# AUTOMATIZACIJA POKRETNE TRAKE

**Svirac, Damian**



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

**Sveučilišni studij**

# AUTOMATIZACIJA POKRETNE TRAKE

**Završni rad**

**Damian Svirac**

**Osijek, 2019.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju**

**Osijek, 19.09.2019.**

**Odboru za završne i diplomske ispite**

# Prijedlog ocjene završnog rada

| | |
|---|---|
| **Ime i prezime studenta:** | Damian Svirac |
| **Studij, smjer:** | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3987, 23.09.2018. |
| **OIB studenta:** | 14210449326 |
| **Mentor:** | Prof.dr.sc. Robert Cupec |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | Marko Španović |
| **Naslov završnog rada:** | Automatizacija pokretne trake |
| **Znanstvena grana rada:** | **Automatizacija i robotika (zn. polje elektrotehnika)** |
| **Predložena ocjena završnog rada:** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene mentora:** | 19.09.2019. |
| **Datum potvrde ocjene Odbora:** | 25.09.2019. |

| | |
|---|---|
| Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija: | Potpis: |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 02.10.2019.**

| | |
|---|---|
| **Ime i prezime studenta:** | Damian Svirac |
| **Studij**: | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3987, 23.09.2018. |
| **Ephorus podudaranje [%]:** | 5 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Automatizacija pokretne trake**

izrađen pod vodstvom mentora Prof.dr.sc. Robert Cupec

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# SADRŽAJ

# 1. UVOD

Pokretna traka jedan je od najstarijih i najzastupljenijih elemenata industrije. U prošlosti, pokretne trake su bile napravljene od više balvana u obliku valjaka preko kojih se prenosio teret. Suvremene pokretne trake koriste se u gotovo svim oblicima industrija, primjerice kemijska, automobilska i metalna industrija; zatim u proizvodnji hrane i pića, poljoprivredi i brojim drugim granama industrija. Poboljšavanjem pokretnih traki, danas se osim za prijevoz materijala koriste i za prijevoz ljudi, npr. pokretne stepenice u trgovačkim centrima ili pokretne trake u zračnim lukama. Zbog jednostavne konstrukcije, pokretna traka je izdržljiva oprema koje se ne kvari često, ne stvara velike probleme i ne troši mnogo energije s obzirom na masu prevezenog tereta.

Cilj ovog završnog rada je izrada programa za upravljanje pokretnom trakom. Program treba omogućiti pokretanje trake u različitim načinima rada te sadržavati mogućnost praćenja proizvoda. Alati i programsko okruženje korišteni za izvedbu ovog rada navedeni su u drugom poglavlju. U istom poglavlju opisane su osnove rada u takvom programskom okruženju te dodatno pojašnjeni određeni elementi korišteni u izradi programa. U trećem poglavlju nalazi se opis i analiza pokretne trake i njenih elemenata. Četvrto poglavlje prikazuje rješenja problema i algoritme iz programa uz njihova objašnjenja. Rezultati programa prikazani su u petom poglavlju.

## 1.1. Zadatak završnog rada

Glavni zadatak završnog rada je izrada programa za upravljanje pokretnom trakom. Program treba sadržavati rješenje ručnog, automatskog i poluautomatskog pokretanja trake te praćenje više proizvoda istovremeno. Završni rad uključuje razvoj programa i simulaciju te vizualizaciju u alatu *TIA* Portal.

Dodatni zadatak je izrada web aplikacije koja komunicira s programom.

# 2. TEHNOLOGIJE I PROGRAMSKO OKRUŽENJE

Izrada programa za upravljanje pokretnom trakom odrađena je u programskom okruženju pod nazivom *TIA* Portal (*Totally Integrated Automation Portal*). *TIA* Portal je alat koji je kreirala tvrtka *Siemens*, ujedno jedna od najvećih svjetskih dobavljača inovativnih proizvoda i rješenja u području industrije te vodeći svjetski proizvođač sustava za automatizacijsku tehnologiju.

Sam alat *TIA* Portal nije dovoljan da bi mogao pokrenuti pokretnu traku. Potreban je uređaj koji će preuzeti naredbe i operacije napisane u programu te izdavati komande pokretnoj traci. Taj uređaj naziva se programirljivi logički kontroler, engl. *Programmable Logic Controller* (*PLC*).

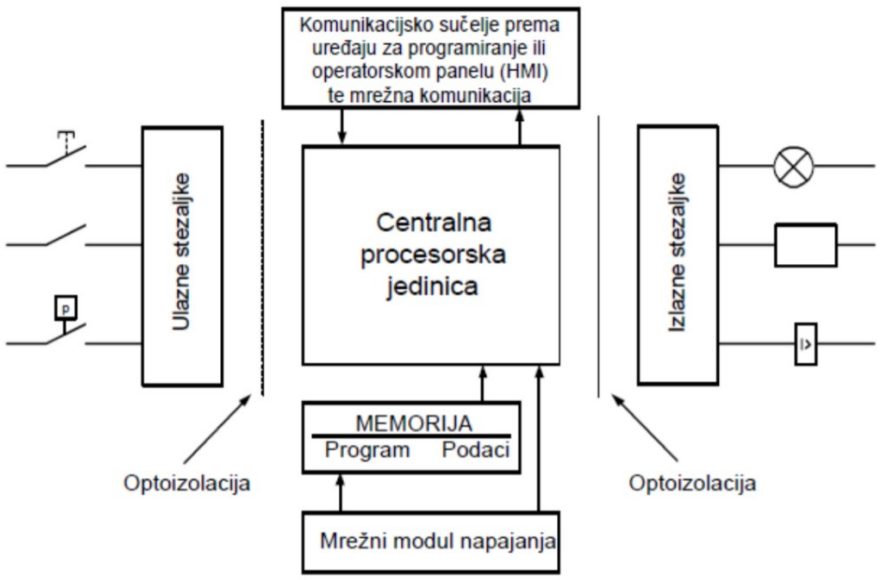
U nastavku ovoga poglavalja, navest će se i opisati tehnologija i alati korišteni za izradu ovoga rada.

## 2.1. *PLC*

*PLC* je industrijsko računalo koje se koristi za sustave upravljanja u proizvodnim pogonima. Često ga se naziva i srcem upravljačkog sustava. Izumljen je krajem šezdesetih godina kao zamjena za tadašnje sekvencijalne relejne krugove u upravljanju pogonima u industriji, a tvrtka koja je prva počela koristiti *PLC* uređaje je *General Motors*.

Osnovni dijelovi jednog *PLC* uređaja su centralna procesorska jedinica (*CPU – Central Processing Unit)*, memorijski blok, zatim ulazni i izlazni dijelovi koji mogu biti analogni ili digitalni, mrežni modul za napajanje i komunikaciju te modul za proširenje.

Shema *PLC* uređaja prikazana je slikom 2.1.




*Slika 2.1. Shema PLC uređaja*

*CPU* je najvažnija komponenta *PLC* uređaja koja upravlja čitavim sustavom i izvršava program. Kako bi *CPU* mogao neometano izvoditi operacije, zaslužan je modul napajanja. Osim što osigurava potreban napon za sustav, regulira i napon koji se dovodi te upozorava *CPU* ako postoji bilo kakvo odstupanje. *PLC* uređaji uglavnom rade na izmjeničnom izvoru napajanja, s naponom od 220V. Memoriju *PLC* uređaja čine izvršna i korisnička memorija. Izvršna se memorija sastoji od trajno pohranjenih programa i potprograma koji su zaduženi za izvođenje korisničkih programa i komunikaciju s ulazno/izlaznim uređajima. Ova memorija sastavni je dio svakog *PLC* uređaja te se ne može mijenjati i nije dostupna krajnjem korisniku. Za razliku od izvršne memorije, korisničkoj memoriji korisnik može pristupiti, spremati programe i podatke te ih mijenjati. Svaki *PLC* ima ograničen broj izlaza i ulaza. Međutim, modulima za proširenje može se ostvariti dovoljan broj ulaza i izlaza za sustav, pri čemu moduli mogu sadržavati digitalne i analogne ulaze i izlaze. Modul za proširenje povezuje se komunikacijskim kabelom na *PLC*. Broj modula koji se povezuje na jedan *PLC* nije ograničen. Ulaze u *PLC* čine stezaljke spojene na različite senzore iz sustava kojim taj *PLC* upravlja, pri čemu se vanjski signal pretvara u signal koji procesor *PLC* uređaja može čitati. Primjer senzora koji se koriste u automatiziranim sustavima upravljanja su toplinski, magnetski, mehanički i svjetlosni senzori, zatim fotoćelije, akcelometri te žiroskopi. Ulaze također čine i ulazni uređaji kojima čovjek upravlja i šalje signale *PLC* uređaju, što je ostvareno pomoću sklopki, tipkala i drugih uređaja. Izlazi iz *PLC* uređaja čine stezaljke spojene na aktuatore. Aktuatori su izvršni uređaji, primjerice

motori, ventili, elektromagneti i grijači, koji dalje upravljaju velikim industrijskim strojevima i njihovim dijelovima. Pomoću signala koje *PLC* dobije iz senzora, procesor *PLC* uređaja izvršava određene akcije koje su zadane korisničkim programom, a po potrebi šalje upravljačke signale izvršnim uređajima. Kao primjer uzmimo slučaj lifta. Ulaz u sustav će biti senzor koji procesoru javlja na kojem se katu lift nalazi, te tipkalo koje osoba pritisne i pozove lift. Na temelju dobivenih signala, procesor će poslati signal aktuatoru, u ovom slučaju motoru, koji će pogoniti lift u smjeru potrebnom da dođe do osobe koja ga je pozvala. Pretpostavimo da se dizalo kreće prema gore i pritom došavši na pozvani kat, upalit će se senzor koji će procesoru javiti da je lift stigao na odredište i procesor će narediti motoru da se zaustavi, te drugom motoru da otvori vrata kako bi osoba mogla ući.

U svijetu postoje razni proizvođači *PLC* uređaja kao što su *Siemens, ABB, Allen-Bradley, Bosch, Mitsubishi i General Electric*, pri čemu su najzastupljeniji *PLC* uređaji od tvrtke *Siemens*. Ovaj rad je napravljen koristeći *Siemens PLC*.

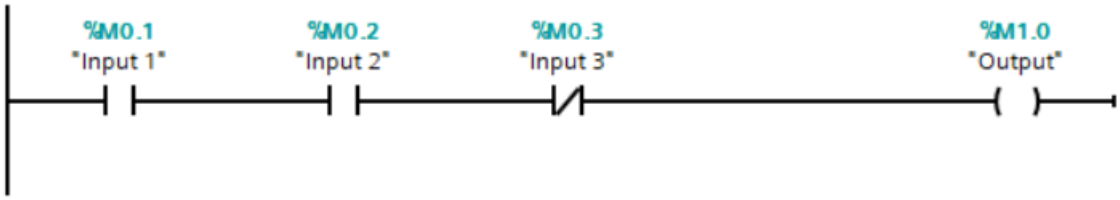
## 2.2. *Step* 7

*Step* 7 je softverski paket korišten za konfiguriranje sklopovlja, programiranje *PLC* uređaja i uspostavu komunikacije između uređaja. Integriran je unutar samog programskog okruženja *TIA* Portala. Programski jezici koje podržava za pisanje korisničkih programa su:

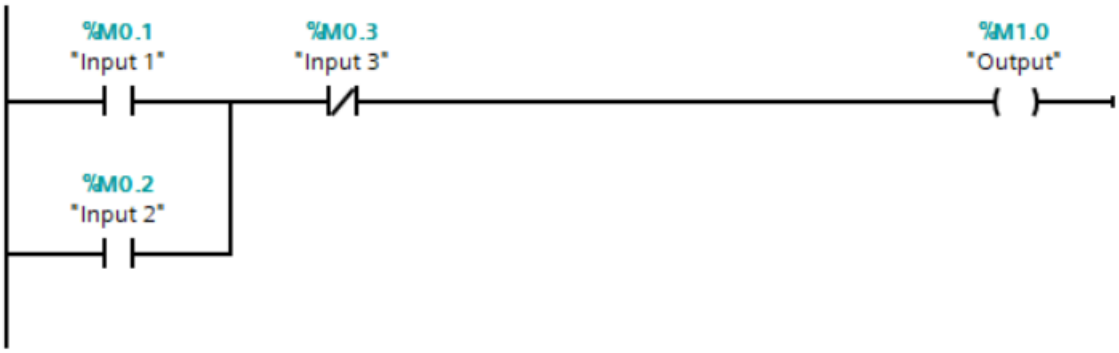
- *LAD* – logika ljestve (engl. *Ladder Logic*)
- *FBD* – funkcijski blokovski dijagram (engl. *Function Block Diagram*)
- *SCL* – strukturirani upravljački jezik (engl. *Structured Control Language*)
- *STL* – lista naredbi (engl. *Statement List*)
- *GRAPH* – slijedno upravljanje (dostupno samo za *Step* 7 *Professional* unutar *TIA* Portala)

### 2.2.1. *LAD* (*Ladder Logic*)

Programiranje *LAD* jezikom može se gledati kao protok struje kroz žicu. U serijskom spoju sve sklopke moraju biti zatvorene u strujnom krugu da bi tekla struja, dok je u paralelnom spoju dovoljna jedna zatvorena sklopka kako bi struja potekla. Na sljedećim slikama prikazano je programiranje *LAD* jezikom za dvije mreže (engl. *Network*). Struja teče žicom slijeva na desno, stoga se *Inputi* nalaze na lijevoj strani, a *Output* na desnoj.

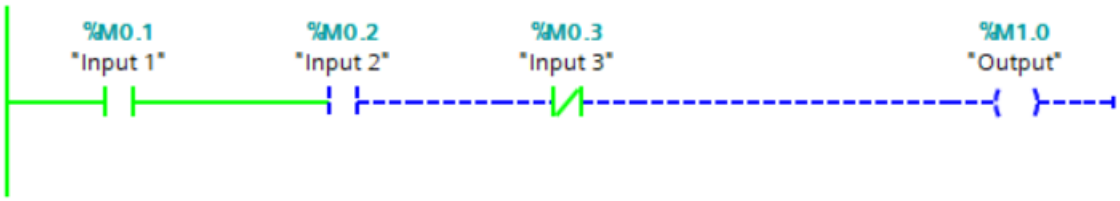



*Slika 2.2. Serijski spoj (LAD)*

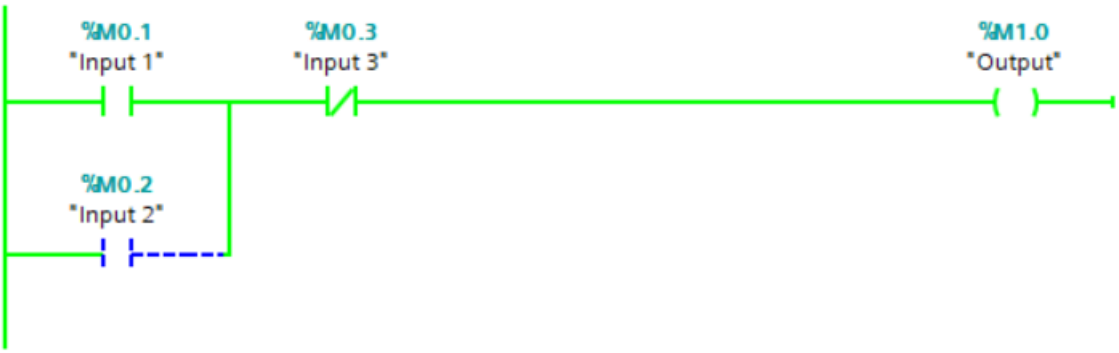



*Slika 2.3 Paralelni spoj (LAD)*

Na slikama 2.2. i 2.3. možemo uočiti dva tipa *Inputa*. *Inputi* 1 i 2 su ulazi i predstavljaju sklopke koju se otvorene kada na njih ne djeluje signal (engl. *Normally Open Switch*), a *Input* 3 predstavlja sklopku koja je zatvorena kada na nju ne djeluje nikakav signal (engl. *Normally Closed Switch*). Kako bi upalili *Output*, moramo dovesti struju do njega. Dakle, u slučaju na slici 2.2., potrebno je aktivirati *Input*e 1 i 2, a deaktivirati *Input* 3. U slučaju na slici 2.3. dovoljno je aktivirati jedan od *Inputa* 1 ili 2, a deaktivirati *Input* 3.




*Slika 2.4.. Serijski spoj (LAD) - aktiviran samo Input 1*




*Slika 2.5.. Paralelni spoj (LAD) - aktiviran samo Input 1*

Na slikama 2.4. i 2.5. prikazan je protok struje kroz serijski i paralelni spoj u slučaju kada je aktiviran samo *Input* 1. Zelena boja žice predstavlja tok struje. U prvom slučaju vidi se da struja nije stigla do *Outputa* jer *Input* 2 nije aktiviran. U drugom slučaju struja je stigla do *Outputa* iako *Input* 2 nije aktiviran, ali budući da je spoj paralelan, dovoljno je da jedan od *Inputa* 1, odnosno 2 bude aktiviran.

### 2.2.2. *FBD* (*Function Block Diagram*)

*FBD* jezik koristi blokovski sustav programiranja. *Inputi* se povezuju u blokove koji daju različiti izlaz ovisno o operaciji koju blok predstavlja te *Inputima* (jesu li aktivirani ili ne). Na sljedećim slikama koristit će se prethodni primjeri serijskog i paralelnog spoja, ali napisani *FBD* jezikom.


*Slika 2.6. Serijski spoj (FBD)*


*Slika 2.7. Paralelni  spoj (FBD)*

### 2.2.3. *SCL* (*Structured Control Language*)

*SCL* jezik, za razliku od prethodnih jezika, predstavlja tekstualni programski jezik korišten za pisanje programa i algoritama. Svojom strukturom je nalik *C* programskom jeziku jer koristi slične izraze i ključne riječi. U mrežama s mnogo varijabli, ulaza i izlaza, bilo bi teško pisati i pratiti tok mreže koristeći prethodno spomenute blokovske jezike. *SCL* je zbog svoje tekstualne forme pogodan za takve uvjete i olakšava pisanje programa. Na sljedećim slikama prikazani su isti primjeri slučajeva kao u prethodnim jezicima.

```
1 IF "db".i1 = TRUE AND
2     "db".i2 = TRUE AND
3     "db".i3 = FALSE
4 THEN
5     "db".o1 := TRUE;
6 ELSE
7     "db".o1 := FALSE;
8 END_IF;
```

*Slika 2.8.. Serijski spoj (SCL)*

```
1 IF ("db".i1 = TRUE OR "db".i2) = TRUE AND
2     "db".i3 = FALSE
3 THEN
4     "db".o1 := TRUE;
5 ELSE
6     "db".o1 := FALSE;
7 END_IF;
```

*Slika 2.9.. Serijski spoj (SCL)*

### 2.2.4. *STL* (*Statement List*)

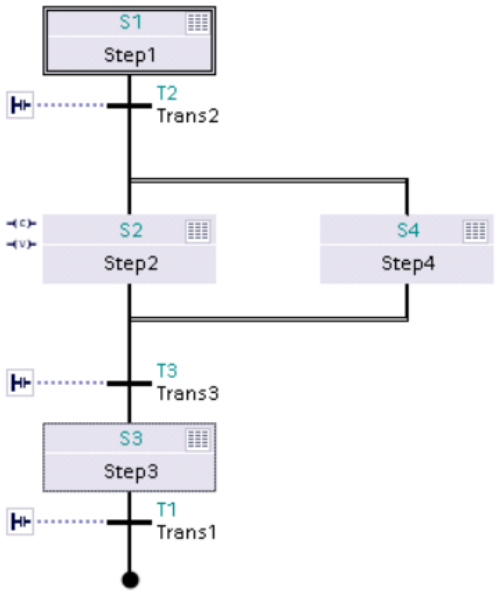
*STL* je tekstualni jezik koji se sastoji od pojedinačnih naredbi navedenih redak po redak, pri čemu se piše odozgo prema dolje. U *STL*-u su implementirane logičke operacije koje obrađuju stanja signala i vrijednosti adresa i varijabli. Slika 2.10. prikazuje jedan *Network* napisan *STL* jezikom, s objašnjenjem naredbi.

| STL | Explanation |
|---|---|
| A"Tag_Input_1" | // Check whether the signal state of the operand is "1" and AND with current RLO |
| A "Tag_Input_2" | // Check whether the signal state of the operand is "1" and AND with current RLO |
| S "Tag_Output" | // Sets the operand to "1" if the RLO is "1" |

*Slika 2.10. STL jezik*

### 2.2.5. *GRAPH*

*GRAPH* je programski jezik koji se koristi za upravljanje slijednim procesima, odnosno sekvencama (engl. *sequence*). Proces se rastavi na korake (engl. *step*) koji se izvršavaju jedan po jedan. U svakom koraku su definirane akcije koje će se izvršiti. Između koraka se nalaze uvjeti prijelaza (engl. *transitions)* koji određuju kada će se jedan korak završiti, a drugi započeti.




*Slika 2.11. GRAPH jezik*

## 2.3. *TIA* Portal (Totally Integrated Automation Portal)

*TIA* Portal (*Totally Integrated Automation Portal*) programsko je okruženje za programiranje i konfiguriranje *PLC* uređaja. Programirati je moguće u bilo kojem od prethodno navedenih programskih jezika. Prije samog pisanja programa, potrebno je odabrati uređaj na kojem će se program izvršavati. *TIA* Portal podržava *SIMATIC* (***Siemens*** *Auto**matic***) uređaje serija S7-1200, S7-1500, S7-300 i S7-400. Nakon odabira uređaja, moguće je podešavati konfiguraciju uređaja (engl. *hardware configuration*) i povezivati ga s drugim uređajima u sustavu. Odabirom uređaja, kreira se mapa koja sadrži programske blokove u kojima će se pisati program. Postoje četiri osnovna tipa programskih blokova:

- ***OB* (*Organization block*)**

  *OB* blokovi su organizacijski blokovi koji upravljaju tokom izvođenja programa. Jasno su definirani tipovi *OB* blokova s njihovim ulogama. Najbitniji *OB* blok je OB1, a ostali su primjerice OB100 i OB30. *OB*1 (*Program Cycle*) glavni je blok programa i izvršava se ciklički. U njemu se najčešće nalaze pozivi drugih funkcija i funkcijskih blokova. *OB*100 (*Startup*) je blok koji će se izvršiti samo jednom prilikom promjene načina rada *PLC*-a iz *STOP* u *RUN*. Nakon njega će se krenuti izvršavati blok *OB*1. *OB*30 (*Cyclic Interrupt*) je blok koji će se izvršavati u korisnički zadanim periodnim razmacima (npr. svakih 100ms), neovisno o cikličkom izvršavanju programa.
- ***DB* (*Data block*)**

  *DB* blokovi su blokovi koji pohranjuju podatke iz programa. Sve varijable u programu, ulaze i izlaze potrebno je definirati unutar *DB* bloka. Moguće je definirati različite tipove podataka, kao npr. *bool* (logička varijabla – *true/false*), *int* (cijeli broj), *dint* (*double int*), *word* (realan broj), *time* (vremenska varijabla). Moguće je definirati i nizove (*arrays*) različitih tipova podataka.
- ***FC* (*Function*)**

  *FC* blokovi su funkcije bez memorije i sadrže program koji će se izvršiti prilikom poziva funkcije. Svrha *FC* blokova je podjela programa na više smislenih dijelova kako bi program bio jasniji i pregledniji. Funkcija će se izvršiti kada se dogodi njen poziv koji može biti u *OB* bloku ili nekom drugom *FC* ili *FB* bloku. Funkcija prilikom poziva može vratiti vrijednost bloku iz kojeg je pozvana ili odraditi određenu operaciju.

- ***FB* (*Function block*)**
  *FB* blokovi su funkcijski blokovi koji „pamte“ prethodno odrađene operacije. Prilikom poziva *FB* bloka, stvorit će se instanca *DB* bloka u kojoj će se čuvati vrijednosti iz tog *FB* bloka nakon što se on izvrši. Sličan je *FC* bloku, s razlikom u memoriji koju *FB* blok sprema, a *FC* blok ne sprema.

U sklopu *TIA* Portala, mogu se instalirati dodatni alati koji se koriste pri izradi programa. Najznačajnije je spomenuti alate *WinCC* i *PLCSIM*.

*PLCSIM* je alat koji omogućuje simulaciju programa. Umjesto na stvarni *PLC* uređaj, program se preuzima na virtualni uređaj koji simulira stvarni *PLC*. Simulacija je vrlo bitna stavka u programiranju rješenja za upravljanje procesima nekog stroja. Vrlo je važno testirati program prije puštanja u pogon jer jedna pogreška u programu može imati vrlo velike posljedice. U industriji se često radi sa strojevima velikih dimenzija u čijoj blizini se nalaze radnici. Može se reći da inženjeri koji programiraju *PLC* uređaje imaju visoku odgovornost za ljudsku sigurnost.

WinCC je alat koji se koristi za izradu HMI (*Human Machine Interface*) aplikacija.

## 2.4. HMI (Human Machine Interface)

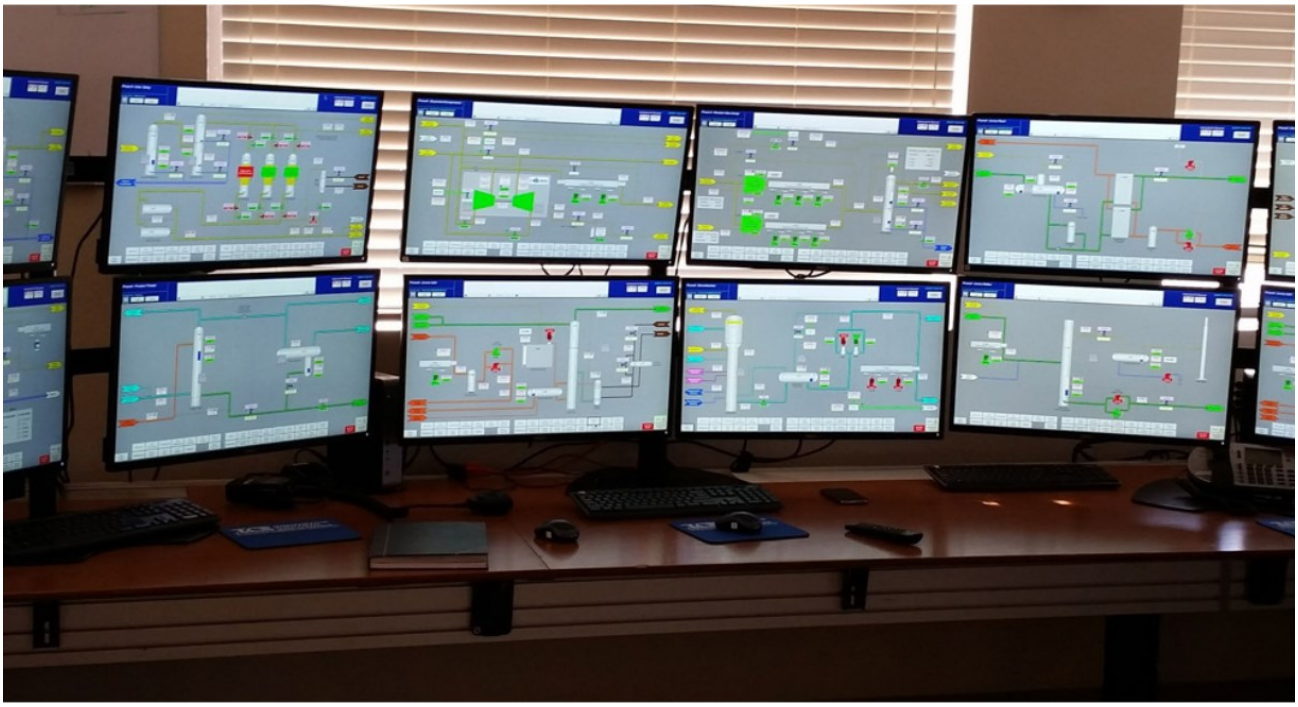
*HMI (Human Machine Interface)* korisničko je sučelje koje služi kao posrednik u komunikaciji čovjeka sa strojem. Kao što je ranije spomenuto, izrađuje se u alatu *WinCC*. *HMI* omogućuje vizualizaciju toka izvođenja raznih procesa, vizualizaciju rada nekog stroja i mogućnost upravljanja nekim strojem. Može biti prikazano na računalu, tabletu ili raznim panelima što je vrlo čest slučaj u industriji, a *TIA* Portal omogućuje izradu *HMI* sučelja za zaslone veličina od 3 do 22 inča.

Jedan *HMI* može sadržavati jedan ili više zaslona. Potrebno je definirati početni zaslon (engl. *root screen*) koji će se prikazati prilikom pokretanja *HMI*-a. Prelazak u druge zaslone potrebno je definirati gumbovima na početnom zaslonu. *TIA* Portal nudi mogućnost korištenja različitih objekata na zaslonima *HMI*-a, od osnovnih geometrijskih oblika poput pravokutnika, kružnice i linije, zatim teksta, gumbova i sklopki do gotovih grafičkih prikaza industrijskih dijelova kao što su senzori,

motori, pumpe, ventili pa čak i sama pokretna traka. Objekti se dodaju na zaslon *drag-and-drop* metodom.

Svakom objektu na zaslonu može se dodijeliti odgovarajuća oznaka (engl. *tag*) koja je referenca na varijablu iz programa na *PLC*-u. Pomoću te oznake, objekt se može animirati. Primjerice kada je vrijednost neke logičke varijable 0 (*false*), objekt postane nevidljiv, a kada se vrijednost varijable promijeni, objekt opet postane vidljiv. Također, moguće je i prikazati vertikalne, horizontalne ili dijagonalne kretnje. Za određenu vrijednost reference, objekt će se na zaslonu kretati u zadanom smjeru.

Strojem je moguće upravljati preko *HMI* sučelja. To je ostvareno gumbovima i sklopkama kojima su dodijeljene oznake iz programa što predstavljaju stvarne gumbove i sklopke na strojevima. Zbog toga u velikim industrijskim pogonima postoje posebne prostorije u kojima se prati i kontrolira sustav na *HMI* sučeljima. Za slučaj da nešto sa strojem nije u redu, sigurnije je iz daljine poduzeti određene mjere, nego da operater zadužen za upravljanje strojem prilazi stroju i pokušava riješiti problem.



*Slika 2.12. HMI kontrolna soba*

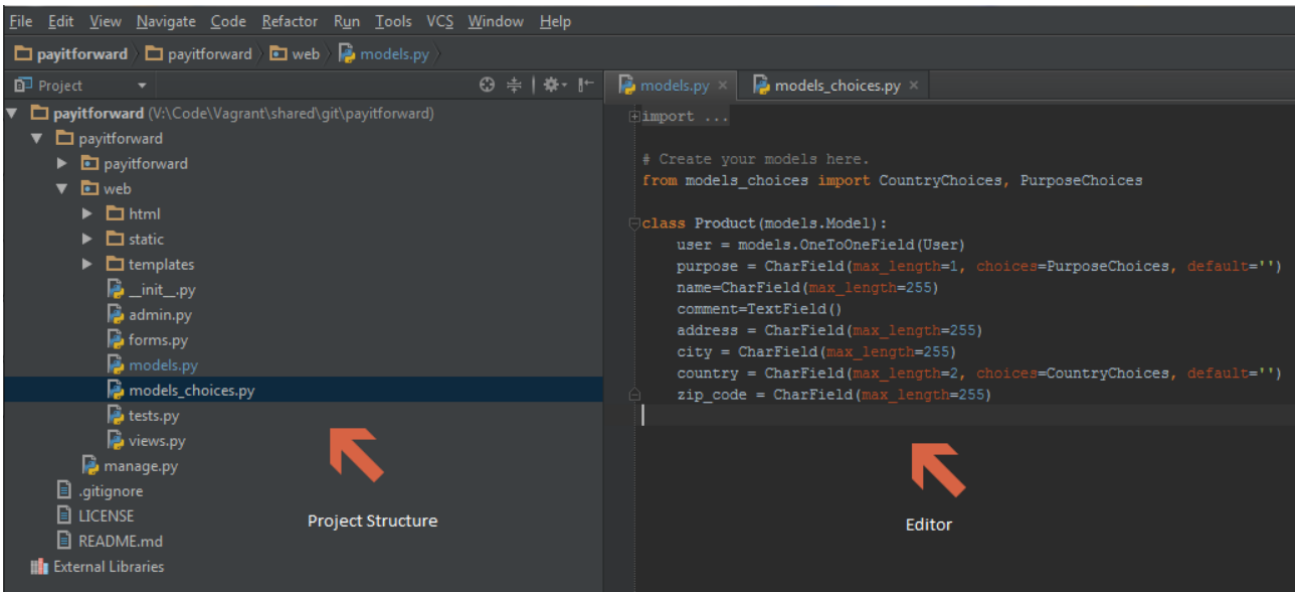
## 2.5. PyCharm

*PyCharm* je integrirano razvojno okruženje (engl. *integrated development environment - IDE*) korišteno za skriptne jezike *Pythona*. Napravila ga je Češka tvrtka *JetBrains*. Svojstva koja podržava *PyCharm* su:

- automatsko dovršavanje koda (engl. *code completion*)
- napredno otklanjanje pogrešaka (engl. *advanced debugging*)
- refaktoriranje koda
- podrška za programiranje web stranica

*PyCharm* je besplatan za preuzimanje i može se pronaći na službenim stranicama *JetBrains*-a.

Sučelje izgledom je nalik na *Android Studio*, a prikazano je na slici ispod.




*Slika 2.13. Sučelje PyCharm alata*

## 2.6. MongoDB

*MongoDB* je baza podataka koja je besplatna za korištenje. Poziva se iz *Command Prompta* i omogućuje spremanje podataka iz programa *PLC*-a u bazu.

## 2.7. Landef

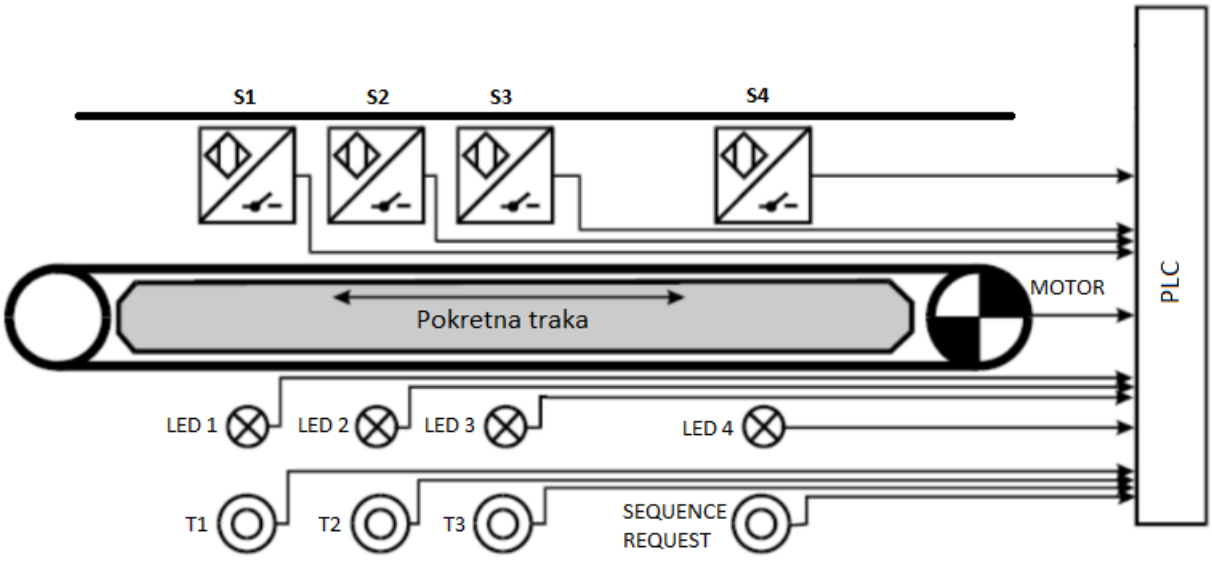
Landef je *Excel* tablica koja definira komunikaciju između *PLC*-a i *HMI*-a. U njoj se nalaze podatci koji se koriste na *HMI* sučelju te njihova veza s odgovarajućim podatcima na *PLC*-u. Vrlo je važno voditi računa o ažuriranosti Landefa. Svaka promjena na *PLC*-u mora se također promijeniti i u Landefu.

Landef tablica ovog završnog rada priložena je na CD-u.

# 3. POKRETNA TRAKA

Stvarna pokretna traka korištena za izradu ovog završnog rada prikazana je na slici 3.1., a na slici 3.2. prikazana je shema pokretne trake.


*Slika 3.1. Pokretna traka*




*Slika 3.2. Shema pokretne trake*

Kao što je prikazano na prethodnim slikama, pokretna traka sastoji se od:

- četiri senzora (*S1, S2, S3, S4*)
- četiri tipkala (*T1, T2, T3, SEQUENCE REQUEST*)
- četiri *LED* žaruljice (*LED1, LED2, LED3 i LED4*)
- objekta
- dodatnih tipkala *ON* i *OFF* kojima se pali i gasi pokretna traka te dodatne *LED* žaruljice (*H5*) koja svijetli kada je pokretna traka uključena (*ON*)

Pokretna traka mora imati mogućnost praćenja (engl. *tracking*) više proizvoda koji se istovremeno nalaze na traci. Traka ima četiri senzora, što znači da postoji sedam različitih pozicija na kojima se objekt može nalaziti. Pozicije su prikazane tablicom 3.1.

*Tablica 3.1. Pozicije objekata na pokretnoj traci*

| S1 | | S2 | | S3 | | S4 |
|---|---|---|---|---|---|---|
| pozicija 1 | pozicija 2 | pozicija 3 | pozicija 4 | pozicija 5 | pozicija 6 | pozicija 7 |

Kada promatramo pokretnu traku na slici 3.1., kretanje trake slijeva na desno označeno je kao kretanje trake prema naprijed (*FWD* – engl. *forward*), a obrnuto kretanje je prema nazad (*REV* – engl. *reverse*). Prema toj logici, objekti dolaze slijeve strane (predmontažni dio) i kreću se prema desnoj strani (postmontažni dio). Ulazi u *PLC* će biti senzori i tipkala, a izlazi će biti vrtnja motora i *LED* žaruljice.

Tipkala *T1, T2 i T3* pokreću traku prema naprijed ako nema objekta na pokretnoj traci. Dolaskom objekta na poziciju *T1, T2* ili *T3* traka se zaustavlja. Primjerice, stisnuta je tipka *T2*, traka će se kretati naprijed sve dok objekt ne dođe do pozicije *T2*. U slučaju da je stisnuta tipka *T2*, a objekt se nalazi na poziciji *T3*, traka će krenuti prema nazad dok ne transportira objekt do pozicije *T2*. Kako bi pokretna traka znala koji smjer mora koristiti da bi transportirala proizvod do određene pozicije koriste se senzori.

Tipkalo *SEQUENCE REQUEST* služi za pozivanje sekvence. Sekvenca se može izvršiti samo ako se jedan objekt nalazi na traci i to na poziciji 1. Koraci sekvence su:

1. transportiraj objekt na poziciju 5, čekaj 3 sekunde čekaj tri sekunde,
2. transportiraj objekt na poziciju 3, čekaj 3 sekunde čekaj tri sekunde,

3. transportiraj objekt na poziciju 7, čekaj 3 sekunde čekaj tri sekunde,
4. transportiraj objekt na poziciju 1

Dodatni uvjet za izvršavanje sekvence zahtjeva automatski način rada pokretne trake. Zbog ograničenosti tipkala na samoj pokretnoj traci, ne može se direktno na njoj promijeniti način rada. Dodatne funkcije nalaze se na *HMI* sučelju za upravljanje pokretnom trakom.

## 3.1. Zadatak upravljanja pokretnom trakom

Pokretna traka ima dva smjera kretanja: prema naprijed i prema nazad. Traka mora imati tri načina rada: ručni (*manual*), poluautomatski (*semiautomatic*) i automatski (*automatic*). Tipke za pokretanje trake prema naprijed ili nazad, te tipke za promjenu načina rada nalaze se na *HMI* sučelju. Aktiviranjem jednog načina rada, automatski isključuje prethodni način rada. Ručni način rada radi na principu da se pokretna traka kreće u određenom smjeru dokle god je tipka za kretanje trake prema naprijed ili nazad pritisnuta. Puštanjem tipke, traka se zaustavlja. Kada je upaljen poluautomatski način rada, pritiskom na tipku za kretanje trake, traka će se kretati u određenom smjeru dokle god objekt ne stigne do bilo kojeg senzora. Kada se senzor upali, traka se zaustavlja. Ako je upaljen automatski način rada, pritiskom na tipku za kretanje, traka će se kretati u određenom smjeru sve dok se ne pritisne tipka za drugi smjer koja će ujedno nastaviti kretanje trake mijenjajuću joj pritom smjer. Traka se zaustavlja gašenjem automatskog načina rada tako da se upali ručni ili poluautomatski način rada. Automatski način rada također omogućava izvršavanje sekvence kao što je spomenuto u prethodnom poglavlju (str. 16).

Kretanje pokretne trake prilikom pritiska na tipku *T1, T2* ili *T3* popraćeno je treperenjem odgovarajuće žaruljice *LED1, LED2 ili LED3* koja se nalazi uz pritisnutu tipku. Također , ako je sekvenca aktivna, treperit će žaruljica *LED4*.

Svaki stroj mora imati tipku za gašenje stroja pa tako i pokretna traka. Tipka *STOP* će zaustaviti traku bez obzira u kojem smjeru se traka kreće, koji je način rada upaljen ili ako je upaljena sekvenca. Traka se obavezno zaustavlja i pritom trepere sve četiri žaruljice istovremeno. Ponovnim pritiskom na tipku *STOP*, traka se može ponovno koristiti za normalan rad i žaruljice prestaju treperiti. Zbog nedovoljno tipki na stvarnoj pokretnoj traci, tipka za *STOP* nalazi se na *HMI* sučelju.

# 4. PROGRAM UPRAVLJANJA POKRETNOM TRAKOM

U ovom poglavlju je prikazano programsko rješenje za upravljanje pokretnom trakom. Najprije će biti objašnjeno programsko rješenje (engl. *software*) za upravljanje pokretnom trakom, a nakon toga rješenje vizualizacije sustava.

## 4.1. Programsko rješenje

Program se izvodi na *PLC* [*CPU 1214 DC/DC/DC*] uređaju, a pisan je uglavnom *LAD* jezikom, uz ponešto *SCL*-a. Glavni program je napisan u *FC* blokovima, a *OB*1 sadrži samo pozive tih blokova. Programski blokovi su grupirani po funkcionalnosti u više mapa radi preglednosti. Nazivi mapa su: *I/O MAP, MOTOR, LEDs, TRACKING, REMOVE OBJECT, SEQUENCE, HMI, MAPPING*, *MODBUS* i *ALARMS*. U nastavku slijede funkcionalnosti i rješenja za svaku mapu pojedinačno.

### 4.1.1. *I/O MAP*

Mapa *I/O MAP* sadrži *DB* blokove u kojima se nalaze ulazi i izlazi pokretne trake. Kao što je već spomenuto, ulazi su senzori i tipkala, a izlazi *LED* žaruljice i motor.

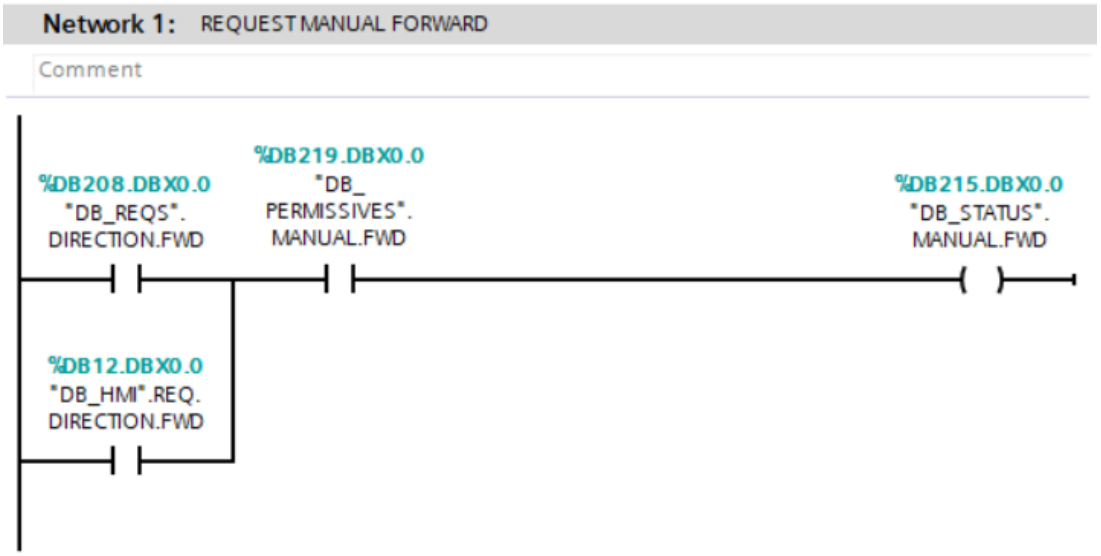
### 4.1.2. *MOTOR*

Mapa *MOTOR* sadrži programske blokove za upravljanje motorom pokretne trake, a upravljanje se odvija na sljedeći način. Pritiskom tipke za pokretanje trake u nekom smjeru stvara se zahtjev (engl. *request*) za pokretanje trake. Zahtjev se predaje statusu, a status se predaje komandi (engl. *command*). Komanda je ta koja motoru naređuje i pokreće ga. Za svaki zahtjev postoje određeni uvjeti (engl. *permissive*) koji moraju biti zadovoljeni da bi se signal prenio od zahtjeva do izvršne komande. U nastavku su dani dijelovi iz programa za određene zahtjeve.

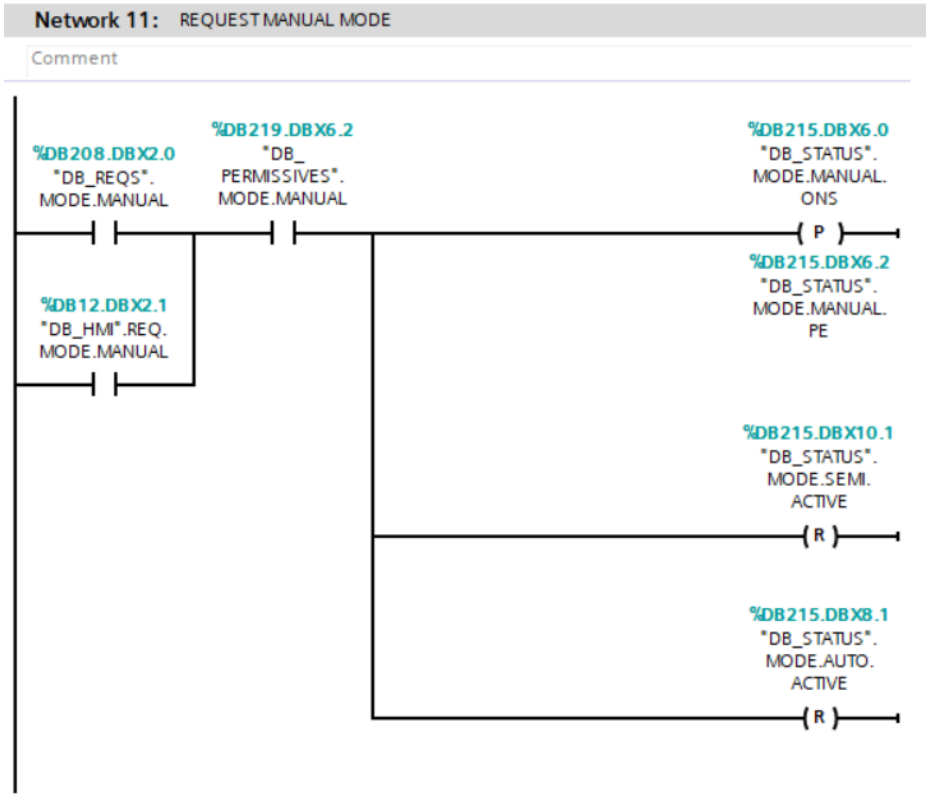
**DB_REQS**

| | Name | Data type | Offset | Start value |
|---|---|---|---|---|
| 1 | Static | | | |
| 2 | DIRECTION | Struct | 0.0 | |
| 3 | FWD | Bool | 0.0 | false |
| 4 | REV | Bool | 0.1 | false |
| 5 | MODE | Struct | 2.0 | |
| 6 | MANUAL | Bool | 2.0 | false |
| 7 | AUTO | Bool | 2.1 | false |
| 8 | SEMI | Bool | 2.2 | false |
| 9 | UNLOAD | Struct | 4.0 | |
| 10 | T1 | Bool | 4.0 | false |
| 11 | T2 | Bool | 4.1 | false |
| 12 | T3 | Bool | 4.2 | false |
| 13 | STOP | Bool | 6.0 | false |

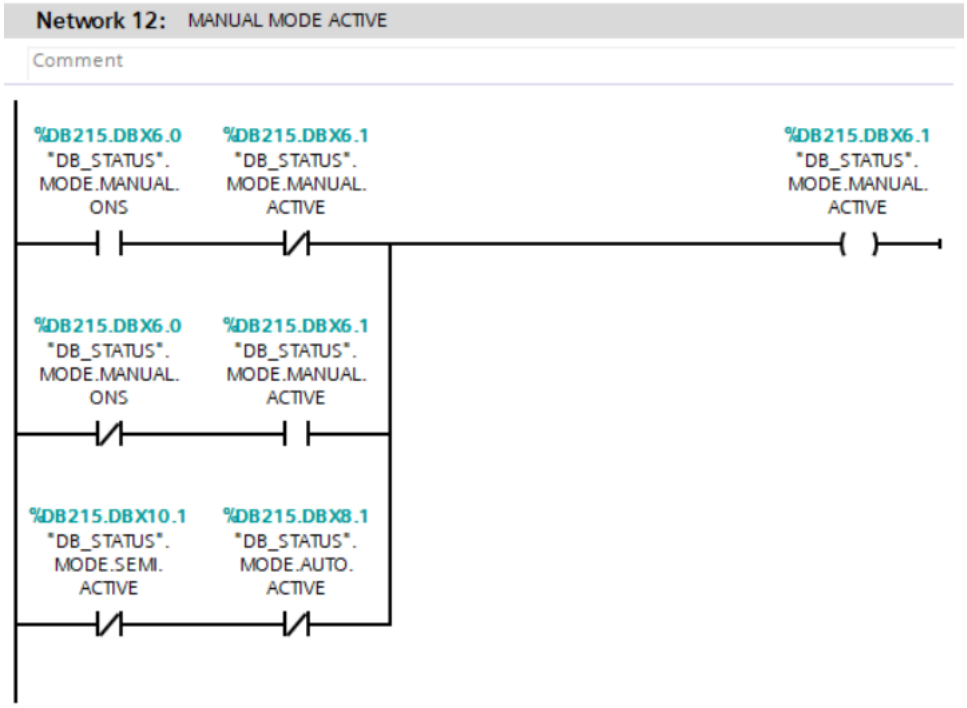
*Slika 4.1. DB blok za pohranu request varijabli*




*Slika 4.2. Zahtjev za kretanje trake prema naprijed u ručnom načinu rada*




*Slika 4.3. Zahtjev za ručni način rada*




*Slika 4.4. Održavanje ručnog načina rada aktivnim*

Na slici 4.1. prikazan je *DB* blok u kojem su spremljene varijable koje služe za zahtjeve. Varijable je moguće dodatno podijeliti unutar istog *DB*-a strukturama, što je i prikazano na slici. U tablici 4.1. prikazane su varijable koje su povezane s *PLC*-om, uz njihova objašnjenja.

*Tablica 4.1. Varijable povezane s PLC-om*

| | |
|---|---|
| *DB_REQS.DIRECTION.FWD** | zahtjev za pokretanje trake prema naprijed |
| *DB_REQS.DIRECTION.REV** | zahtjev za pokretanje trake prema nazad |
| *DB_REQS.MODE.MANUAL** | zahtjev za ručni način rada |
| *DB_REQS.MODE.AUTO** | zahtjev za automatski način rada |
| *DB_REQS.MODE.SEMI** | zahtjev za poluautomatski način rada |
| *DB_REQS.UNLOAD.T1* | zahtjev za iskrcavanje objekta na poziciji *T1* |
| *DB_REQS.UNLOAD.T2* | zahtjev za iskrcavanje objekta na poziciji *T2* |
| *DB_REQS.UNLOAD.T3* | zahtjev za iskrcavanje objekta na poziciji *T3* |
| *DB_REQS.STOP** | zahtjev za *STOP* |
| *DB_SEQ.REQ* | zahtjev za izvršavanje sekvence |
| *DB_INPUT.S1* | senzor *S1* |
| *DB_INPUT.S2* | senzor *S2* |
| *DB_INPUT.S3* | senzor *S3* |
| *DB_INPUT.S4* | senzor *S4* |
| *DB_LEDS.LED.1* | *LED* žaruljica 1 |
| *DB_LEDS.LED.2* | *LED* žaruljica 2 |
| *DB_LEDS.LED.3* | *LED* žaruljica 3 |
| *DB_LEDS.LED.4* | *LED* žaruljica 4 |
| *DB_OUTPUT.FWD* | smjer kretnje pokretne trake prema naprijed |
| *DB_OUTPUT.REV* | smjer kretnje pokretne trake prema nazad |

*zahtjevi označeni zvjezdicom(*) mogu se poslati isključivo putem *HMI*-a, jer ne postoji dovoljno tipki na stvarnoj traci. Međutim, ostavljena je mogućnost povezivanja svih zahtjeva s *PLC*-a na pokretnu traku.

Zahtjevi koji dolaze s *HMI*-a mogu se lako prepoznati jer se nalaze u *DB*-u pod nazivom *DB_HMI*, primjerice *DB_HMI.REQ.DIRECTION.FWD.*

Zahtjeve su *inputi* koje korisnik šalje *PLC*-u putem tipki na pokretnoj traci ili *HMI*-u. Senzori su *inputi* koje *PLC* šalje programu prilikom dolaska nekog objekta na poziciju senzora. *LED* žaruljice su *outputi* programirani da se aktiviraju na *PLC*-u u određenim uvjetima. Smjer kretanja pokretne trake je *output* koji na određene zahtjeve pokreće motor na stvarnoj pokretnoj traci u određenom smjeru.

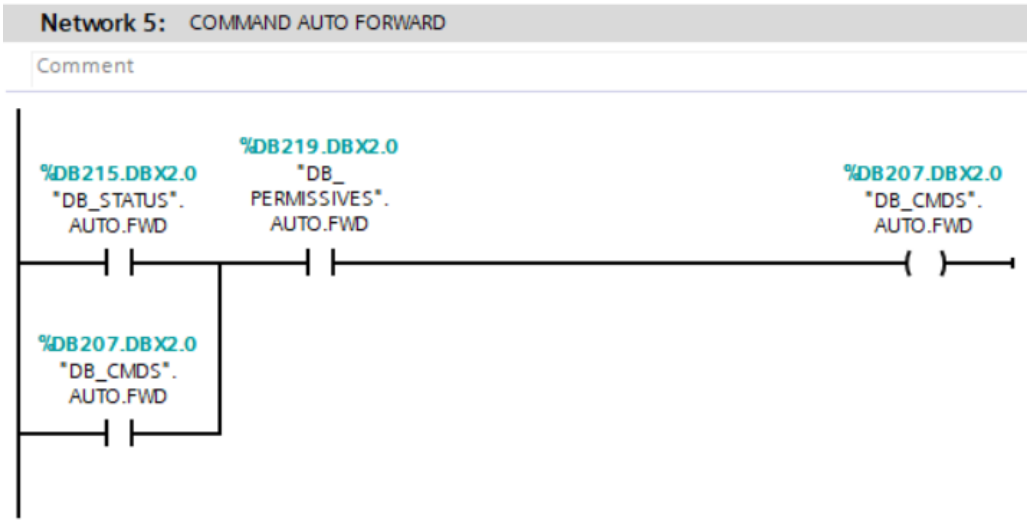
Slika 4.2. prikazuje zahtjev za pokretanje trake prema naprijed u ručnom načinu rada. Može se uočiti da postoje 2 načina za zahtjev. Prvi je zahtjev koji moguće ostvariti tipkom na stroju, a drugi je zahtjev iz *HMI* sučelja. Zahtjev će poslati signal statusu ako su uvjeti zadovoljeni. Na isti način funkcioniraju zahtjevi za pokretanje trake unazad ili u drugom načinu rada.

Na slici 4.3. prikazan je zahtjev za ručni način rada koji se nastavlja na slici 4.4. Kratica *ONS* u imenu varijable simbolizira varijablu koja će se okinuti (engl. *one shot*) samo u jednom ciklusu na rastući brid (engl. *positive edge – PE*) ili padajući brid (engl. *negative edge – NE*) signala. Prikaz rastućeg i padajućeg brida može se vidjeti na slici 4.5. Element ( P ) je instrukcija koja će postaviti bit u jednom ciklusu kada se dogodi rastući brid, a element ( N ) je obratni slučaj. Element ( R ) je instrukcija *Reset Output* koja mijenja vrijednost zadane varijable u 0 (*False*).


*Slika 4.5. Rastući i padajući brid*

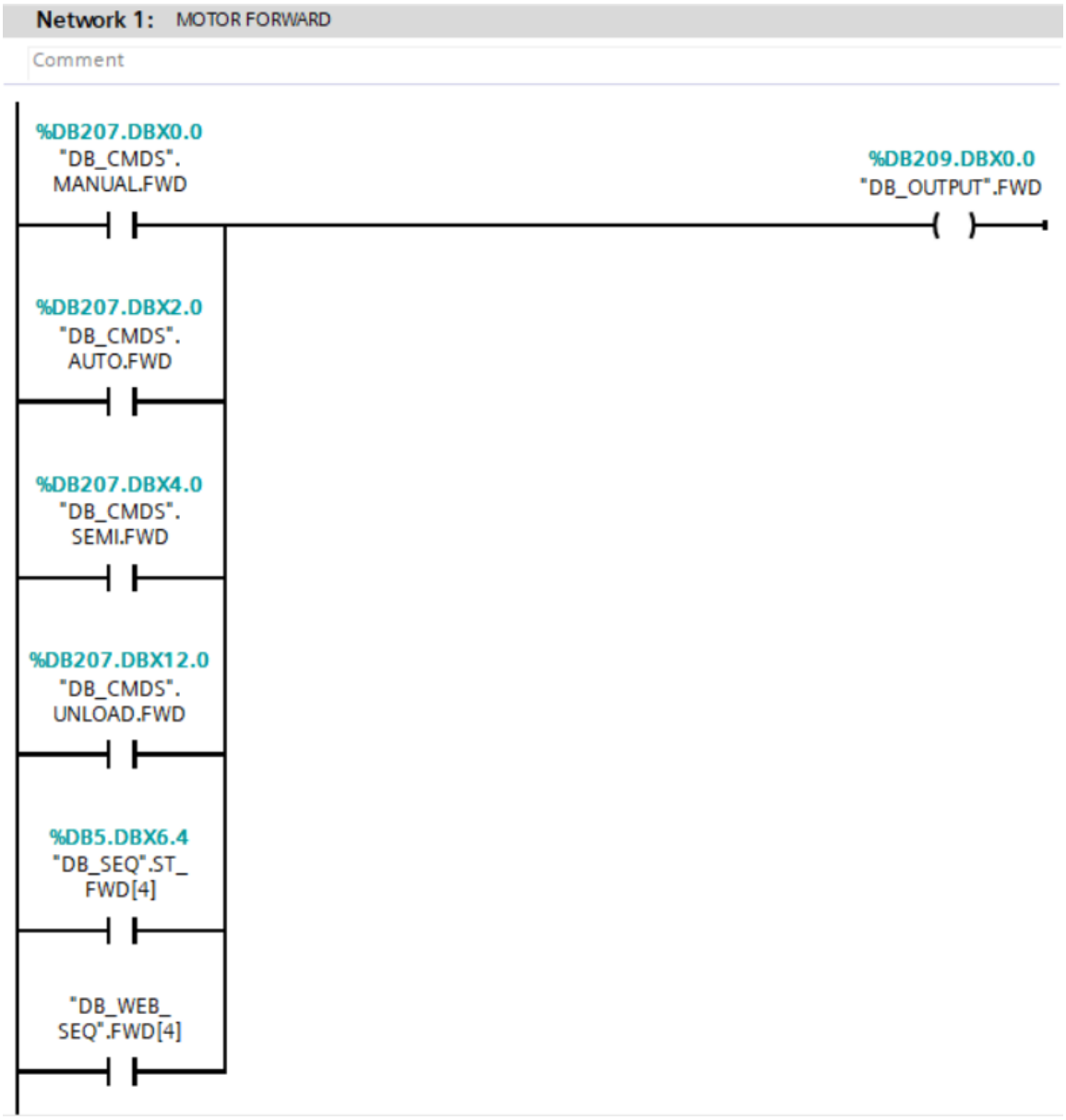
Za primjer prikazan na prethodnim slikama (4.3. i 4.4.), pretpostavimo da nije upaljen ručni način rada. Kada se pritisne tipka za ručni način rada, dogodit će se rastući brid i vrijednost *True* će biti spremljena u varijablu *"DB_STATUS".MODE.MANUAL.ONS*. U tom istom ciklusu, u *Networku 12,* prvi redak žice će propustiti signal i upaliti ručni način rada. U idućem ciklusu će se vrijednost varijable *"DB_STATUS".MODE.MANUAL.ONS* postaviti u *False*, što će aktivirati drugi redak *Networka 12* koji će održavati ručni način rada upaljenim. Ponovni pritisak na tipku za odabir ručnog načina rada ugasio bi ručni način rada, ali to se neće dogoditi zbog zadnjeg retka u *Networku 12* koji govori da ako nije uključen poluautomatski ili automatski način rada, traka mora biti u ručnom načinu rada.


*Slika 4.6. FWD komanda u ručnom načinu rada*



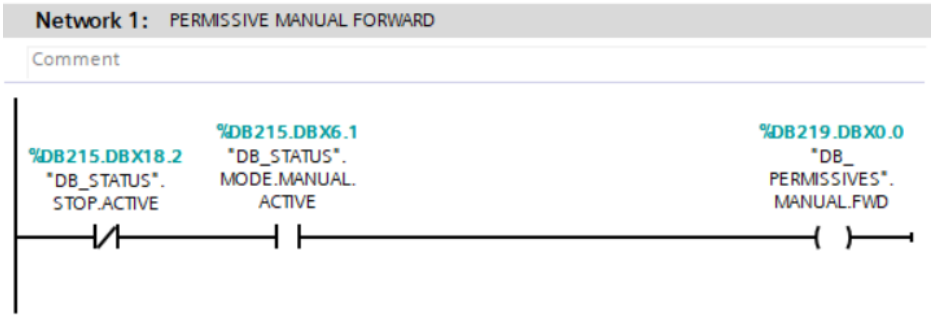

*Slika 4.7. FWD komanda u automatskom načinu rada*

Slike 4.6. i 4.7. prikazuju slanje signala iz statusa u komandu u slučaju ručnog i automatskog načina rada. Opet mora postojati uvjet koji će dozvoliti slanje signala u komandu. Za razliku od ručnog načina rada, automatski način podrazumijeva „samoodržavanje“ komande, što znači da će komanda samu sebe održavati aktivnom i pritom pokretati traku u određenom smjeru i kada tipka za kretanje trake nije stisnuta.

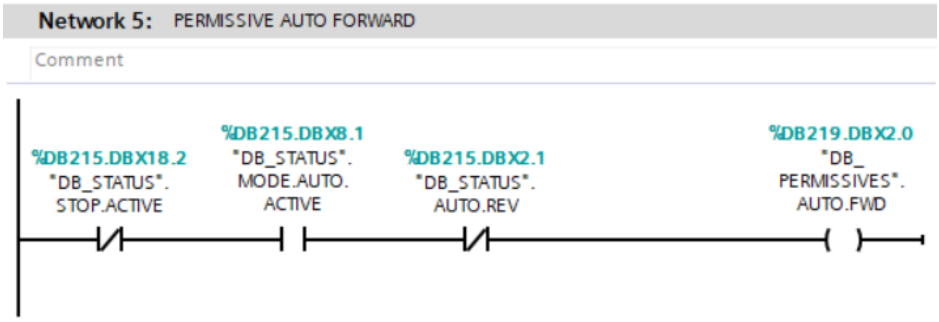



*Slika 4.8. Pokretanje motora komandama*

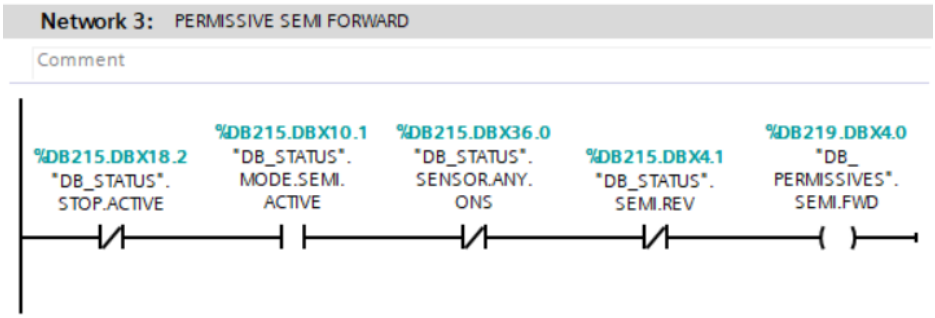
Na slici iznad, prikazane su sve komande koje pokreću motor prema naprijed; varijabla *"DB_OUTPUT".FWD* predstavlja motor. Prve tri komande su pokretanje motora u različitim načinima rada. Četvrta komanda je *UNLOAD* koja će biti objašnjena kasnije u radu. Komande *DB_SEQ.ST_FWD[4]* i *DB_WEB_SEQ.FWD[4]* su koraci sekvenci koji pokreću traku prema naprijed.




*Slika 4.9. Uvjeti za kretanje trake naprijed u ručnom načinu rada*




*Slika 4.10. Uvjeti za kretanje trake naprijed u ručnom automatskom rada*




*Slika 4.11. Uvjeti za kretanje trake naprijed u poluautomatskom načinu rada*

Na gornjim slikama prikazani su uvjeti koji omogućuju izvršavanje komande.

Varijabla *"DB_STATUS".SENSOR.ANY.ONS* predstavlja okidač na rastući brid bilo kojeg senzora.

Vrlo je važno da svaka komanda ima u sebi uvjet za *STOP*. Kada je *STOP* aktivan, stroj se mora zaustaviti i nijedna komanda ne smije biti aktivna.

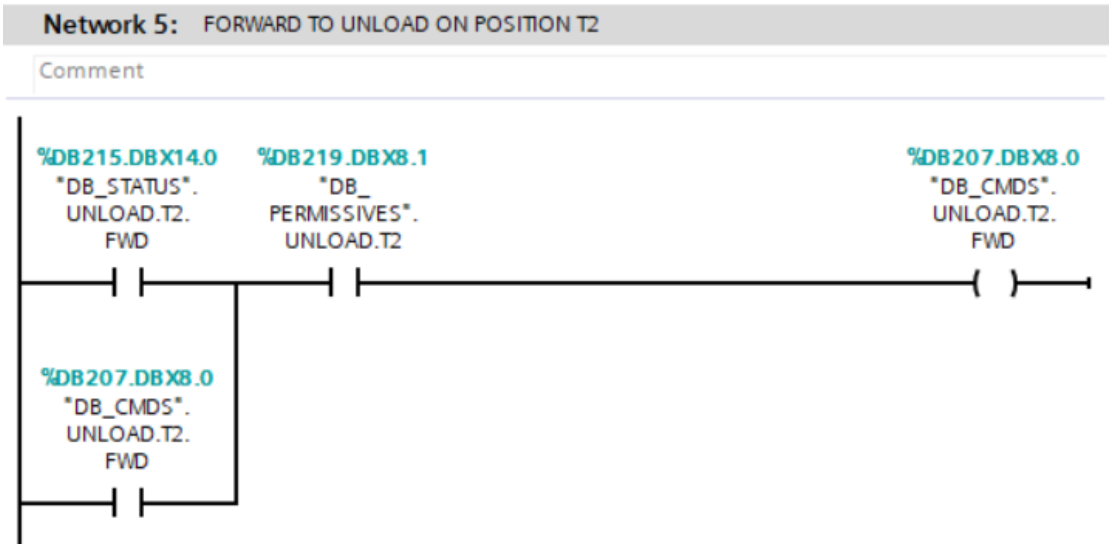
Network 4: REQUEST UNLOAD ON POSITION T2

Comment

```
IF "DB_PERMISSIVES".UNLOAD.T2 = TRUE
THEN
    IF "DB_REQS".UNLOAD.T2 = TRUE OR "DB_HMI".REQ.UNLOAD.T2 = TRUE
    THEN
        IF "DB_TRACKING".POSITION4 = TRUE OR
            ("DB_TRACKING".POSITION5 = TRUE AND "DB_TRACKING".POSITION2 = FALSE) OR
            ("DB_TRACKING".POSITION6 = TRUE AND "DB_TRACKING".POSITION2 = FALSE AND "DB_TRACKING".POSITION1 = FALSE) OR
            ("DB_TRACKING".POSITION7 = TRUE AND "DB_TRACKING".POSITION2 = FALSE AND "DB_TRACKING".POSITION1 = FALSE)
        THEN
            "DB_STATUS".UNLOAD.T2.REV := TRUE;
        END_IF;

        IF ("DB_TRACKING".POSITION2 = TRUE AND "DB_TRACKING".POSITION4 = FALSE) OR
            ("DB_TRACKING".POSITION1 = TRUE AND "DB_TRACKING".POSITION4 = FALSE AND "DB_TRACKING".POSITION5 = FALSE) OR
            ("DB_TRACKING".POSITION4 = FALSE AND "DB_TRACKING".POSITION5 = FALSE AND "DB_TRACKING".POSITION6 = FALSE AND
            "DB_TRACKING".POSITION7 = FALSE)
        THEN
            "DB_STATUS".UNLOAD.T2.FWD := TRUE;
        END_IF;
    END_IF;
ELSE
    "DB_STATUS".UNLOAD.T2.REV := FALSE;
    "DB_STATUS".UNLOAD.T2.FWD := FALSE;
END_IF;
```

*Slika 4.12. Zahtjev za iskrcavanje na poziciji T2*

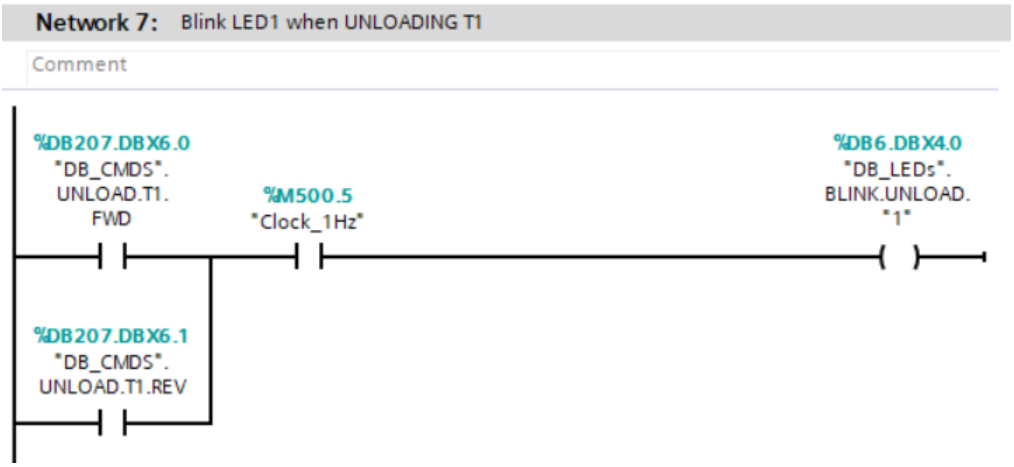



*Slika 4.13. Komanda za iskrcavanje na poziciji T2*

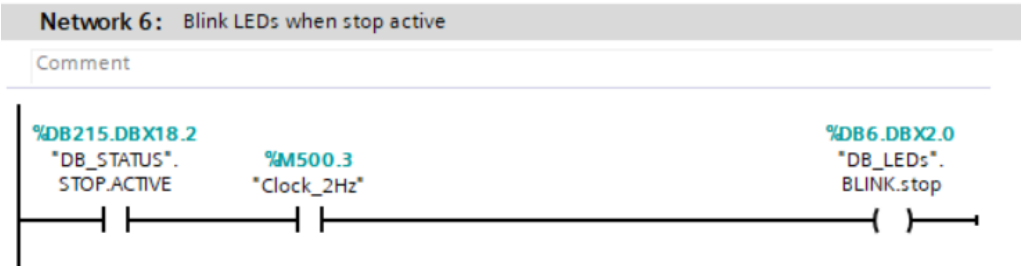
Zahtjevi za iskrcavanje (engl. *unload*) napisani su *SCL* jezikom zbog velike količine varijabli i složenosti algoritma. Iskrcavanje se zahtijeva tipkama *T1*, *T2* i *T3*, iskrcavajući pritom objekt najbliži odabranoj poziciji iskrcavanja.

### 4.1.3. *LEDs*

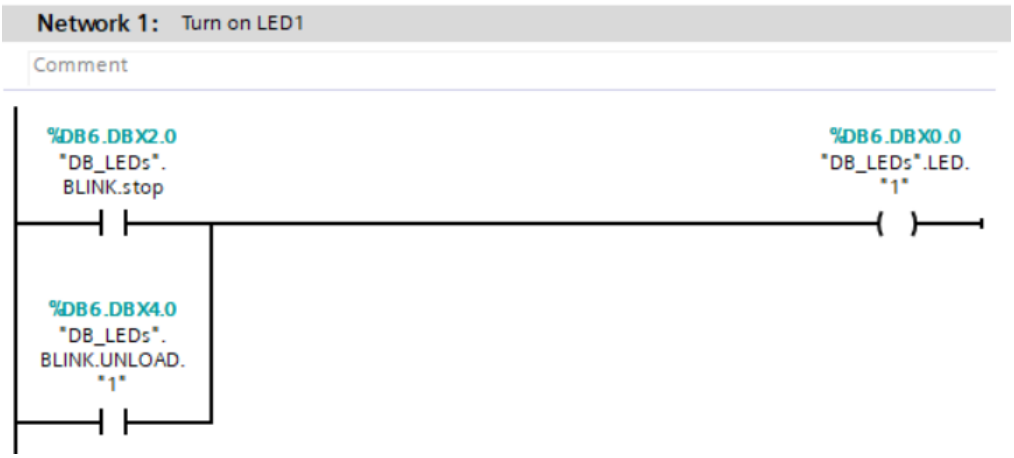
Mapa *LEDs* sadrži programske blokove koji upravljaju ponašanjem *LED* žaruljica na pokretnoj traci.




*Slika 4.14. Treptanje LED žaruljice pri iskrcavanju objekta na poziciji T1*




*Slika 4.15. Treptanje LED žaruljice kada je stop aktiviran*

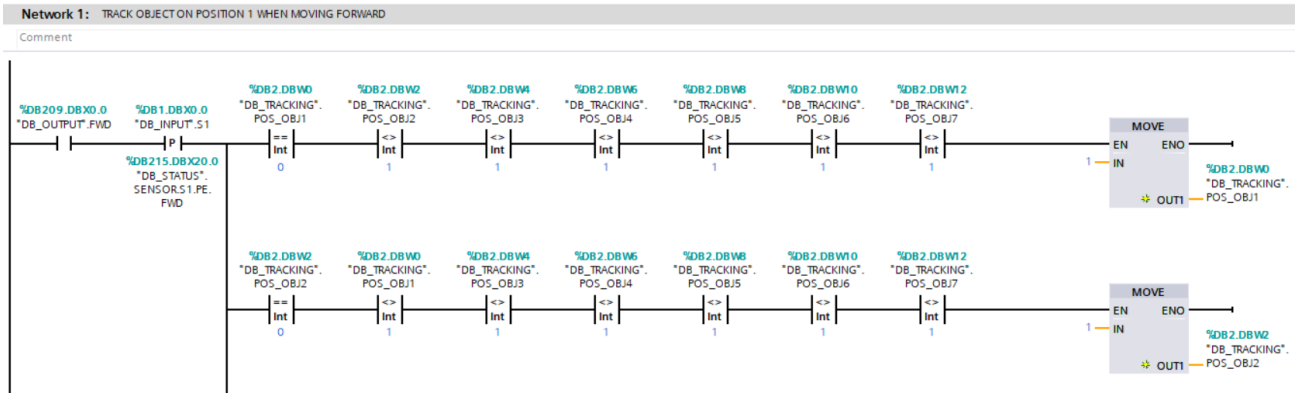



*Slika 4.16. Paljenje LED1 žaruljice*

Treptanje *LED* žaruljica ostvareno je koristeći posebne memorijske bitove (*clock memory bit*) koji se nalaze unutar *PLC*-a. Svrha tih bitova je promjena stanja bita, a moguće je koristiti bitove s frekvencijama izmjene stanja od 10Hz, 5Hz, 2Hz itd.

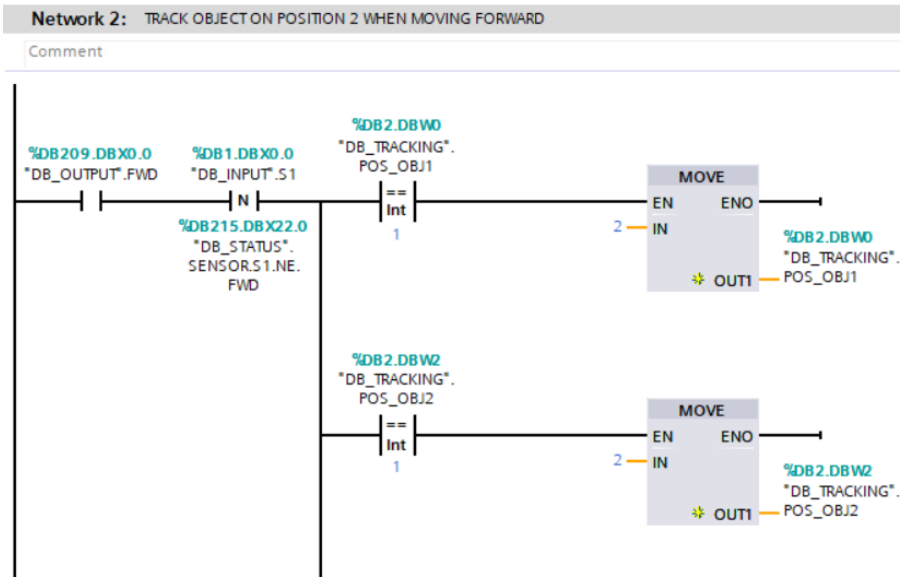
### 4.1.4. *TRACKING*

Mapa *TRACKING* rješava problem praćenja proizvoda. Algoritam je prikazan na sljedećim slikama.




*Slika 4.17. Praćenje objekta na poziciji 1 u slučaju smjera kretanja trake prema naprijed*

Zbog veličine *Networka*, prikazan je slučaj samo za prva dva objekta.




*Slika 4.18. Praćenje objekta na poziciji 2 u slučaju smjera kretanja trake prema naprijed*

Na slikama 4.17. i 4.18. jasno se vidi razlika u praćenju proizvoda na pozicijama 1 i 2. Prisjetimo se, pozicije 1, 3, 5 i 7 su na senzorima, a pozicije 2, 4 i 6 su između senzora. Prilikom dolaska objekta na poziciju 1, događa se rastući brid senzora 1. Taj senzor je mogao okinuti bilo kojih od 7 različitih objekata koje je moguće pratiti na pokretnoj traci. Zbog cikličkog izvršavanja programa odozgo prema dolje, ispituju se pozicije objekata počevši od 1 pa do 7. Na taj način će se pozicije objekata pomicati ispravnim redoslijedom. Na senzoru se može nalaziti samo jedan objekt što znači da je potrebno ispitati za svaki slučaj ako postoji objekt na poziciji 1 prije nego se pozicija nekog objekta stavi na 1. Praćenje na poziciji 2 je puno jednostavnije jer objekt koji je okinuo padajući brid senzora 1 može biti samo taj jedan objekt koji se nalazio na senzoru.

Na slici 4.19. prikazano je održavanje neke pozicije aktivnom ako se objekt nalazi na njoj. Varijable koje spremaju te vrijednosti su korisne zbog ispitivanja uvjeta u nekim drugim algoritmima.

**Network 1:** POSITION 1 ACTIVE

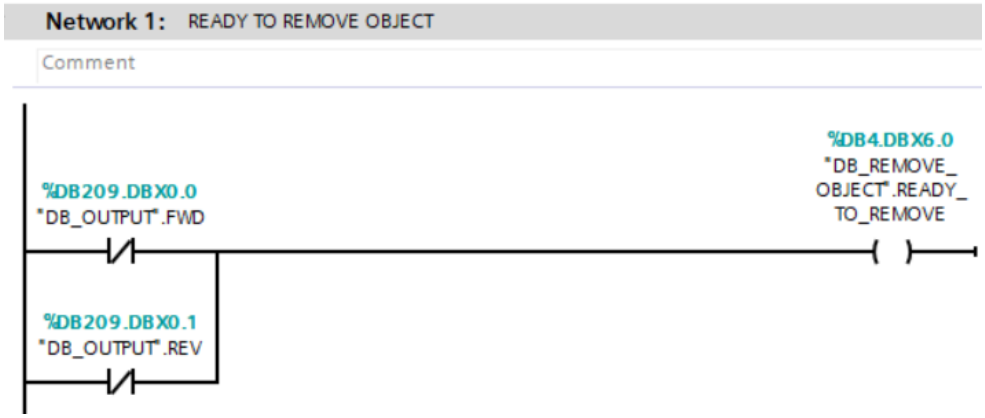
Comment

```
1  IF "DB_TRACKING".POS_OBJ1=1 OR
2      "DB_TRACKING".POS_OBJ2=1 OR
3      "DB_TRACKING".POS_OBJ3=1 OR
4      "DB_TRACKING".POS_OBJ4=1 OR
5      "DB_TRACKING".POS_OBJ5=1 OR
6      "DB_TRACKING".POS_OBJ6=1 OR
7      "DB_TRACKING".POS_OBJ7=1
8     THEN
9         "DB_TRACKING".POSITION1 := TRUE;
10
11    ELSE
12        "DB_TRACKING".POSITION1 := FALSE;
13 END_IF;
```

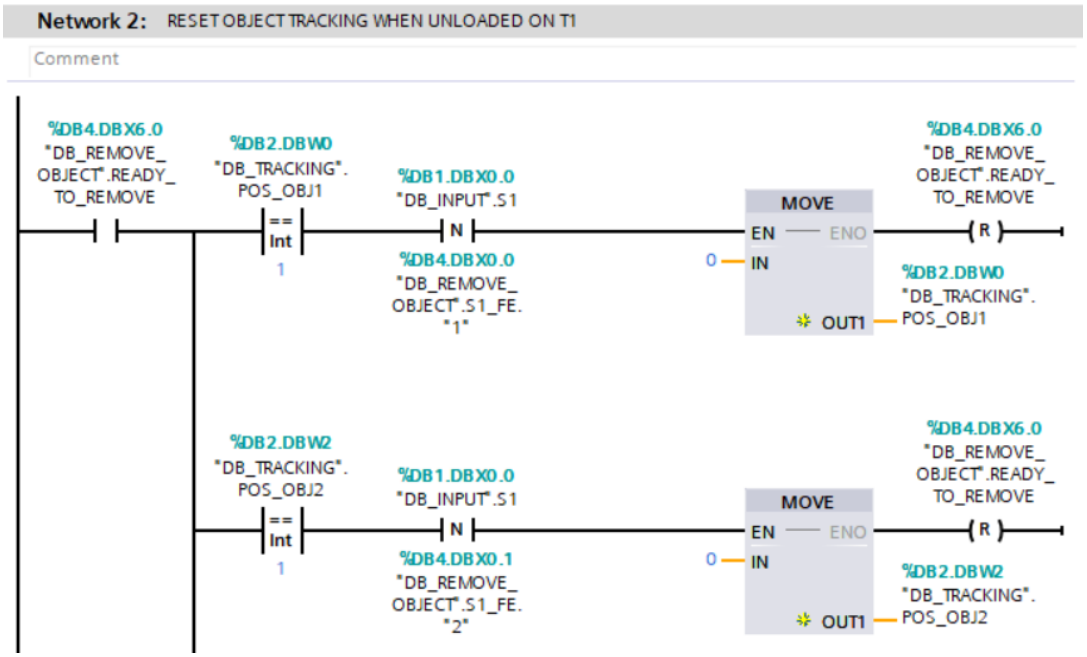
*Slika 4.19. Držanje pozicije 1 aktivnom ako se neki objekt nalazi na njoj*

### 4.1.5. *REMOVE OBJECT*

Mapa *REMOVE OBJECT* sadrži funkciju koja resetira pozicije objekata nakon što su isti uklonjeni s trake. Objekt je moguće ukloniti s trake samo ako se nalazi na jednom od senzora, inače program neće registrirati uklanjanje objekta. Također, objekt je moguće ukloniti samo ako je pokretna traka u stanju mirovanja. Na sljedećim slikama je prikazan dio algoritma koji rješava problem uklanjanja objekta.




*Slika 4.20. Uklanjanje objekta moguće samo ako traka nije u pokretu*

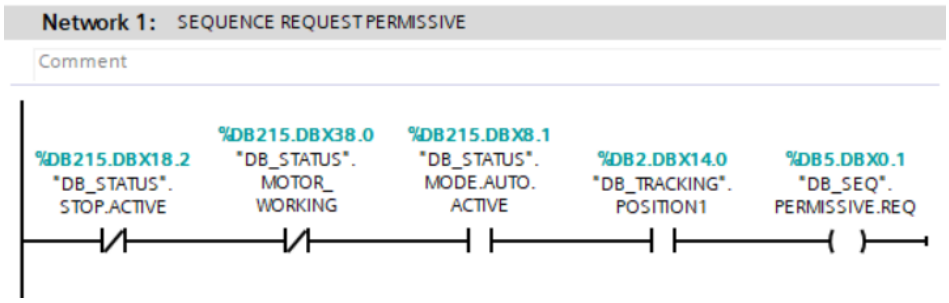



*Slika 4.21. Resetiranje pozicije objekta prilikom uklanjanja na poziciji T1*

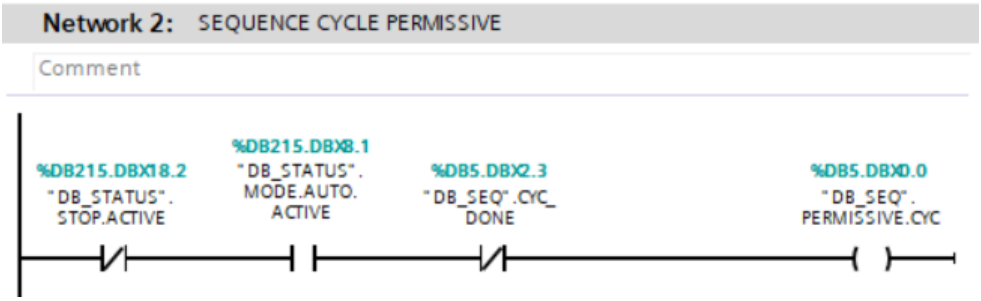
Zbog veličine *Networka*, prikazan je slučaj samo za prva dva objekta.

### 4.1.6 *SEQUENCE*

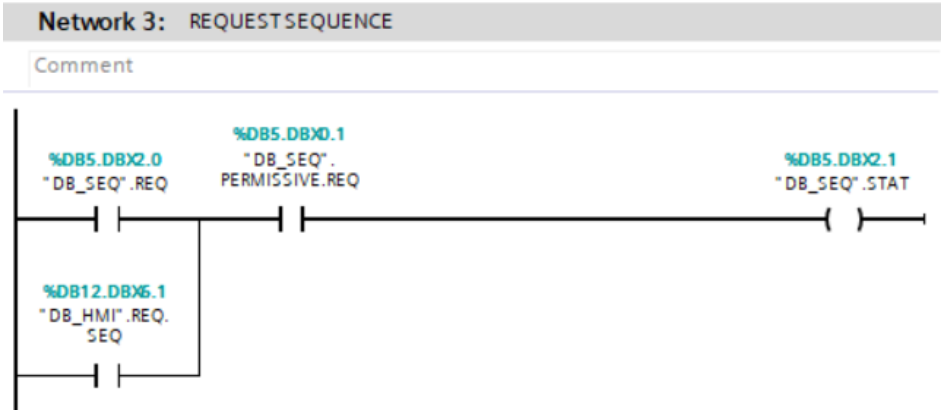
Mapa *SEQUENCE* sadrži rješenje upravljanja sekvencom. Na idućim slikama prikazan je dio *FC* bloka u kojemu je program za sekvencu.



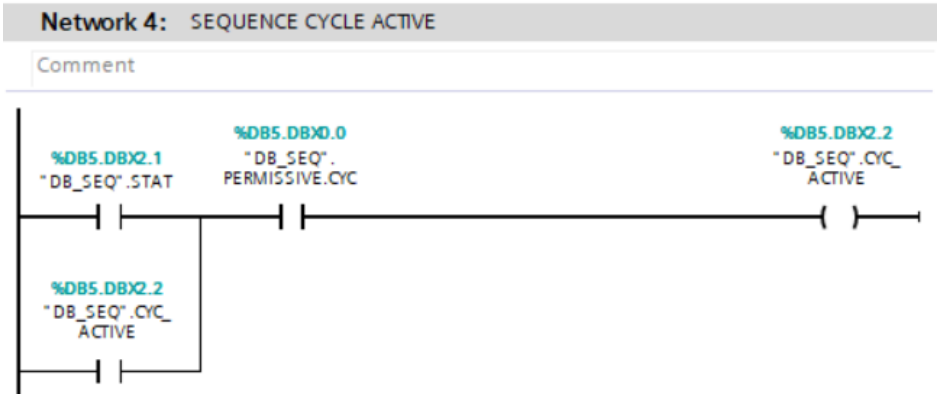

*Slika 4.22. Uvjeti za zahtjev sekvence*



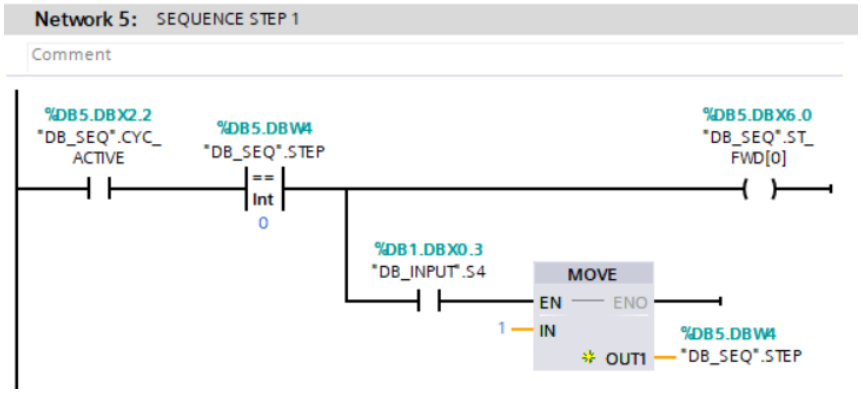

*Slika 4.23. Uvjeti za održavanje ciklusa sekvence aktivnim*



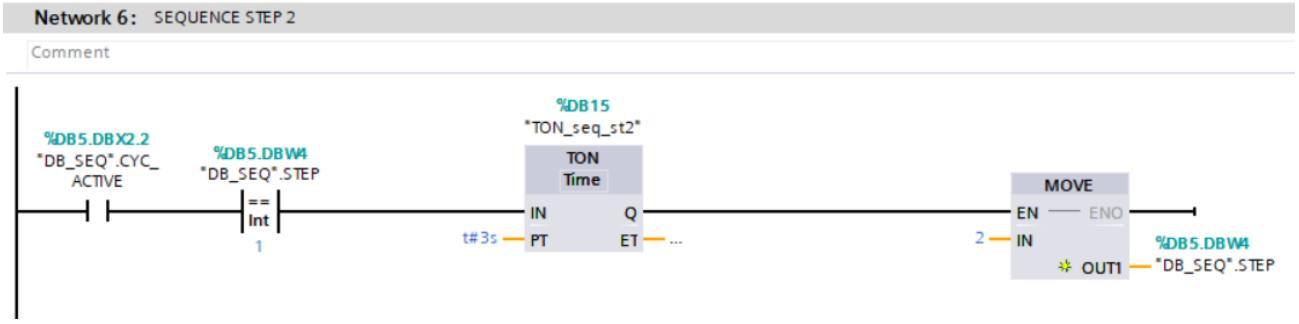

*Slika 4.24. Zahtjev za sekvencu*




*Slika 4.25. Održavanje ciklusa sekvence aktivnim*

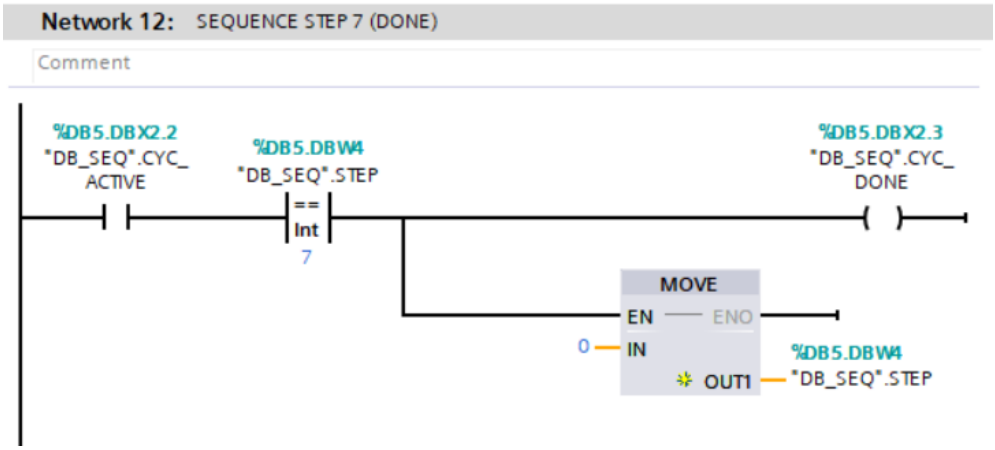



*Slika 4.26. Prvi korak sekvence*




*Slika 4.27 Drugi korak sekvence*

Element "*TON_seq_st2"* predstavlja *timer* koji odbrojava 3 sekunde, a zatim propušta signal.



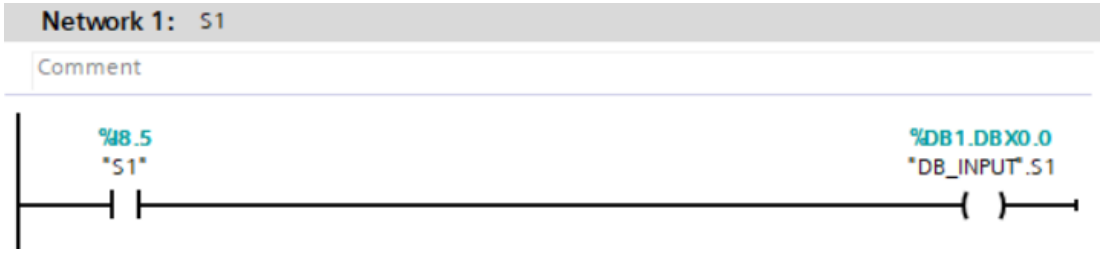

*Slika 4.28. Sedmi (zadnji) korak sekvence*

### 4.1.7. *HMI*

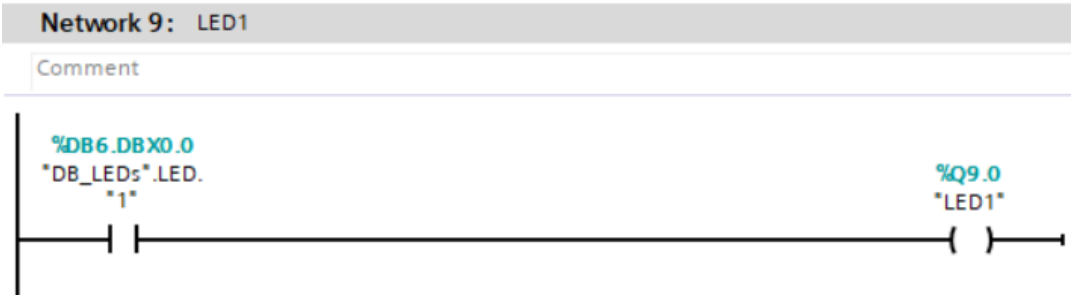
*HMI* mapa sastoji se od *DB* bloka u kojoj se nalaze zahtjevi te *FC* bloka gdje se nalazi *resetovi* na PLC-u. Omogućuje upravljanje pokretnom trakom iz *HMI* sučelja tako što su zahtjevi podijeljeni na zahtjeve s pokretne trake i zahtjeve iz sučelja.

### 4.1.8. *MAPPING*

Mapa *MAPPING* povezuje ulazne i izlazne varijable u programu sa stvarnim ulazima i izlazima na pokretnoj traci. Ulaz se označava slovom *I*, a izlaz slovom *Q* iza kojih slijedi adresa ulaza i izlaza, npr. *I4.0, Q20.3.* U nastavku je prikazan primjer mapiranja za jednu ulaznu i izlaznu varijablu, te tablica prema kojoj su povezani svi ostali ulazi i izlazi.




*Slika 4.29. Mapiranje ulazne varijable*




*Slika 4.30. Mapiranje izlazne varijable*

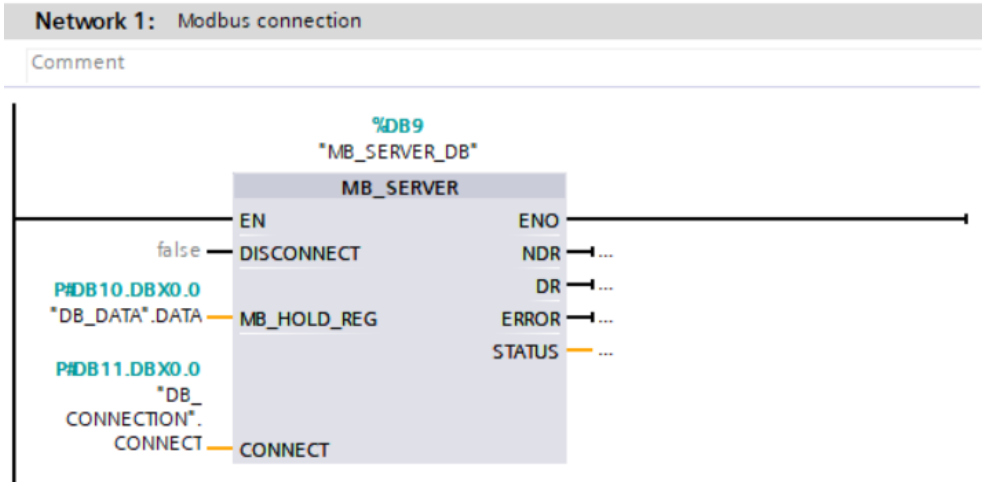
*Tablica 4.2. Adrese ulaza i izlaza*

| OPERAND | SIMBOL |
|---|---|
| ***I8.8*** | senzor *S1* |
| ***I8.6*** | senzor *S2* |
| ***I8.7*** | senzor *S3* |
| ***I8.0*** | senzor *S4* |
| ***I8.1*** | tipka *T1* |
| ***I8.2*** | tipka *T2* |
| ***I8.3*** | tipka *T3* |
| ***I8.4*** | tipka *SEQ.REQ* |
| ***Q9.0*** | žaruljica *LED1* |
| ***Q9.1*** | žaruljica *LED2* |
| ***Q9.2*** | žaruljica *LED3* |
| ***Q9.3*** | žaruljica *LED4* |
| ***Q9.5*** | *Output* kretnje motora prema naprijed (*FWD*) |
| ***Q9.6*** | *Output* kretnje motora prema naprijed (*REV*) |

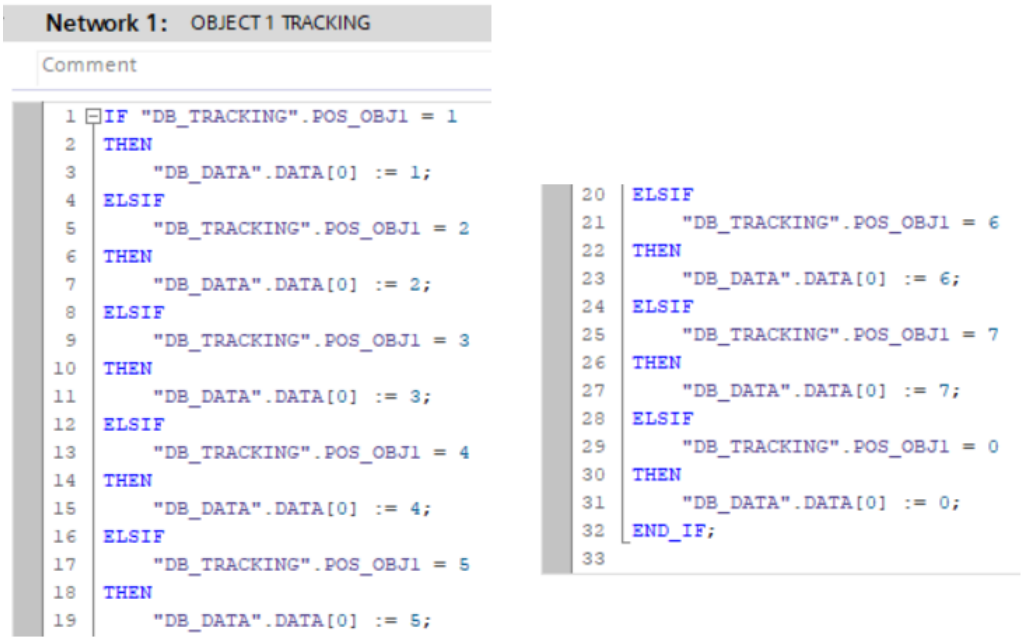
### 4.1.9. *MODBUS*

*Modbus* predstavlja serijski komunikacijski protokol korišten u programiranju *PLC* uređaja. *PLC* s7-1200 korišten za ovaj završni rad podržava Modbus *TCP* protokol, stoga je moguće napraviti *Modbus* poslužitelj (*server*) koji će pohranjivati podatke u registre. Zadana port adresa za *Modbus TCP* je 502.

U mapi *MODBUS* nalazi se ostvarena komunikacija s *Modbus*-om i spremanje podataka u registre.

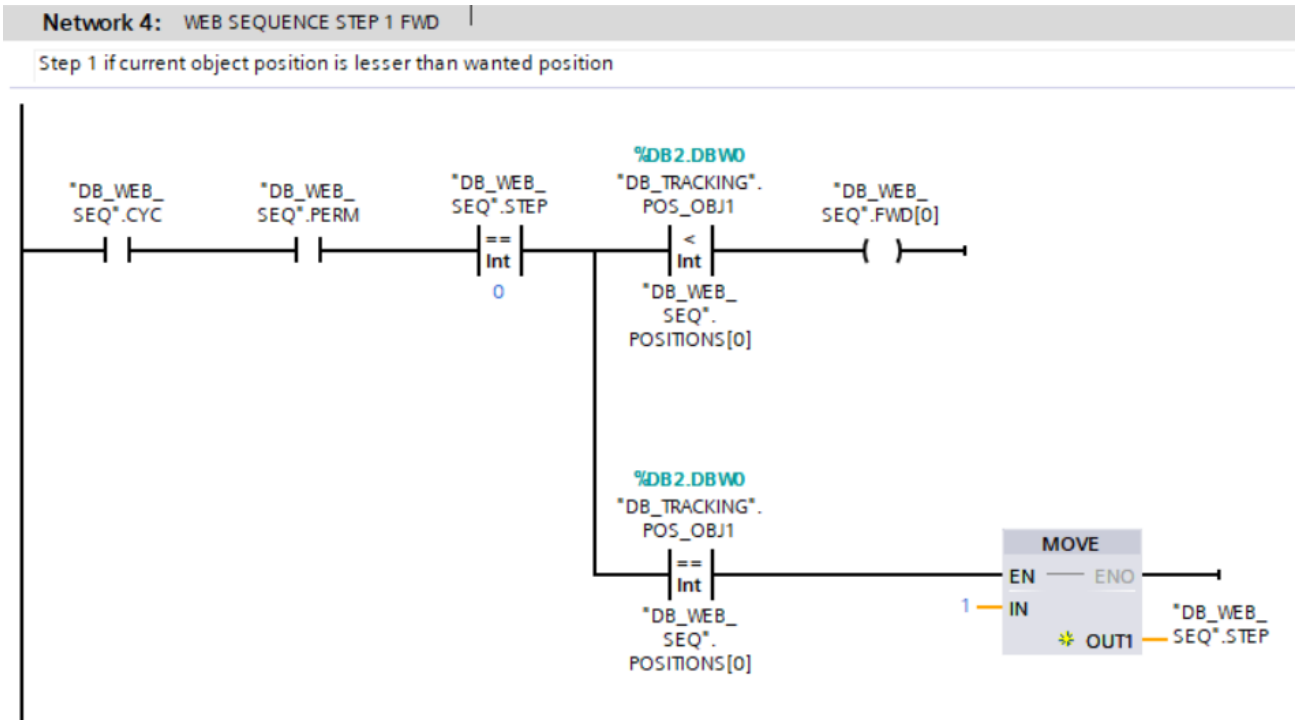



*Slika 4.31. Komunikacija s Modbus poslužiteljem*




*Slika 4.32. Spremanje podataka u Modbus registre*

Osim ostvarene komunikacije, u mapi se još nalazi funkcija za sekvencu dobivena iz web aplikacije. Na slici ispod prikazana je mreža iz te funkcije.



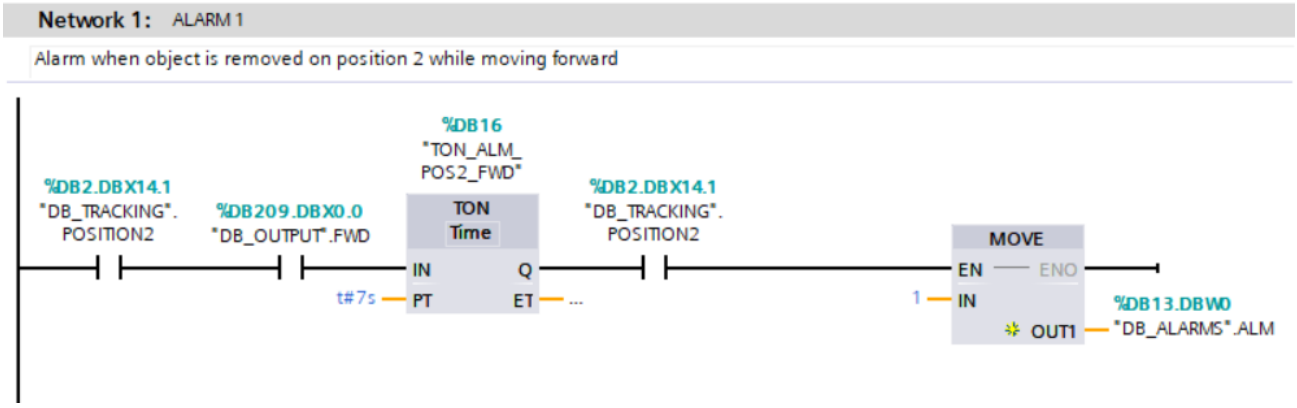

*Slika 4.33. Prvi korak web sekvence*

Web sekvenca je vrlo slična ranije spomenutoj običnoj sekvenci, osim što u web sekvenci postoje dodatni uvjeti koji omogućuju zadavanje različitih koraka sekvence.

### 4.1.10. *ALARMS*

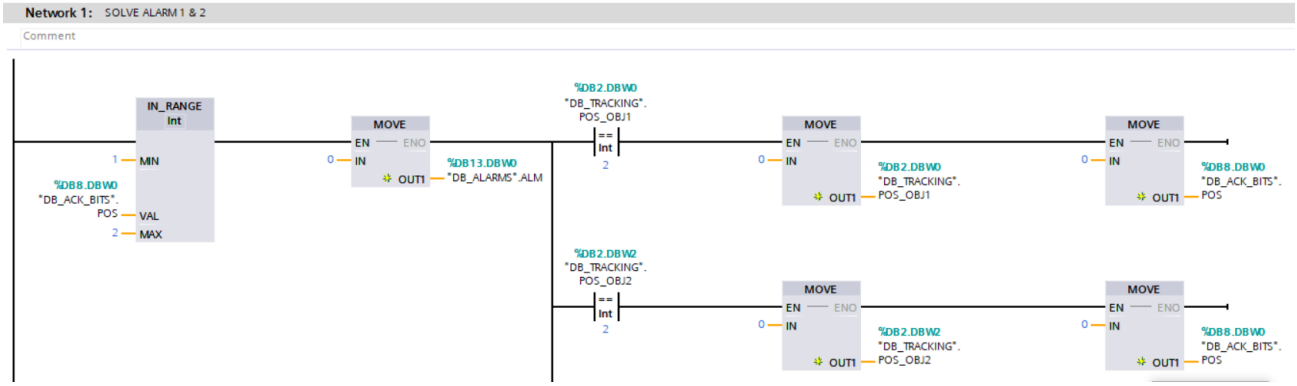
Na pokretnoj traci može se dogoditi da objekt „nestane“ između senzora. Kako objekt nije bio na senzoru u trenutku skidanja s pokretne trake, potrebno je napraviti alarm. U algoritmu je izrađen alarm koji odbrojava sedam sekundi kada je objekt sišao s nekog senzora. Ako u roku od sedam sekundi objekt ne stigne do idućeg senzora, aktivirat će se alarm i javit će grešku na *HMI* sučelju, pri čemu će se dodatno upaliti *STOP* i traka će prestati s radom. Potvrdom da je alarm viđen, skinutom objektu će se resetirati praćena vrijednost na nulu.

Na slici ispod prikazan je *ALARM 1* koji će se aktivirati ako se objekt na poziciji 2 kreće prema naprijed i u roku od 7 sekundi ne stigne na poziciju 3.




*Slika 4.34. Alarm 1*

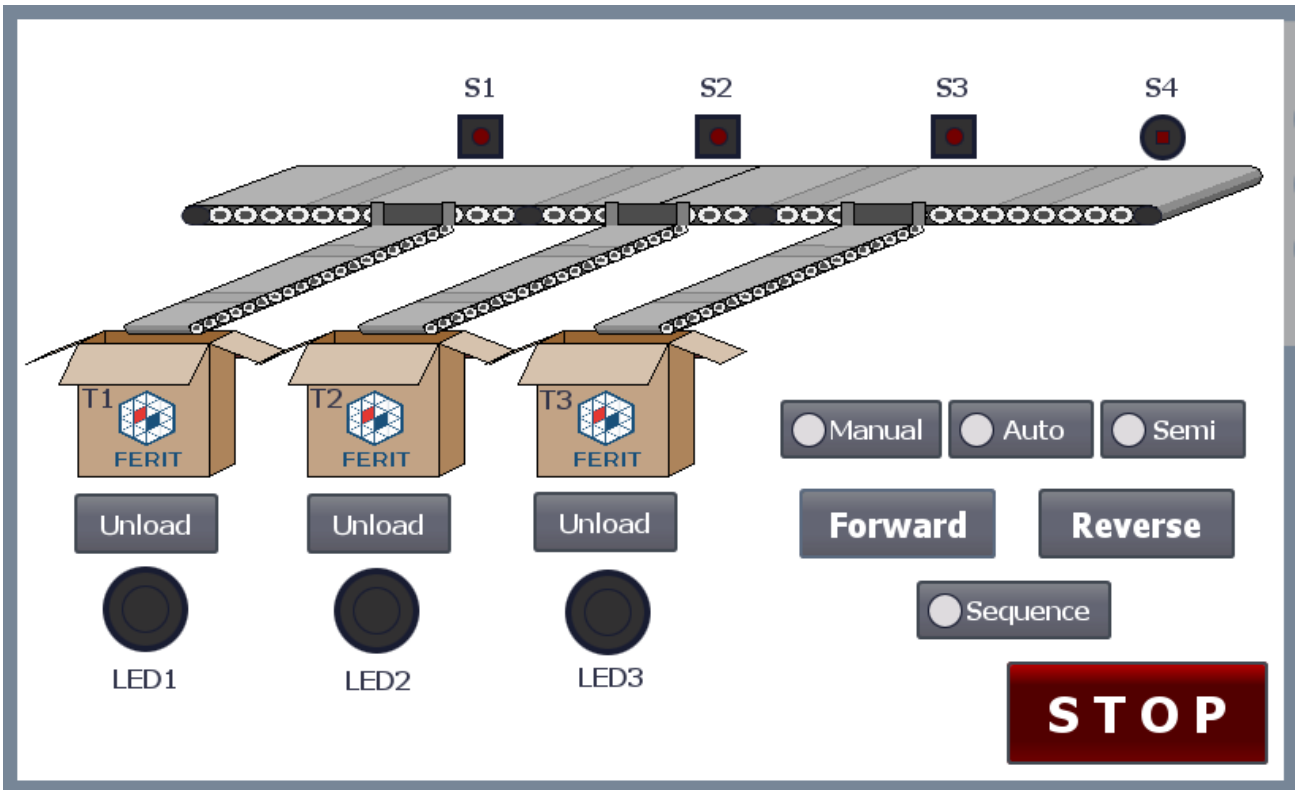
Na slici ispod prikazano je rješenje (engl. *acknowledge*) *ALARMA 1* i *2*. Klikom na potvrdu alarmu unutar *HMI*-a, resetirat će se praćena vrijednost objekta koji je upalio alarm. Na slici se mogu vidjeti rješenja za objekte 1 i 2.




*Slika 4.35. Potvrđivanje alarma*

## 4.2. Vizualizacija

Vizualizacija je napravljena u *WinCC* alatu unutar *TIA* Portala. *HMI* sučelje za vizualizaciju pokretne trake je prikazano na slici ispod.




*Slika 4.36. HMI sučelje za pokretnu traku*

Na prikazanom *HMI* sučelju moguće je pratiti smjer kretanja pokretne trake. Uz tipke *Manual, Auto* i *Semi* nalaze se mali simboli žaruljica koji će pokazivati način rada pokretne trake. Također, svaki objekt koji se nalazi na pokretnoj traci bit će prikazan i na *HMI*-u nalazeći se na odgovarajućoj poziciji.

## 4.3. Web aplikacija

Web aplikacija sastoji se od dvije datoteke. Datoteka *PLCtoMongo* čita podatke iz *PLC*-a i zapisuje ih u *Mongo* bazu podataka, a *WebbApp* uzima podatke iz baze i stavlja ih na raspolaganje web pregledniku. Podatci s *PLC*-a se dohvaćaju u registre koji se dalje koriste za prikaz određenih stanja pokretne trake. Primjerice, aplikacija može prikazati miče li se traka u kojem smjeru, te prikazuje praćenje objekata. Omogućeno je i unošenje sekvence koja će biti poslana i izvršena na *PLC*-u. Pri tome je potrebno unijeti četiri koraka sekvence, ali i četiri *timera* (u sekundama) koji određuju koliko će se objekt zadržavati na pojedinom mjestu zadane sekvence.

Cijela web aplikacija priložena je na CD-u.

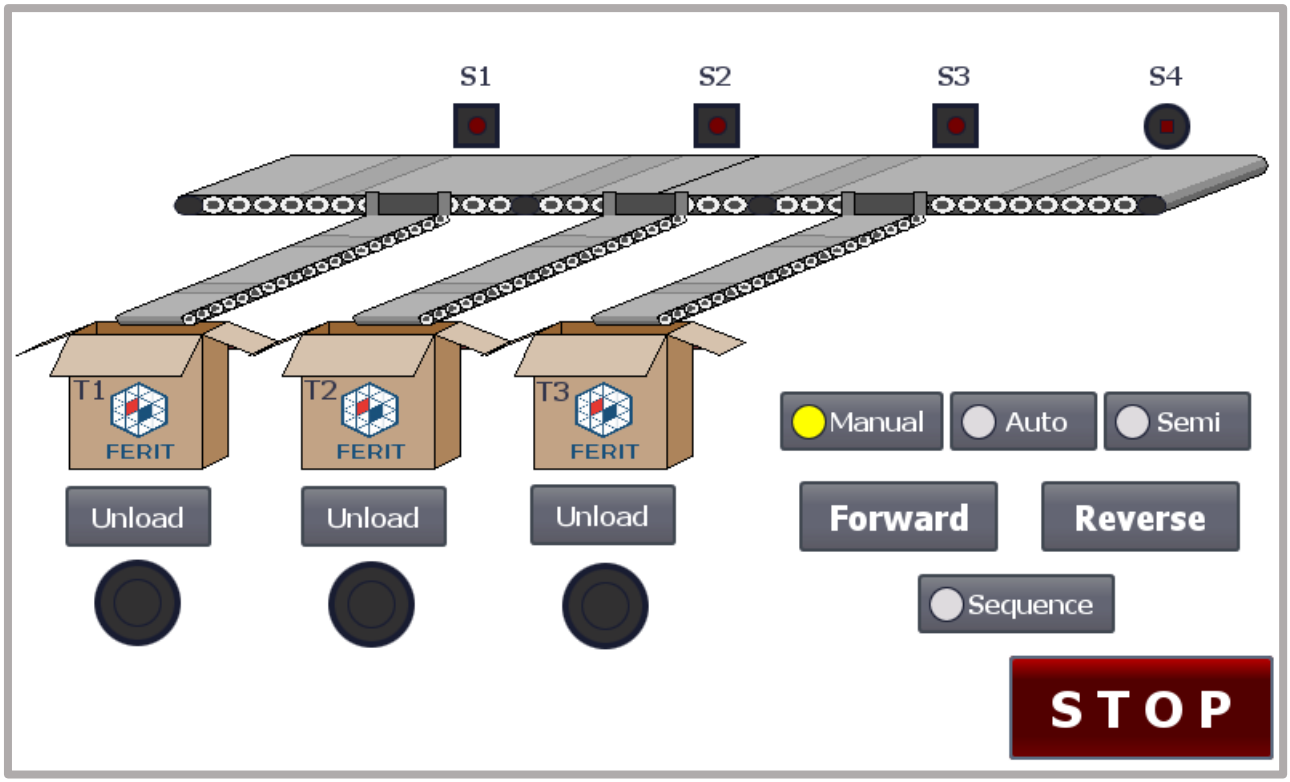
# 5. REZULTATI

U ovom poglavlju prikazani su rezultati programa upravljanja pokretnom trakom. Na sljedećim slikama prikazat će se različiti slučajevi  načina rada pokretne trake, smjera kretanja, količine objekata na pokretnoj traci i njihovih pozicija. Svaki slučaj prikazan je trima odgovarajućim slikama: stanje na fizičkoj traci, vizualizacija na *HMI*-u i stanje web aplikacije.

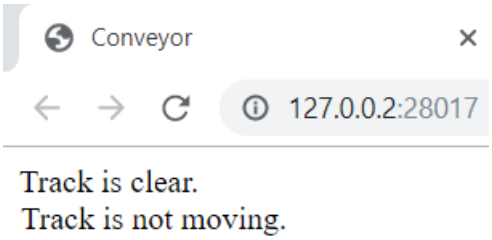
- **Prvi slučaj:**

  na pokretnoj traci nema objekta, traka miruje i upaljen je ručni način rada


*Slika 5.1. Pokretna traka u prvom slučaju*




*Slika5.2. HMI sučelje u prvom slučaju*



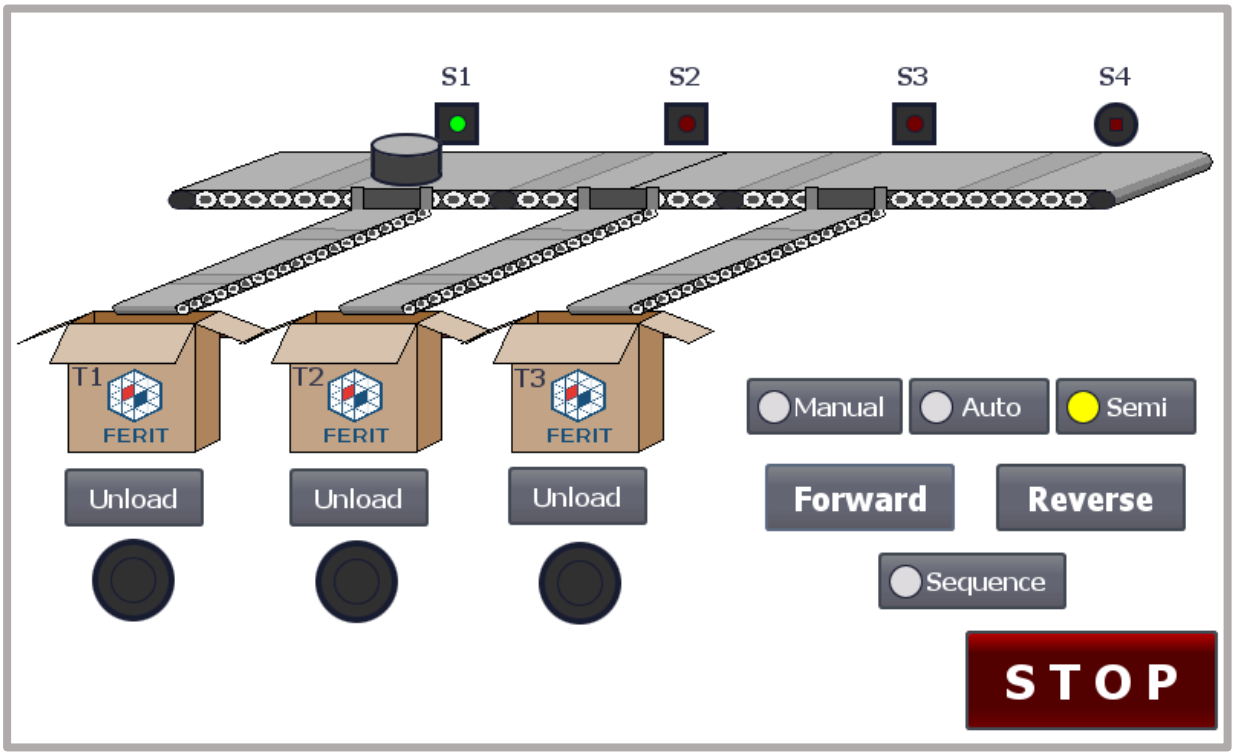

*Slika 5.3. Web aplikacija u prvom slučaju*

Na prethodnim slikama uočava se kako *HMI* i web aplikacija zorno predočuju stanje pokretne trake. Način rada je vidljiv na *HMI*-u i prikazan je žutom lampicom pokraj tipke za ručni način rada. U web aplikaciji moguće je vidjeti da je traka bez tereta i da se ne kreće.

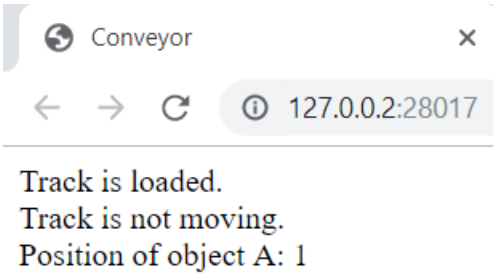
- **Drugi slučaj:**

  objekt se nalazi na poziciji 1, traka miruje, uključen je poluautomatski način rada


*Slika 5.4. Pokretna traka u drugom slučaju*




*Slika 5.5. HMI u drugom slučaju*



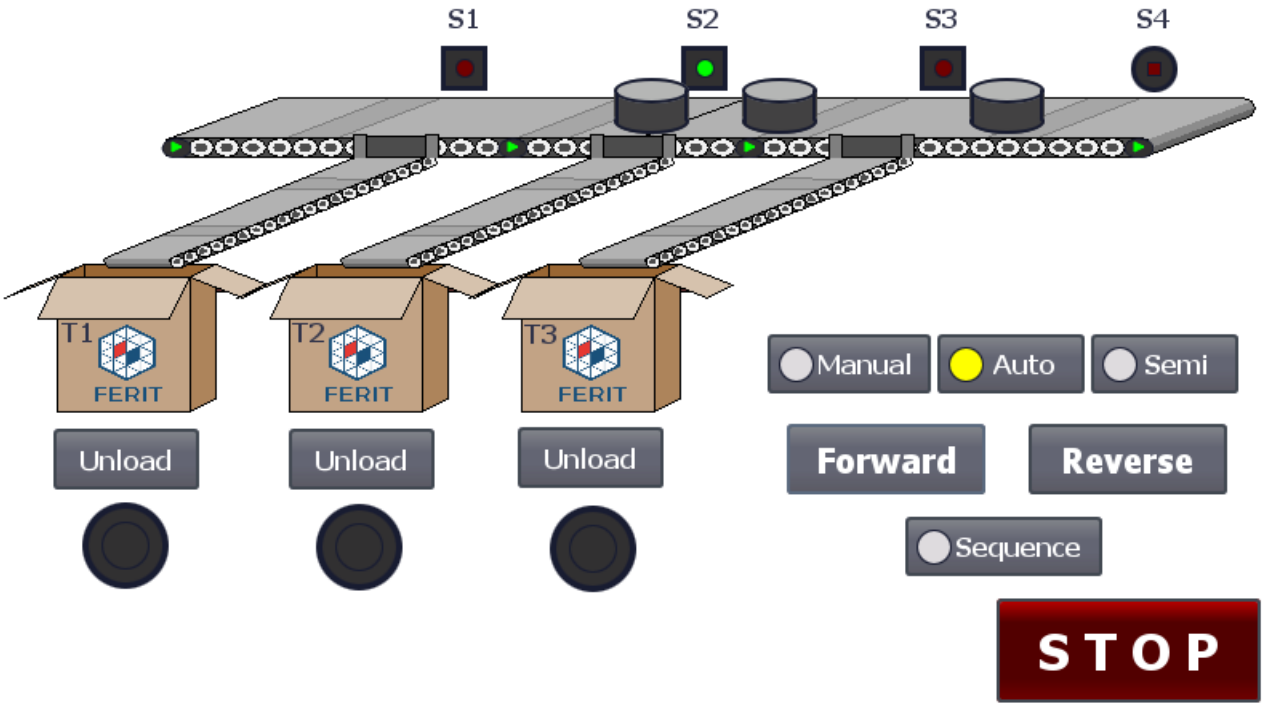

*Slika 5.6. Web aplikacija u drugom slučaju*

Na slikama u drugom slučaju vidimo da se pojavio jedan objekt. Na *HMI*-u može se uočiti da se aktivirao prvi senzor jer se na njemu nalazi objekt, te da je uključen poluautomatski način rada. Web aplikacija će prikazati da pokretna traka ima teret te će označavati objekte redom kako dolaze slovima A – G. U ovom slučaju objekt će biti označen slovom A i bit će prikazana njegova pozicija.

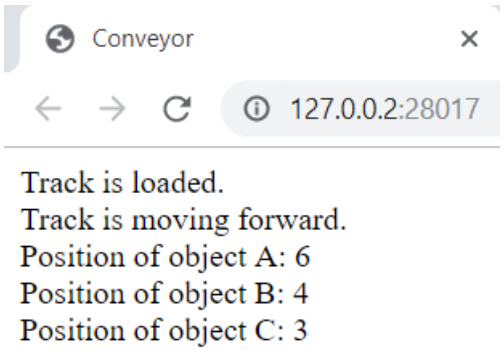
- **Treći slučaj:**

  na pokretnoj traci se nalaze 3 objekta, pokretna traka se kreće prema naprijed, uključen je automatski način rada


*Slika 5.7 Pokretna traka u trećem slučaju*




*Slika 5.8. HMI u trećem slučaju*




*Slika 5.9. Web aplikacija u trećem slučaju*

Radi lakšega snalaženja, na slici 5.7. napisane su pozicije na pokretnoj traci. U *HMI*-u može se vidjeti da se na traci nalaze tri objekta na odgovarajućim pozicijama. Također se vidi da je uključen automatski način rada i da se traka kreće prema naprijed. U web aplikaciji objektima su dodijeljena slova redom kako su dolazili, stoga je prvi objekt A na poziciji 6, drugi objekt B na poziciji 4 itd.

# 6. ZAKLJUČAK

Pokretna traka jedna je od najzastupljenijih automatski upravljanih strojeva u različitim industrijama, od velikih proizvodnih pogona, do malih trgovina. Postoje različite vrste pokretnih traka s različitim elementima, ali njihov zadatak je uvijek isti: prevesti objekt od točke A do točke B.

U ovom završnom radu dana je zaokružena slika industrijskog pogona, te osnovni dijelovi koje on može sadržavati, od električnih shema koje prethode ožičavanju i puštanju u pogon, preko PLC *softwarea* i vizualizacije. Viša razina upravljanja ostvarena je *Modbus* komunikacijom i aplikacijom napisanom u Pythonu. *Software* je napisan s mogućnošću upravljanja pokretnom trakom u ručnom, poluautomatskom i automatskom načinu rada. Aplikacija razvijena u višem programskom jeziku daje mogućnost upravljanja pokretnom trakom s kompleksnijim algoritmima i spremanje podataka u bazu podataka.

Rezultati testiranja programa bili su očekivani. Stvarna pokretna traka uspješno je interpretirala sve što je u programu napisano i davala je točne rezultate na sve testove.

# LITERATURA

[1] H. Jack, Automating Manufacturing Systems with PLCs, Version 5.0, 2007.

[2] D. Maršić, G. Malčić, I.Vlašić, Izrada programskih komponenti u TIA Portal programskom okruženju, Tehničko veleučilište u Zagrebu/Elektrotehnički odjel, Zagreb

[3] H. Berger, SIMATIC automatizacijski sustavi, Graphis d.o.o., Zagreb, 2013.

[4] TIA Portal Manual

[5] https://support.industry.siemens.com/tf/WW/en/posts/difference-between-fb-and-fc-s/18334?page=0&pageSize=10, 28.6.2019.

[6] https://www.tutorialspoint.com/pycharm/pycharm_introduction.htm, 29.6.2019.

[7] Marin Šepac, Programirljivi logički kontroleri (PLC), Završni rad, Sveučilište u Rijeci, Filozofski fakultet u Rijeci, odsjek za politehniku

# SAŽETAK

U ovom radu dane su osnove programiranja *PLC*-a. Navedeni su i objašnjeni alati u kojima se *PLC* programira. Analizirana je pokretna traka s opisima njenih pojedinih dijelova. Izrađen je program za upravljanje pokretnom trakom u alatu *TIA* Portal. Traku je moguće pokretati u ručnom, automatskom i poluautomatskom načinu rada. Omogućeno je praćenje do sedam proizvoda istovremeno na pokretnoj traci. Izrađena je mrežna aplikacija u alatu *PyCharm* koja je povezana s programom i prikazuje određene podatke iz programa na web stranici. Izrađeno je *HMI* sučelje koje omogućuje vizualizaciju rada programa.

Ključne riječi: pokretna traka, *PLC*, *TIA* Portal, *Step* 7

## ABSTRACT

In this paper basics of *PLC* programming are described. The programming tools are listed and explained. A conveyor belt is analyzed with a description of its components. A software for controlling a conveyor belt is made using *TIA* Portal. The conveyer can operate in manual, automatic and semiautomatic mode. It is possible to track up to seven different objects on the conveyor belt at the same time. A web application is created in PyCharm. It is connected with the *PLC* software and it displays certain data on a web page. An *HMI* is created which allows visualization of the program.



Key words: conveyor belt, conveyer, *PLC*, *TIA* Portal, *Step* 7

# ŽIVOTOPIS

Damian Svirac rođen je 1.9.1997. godine u Slavonskom Brodu. Pohađao je osnovnu školu „Đure Pilara“ nakon koje upisuje „Prirodoslovno-matematičku gimnaziju Matija Mesić“. Uz školske obveze aktivno je sudjelovao u Kulturno umjetničkom društvu Luka Lukić Brodski Varoš, te nekoliko godina svirao u Brodskom tamburaškom orkestru. Nakon završetka srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer računarstvo. Tijekom 2019. godine odrađivao je stručnu praksu u Danieli-Systec d.o.o., gdje je i pisao završni rad.

**PRILOZI**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

DANIELI SYSTEC
Member of Danieli Automation

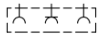
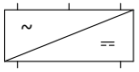
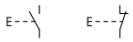
| | |
|---|---|
| *Customer* | FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK |
| *Project description* | SIEMENS POKRETNA TRAKA |
| *Revision* | **FIRST ISSUE [00]** |
| *Revision date* | **10-05-2019** |
| *Revised by* | **DSYS** |

*General notes*

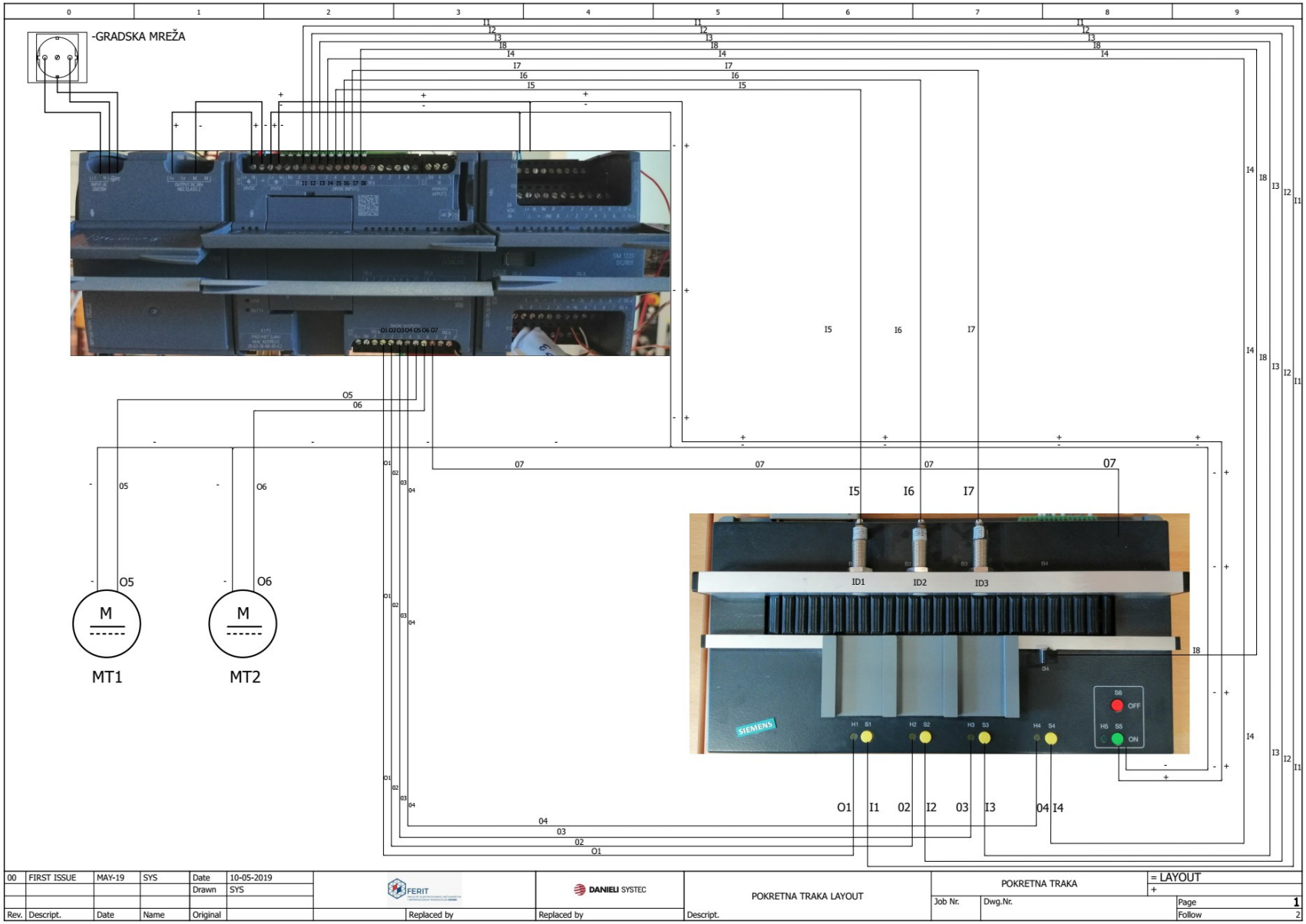
| *Circuit* | *Voltage* | *Common wire color* |
|---|---|---|
| Main circuit | 1~230VAC 50 Hz | Black |
| Auxiliary AC | | Red |
| Auxiliary DC | 24VDC | Blue |
| External circuit | | Orange |
| Neutral | | Light blue |
| Protective earth | TN/TN-C | Green-Yellow |

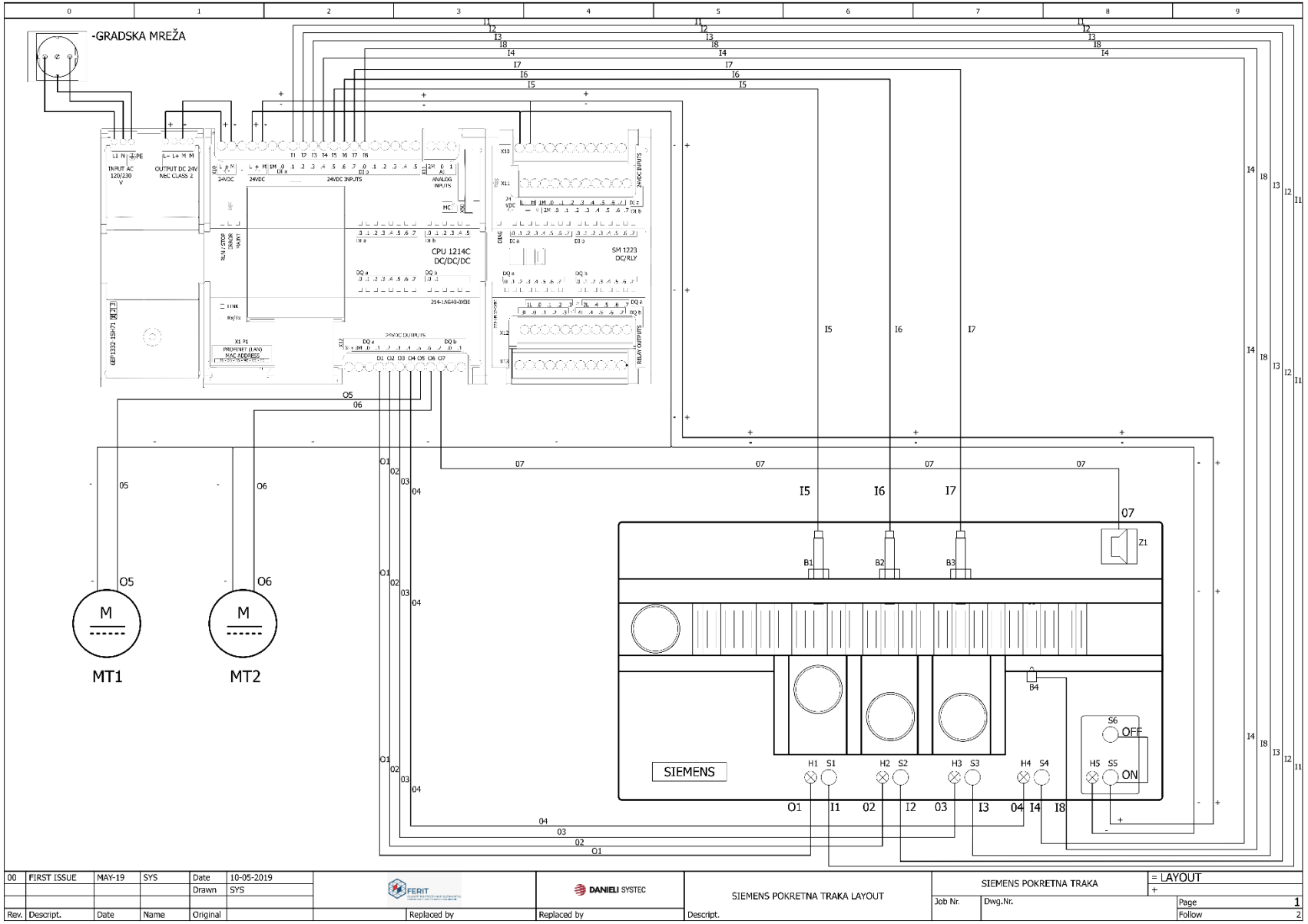
SIEMENS POKRETNA TRAKA

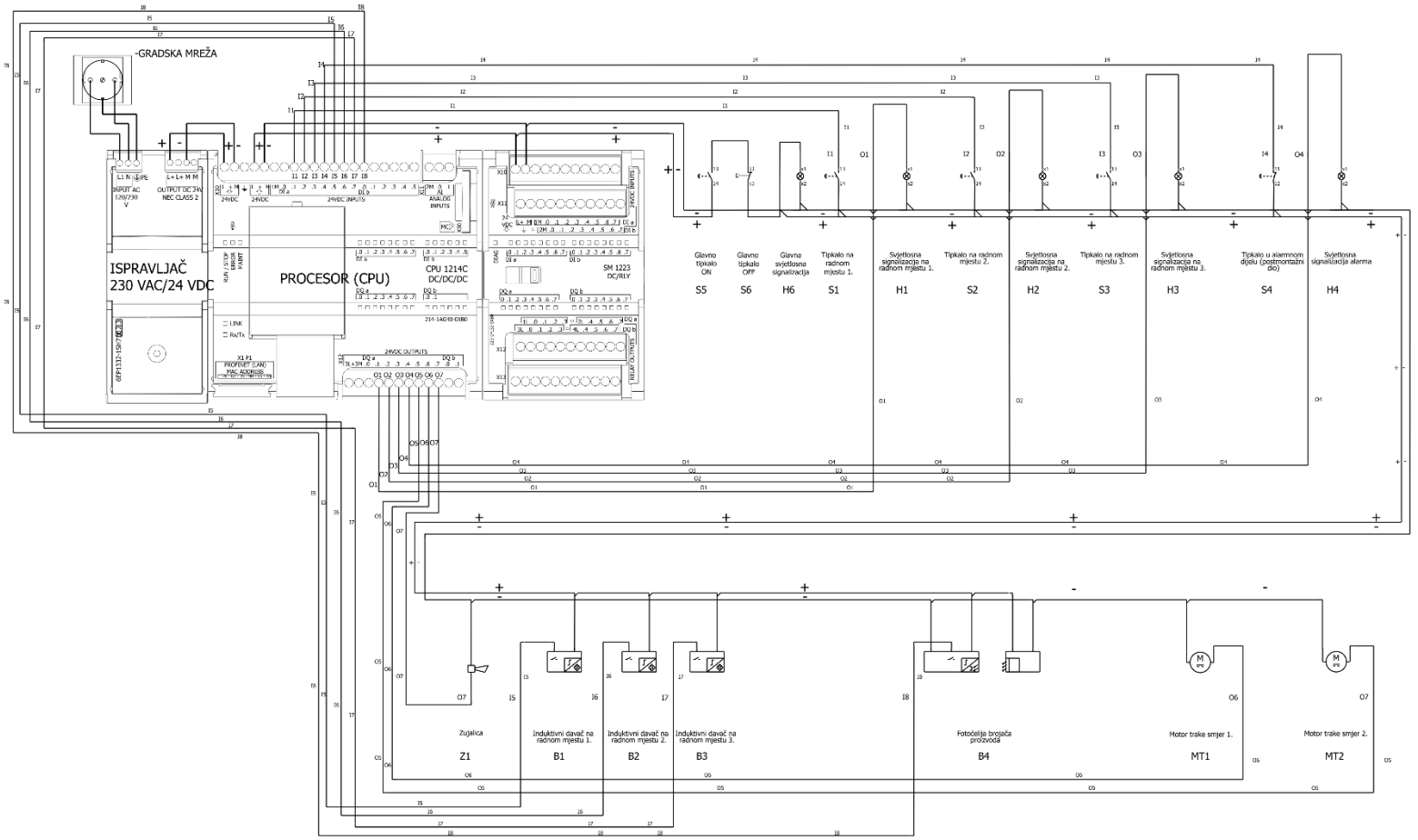
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

| | | |
|---|---|---|
| Socket<br>(Utičnica)  | | |
| Power supply unit<br>(Napajanje)  | | |
| Pushbutton operated by pushing<br>(Tipkalo)  | | |
| Lamp<br>(Lampica)  | | |
| Horn<br>(Zujalica)  | | |
| Proximity switch<br>(Induktivni davač)  | | |
| Photocell, light barrier<br>(Fotoćelija)  | | |
| Motor<br>(Elektromotor)  | | |
| | | |

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


-GRADSKA MREŽA
I1 I2 I3 I8 I4 I7 I6 I5
O5
O6
07
M
MT1
MT2
ID1
ID2
ID3
SIEMENS
S6
OFF
H5 S5
ON
O1 I1 02 I2 03 I3 04 I4
POKRETNA TRAKA LAYOUT
DANIELI SYSTEC
FERIT
00 FIRST ISSUE MAY-19 SYS Date 10-05-2019
Drawn SYS
Rev. Descript. Date Name Original
Replaced by
Replaced by
Descript.
POKRETNA TRAKA
= LAYOUT
Job Nr.
Dwg.Nr.
Page 1
Follow 2


-GRADSKA MREŽA
CPU 1214C DC/DC/DC
SM 1223 DC/RLY
MT1
MT2
SIEMENS
ON
OFF
SIEMENS POKRETNA TRAKA LAYOUT
DANIELI SYSTEC
FERIT
FIRST ISSUE
MAY-19
10-05-2019
SIEMENS POKRETNA TRAKA
LAYOUT




| Rev. | Descript. | Date | Name | Date / Drawn / Original | Replaced by | Replaced by | Descript. |
|---|---|---|---|---|---|---|---|
| | | | | | FERIT | DANIELI SYSTEC | SHEMA |

| SIEMENS POKRETNA TRAKA | = LAYOUT + |
|---|---|
| Job Nr. / Dwg.Nr. | Page 2 / Follow |