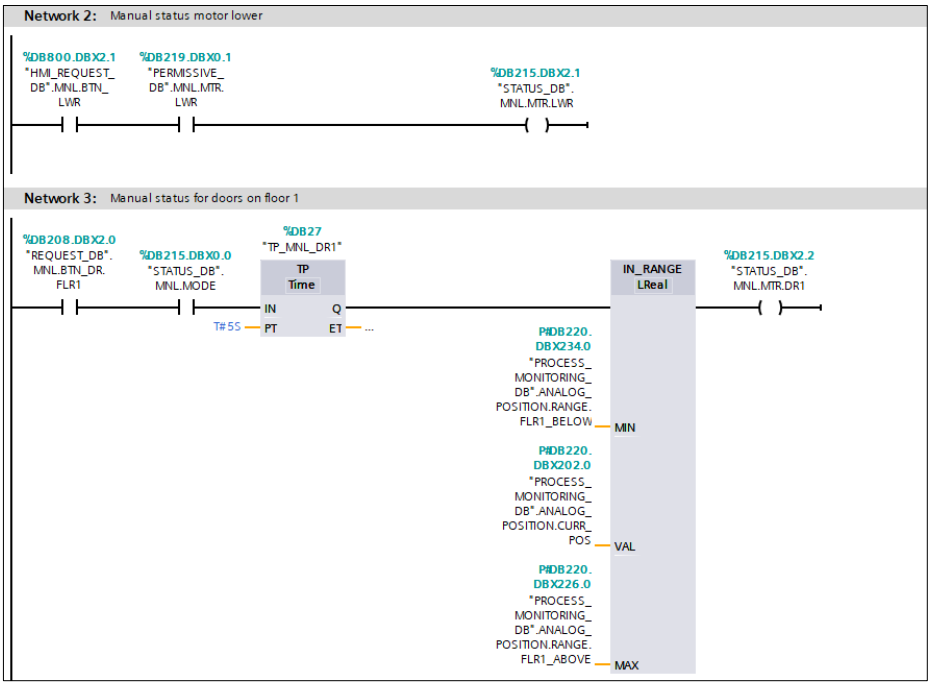
Slika 4.9. Prikaz spajanja varijabli stanja s pripadajućom varijablom naredbe

4.1.4. Ručni način rada

Ručni način rada (engl. Manual mode) je način rada koji je u ovom radu sličan *attendant service* načinu rada. *Attendant service* je način rada u kojemu dizalo ne radi više u poluautomatskom načinu rada, već sve operacije preuzima čovjek unutar dizala. U ovom slučaju, način rada je promijenjen jer dizalom nije moguće upravljati na maketi, već se dizalom upravlja preko HMI-ja. Radi na način da kada se upali, dizalo se spušta do prvog kata i otvara vrata na 5s, osim ako već nije na prvom katu, tad se samo vrata otvaraju. Nakon toga, držanjem tipki „Lift“ ili „Lower“ u HMI-ju je moguće dizati ili spuštati dizalo. Vrata se otvaraju pomoću tipki koje stoje na vertikalnom stupu dizala HMI-ja i mogu se otvoriti samo ako je dizalo na određenom katu (slika 4.11.). Nakon što se ručni način rada upali, svjetlosna dioda DA počne svijetliti. Svjetlosne diode D4 i D3 pale se i gase frekvencijom 1,25Hz kad se dizalo diže, odnosno spušta.



Slika 4.10. Uvjeti za spuštanje dizala u ručnom načinu rada

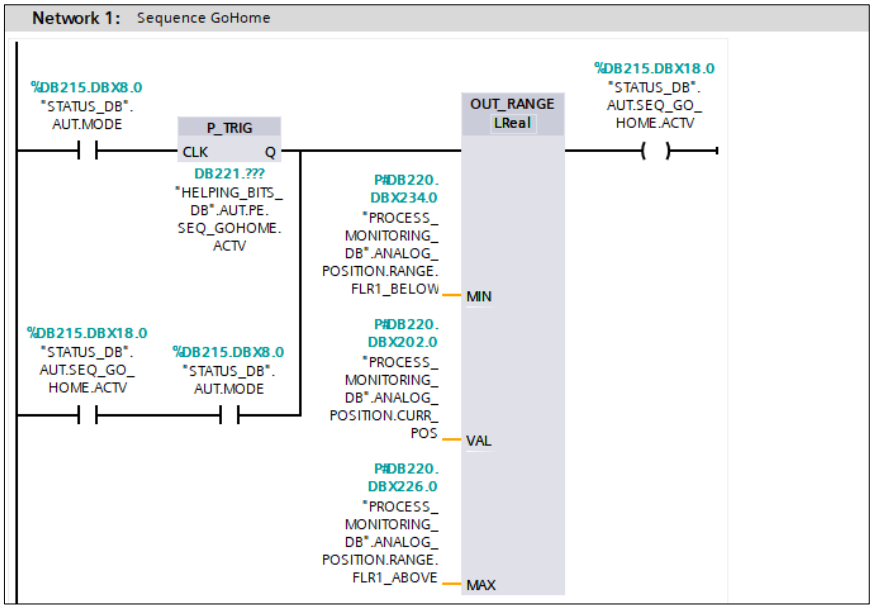


Slika 4.11. Stanja za spuštanje dizala i otvaranje vrata na prvom katu u ručnom načinu rada

4.1.5. Automatski način rada

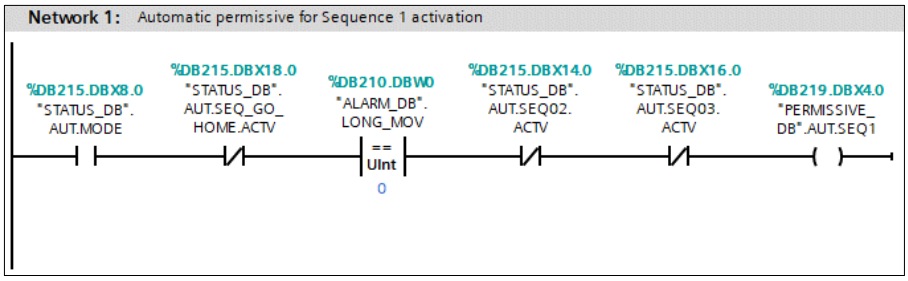
Automatski način rada (engl. Automatic mode) je način rada u kojemu dizalo obavlja slijed naredbi bez upita korisnika, osim kod pokretanja određenog slijeda naredbi. U ovom završnom radu napravljena su tri slijeda naredbi, s dodatnim slijedom koji se izvodi iz web aplikacije. Sva tri slijeda naredbi su slična, no ipak su dovoljno različiti da svaki od njih bude zasebni slijed. Budući da dizalo nema previše mogućnosti što se tiče kretanja i ostalih naredbi, puno koraka se ponavlja.

Prilikom pokretanja automatskog načina rada, slično kao i kod ručnog, dizalo se spušta na prvi kat, u slučaju da već nije (slika 4.12.). Pritom se ne otvaraju vrata, već je dizalo spremno za pokretanje jednog od sljedova naredbi. Dok je dizalo u stanju čekanja pokretanja slijeda naredbi, sve svjetlosne diode se pale i gase frekvencijom 1,25Hz.



Slika 4.12. "GoHome" slijed naredbi koji se sastoji od samo jednog koraka

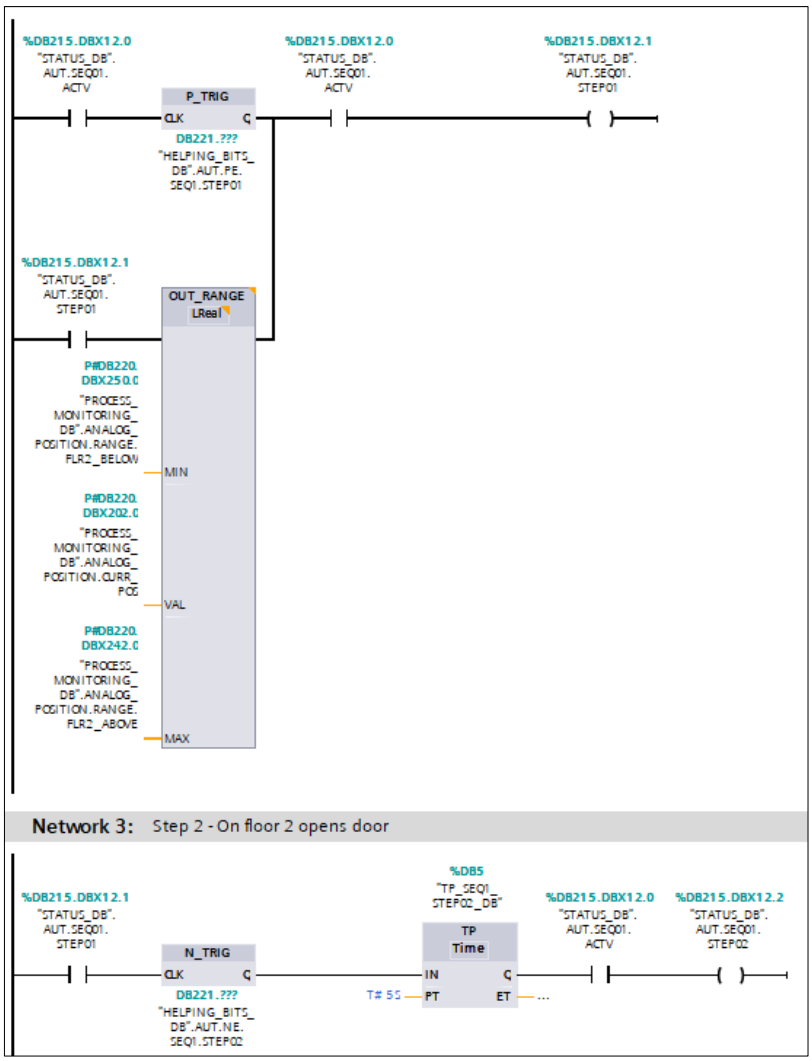
Prvi slijed naredbi pali se tipkom prvog kata na upravljačkoj ploči na HMI-ju, čiji su uvjeti prikazani na slici 4.13. Pri tome se gase svjetlosne diode D2, D3 i D4, a D1 počne trajno svijetliti.



Slika 4.13. Uvjeti za paljenje stanja aktivnosti prvog slijeda naredbi

Tek kad je dizalo na prvom katu i kad je poslan upit za pokretanjem slijeda naredbi, upalit će se stanje aktivnosti slijeda. U idućem ciklusu dolazi do samoodržanja gdje je bitno negiranje signala koji dolazi iz bloka padajućeg brida. Stanja aktivnosti i uvjeti ostalih sljedova naredbi su napravljeni na isti način. Prva dva koraka prikazana su na slici 4.14. Kad dizalo završi slijed u 8. koraku i taj se bit ugasi, blok za padajući brid daje logičku jedinicu na svom izlazu koja se invertira i gasi samoodržanje. U ovom slijedu naredbi, dizalo kreće s prvog kata i diže se do svakog kata, na kojem onda otvara vrata na 5s. Nakon što dođe do četvrtog kata i otvori vrata, dizalo se počinje spuštati do prvog kata. Kada dođe na prvi kat, otvaraju se vrata na 5s. Nakon što se zatvore vrata na prvom katu, zadnji korak završava i samoodržanje na varijabli stanja koje označava da je prvi

slijed naredbi aktivan se mijenja iz logičke jedinice u nulu. Time prvi slijed naredbi završava i sve svjetlosne diode se opet pale i gase.



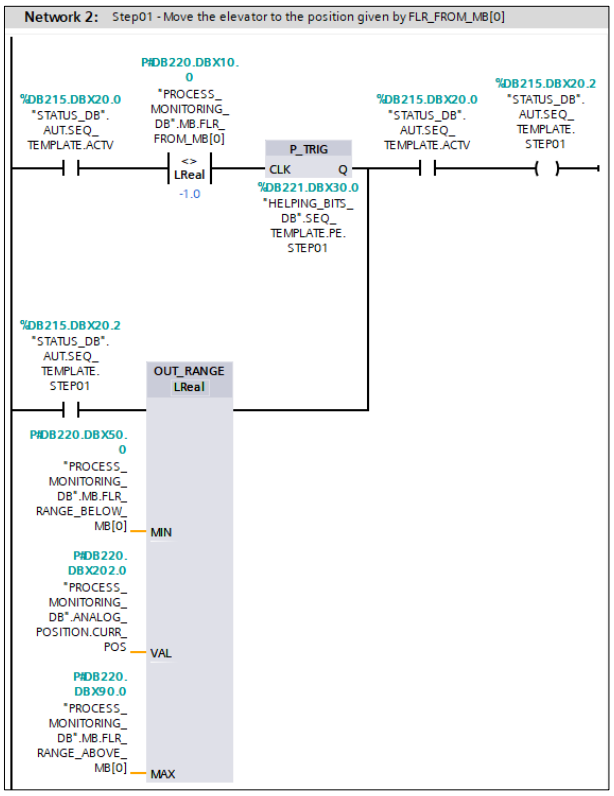
Slika 4.14. Prva dva koraka u prvom slijedu naredbi

Drugi slijed naredbi pali se tipkom drugog kata na upravljačkoj ploči na HMI-ju. Ovaj slijed naredbi je zapravo nadograđeni prvi slijed naredbi. Dodani su koraci u kojima se dizalo, nakon zatvaranja vrata na svakom katu, vraća svaki put na prvi kat i otvara vrata na prvom katu. Nakon što se dizalo spusti s četvrtog kata na prvi i nakon što otvori vrata, završava zadnji korak drugog slijeda naredbi. Time se, isto kao i u prvom slijedu, gasi stanje aktivnosti drugog slijeda naredbi.

Treći slijed naredbi je najjednostavniji i ima najmanje koraka. Pali se tipkom trećeg kata na upravljačkoj ploči na HMI-ju. Ima svega dva koraka u kojima se dizalo iz početne pozicije diže do četvrtog kata. Na četvrtom katu čeka 5s bez otvaranja vrata i nakon toga se vraća na prvi kat,

gdje također ne otvara vrata. Dolaskom na prvi kat, stanje aktivnosti ovog slijeda naredbi se gasi, čime završava i treći slijed naredbi.

Zadnjim slijedom naredbi se upravlja iz web aplikacije i on je napravljen kao predložak koji može imati različite katove i vremena stajanja. Nakon upisa i slanja željenih katova u odgovarajuće registre MODBUS-a, provjerava se je li barem jedan upisani registar drukčiji od ranije zapisanih. Ako jest, bit za paljenje sekvence se okida i pokreće sekvencu. U web aplikaciji je moguće upisati željene katove i vrijeme stajanja na svakom katu, oba kao višeznamenkasti broj. Moguće je upisati najviše 5 katova i 5 vremena stajanja. Primjer upisa katova i stajanja je prikazan na slici 4.34. Budući da svaki kat ima svoj raspon vrijednosti, dizalo će se kretati dok ne bude u tom rasponu. Znat će u kojem smjeru treba ići po tome je li trenutna pozicija veća ili manja od željene. Stajanje na svakom katu (u sekundama) upisuje se u web aplikaciji. Nakon što se pokupe s MODBUS-a, te vrijednosti se pretvaraju u varijable tipa *TIME* i predaju se *timeru*. Prvi korak predloška slijeda naredbi je prikazan slikom 4.15.



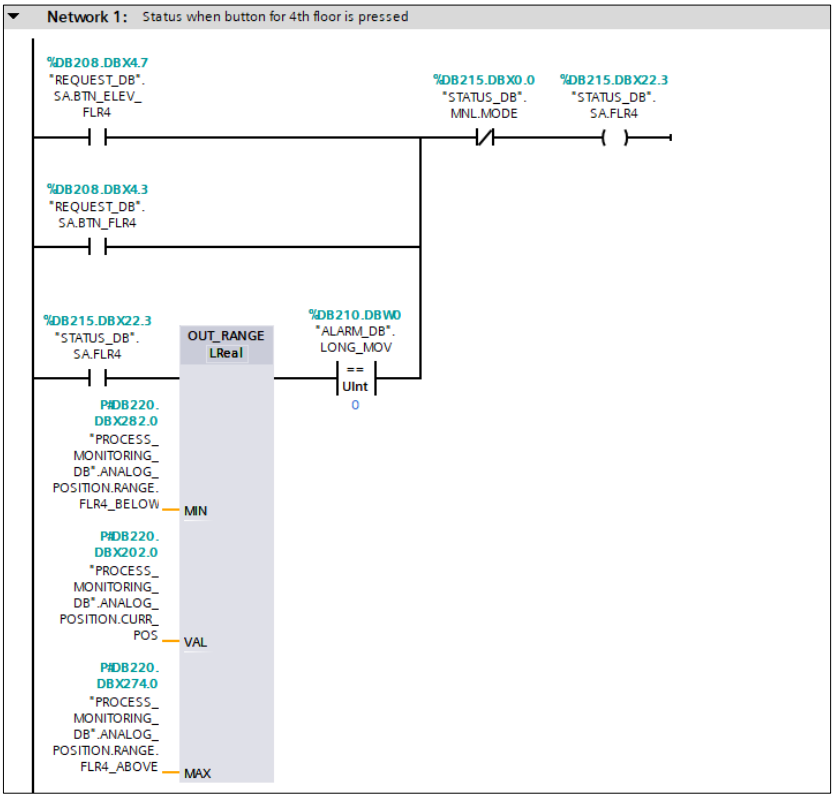
Slika 4.15. Prvi korak slijeda naredbi iz web aplikacije

4.1.6. Poluautomatski način rada

Poluautomatski način rada je način rada u kojemu dizalo provede najviše vremena. To je način u kojemu dizalo odgovara na upite korisnika na svim katovima. Postoje mnogi algoritmi koji se bave dizalima, dizajnirani su za jedno ili više njih u zgradi i koriste puno parametara.

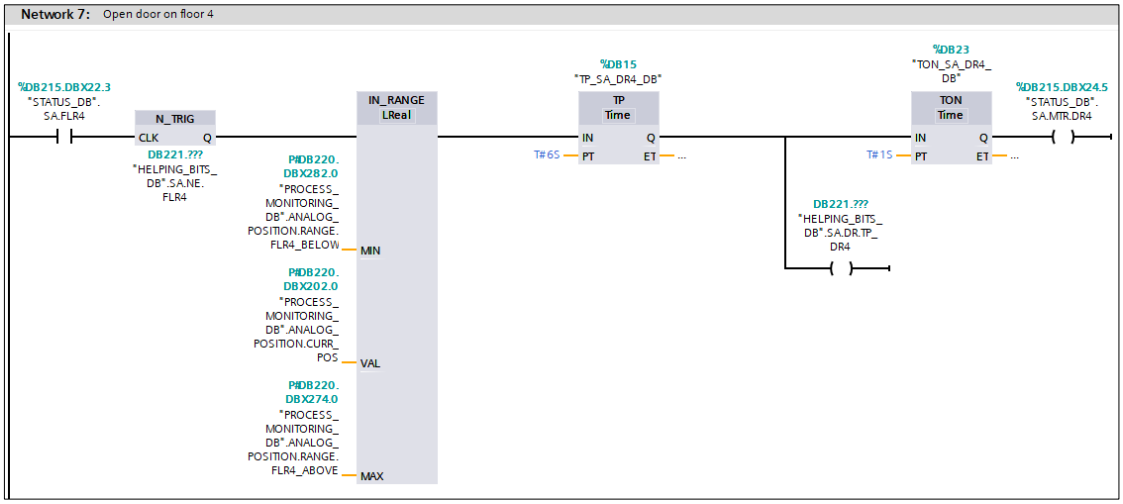
Algoritam upotrijebljen u ovom završnom radu je prilagođena verzija Karpovog algoritma za dizala. Karp se bavio sortiranjem zapisa na magnetnoj traci i njegov se algoritam često koristi kod čvrstih diskova. Algoritam je vrlo složen, ali u osnovi dizalo ide u jednom smjeru sve dok ne ispuni sve upite. Nakon što ispuni sve upite u jednom smjeru, ostaje mirovati ili se počinje kretati u drugom smjeru. Na primjer, ako je dizalo na prvom katu i dobije upit s trećeg i četvrtog kata, počinje se kretati gore. Nakon što obavi upit na trećem katu, kreće na četvrti kat. Ako netko na trećem katu pritisne tipku za prvi ili drugi kat, dizalo će prvo riješiti upit na četvrtom katu jer je krenulo u tom smjeru i tek onda se vratiti na drugi, a zatim i prvi kat.

Tipke za pozivanje katova na upravljačkoj ploči, tipka pokraj vrata na maketi dizala i odgovarajuće tipke na HMI-ju se promatraju na isti način, odnosno nijedan skup tipki nema prednost nad drugim. Pritiskom tipke za određeni kat, šalje se upit dizalu da dođe na taj kat. Svjetlosna dioda za taj kat svijetli sve dok dizalo ne dođe do njega.



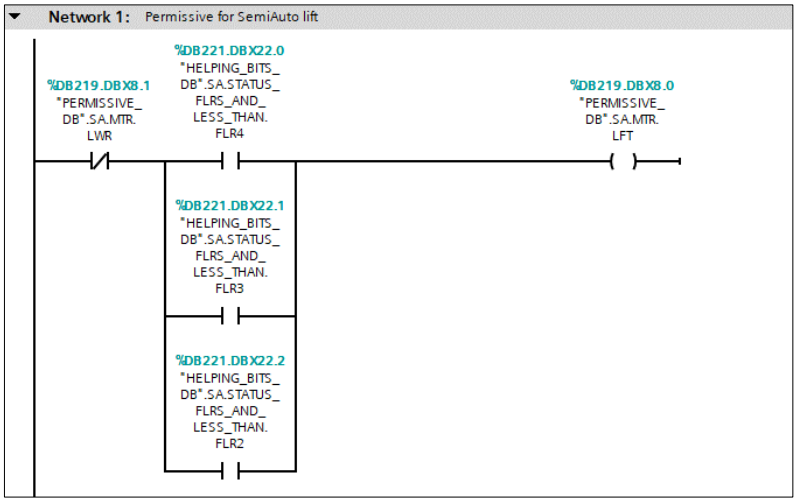
Slika 4.16. Prikaz stanja za četvrti kat u poluautomatskom načinu rada

Nakon što dizalo stane na katu, postoji zadrška od 1s prije nego što se vrata otvore. Vrata se otvaraju na 5s i nakon što se zatvore vrata, dizalo kreće na sljedeći upit. Zbog zadrške od 1s, stvoreni su novi pomoćni bitovi koji sprječavaju dizalo da samo produži na idući upit bez otvaranja vrata.



Slika 4.17. Prikaz stanja za otvaranje vrata na četvrtom katu u poluautomatskom načinu rada

Kretanje dizala riješeno je na način da ako dizalo krene u jednom smjeru, kretanje u drugom smjeru je odmah onemogućeno postavljenim uvjetom. Dizalo prati u kojem smjeru mora ići pomoću trenutne pozicije i pozicije određenog kata. Na primjer, ako je aktivno stanje na četvrtom katu (slika 4.16.), a pozicija dizala je manja od izračunate pozicije četvrtog kata, dizalo se kreće gore sve dok trenutna pozicije ne bude u rasponu vrijednosti za četvrti kat. Tad se gasi stanje na četvrtom katu i otvaraju se vrata. Slično je i s kretanjem prema dolje, samo se provjerava je li dizalo na poziciji većoj od pozicije kata i je li stanje na tom katu aktivno.



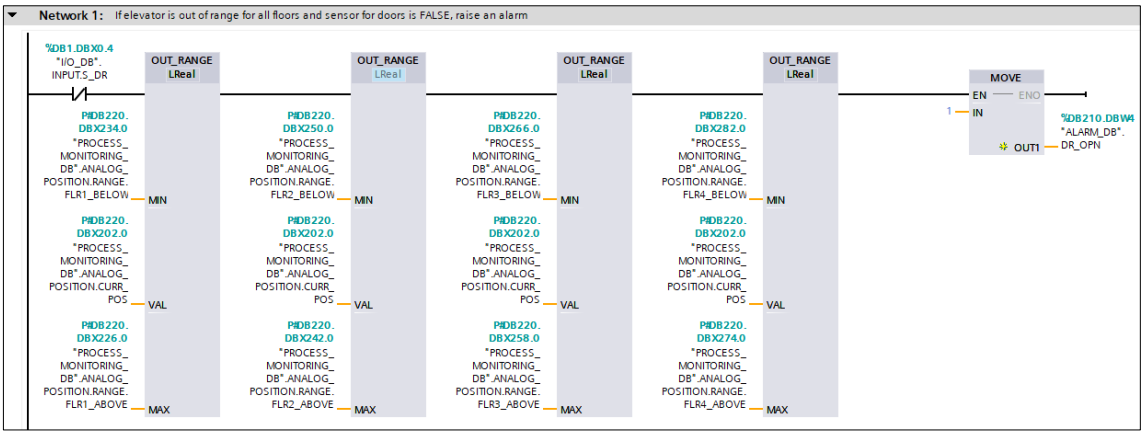
Slika 4.18. Uvjeti za kretanje dizala prema gore u poluautomatskom načinu rada

4.1.7. Alarmi

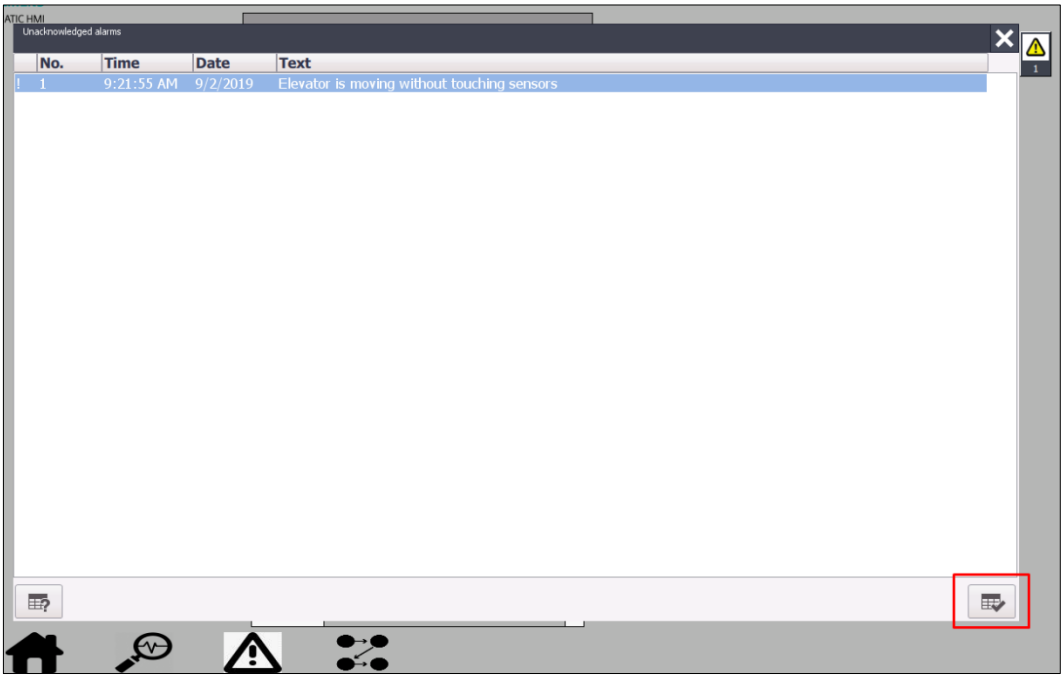
Alarm je dio koda koji se bavi greškama i neočekivanim radnjama u radu dizala. U ovom završnom radu, izrađena su dva alarma i dva upozorenja. Razlika između samog alarma i upozorenja je da alarm signalizira grešku u radu dizala i sprječava daljnji rad dok se problem ne otkloni ili se ne

potvrdi u HMI-ju, a upozorenje je obavijest koja prikazuje neočekivanu radnju u normalnom radu dizala i ne utječe na rad dizala.

Prvi alarm se podiže ako su vrata otvorena, a dizalo nije ni na jednom katu. U tom slučaju, kretanje dizala je onemogućeno sve dok se vrata ne zatvore i alarm se ne potvrdi u HMI-ju. Potreban kod za alarm se nalazi na slici 4.19., a prikaz potvrđivanja alarma (označeno crvenim pravokutnikom) se nalazi na slici 4.21.



Slika 4.19. Potreban kod za podizanje alarma vrata



Slika 4.20. Prikaz potvrđivanja alarma u HMI-ju

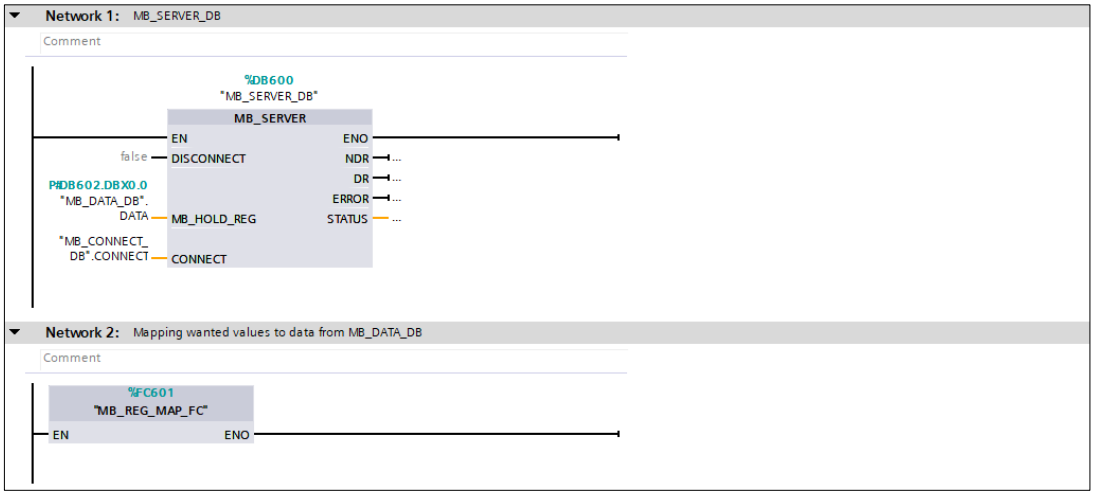
Drugi alarm se podiže ako se dizalo predugo kreće u bilo kojem smjeru. Budući da dizalu treba oko 7s da dođe od prvog do trećeg kata (isto tako od drugog do četvrtog) i obrnuto, ono se mora

kretati 8s da se alarm podigne. U tom slučaju, svaka kretnja je obustavljena dok se alarm ne potvrdi.

Prvo upozorenje se podiže kad se upali ručni način rada jer je njega moguće upaliti iz dizala. Drugo upozorenje se podiže u slučaju da se slijed naredbi prekinuo prije izvršavanja zadnjeg koraka u automatskom načinu rada. Oba upozorenja nije potrebno potvrditi, već samo spustiti podignuti prozor.

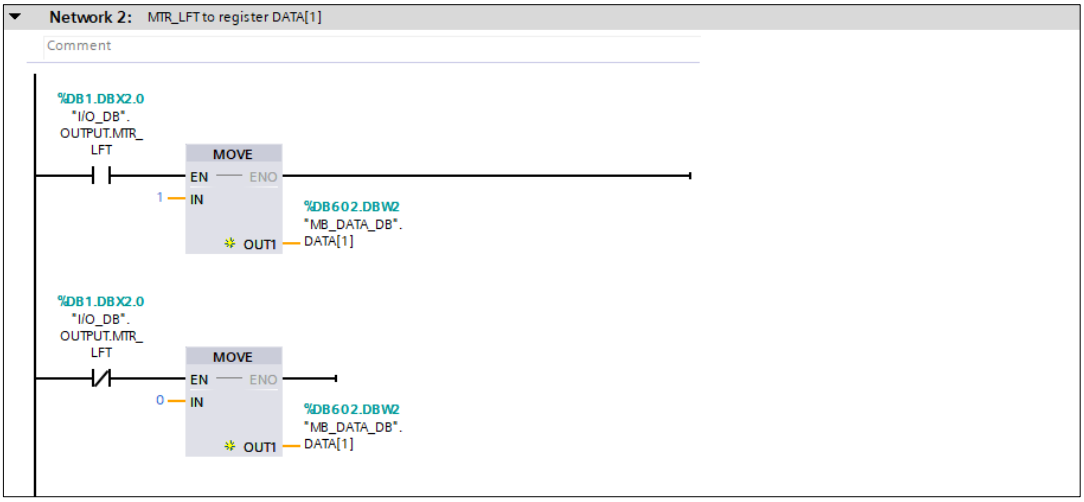
4.1.8. MODBUS server

Kako bi se mogla uspostaviti komunikacija između Python aplikacije i PLC-a, potreban je MODBUS server. MODBUS server se u TIA Portalu može dobiti pomoću bloka MB_SERVER (slika 4.21.). MB_SERVER se u komunikaciji ponaša kao MODBUS TCP server i zahtijeva PROFINET vezu za pravilan rad. Kao što je već ranije opisano, MB_SERVER prima i obrađuje upite klijenta, a zatim šalje odgovor. U ovom radu korištena je verzija ugrađene programske podrške 4.2 kako bi se mogle koristiti naredbe svih verzija biblioteka. Nakon postavljanja bloka u funkciji, potrebno je odrediti zaseban podatkovni blok. Blok ima nekoliko ulaznih, ulazno/izlaznih i izlaznih parametara, ali su korišteni samo *DISCONNECT, MB_HOLD_REG* i *CONNECT*. Parametar DISCONNECT se koristi za ulazak u pasivnu vezu s partnerskim modulom. Parametar može primiti vrijednosti logičke jedinice i logičke nule, no odabrana je logička nula jer se time postiže pasivna veza kada nema komunikacije. MB_HOLD_REG je parametar koji služi kao pokazivač na međuspremnik za spremanje podataka koji se čitaju ili pišu u MODBUS server. Za ovaj parametar stvoren je novi podatkovni blok koji sadrži niz od 31 podatka (od 0 do 30) tipa *WORD*. Parametar *CONNECTION* služi za strukturu opisa veze. U radu je korištena struktura „TON_IP_v4“ u kojoj su se postavili ID, IP adresa partnerskog uređaja (postavljena je adresa 0.0.0.0 kako bi se mogao povezati bilo koji uređaj) i priključak na kojem se nalazi partnerski uređaj.



Slika 4.21. Prikaz bloka MB_SERVER i mapiranja podataka u MB_DATA_DB

Budući da se za prijenos podataka koristi međuspremnik, željene bitove potrebno je pretvoriti u tip podatka WORD. To se napravilo na način da ako je bit u stanju logičke jedinice, u registar se sprema broj 1, a ako je u stanju logičke nule, u registar se sprema broj 0. Za podatke koji već jesu tipa WORD, blokom MOVE se jednostavno sprema broj u željeni registar. Podaci koji se šalju i spremaju u registre su: broj kata, naredbe za kretanje gore i dolje i naredbe za otvaranje vrata. Ovi podaci zauzimaju prvih 7 registara (od 0 do 6) međuspremnika.



Slika 4.22. Prikaz spremanja podataka u registre međuspremnika

Također, preko MODBUS-a je moguće i primati podatke. U ovom radu, primljeni podaci služe kao katovi i vremena stajanja za predložak slijeda naredbi opisan u automatskom načinu rada. Podaci se zapisuju u registre od 10 do 19 (od 10 do 14 za katove i od 15 do 19 za vremena stajanja na katu). Nakon čitanja se pretvaraju u raspone vrijednosti za svaki kat i milisekunde za stajanje.

4.2. Vizualizacija

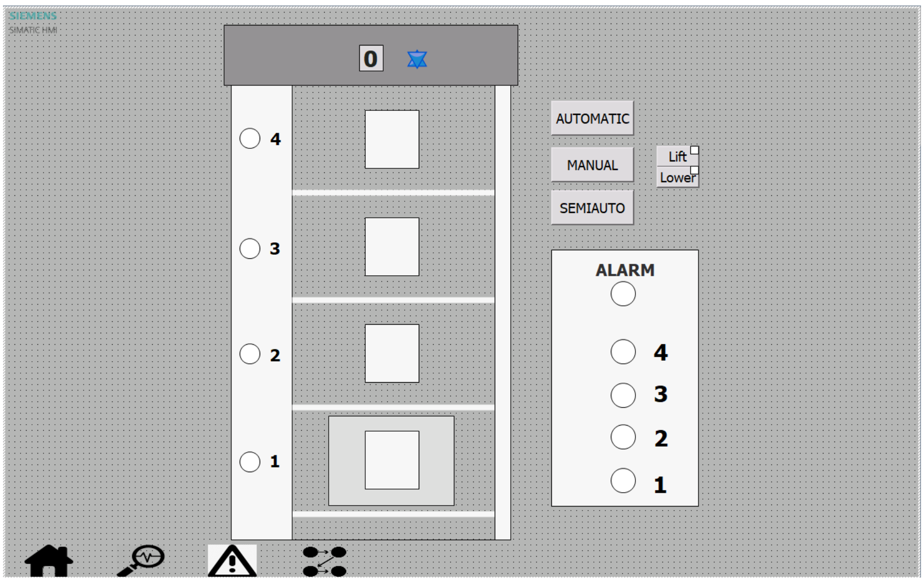
Vizualizacija dizala odrađena je u programu WinCC koji se nalazi u sklopu razvojnog okruženja TIA Portal. Korišten je KTP1200 Basic PN panel gdje su, uz glavni ekran, napravljena još tri dodatna ekrana za prikazivanje određenih informacija. Budući da nema pristupa fizičkom panelu na koji bi se mogao preuzeti HMI, on će se koristiti samo u simulaciji. Vrlo bitan dio kod izrade HMI-ja su HMI oznake. One povezuju sve oznake koje se koriste pri izradi sučelja i povezuju se s varijablama iz DB-ova, a dio njih je prikazan slikom 4.23.

HMI tags

Name	Tag table	Data type	Connection	PLC name	PLC tag	...	Access mode	Acquisition cycle	Source comment
CHK_STATUS_AND_GREATER_T...	HELPING_BITS_TAG	Bool	HMI_Connectio...	PLC_1	HELPING_BITS_DB.SA.ST...		<symbolic access>	100 ms	Bit active when there is a status fo
CHK_STATUS_AND_GREATER_T...	HELPING_BITS_TAG	Bool	HMI_Connectio...	PLC_1	HELPING_BITS_DB.SA.ST...		<symbolic access>	100 ms	Bit active when there is a status fo
CHK_STATUS_AND_GREATER_T...	HELPING_BITS_TAG	Bool	HMI_Connectio...	PLC_1	HELPING_BITS_DB.SA.ST...		<symbolic access>	100 ms	Bit active when there is a status fo
CHK_STATUS_AND_LESS_THAN...	HELPING_BITS_TAG	Bool	HMI_Connectio...	PLC_1	HELPING_BITS_DB.SA.ST...		<symbolic access>	100 ms	Bit active when there is a status fo
CHK_STATUS_AND_LESS_THAN...	HELPING_BITS_TAG	Bool	HMI_Connectio...	PLC_1	HELPING_BITS_DB.SA.ST...		<symbolic access>	100 ms	Bit active when there is a status fo
CHK_STATUS_AND_LESS_THAN...	HELPING_BITS_TAG	Bool	HMI_Connectio...	PLC_1	HELPING_BITS_DB.SA.ST...		<symbolic access>	100 ms	Bit active when there is a status fo
DR_OPN	ALARM_TAG	UInt	HMI_Connectio...	PLC_1	ALARM_DB.DR_OPN		<symbolic access>	100 ms	Doors opened when the elevator i
DR_OPN_ACK	ALARM_TAG	Int	HMI_Connectio...	PLC_1	ALARM_DB.ACKNOWLED...		<symbolic access>	100 ms	Doors open acknowledgement
HMI_ANALOG_POS	PROCESS_MONITORING_TAG	LReal	HMI_Connectio...	PLC_1	PROCESS_MONITORING_...		<symbolic access>	100 ms	Current analog position
HMI_AUT_MODE	STATUS_TAG	Bool	HMI_Connectio...	PLC_1	STATUS_DB.AUT.MODE		<symbolic access>	100 ms	Automatic alarm mode status
HMI_AUT_SEQ_GOHOME_ACTV	STATUS_TAG	Bool	HMI_Connectio...	PLC_1	STATUS_DB.AUT.SEQ_GO...		<symbolic access>	100 ms	Sequence GoHome status active
HMI_BTN_ALM	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_A...		<symbolic access>	100 ms	HMI button alarm
HMI_BTN_AUT	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_A...		<symbolic access>	100 ms	HMI button automatic
HMI_BTN_ELEV_FLR1	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_E...		<symbolic access>	100 ms	HMI button in Elevator floor 1
HMI_BTN_ELEV_FLR2	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_E...		<symbolic access>	100 ms	HMI button in Elevator floor 2
HMI_BTN_ELEV_FLR3	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_E...		<symbolic access>	100 ms	HMI button in Elevator floor 3
HMI_BTN_ELEV_FLR4	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_E...		<symbolic access>	100 ms	HMI button in Elevator floor 4
HMI_BTN_FLR1	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_F...		<symbolic access>	100 ms	HMI button floor 1
HMI_BTN_FLR2	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_F...		<symbolic access>	100 ms	HMI button floor 2
HMI_BTN_FLR3	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_F...		<symbolic access>	100 ms	HMI button floor 3
HMI_BTN_FLR4	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_F...		<symbolic access>	100 ms	HMI button floor 4
HMI_BTN_MNL	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_...		<symbolic access>	100 ms	HMI button manual
HMI_BTN_SA	REQUESTS_TAG	Bool	HMI_Connectio...	PLC_1	HMI_REQUEST_DB.BTN_SA		<symbolic access>	100 ms	HMI button manual OFF
HMI_CHK_RANGE_FLR1	STATUS_TAG	Bool	HMI_Connectio...	PLC_1	STATUS_DB.FLR_POS.CH...		<symbolic access>	100 ms	Check range for floor 1
HMI_CHK_RANGE_FLR4	STATUS_TAG	Bool	HMI_Connectio...	PLC_1	STATUS_DB.FLR_POS.CH...		<symbolic access>	100 ms	Check range for floor 4
HMI_DR1	OUTPUT_TAG	Bool	HMI_Connectio...	PLC_1	"I/O_DB".OUTPUT.MTR_DR1		<symbolic access>	100 ms	Motor door floor 1
HMI_DR2	OUTPUT_TAG	Bool	HMI_Connectio...	PLC_1	"I/O_DB".OUTPUT.MTR_DR2		<symbolic access>	100 ms	Motor door floor 2
HMI_DR3	OUTPUT_TAG	Bool	HMI_Connectio...	PLC_1	"I/O_DB".OUTPUT.MTR_DR3		<symbolic access>	100 ms	Motor door floor 3

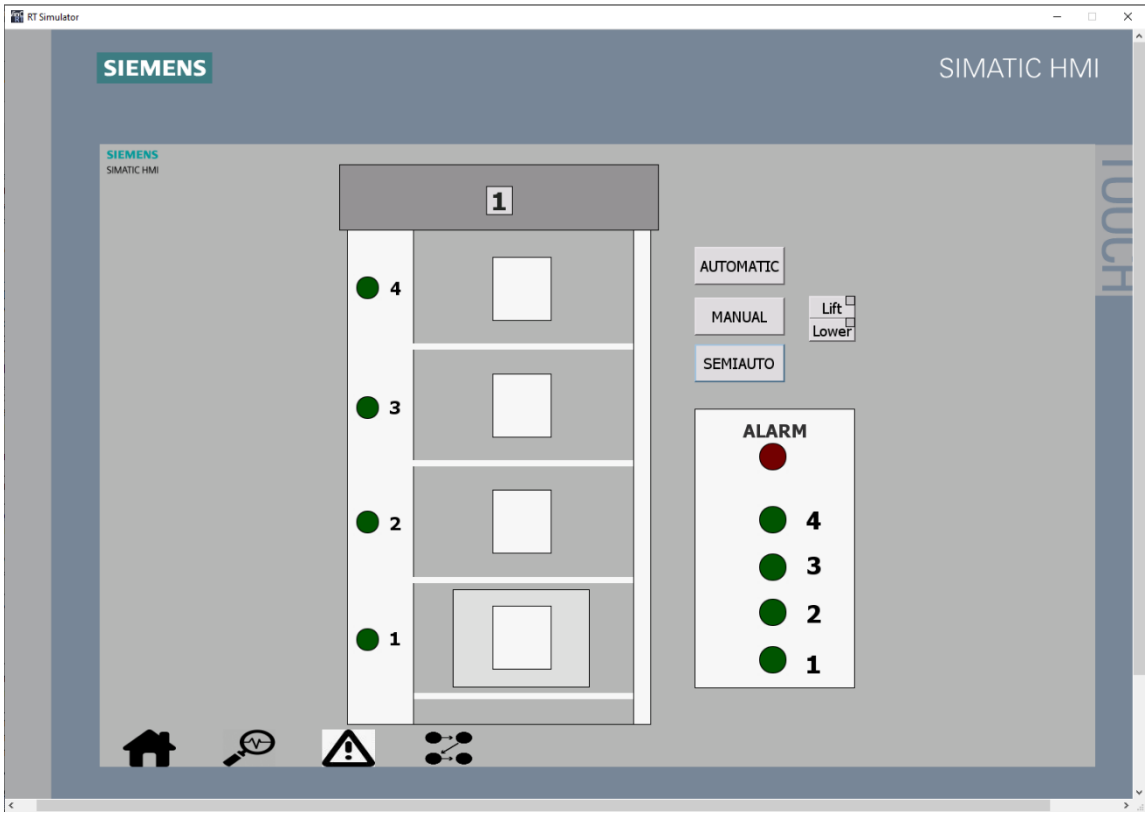
Slika 4.23. HMI oznake

Pomoću tih HMI oznaka, izrađene su animacije i događaji gdje elementi najčešće mijenjaju boju, vidljivost ili poziciju.



Slika 4.24. Glavni ekran u vizualizaciji

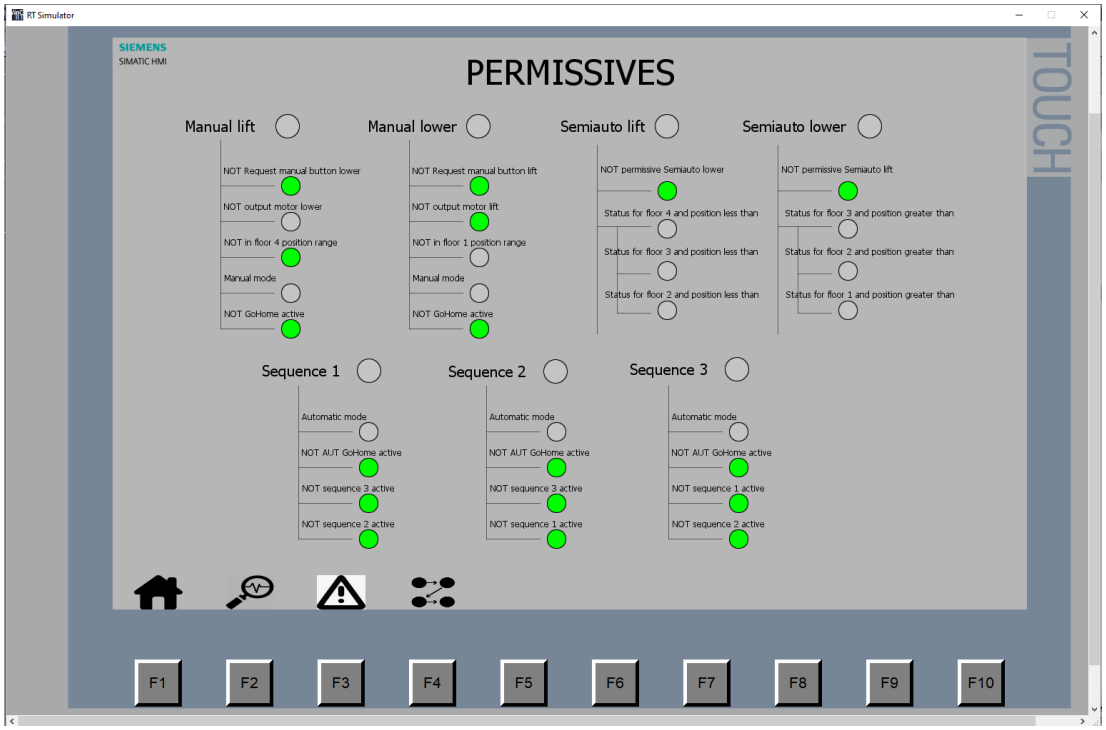
Budući da u korištenoj verziji WinCC-a ne postoji tipka okruglog oblika, sve napravljene okrugle tipke su spoj običnog kruga ispred kojeg se nalazi potpuno transparentna tipka četvrtastog oblika. Sve okrugle tipke su tamnozelene boje dok je izlaz na tu svjetlosnu diodu na maketi u stanju logičke nule, a svijetlozelene dok je u stanju logičke jedinice. Izuzetak od toga je tipka za alarm, koja izmjenjuje tamnocrvenu i svijetlocrvenu boju. Pravokutnik koji predstavlja kabinu dizala ima animaciju pokretanja u kojoj povezana varijabla ima raspon od 0 do 453 iz razloga što je u vizualizaciji udaljenost između prvog i četvrtog kata 453 piksela. Vrijednosti te varijable su skalirane u kodu kako bi vizualizacija mogla pratiti poziciju dizala na maketi. Pravokutnici koji predstavljaju vrata imaju animaciju gdje mijenjaju boju u tamnosivu kad se otvore vrata na maketi. Također, dodan je i element koji predstavlja 7-segmentni pokaznik i mijenja vrijednost ovisno o katu. Kraj pokaznika se nalaze dvije strelice koje se naizgled preklapaju, ali dok je jedna vidljiva, druga nije. Te strelice prikazuju kreće li se dizalo gore ili dolje. Ako se dizalo ne kreće, ne vidi se nijedna strelica. Iznad upravljačke ploče dodane su i tipke za paljenje automatskog, ručnog i poluautomatskog načina rada. Uz tipku za ručni način rada, postavljene su i dvije tipke za pokretanje dizala u ručnom načinu rada. Te dvije tipke su vrlo bitne, budući da se jedino pomoću njih može dizalo kretati gore i dolje.



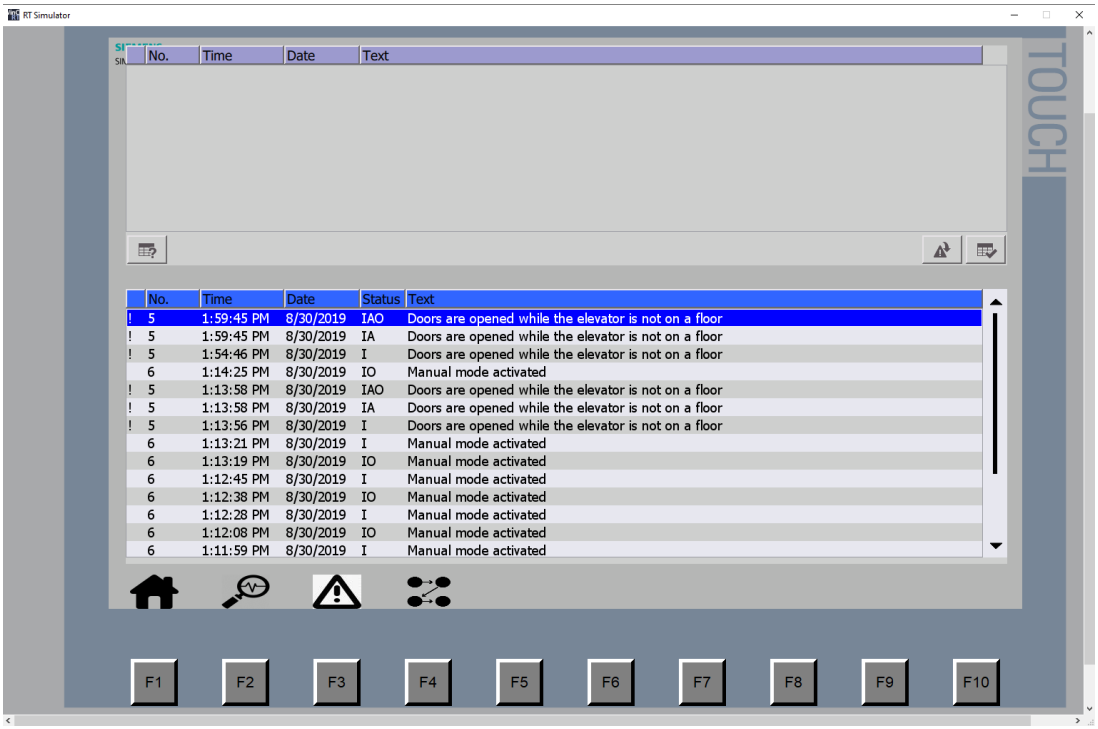
Slika 4.25. Izgled glavnog ekrana pri radu dizala

Pri dnu ekrana se mogu vidjeti ikone koje vode na ostale ekrane. Njima se pristupa tipkama F1 do F4 na tipkovnici.

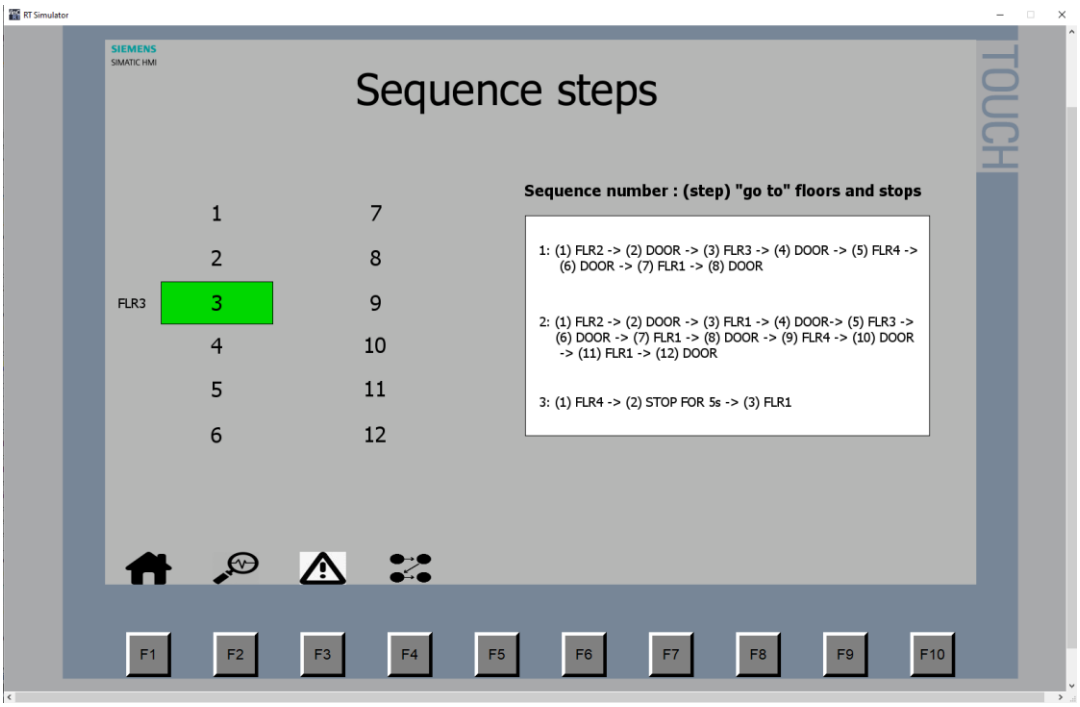
Druga ikona slijeva prikazuje dopuštenja kretanje gore i dolje u poluautomatskom i ručnom načinu rada te dopuštenja za aktivnosti sljedova naredbi (slika 4.26.). Treća ikona vodi na ekran za alarme gdje gornji dio prikazuje trenutne alarme i upozorenja, a donji dio služio kao spremnik svih alarma, upozorenja i obavijesti (slika 4.27.). Zadnja ikona vodi na ekran za prikazivanje koraka u sljedovima naredbi. S desne strane je legenda koja pojašnjava korake sljedova naredbi, a s lijeve strane se prikazuju trenutni koraci (slika 4.28.).



Slika 4.26. Ekran koji prikazuje dopuštenja



Slika 4.27. Ekran koji prikazuje alarme, upozorenja i obavijesti



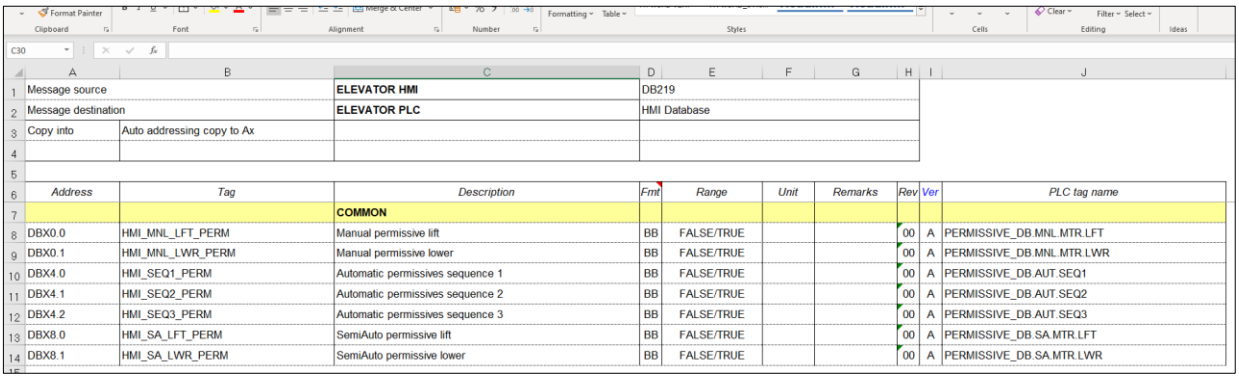
Slika 4.28. Ekran koji prikazuje korake sljedova naredbi

4.3. LANDEF tablica

LANDEF je tablica pisana u MS Excelu ili sličnom programu za tablično računanje. Ona definira komunikaciju između PLC-a i ostalih uređaja koji se koriste (HMI ili drugi PLC-ovi). Definiranje

komunikacije znači da ona sadrži sve varijable koje se koriste između dvaju uređaja u projektu kako ne bi došlo do nesporazuma pri pisanju programa ili izrade vizualizacije jer najčešće na projektu radi više ljudi.

Većinom sadrži više stranica koje su podijeljene po podatkovnim blokovima, a prikazuje podatke poput adrese u PLC-u, oznake koja se koristi u HMI-ju ili drugom PLC-u, opisa varijable, tipa varijable, oznake varijable u PLC-u itd. Na početnoj stranici nalazi se tablica sadržaja koja prikazuje imena svih stranica u LANDEF-u. Izgled jedne stranice nalazi se na slici 4.29., a cijela tablica se nalazi u prilogu.



Message source		ELEVATOR HMI	DB219						
Message destination		ELEVATOR PLC	HMI Database						
Copy into	Auto addressing copy to Ax								
Address	*Tag*	*Description*	*Fmt*	*Range*	*Unit*	*Remarks*	*Rev*	*Ver*	*PLC tag name*
		COMMON							
DBX0.0	HMI_MNL_LFT_PERM	Manual permissive lift	BB	FALSE/TRUE			00	A	PERMISSIVE_DB.MNL.MTR.LFT
DBX0.1	HMI_MNL_LWR_PERM	Manual permissive lower	BB	FALSE/TRUE			00	A	PERMISSIVE_DB.MNL.MTR.LWR
DBX4.0	HMI_SEQ1_PERM	Automatic permissives sequence 1	BB	FALSE/TRUE			00	A	PERMISSIVE_DB.AUT.SEQ1
DBX4.1	HMI_SEQ2_PERM	Automatic permissives sequence 2	BB	FALSE/TRUE			00	A	PERMISSIVE_DB.AUT.SEQ2
DBX4.2	HMI_SEQ3_PERM	Automatic permissives sequence 3	BB	FALSE/TRUE			00	A	PERMISSIVE_DB.AUT.SEQ3
DBX8.0	HMI_SA_LFT_PERM	SemiAuto permissive lift	BB	FALSE/TRUE			00	A	PERMISSIVE_DB.SA.MTR.LFT
DBX8.1	HMI_SA_LWR_PERM	SemiAuto permissive lower	BB	FALSE/TRUE			00	A	PERMISSIVE_DB.SA.MTR.LWR

Slika 4.29. Izgled jedne stranice LANDEF tablice

4.4. Python aplikacije

Kako bi se podaci iz međuspremnika MODBUS servera mogli pokupiti i poslati, korištene su dvije aplikacije izrađene u programskom jeziku Python.

Aplikacija Mongo2PLCapp.py služi kao MODBUS klijent koji pomoću biblioteke pyModbusTCP i razreda „client“ šalje zahtjeve za čitanje registara međuspremnika (slika 4.30.) i pisanje u njih (slika 4.31.).

```
    # if open() is ok, read register (modbus function 0x03)
    if modbus_client.is_open():
        # read 7 registers at address 0, store result in regs list
        regs = modbus_client.read_holding_registers(0, 7)

        # if success display registers
        if regs:
            print("reg: " + str(regs))
            Database.clean_collection(collection=collection)
            for reg in regs:
                Database.to_mongo(collection=collection, data={'value_reg': reg})
```

Slika 4.30. Isječak koda aplikacije app2c.py koji čita registre iz MODBUS-a i sprema u Mongo bazu podataka

```
231         modbus_client.write_multiple_registers(10, floors)
232         modbus_client.write_multiple_registers(15, times)
```

Slika 4.31. Isječak koda aplikacije Mongo2PLCapp.py koji piše u registre MODBUS-a

Za čitanje registara međuspremnika, koristi se metoda „read_holding_registers“ koja, u ovom slučaju, čita 7 registara koji kreću s nulte adrese i sprema u niz „regs“. Nakon toga se svi podaci spremaju u Mongo bazu. Za pisanje se koristi metoda „write_multiple_registers“ kojoj se predaju početni registar i niz koji se želi upisati.

Aplikacija Mongo2Webapp.py koristi tzv. *microframework* Flask koji omogućava brzu i jednostavnu izradu web aplikacija pomoću programskog jezika Python i HTML-a. Izrađena aplikacija pomoću biblioteke „flask“ i pomoću baze podataka omogućava prikaz željenih podataka na jednostavnoj web aplikaciji.

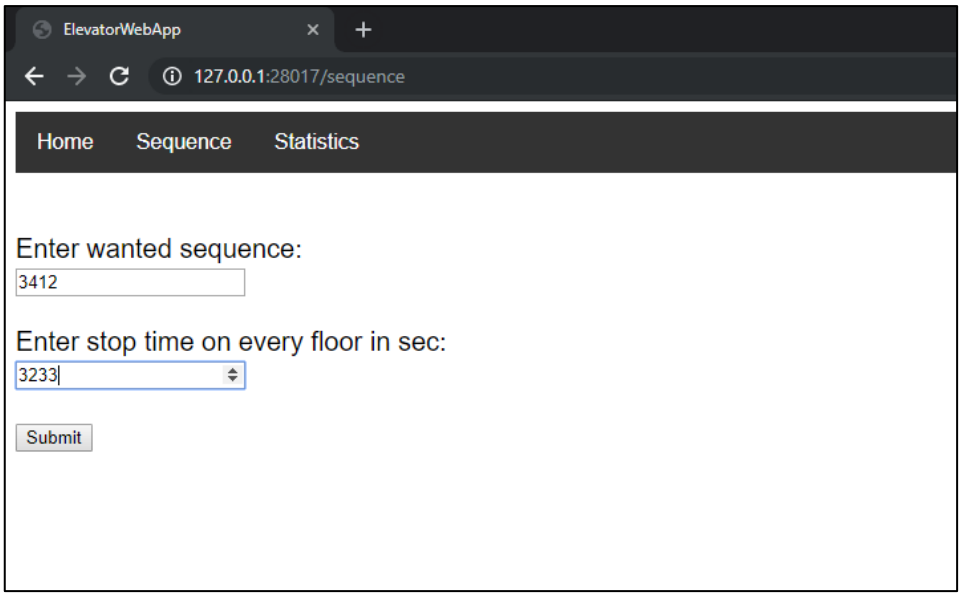
```
23  @app.route('/')
24  def elevator():
25      elev_floor_txt = ""
26      opened_doors_txt = ""
27      elev_movement_txt = ""
28      try:
29          received_data = list(Database.from_mongo(collection=collection_to_webapp, query={}))
30          regs = []
31
32          for n in range(len(received_data)):
33              if 'value_reg' in received_data[n]:
34                  regs = received_data[n].get('value_reg')
35          if regs:
36              regs = list(map(int, regs))
37
38          if len(regs) == 7:
39              elev_floor_txt = 'Current floor: ' + str(regs[0])
40              if regs[1] == 1:
41                  elev_movement_txt = 'Elevator is going up.'
42              elif regs[2] == 1:
43                  elev_movement_txt = 'Elevator is going down.'
44              else:
45                  elev_movement_txt = 'Elevator is idle.'
46
47              if regs[3] == 1:
48                  opened_doors_txt = 'Door on floor 1 is open.'
49              elif regs[4] == 1:
50                  opened_doors_txt = 'Door on floor 2 is open.'
51              elif regs[5] == 1:
52                  opened_doors_txt = 'Door on floor 3 is open.'
53              elif regs[6] == 1:
54                  opened_doors_txt = 'Door on floor 4 is open.'
55              else:
56                  opened_doors_txt = 'No doors are open.'
57      except:
58          elev_floor_txt = "Couldn't get data from PLC"
59      return render_template("elevator.html", elevator_floor_txt=elev_floor_txt,
60                             elevator_movement_txt=elev_movement_txt,
61                             doors_opened_txt=opened_doors_txt)
```

Slika 4.32. Isječak koda aplikacije Mongo2Webapp.py koji čita podatke iz baze podataka, formira ih i slaže u HTML dokument

Slika 4.32. prikazuje kako se formiraju primljeni podaci iz baze podataka. Iz pročitanih registara se formiraju tekstualni podaci koji se spremaju u varijable i predaju kao parametri funkciji „render_template“. Ova funkcija omogućava kreiranje i formiranje podataka u HTML datoteci „elevator.html“. Prikaz podataka je pokazan na slici 4.33.

Slika 4.33. Prikaz glavne stranice web aplikacije

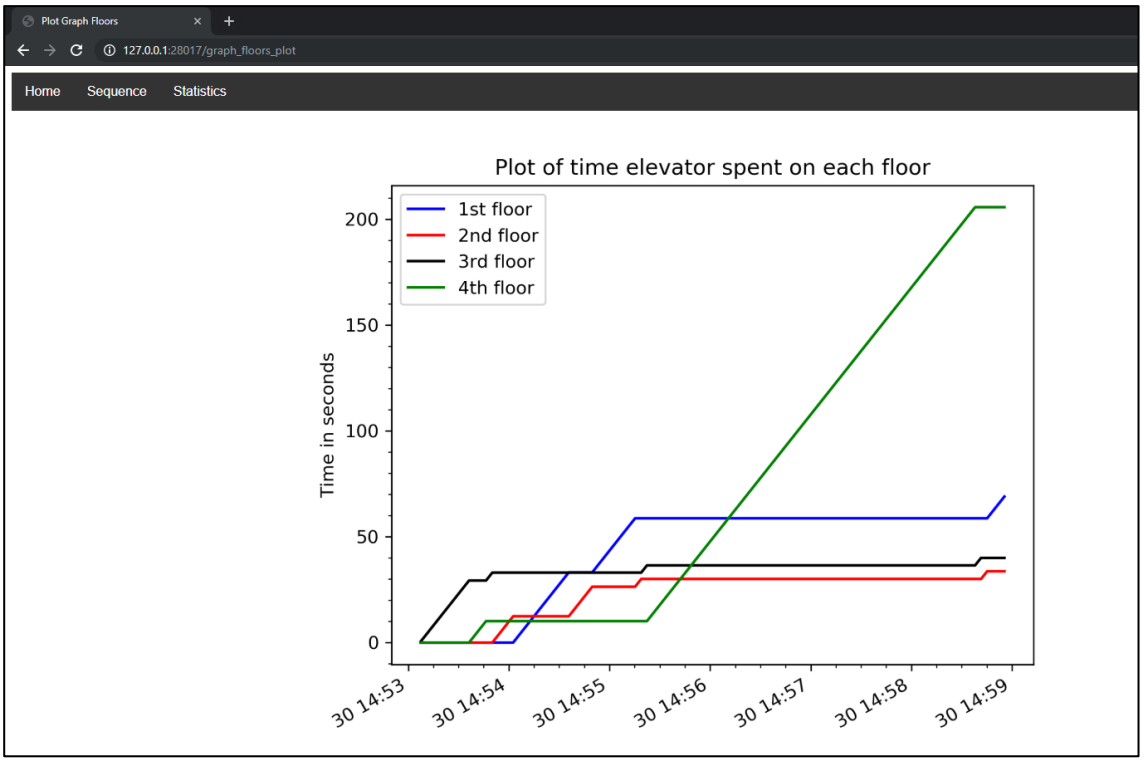
Uz glavnu stranicu, postoje još i stranica za unos slijeda naredbi te stranice koja prikazuju statistiku rada dizala.



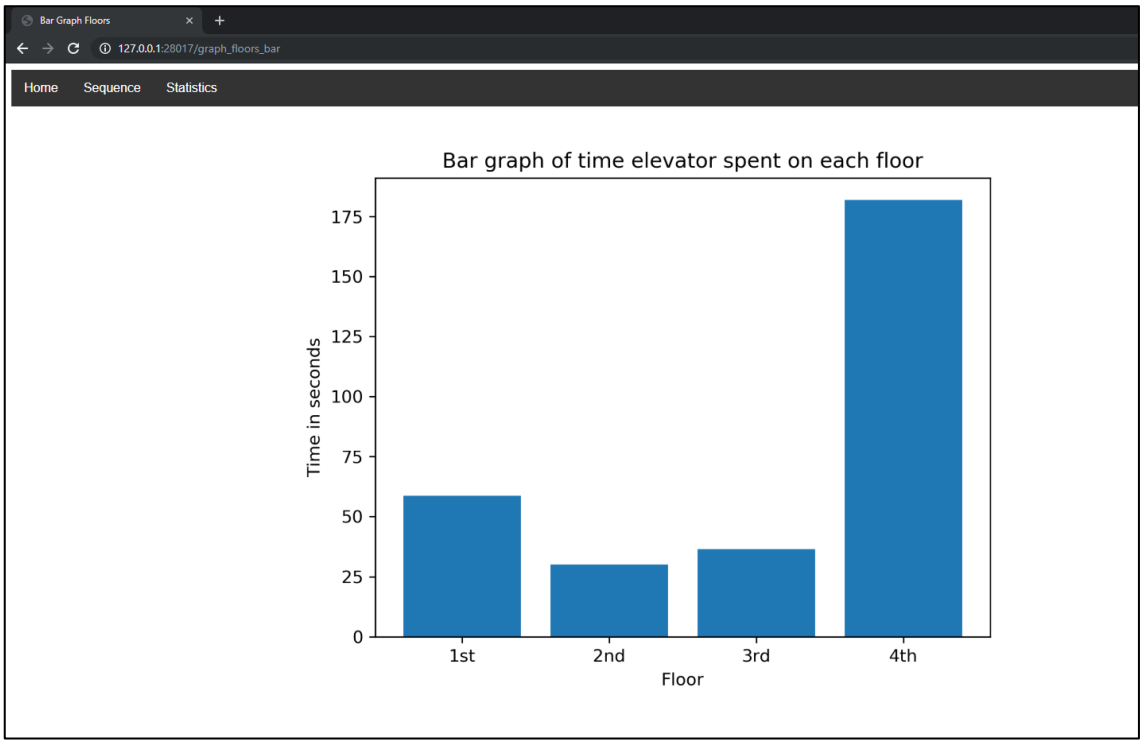
Slika 4.34. Stranica za unošenje slijeda naredbi

Slika 4.34. prikazuje unošenje željenih katova i vremena stajanja za slijed naredbi koji se izvrši na dizalu. Katovi i vremena stajanja se unose kao jedan višeznamenkasti broj gdje je prva znamenka slijeva ujedno i prvi kat na koji će dizalo krenuti, odnosno prvo vrijeme stajanja u sekundama.

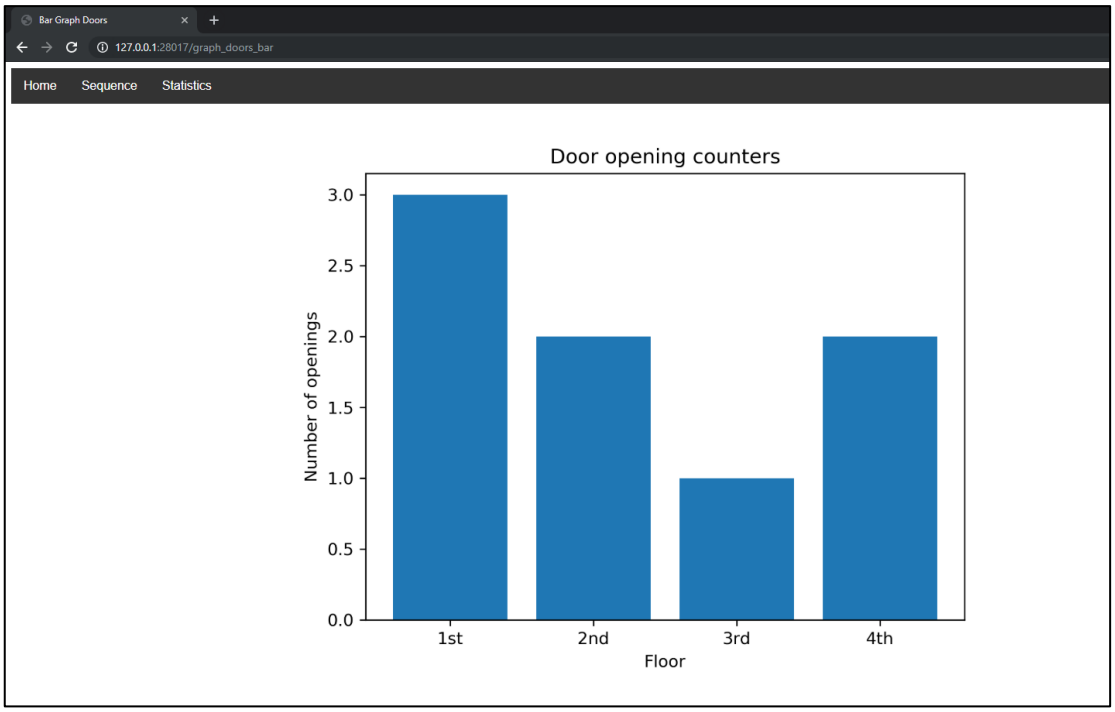
Slike 4.35., 4.36. i 4.37. prikazuju statistiku rada dizala. Slika 4.35. prikazuje linijski graf koji predstavlja vrijeme koje je dizalo provelo na svakom od katova. Y os označava provedeno vrijeme, a x os označava dan u mjesecu, sat i minute u stvarnom vremenu. Na slici 4.36. se također može vidjeti provedeno vrijeme na katovima, ali u stupčastom grafu, a slika 4.37. prikazuje broj otvaranja vrata na svakom katu.



Slika 4.35. Linijski graf koji prikazuje vrijeme provedeno na svakom katu



Slika 4.36. Stupčasti graf koji prikazuje vrijeme provedeno na svakom katu



Slika 4.37. Stupčasti graf koji prikazuje broj otvaranja vrata na svakom katu

5. ZAKLJUČAK

Industrijska automatizacija danas je sve šira i kompleksnija. Velikim dijelom je to omogućeno dolaskom programirljivih logičkih kontrolera. PLC-ovi su omogućili jednostavnije programiranje, ispravljanje i puštanje u pogon strojeva uz nisku cijenu. U ovom završnom radu htjela se dati zaokružena slika industrijskog pogona te osnovni dijelovi koje taj pogon može sadržavati: od električnih shema koje prethode ožičavanju i puštanju u pogon, preko programa PLC-a i vizualizacije do više razine upravljanja. Viša razina upravljanja ostvarena je pomoću MODBUS komunikacije i aplikacija napisanih u programskom jeziku Python. U ovom slučaju, taj pogon predstavlja dizalo. Dizalo je u mogućnosti raditi u tri načina rada: ručni, poluautomatski i automatski. Program PLC-a je napisan na način da se temelji na stanju mašine te ovisno je li mašina u stanju ručnog, poluautomatskog ili automatskog načina rada, tako će se i ponašati. Ručni način rada je način rada čije je izvođenje zamišljeno u slučaju kvara ili izvanredne situacije. Poluautomatski način rada omogućava prihvaćanje više upita odjednom, čime se postiže rad stvarnog dizala, a automatski način rada omogućava izvođenje sveukupno četiri različita slijeda naredbi. Napisani program je simuliran na maketi dizala smještenoj u zgradi fakulteta. Uz napisani program, izrađena je vizualizacija (HMI) u programu WinCC u sklopu TIA Portala. Aplikacije izrađene u programskom jeziku Python daju mogućnost upravljanja dizalom s kompleksnijim algoritmima, spremanje podataka u bazu podataka te kreiranje izvješća različitih vrsta.

LITERATURA

[1] History of the PLC, https://library.automationdirect.com/history-of-the-plc/, 26.6.2019.

[2] History of the Programmable Logic Controller (PLC), https://www.plcmentor.com/Articles/Newsletters/Programmable-Logic-Controller-PLC-History, 26.6.2019.

[3] W., Bolton, Programmable Logic Controllers, Elsevier Newnes, 2006.

[4] D., Marinković, Programabilni logički kontroleri : uvod u programiranje i primenu, Mikro knjiga, Beograd, 2013.

[5] PROFIBUS & PROFINET International (PI), PROFINET System Description Technology and Application, 2014.

[6] MODBUS PROTOCOL, http://www.modbus.org/specs.php, 25.6.2019.

[7] Modbus-IDA, MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE, 2006.

[8] Siemens, Data sheet 6ES7214-1AG40-0XB0, 2019.

[9] Siemens, SIMATIC S7-1200 Programmable Controller – System Manual, 2016.

[10] 6EP1332-1SH71 - POWER SUPPLY S7-1200 PM1207, https://support.industry.siemens.com/cs/pd/399209?pdti=td&dl=en&lc=en-HR, 27.6.2019.

[11] Ladder Logic Tutorial for Beginners, https://www.plcacademy.com/ladder-logic-tutorial/, 20.6.2019.

[12] Ladder Logic Programming Examples, https://ladderlogicworld.com/ladder-logic-programming-examples/, 20.6.2019.

[13] Siemens, Structured Control Language (SCL) for S7-300/S7-400 Programming

[14] Efficient SCL Development in TIA Portal V14, https://www.dmcinfo.com/latest-thinking/blog/id/9540/efficient-scl-development-in-tia-portal-v14, 20.6.2019.

[15] Meet PyCharm, https://www.jetbrains.com/help/pycharm/meet-pycharm.html, 21.6.2019.

[16] Introduction to MongoDB, https://docs.mongodb.com/manual/introduction/, 22.6.2019.

[17] Install MongoDB Community Edition on Windows, https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/#run-mongodb-from-cmd, 22.6.2019.

[18] Attendant service (AS), https://elevation.fandom.com/wiki/Attendant_service_(AS), 25.6.2019.

[19] D. E., Knuth, The art of Computer Programming, Vol. 3, Addison-Wesley Publishing Company, 1973., str. 358

[20] PyModbusTCP documentation, https://pymodbustcp.readthedocs.io/en/latest/, 26.6.2019.

[21] Flask, https://palletsprojects.com/p/flask/, 26.6.2019.

[22] TIA Portal V15 – Help

[23] T., Šurina, Automatska regulacija, 3. izdanje, Školska knjiga, Zagreb, 1987.

[24] N., Perić, Automatsko upravljanje – predavanja, Fakultet elektrotehnike i računarstva, Zagreb, 2004.

[25] D., Slišković, R., Cupec, Osnove automatskog upravljanja, materijali s predavanja, FERIT

SAŽETAK

U ovom radu opisana je automatizacija dizala pomoću programirljivog logičkog kontrolera u programskom okruženju TIA Portal. Dizalo može raditi na tri načina: ručni, poluautomatski i automatski način rada. Za simulaciju je korištena maketa dizala smještena u zgradi fakulteta. Uz program, napravljena je vizualizacija (HMI) dizala u alatu WinCC. Omogućeno je i prikazivanje određenih podataka na web aplikaciji i slanje podataka u PLC pomoću MODBUS protokola i lokalne baze podataka. Korišteni PLC je iz serije S7-1200 tvrtke Siemens na koji je još dodano napajanje i signalni modul.

Ključne riječi: Dizalo, PLC, HMI, TIA Portal, Automatizacija

ABSTRACT

Title: Elevator control with programmable logic controller

The focus of this paper is elevator control using a programmable logic controller in the TIA Portal development environment. The elevator has three operation modes: manual, semiautomatic and automatic mode. A laboratory model of elevator is used for simulation. Additionally, the visualization or the HMI was made in WinCC software. It is also possible to display certain information on a web application and send data to PLC with MODBUS protocol and a local database. The PLC used in this paper is Siemens Simatic S7-1200 series.

Keywords: Elevator, PLC, HMI, TIA Portal, Automation

ŽIVOTOPIS

Valentin Šimundić rođen je 17.3.1998. godine u Osijeku. U Đakovu je pohađao OŠ Josipa Antuna Ćolnića i gimnaziju Antuna Gustava Matoša. Tečno govori engleski jezik. 2016. godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer računarstvo. 2018. godine dobiva STEM stipendiju, a od veljače do ožujka 2019. godine pohađa praksu u tvrtki Danieli Systec d.o.o. za koju je izradio završni rad.

PRILOG

U prilogu se nalazi dio elektroshema koje prikazuju ožičavanje makete dizala i PLC-a.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

DANIELI SYSTEC
Member of Danieli Automation

Customer FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Project description LIFT

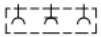
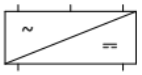
Revision **FIRST ISSUE [00]**

Revision date **23-05-2019** *Revised by* **DSYS**

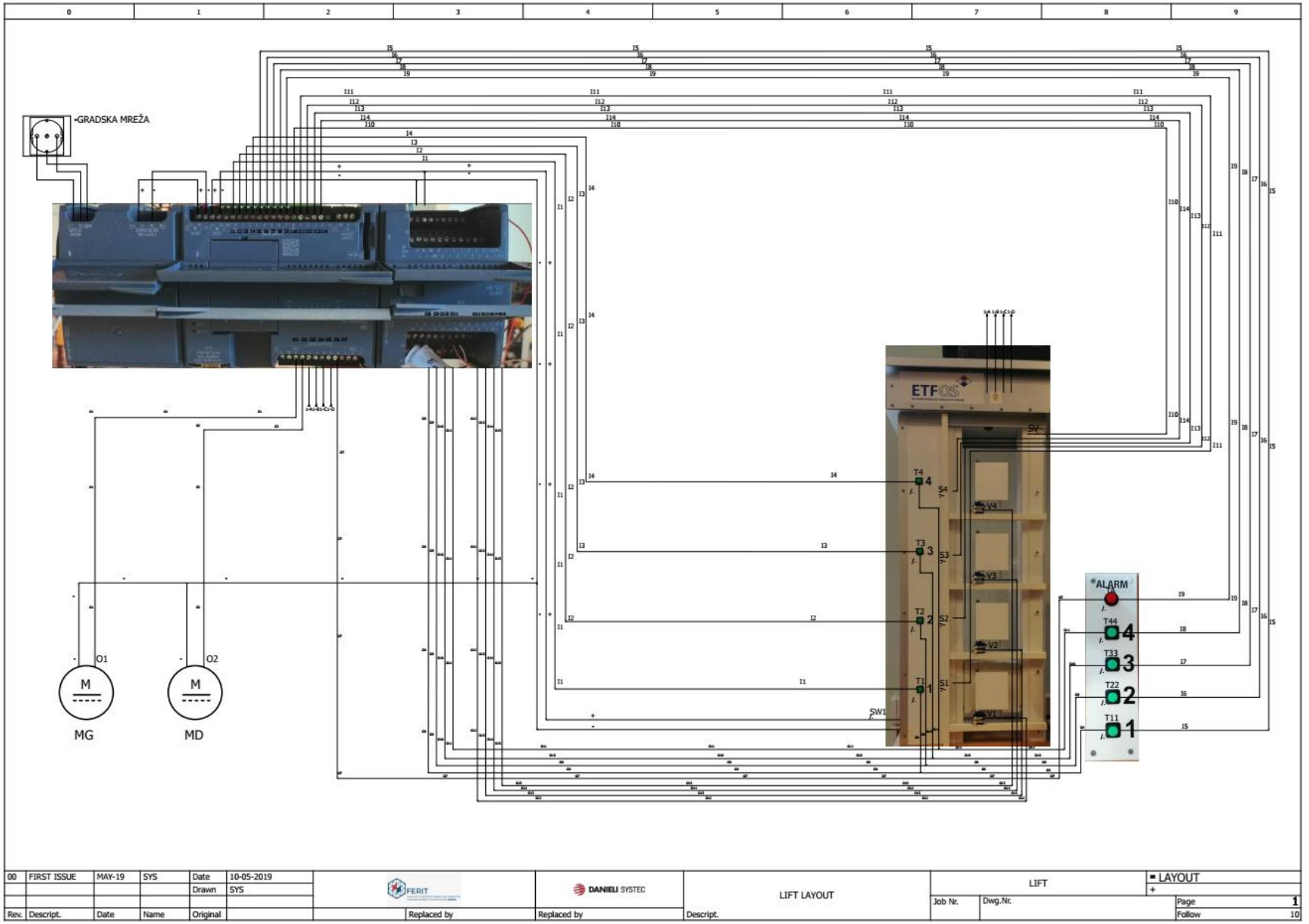
General notes

Circuit	*Voltage*	*Common wire color*
Main circuit	1~230VAC 50 Hz	Black
Auxiliary AC		Red
Auxiliary DC	24VDC	Blue
External circuit		Orange
Neutral		Light blue
Protective earth	TN/TN-C	Green-Yellow

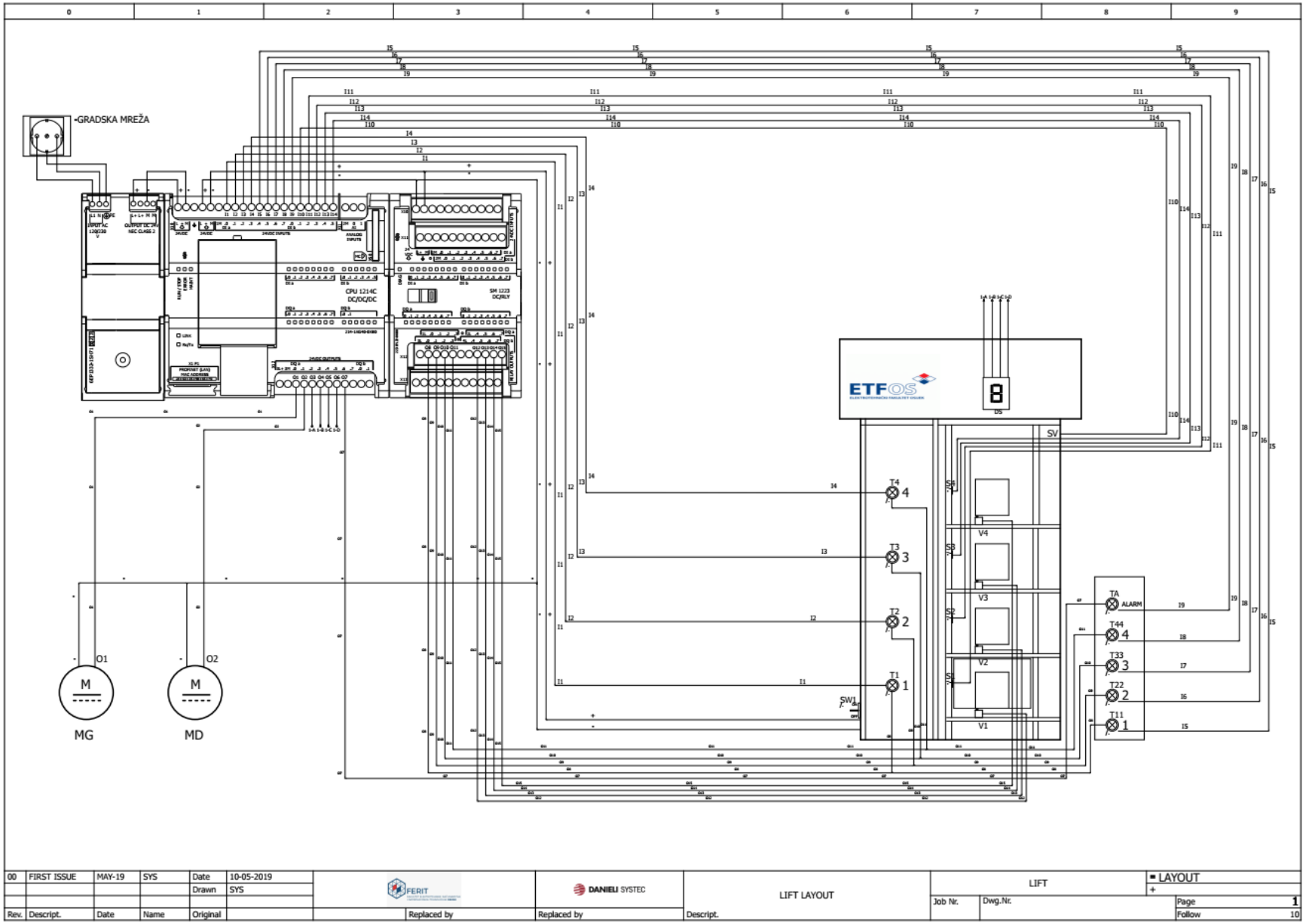
LIFT

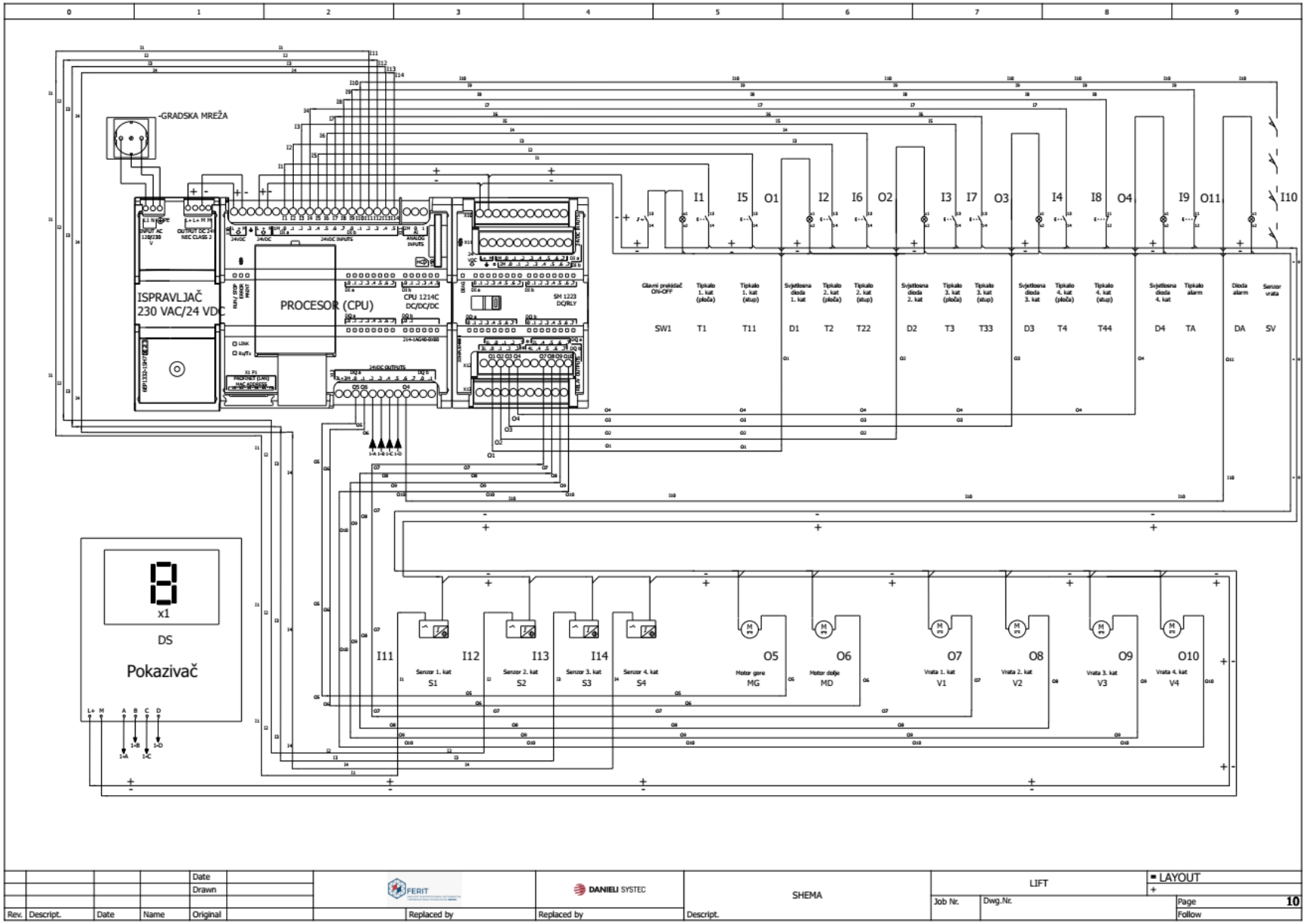
Socket (Utičnica)			
Power supply unit (Napajanje)			
Pushbutton operated by pushing (Tipkalo)			
Switch, operated by turning (Prekidač)			
Lamp (Lampica)			
Proximity switch (Induktivni davač)			
Motor (Elektromotor)			
Limit switch (Mikro prekidač)			
Display (Pokazivač)			

00	FIRST ISSUE	MAY-19	SYS	Date	10-05-2019	FERIT	DANIELI SYSTEC	SIMBOLI KORIŠTENI U SHEMI	LIFT	
				Drawn	SYS					
Rev.	Descript.	Date	Name	Original		Replaced by	Replaced by	Descript.	Job Nr. / Dwg.Nr.	Page 3.f / Follow =LAYOUT/1



00	FIRST ISSUE	MAY-19	SYS	Date	10-05-2019	FERIT	DANIELI SYSTEC	LIFT LAYOUT	LIFT	LAYOUT	
				Drawn	SYS					+	
Rev.	Descript.	Date	Name	Original		Replaced by	Replaced by	Descript.	Job Nr.	Dwg.Nr.	Page 1 / Follow 10


GRADSKA MREŽA
CPU 1214C DC/DC/DC
SM 1223 DC/RLY
ETFOS
SV
T4
T3
T2
T1
V4
V3
V2
V1
SW1
TA
ALARM
T44
T33
T22
T11
O1
O2
M
MG
MD
LIFT LAYOUT
LIFT
LAYOUT
FERIT
DANIELI SYSTEC


GRADSKA MREŽA
ISPRAVLJAČ
230 VAC/24 VDC
PROCESOR (CPU)
CPU 1214C
DC/DC/DC
SM 1223
DC/RLY
Glavni prekidač ON-OFF
Tipkalo 1. kat (ploča)
Tipkalo 1. kat (stup)
Svjetlosna dioda 1. kat
Tipkalo 2. kat (ploča)
Tipkalo 2. kat (stup)
Svjetlosna dioda 2. kat
Tipkalo 3. kat (ploča)
Tipkalo 3. kat (stup)
Svjetlosna dioda 3. kat
Tipkalo 4. kat (ploča)
Tipkalo 4. kat (stup)
Svjetlosna dioda 4. kat
Tipkalo alarm
Dioda alarm
Senzor vrata
SW1
T1
T11
D1
T2
T22
D2
T3
T33
D3
T4
T44
D4
TA
DA
SV
I1
I5
O1
I2
I6
O2
I3
I7
O3
I4
I8
O4
I9
O11
I10
x1
DS
Pokazivač
I11
Senzor 1. kat
S1
I12
Senzor 2. kat
S2
I13
Senzor 3. kat
S3
I14
Senzor 4. kat
S4
O5
Motor gore
MG
O6
Motor dolje
MD
O7
Vrata 1. kat
V1
O8
Vrata 2. kat
V2
O9
Vrata 3. kat
V3
O10
Vrata 4. kat
V4
Date
Drawn
Rev.
Descript.
Date
Name
Original
FERIT
Replaced by
DANIELI SYSTEC
Replaced by
SHEMA
Descript.
LIFT
LAYOUT
Job Nr.
Dwg.Nr.
Page
10
Follow