Dizajn automatiziranog sustava: razvoj PLC logike i HMI sučelja za autonomno upravljanje električnim vlakom

Peša, Valerija

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

DIZAJN AUTOMATIZIRANOG SUSTAVA: RAZVOJ PLC LOGIKE I HMI SUČELJA ZA AUTONOMNO UPRAVLJANJE ELEKTRIČNIM VLAKOM

Rijeka, studeni 2024.

Valerija Peša
0069083575

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

DIZAJN AUTOMATIZIRANOG SUSTAVA: RAZVOJ PLC LOGIKE I HMI SUČELJA ZA AUTONOMNO UPRAVLJANJE ELEKTRIČNIM VLAKOM

Mentor: prof. dr. sc. Dario Matika

Rijeka, studeni 2024.

Valerija Peša
0069083575

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 20.09.2024.

Zavod: Zavod za automatiku i elektroniku

Predmet: Automatizacija postrojenja i procesa

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Valerija Peša (0069083575)**

Studij: Sveučilišni diplomski studij elektrotehnike (1300)

Modul: Elektroenergetika (1332)

Zadatak: **Dizajn automatiziranog sustava: razvoj PLC logike i HMI sučelja za autonomno upravljanje električnim vlakom / Design of an Automated System: Development of PLC Logic and HMI Panel for Autonomous Operation of an Electric Train**

Opis zadatka:

U radu je potrebno razviti konceptualno rješenje sustava automatizacije za autonomno upravljanje električnim vlakom koristeći softverski paket Siemens TIA Portal. Rješenje treba obuhvatiti dizajn PLC logike, razvoj HMI sučelja za korisničku interakciju, te implementaciju upravljanja kretanjem vlaka s integriranim sigurnosnim značajkama. Poseban naglasak potrebno je staviti na sekvencijalno upravljanje kretanjem, integraciju sigurnosnih mehanizama, te simulaciju i validaciju funkcionalnosti sustava.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Predsjednik povjerenstva za diplomski ispit:
prof. dr. sc. Dubravko Franković

Mentor:
prof. dr. sc. Dario Matika

Komentor:
dr. sc. Dominik Cikač

IZJAVA

Sukladno članku 8. Pravilnika o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija Tehničkog fakulteta Sveučilišta u Rijeci od 1. veljače 2020., izjavljujem da sam samostalno izradila diplomski rad prema zadatku preuzetom dana 20. ožujka 2024.

Rijeka, studeni 2024.

Valerija Peša

0069083575

Sadržaj

1. UVOD

Automatizacija sustava omogućuje preciznije, sigurnije i učinkovitije upravljanje složenim procesima, čime se smanjuje potreba za ljudskom intervencijom, a povećava pouzdanost i sigurnost. Jedan od naprednih primjera primjene automatizacije u transportu jest razvoj autonomnih sustava za upravljanje vozilima. Upravljanje takvim sustavima zahtijeva integraciju hardverskih i softverskih komponenti te implementaciju složenih algoritama koji omogućuju sigurno kretanje vozila, nadzor nad njegovim stanjem i prilagodbu različitim vanjskim uvjetima. Cilj ovog diplomskog rada je prikazati cjelokupan proces razvoja, implementacije i testiranja automatiziranog upravljačkog sustava za autonomni električni vlak u vidu kompleksne simulacije. Razvoj ovakvog sustava predstavlja ne samo tehnički izazov nego i priliku za primjenom teorijskih znanja u dizajniranju rješenja.

Strukturu rada čini devet poglavlja, koja detaljno obrađuju ključne aspekte sustava i faze njegove izrade. Prvi dio rada posvećen je funkcionalnim specifikacijama i dizajnu sustava, gdje se definiraju operativni uvjeti, glavne komponente sustava s posebnim naglaskom na korisničke unose, sigurnosne značajke i moguća stanja kvara. Ovaj dio postavlja temelje za daljnji razvoj sustava. U nastavku se opisuje proces odabira konfiguracije hardvera. Prikazani su osnovni koraci za uspostavljanje komunikacije između različitih dijelova sustava, čime se omogućuje realizacija zadane logike upravljanja. Četvrto poglavlje o programiranju funkcijskih blokova, pruža detaljan prikaz stvaranja logike za upravljanje motorom, senzorima i kočnicama, koji čine osnovu za složenije sekvence upravljanja. U petom poglavlju slijedi opis glavne programske sekvence, koja razrađuje prelazak sustava iz jednog operativnog stanja u drugo na temelju korisničkih naredbi. Ova sekvenca osigurava da sustav autonomno izvršava potrebne radnje u skladu s odabranim parametrima. Sigurnosni aspekti sustava prikazani su kroz implementaciju detekcije grešaka i definiranje kodova grešaka koji pokreću zaštitne mehanizme u šestom poglavlju. Sedmo poglavlje objašnjava implementaciju dodatnog koda za simulaciju stvarnih uvjeta u programiranju kretanja vlaka. Osmo poglavlje je usredotočeno na inicijalizaciju osnovnih parametara u PLC programu, kako bi bilo omogućeno pravilno pokretanje i stabilnost sustava. Posljednje, deveto poglavlje obrađuje razvoj HMI sučelja i njegovu prilagodbu, čime se ostvaruje interakcija korisnika s autonomnim sustavom i nadzor nad njegovim radom. U tom poglavlju detaljno su opisani elementi sučelja, uključujući početni zaslon, kontrolne panele i konfiguraciju dodatnih funkcionalnosti. Na kraju, kroz završno funkcionalno testiranje, provjerava se rad cijelog sustava i potvrđuje njegova usklađenost s postavljenim specifikacijama, čime se dokazuje njegova pouzdanost, sigurnost i ispravnost u radu.

2. FUNKCIONALNE SPECIFIKACIJE SUSTAVA

U ovome poglavlju specificiraju se operativni uvjeti i komponente sustava, te se opisuje način rada sustava kojeg razvijamo s naglaskom na korisničke unose, sigurnosne značajke i potencijalna stanja kvara. Detaljno ćemo objasniti kako korisnici mogu upravljati vlakom putem HMI sučelja i koje sigurnosne mjere će sustav sadržavati. Uz to, prikazat će se glavne sekvence koje stroj izvršava kako bi se postigla željena operativna stanja.

Sustav je zamišljen tako da ostvaruje sigurno i učinkovito kretanje vlaka, uz minimalnu ljudsku intervenciju. Ideja je omogućiti operateru da odabere stanicu na koju vlak treba ići, te da pritom postavi željenu brzinu vožnje. Sigurnosne mjere koje su primijenjene u sustavu, osmišljene su kako bi se spriječile opasne situacije. Na primjer, u slučaju potrebe za hitnim zaustavljanjem, korisnik može pritisnuti tipku koja odmah zaustavlja vlak, dok je sigurnosni ključ osmišljen tako da spriječi pokretanje vlaka dok je ključ uklonjen, što je korisno pri održavanju sustava. Senzori postavljeni uzduž rute detektiraju poziciju vlaka, što omogućava sustavu da prati kada se vlak približava stanici, kako bi započeo usporavanje te omogućio zaustavljanje točno na željenoj poziciji. Osim toga, sustav uključuje HMI sučelje putem kojeg korisnik može pratiti rad vlaka i status sustava, kao i resetirati bilo kakve greške kad su ispunjeni potrebni uvjeti. Sustav automatski detektira nepravilnosti, poput nemogućnosti pokretanja motora ili neispravnosti kočnica, te generira kod greške koji korisniku pruža informacije o problemu i omogućuje poduzimanje potrebnih koraka za vraćanje sustava u sigurno stanje. Sve ove funkcionalnosti zajedno omogućuju stabilan i kontroliran rad vlaka, uz sigurnosne mehanizme koji osiguravaju pravovremeno otkrivanje problema.

Korisnički unos se u smislu ovoga rada odnosi na interakciju korisnika s autonomnim vlakom putem HMI sučelja. Operater može odabrati ciljno odredište vlaka (stanicu 1, 2, 3 ili 4), postaviti maksimalnu brzinu vlaka te odabrati stupanj usporavanja i ubrzavanja. Dodatno, operater može resetirati sve eventualne greške ili upozorenja, aktivirati hitno zaustavljanje vlaka („*Emergency-Stop*“) te koristiti sigurnosni ključ u slučaju radova na željezničkoj pruzi. Sve ovo spada u korisničke unose našeg sustava.

2.1 Opis sigurnosnih značajki i glavnih sekvenci sustava

Jasno je da u razvoju ovoga rješenja želimo spriječiti opasne situacije koje mogu dovesti do ozljeda, nesreća ili oštećenja opreme. Stoga, posebnu pažnju moramo posvetiti sigurnosnim

značajkama i kodovima grešaka. Sigurnosne značajke se odnose na funkcije unutar sustava dizajnirane da se spriječe nesreće, zaštite korisnici i oprema te osigura sigurno djelovanje sustava. S druge strane kodovi grešaka služe za prepoznavanje i identificiranje problema u radu sustava. Funkcioniraju tako da aktiviraju upozorenja ili alarme u specifičnim uvjetima kako bi se spriječili kvarovi ili oštećenja te kako bi sustav mogao ispravno funkcionirati. Pregled svih kodova grešaka i sigurnosnih značajki dan je u *tablici 1.*

Kodovi grešaka i sigurnosne značajke	**Objašnjenje**
Greška u pokretanju motora	Ako nakon određenog vremenskog intervala i dalje nema potvrde da je motor pokrenut, nakon što je naredba za pokretanjem već zadana, aktivira se ovaj kod greške
Greška u aktivaciji/otpuštanju kočnice	Ako senzor ne potvrdi aktivaciju ili otpuštanje kočnice, aktivira se kod greške
Greška izrazito visoke vibracije (HH)	Kod greške aktiviran je od strane sustava za mjerenje vibracija vlaka koji prenosi podatak o prekoračenju dopuštene razine vibracije (u oznaci HH – engl. *high-high*)
Upozorenje o visokoj vibraciji (H)	Kada izmjerena vrijednost vibracije prijeđe dozvoljenu granicu, ali nije dovoljno visoka da bi rad sustava predstavljao opasnost koja zahtijeva zaustavljanje - aktivira se upozorenje o povećanju vibracije i mogućem kritičnom stanju (oznaka H – engl. *„high*“)
Greška izrazito niskih vibracija (LL)	Kod greške je aktiviran pri detekciji vibracija koje su ispod dozvoljene donje granice, a alarm se pokreće od strane koda isključivo uz dodatnu potvrdu o radu motora i otpuštenosti kočnica kako bi se utvrdilo da uzrok niskim vibracijama nije mirovanje vlaka (u oznaci LL – engl. „*low-low“*)
Sigurnosna svjetlosna zavjesa	Ako laserski senzor detektira prepreku na tračnicama, vlak se automatski zaustavlja

Hitno zaustavljanje (engl. Emergency-stop)	Zaustavljanje vlaka u hitnim situacijama
Sigurnosni ključ (engl. Safety key)	Kada je sigurnosni ključ uklonjen iz svojeg kućišta, vlak se ne može pokrenuti – sigurnosna mjera u slučaju održavanja ili radova na željeznici
Greška prelaska preko krajnje pozicije (engl. Overrun)	Kod greške označava da vlak nije stao na predviđenom mjestu, što može ukazivati na kvar zaustavnog senzora ili kočnice
Greška u redoslijedu aktivacije senzora	Kod greške povezan s nepravilnim redoslijedom aktivacije senzora blizine

Tablica 1. Sigurnosne značajke i kodovi grešaka

Putem informacija na zaslonu, operater može pratiti rad vlaka u realnom vremenu, upravljati funkcijama kao što su kočenje i pokretanje motora, te imati uvid u status stroja kako bi osigurao siguran i učinkovit rad sustava. Informacije prikazane na zaslonu, koje definiramo kao *izlaz* sustava su:

a) Prikaz upozorenja i grešaka,
b) Prikaz brzine vlaka i ubrzanja,
c) Prikaz razine vibracija,
d) Aktivacija izlaza (kočnica, motor),
e) Status stroja.

Glavne sekvence sustava definiraju korake koje vlak mora izvršiti na temelju određenih naredbi od strane korisnika (*tablica 2.*).

Glavne sekvence sustava	**Objašnjenje**
Inicijalizacija	Kada se sustav pokrene, korisnik treba pritisnuti tipku za inicijalizaciju, čime vlak automatski dolazi na stanicu 1
Odabir stanice	Korisnik odabire stanicu na koju vlak treba ići, nakon čega vlak ubrzava do maksimalne brzine, usporava kada detektira relevantni senzor za usporavanje i zaustavlja se na odabranoj stanici

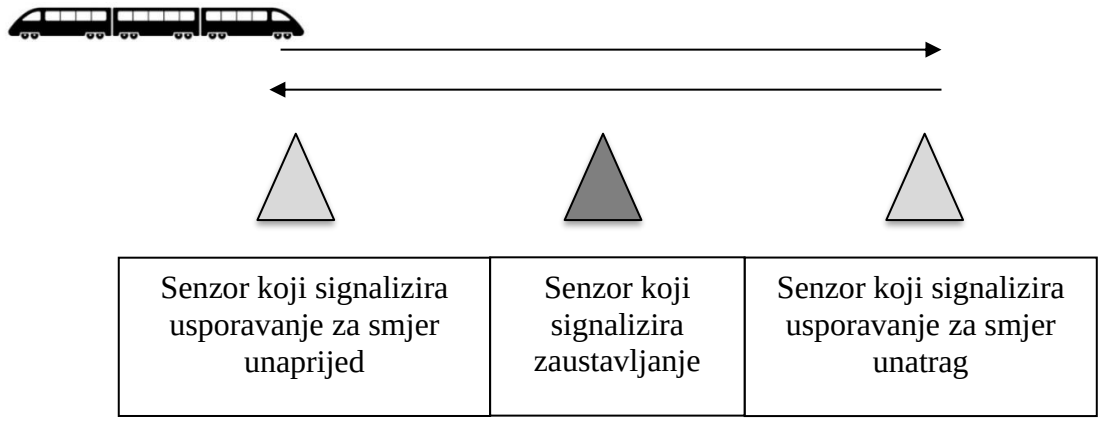
Ubrzanje	Ova sekvenca uključuje otpuštanje kočnice, potvrdu otpuštanja i pokretanje motora.
Usporavanje	Ova sekvenca uključuje zaustavljanje motora i primjenu kočnica

Tablica 2. Glavne sekvence sustava

2.2 Napomene

U sustavu upravljanja kretanjem autonomnog vlaka primjenjuju se senzori blizine (engl. proximity switch) postavljeni duž pruge za signalizaciju usporavanja i zaustavljanja vlaka pri dolasku na stanicu. Senzori su strateški raspoređeni kako bi omogućili zaustavljanje vlaka na predviđenim pozicijama. Oni emitiraju signal kada vlak dođe u njihovu blizinu i taj signal obično služi kao informacija za kontrolni sustav vlaka, koji reagira tako da pokreće naredbu za usporavanje ili zaustavljanje.

Senzori koji signaliziraju usporavanje postavljeni su neposredno ispred svake stanice s obje strane, kako bi oba smjera kretanja vlaka bila uzeta u obzir. Kada se vlak približava stanici iz smjera unaprijed njegovo kretanje opazit će senzor koji signalizira usporavanje u smjeru unaprijed, a kada se vlak istoj stanici približava iz suprotnog smjera signal će mu odaslati senzor za usporavanje za smjer unatrag, smješten s druge strane stanice. Ovo je prikazano na *slici 1.* i bit će važno za postavljanje uvjeta za kretanje vlaka u kasnijim poglavljima.



Slika 1. Raspored senzora blizine za signalizaciju usporavanja ovisno o smjeru kretanja vlaka

Sustav pravi razliku između situacija koje uzrokuju upozorenje i onih koje uzrokuju stanje greške (engl. fault condition). Kada uslijedi događaj na razini upozorenja, poput visoke vibracije, sustav

će obavijestiti korisnika o mogućoj opasnoj situaciji, ali će nastaviti raditi normalno. Ova upozorenja služe kao preventivne mjere i ne zahtijevaju trenutačno djelovanje korisnika.

S druge strane, kada uslijedi događaj na razini greške, sustav automatski pokreće sekvencu zaustavljanja i prelazi u stanje greške. U tom stanju, vlak se ne može pokrenuti dok korisnik ne poništi grešku pritiskom na tipku za resetiranje i zatim pritiskom na tipku za inicijalizaciju. Na primjer, ako sustav uđe u stanje greške zbog izrazito visoke vibracije ili zbog pritiska na tipku za hitno zaustavljanje, vlak će se odmah zaustaviti na mjestu. Korisnik tada mora resetirati grešku, što je moguće samo kada je uzrok greške uklonjen. Primjerice, ako je radnik uklonio sigurnosni ključ iz njegovog kućišta, to će generirati stanje greške, a ključ mora biti vraćen u kućište prije nego što se greška može resetirati i vlak ponovno pokrenuti.

2.3 IO lista

IO lista bi uobičajeno opisivala prave električne *inpute* i *outpute* PLC-a. Ideja ovoga rada je kreirati simulirani sustav koji ne zahtijeva fizičke komponente. Iz ovog razloga, IO lista bit će simulirana na HMI-ju, što znači da se pritiskom tipki na HMI simuliraju električni inputi koje bi inače PLC dobivao s hardvera. Dakle svi senzori će biti simulirani programom, prema *tablici 3.*

Oznaka („TAG“)	**Tip signala**
VT-01_train_vibration	AI
M-01-spd_motor_speed	AO
M-01-accl_motor_acceleration	AO
ZS-01_fault_rev	DI
ZS-02_stop1	DI
ZS-03_slow1_rev	DI
ZS-04_slow2_fwd	DI
ZS-05_stop2	DI
ZS-06_slow2_rev	DI
ZS-07_slow3_fwd	DI
ZS-08_stop3	DI
ZS-09_slow3_rev	DI
ZS-10_slow4_fwd	DI
ZS-11_stop4	DI
ZS-12_fault_fwd	DI

XS-01_e_stop	DI
XS-02_safety_key	DI
LS-01_light_curtain	DI
HS-01_station1	DI
HS-02_station2	DI
HS-03_station3	DI
HS-04_station4	DI
M-01-VSD_flt	DI
M-01-ctrl-flt_controol_loop_fault_sig	DI
M-01-run-cnf_motor_running_confirmation	DI
ZS-13_brake_engaged	DI
XZ-01_brake	DO
M-01-run-cmd_motor_run	DO
M-01-dir_motor_direction	DO

Tablica 3. IO lista

3. KONFIGURACIJA SUSTAVA

U ovome radu primijenjen je *Siemens S7-1500* PLC (CPU 1511-1-PN; 6ES7-511-TAK01-0AB0). IP adresa je po zadanim postavkama postavljena na 192.168.0.1 s maskom podmreže 255.255.255.0. Odabrani HMI je 15 inčni, *Siemens Comfort Panel, TP 1500*. Nakon odabira, u postavkama HMI-ja provjeravamo profinet sučelje, zbog važnosti da adrese uređaja budu različite jer inače neće moći međusobno komunicirati. TIA Portalom je automatski dodijeljena IP adresa 192.168.0.2. Za povezivanje hardvera koristimo *Ethernet* vezu, koja je ovom slučaju „PN/IE_1“. Ovime se omogućava veza PLC-a s HMI-jem, prikazana na *slici 2.*

Slika 2. Odabrani PLC i HMI

Kod ovog specifičnog PLC CPU nema integriranih IO priključaka, pa moramo dodati proširive kartice koje će omogućiti ovom PLC-u da očitava fizičke ulazne i izlazne signale s različitih senzora. Za digitalni ulaz odabiremo karticu od 24 volta DC s 32 digitalna ulaza (DI 32X24VDC BA). Za digitalni izlaz biramo karticu od 32 izlaza, 24 volta DC s 0,5 A (DQ 32X24VDC/0.5ABA). Što se tiče analognih kartica, koristimo onu sa 16 analognih ulaza, te jednu s osam analognih izlaza, odabrane prema zahtjevima projekta (AI 16xIBA, AQ 8XU/I HS). Kada bi postojala fizička implementacija sustava koji razvijamo, morali bismo voditi računa o tome da broj mjesta na kojem se u TIA Portalu nalazi određena kartica, mora odgovarati fizičkom rasporedu proširivih portova na stvarnom PLC-u. Dodavanjem PLC-a i HMI-ja, te proširivih kartica za ulaze i izlaze, provjerom njihovih IP adresa i njihovim umrežavanjem – postavljen je potreban hardver za potrebe sustava, prikazan na *slici 3.*

Slika 3. Hardverska konfiguracija

4. PROGRAMIRANJE FUNKCIJSKIH BLOKOVA

4.1 Upravljanje motorom

Za programiranje funkcijskog bloka upravljanja motorom, ulazi su definirani kao što je prikazano na *slici 4.* Prefiks BI/BO u nazivima, odnosi se na binarne ulaze i binarne izlaze. Ovom funkcijskom bloku potrebni ulazi su:

- zahtjev za pokretanje i zaustavljanje motora,
- VSD signal greške,
- signal greške iz kontrolnog kruga,
- informacija o tome treba li vlak krenuti naprijed ili u suprotnom smjeru,
- signal potvrde, odnosno povratna informacija o radu motora,
- signal resetiranja neispravnog stanja.

FB_Motor_VSD

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
1	Input									
2	BI_Start_Req	Bool	false	Non-retain	☑	☑	☑	☐		
3	BI_Stop_Req	Bool	false	Non-retain	☑	☑	☑	☐		
4	BI_VSD_Fault	Bool	false	Non-retain	☑	☑	☑	☐		
5	BI_Control_Fault	Bool	false	Non-retain	☑	☑	☑	☐		
6	BI_Dirction_Fwd	Bool	false	Non-retain	☑	☑	☑	☐		
7	BI_Running_FB	Bool	false	Non-retain	☑	☑	☑	☐		
8	BI_Reset_Faults	Bool	false	Non-ret...	☑	☑	☑	☐		
9	<Add new>				☐	☐	☐	☐		

Slika 4. Ulazi funkcijskog bloka upravljanja motorom

Potrebni izlazi su:

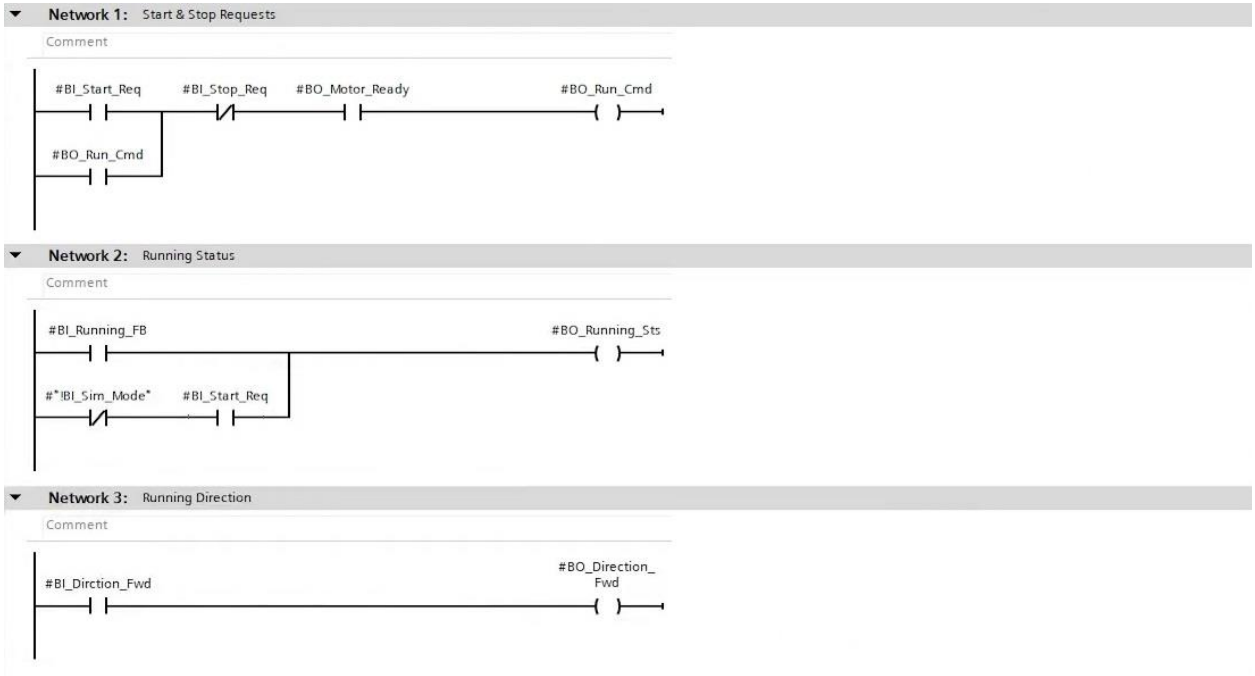
- naredba za smjer motora,
- naredba za pokretanje motora,
- informacija o spremnosti motora za pokretanje,
- status motora.

FB_Motor_VSD

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
10	Output									
11	BO_Direction_Fwd	Bool	false	Non-retain	☑	☑	☑	☐		
12	BO_Run_Cmd	Bool	false	Non-retain	☑	☑	☑	☐		
13	BO_Motor_Ready	Bool	false	Non-retain	☑	☑	☑	☐		
14	BO_Running_Sts	Bool	false	Non-ret...	☑	☑	☑	☐		
15	<Add new>				☐	☐	☐	☐		

Slika 5. Izlazi funkcijskog bloka upravljanja motorom

Programiranje funkcijskog bloka upravljanja motora s VSD-om (engl. variable speed drive), ladder logikom, prikazano je na sljedećim slikama.



Slika 6. Upravljanje motorom; prvi dio koda

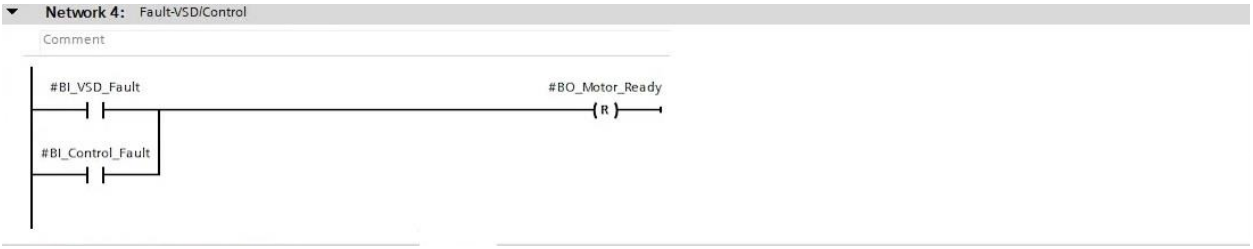
Program za simulaciju rada motora osmišljen je tako da podržava sve ključne funkcije upravljanja motorom: uključivanje, isključivanje, detekciju smjera rada, praćenje statusa, otkrivanje grešaka pri pokretanju i zaustavljanju te funkciju za resetiranje grešaka.

Logika upravljanja pokretanjem i zaustavljanjem motora omogućava da se motor pokrene samo u slučaju kada postoji zahtjev za pokretanjem, kada je motor spreman, i kada nema istovremenog zahtjeva za zaustavljanjem. Jednom kada je motor pokrenut, njegova naredba za rad ostaje aktivna sve dok se ne dobije jasan zahtjev za zaustavljanjem. Time se osigurava stabilan rad bez nepotrebnih zaustavljanja. U slučaju da istovremeno postoje zahtjevi za pokretanjem i zaustavljanjem, motor se neće pokrenuti, čime se dodatno osigurava točnost i kontrola, kao što je prikazano na *slici 6.*

Status rada motora prati se na temelju povratnog signala, koji označava je li motor zaista u pogonu. Ovo je korisno u testiranju sustava, jer simulacija daje priliku za testiranje reakcija sustava bez fizičkog motora. Smjer kretanja vlaka definiran je prema odgovarajućem ulaznom signalu koji određuje da se stroj kreće u pravom smjeru. Tako se osigurava da motor uvijek slijedi točno zadani smjer rada, što je važno za pravilno izvršavanje operacija u stvarnim uvjetima. Sve ovo je prikazano na *slici 6.*

U dijelu koda namijenjenom za praćenje grešaka koje mogu utjecati na rad motora, sustav provjerava moguće greške povezane s upravljanjem (ili sustavom VSD). Ako dođe do neke od tih grešaka, rad motora automatski prestaje, što je sigurnosna mjera koja sprječava daljnje

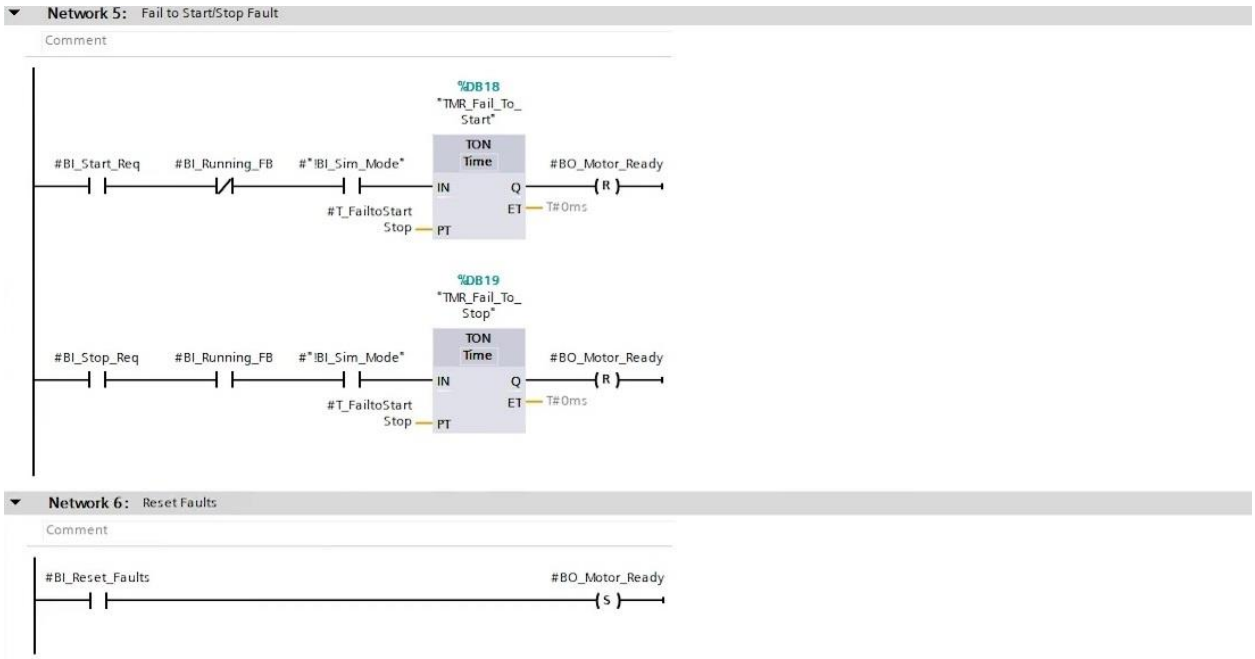
potencijalno oštećenje. Na taj način sustav detektira i reagira na sve kritične pogreške, čime se povećava sigurnost i pouzdanost, kao što je prikazano na *slici 7.*



Slika 7. Kontrola grešaka

Kod za detekciju grešaka pri pokretanju i zaustavljanju dopušta da sustav prati vremensko kašnjenje između zahtjeva za pokretanje ili zaustavljanje i stvarnog odgovora motora. Ako motor ne biva pokrenut u očekivanom vremenskom intervalu nakon zahtjeva za pokretanje ili se ne zaustavi u predviđenom vremenskom intervalu nakon zahtjeva za zaustavljanje, to se bilježi kao greška. Ova funkcija osigurava da motor uvijek reagira u realnom vremenu na naredbe, a u suprotnom slučaju automatski ulazi u sigurnosni režim rada. Opcija simulacije omogućava ignoriranje ovih grešaka kada se sustav testira, što je korisno za razvoj i analizu bez hardverskih ograničenja.

Program također sadrži mogućnost resetiranja grešaka (*slika 8.* - mreža 6.), koja omogućava poništavanje svih detektiranih grešaka i ponovno postavljanje motora u stanje spremno za pokretanje. Ovo omogućava korisniku da brzo i jednostavno vrati sustav u radno stanje nakon otklanjanja uzroka greške, što je važno za kontinuitet rada i brzinu intervencije.



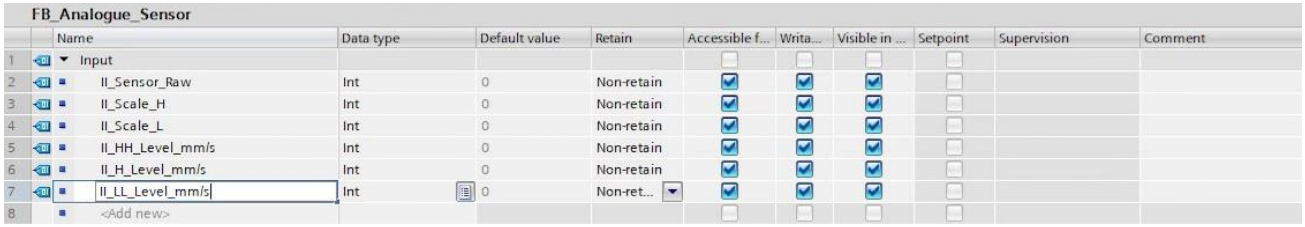
Slika 8. Greška pokretanja/zaustavljanja te reset grešaka

4.2 Upravljanje analognim senzorima

Na slikama 9. i 10. su prikazane dvije tablice koje definiraju ulaze i izlaze funkcijskog bloka za upravljanje analognim signalom, za sustav koji mjeri vibracije vlaka. Ovaj funkcijski blok koristi se za skaliranje i obradu analognog signala te generiranje alarma na temelju izmjerene vrijednosti vibracija.

Ulazne vrijednosti funkcijskog bloka su:

- očitanja senzora, koja dolaze s analognog ulaza PLC-a,
- ulazne vrijednosti za skaliranje, koje se koriste za pretvaranje očitane vrijednosti u potrebne jedinice,
- razina visokog alarma za vibracije u milimetrima po sekundi,
- razina upozorenja za visoke vibracije,
- razina niskog alarma za izmjerene vibracije.



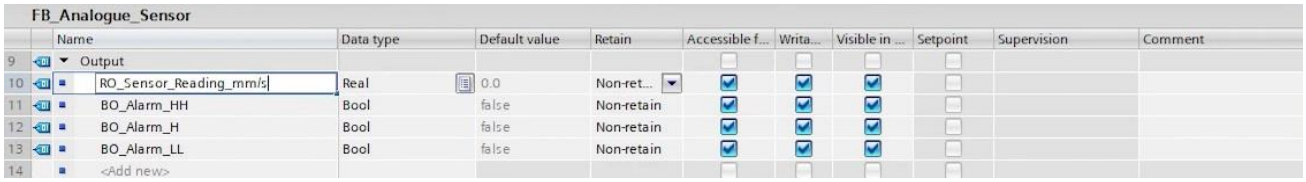
FB_Analogue_Sensor

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
1	Input				☐	☐	☐	☐		
2	II_Sensor_Raw	Int	0	Non-retain	☑	☑	☑	☐		
3	II_Scale_H	Int	0	Non-retain	☑	☑	☑	☐		
4	II_Scale_L	Int	0	Non-retain	☑	☑	☑	☐		
5	II_HH_Level_mm/s	Int	0	Non-retain	☑	☑	☑	☐		
6	II_H_Level_mm/s	Int	0	Non-retain	☑	☑	☑	☐		
7	II_LL_Level_mm/s	Int	0	Non-ret...	☑	☑	☑	☐		
8	<Add new>				☐	☐	☐	☐		

Slika 9. Ulazne vrijednosti funkcijskog bloka upravljanja analognim senzorom

Izlazne vrijednosti funkcijskog bloka su:

- RO_Sensor_Reading_mm/s: izlaz koji prikazuje očitanu vrijednost senzora u milimetrima po sekundi (tip podataka je *Real*),
- BO_Alarm_HH: binarni izlaz (tip podataka *Bool*) koji označava da je dostignuta razina visokog alarma,
- BO_Alarm_H: binarni izlaz (tip podataka *Bool*) koji označava da je dostignuta razina upozorenja za visoke vibracije,
- BO_Alarm_LL: binarni izlaz (tip podataka *Bool*) koji označava da je dostignuta razina niskog alarma.

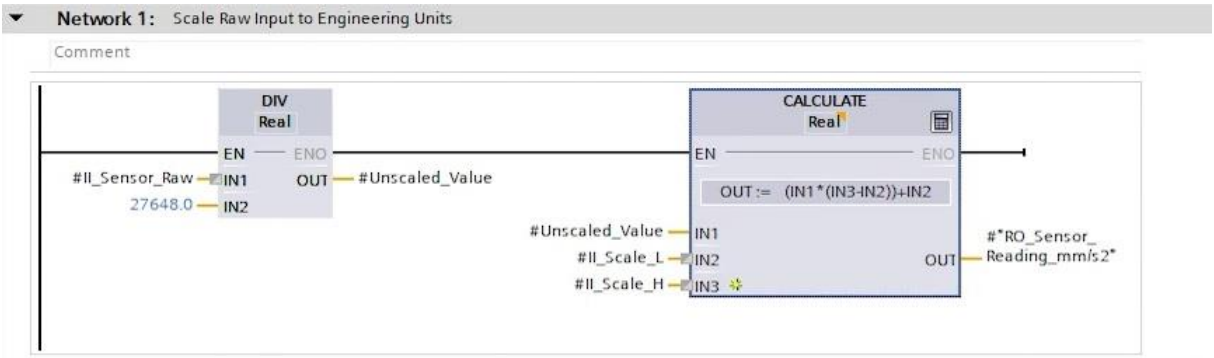


FB_Analogue_Sensor

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
9	Output				☐	☐	☐	☐		
10	RO_Sensor_Reading_mm/s	Real	0.0	Non-ret...	☑	☑	☑	☐		
11	BO_Alarm_HH	Bool	false	Non-retain	☑	☑	☑	☐		
12	BO_Alarm_H	Bool	false	Non-retain	☑	☑	☑	☐		
13	BO_Alarm_LL	Bool	false	Non-retain	☑	☑	☑	☐		
14	<Add new>				☐	☐	☐	☐		

Slika 10. Izlazne vrijednosti funkcijskog bloka upravljanja analognim senzorom

Ovaj funkcijski blok uzima vrijednost očitanja senzora, skalira je u potrebne jedinice (mm/s), te generira signale upozorenja i alarma na temelju definiranih graničnih vrijednosti. Na kraju, funkcijski blok daje izlaznu vrijednost koja se može koristiti za prikaz vibracija, zajedno s binarnim signalima koji označavaju stanje alarma.

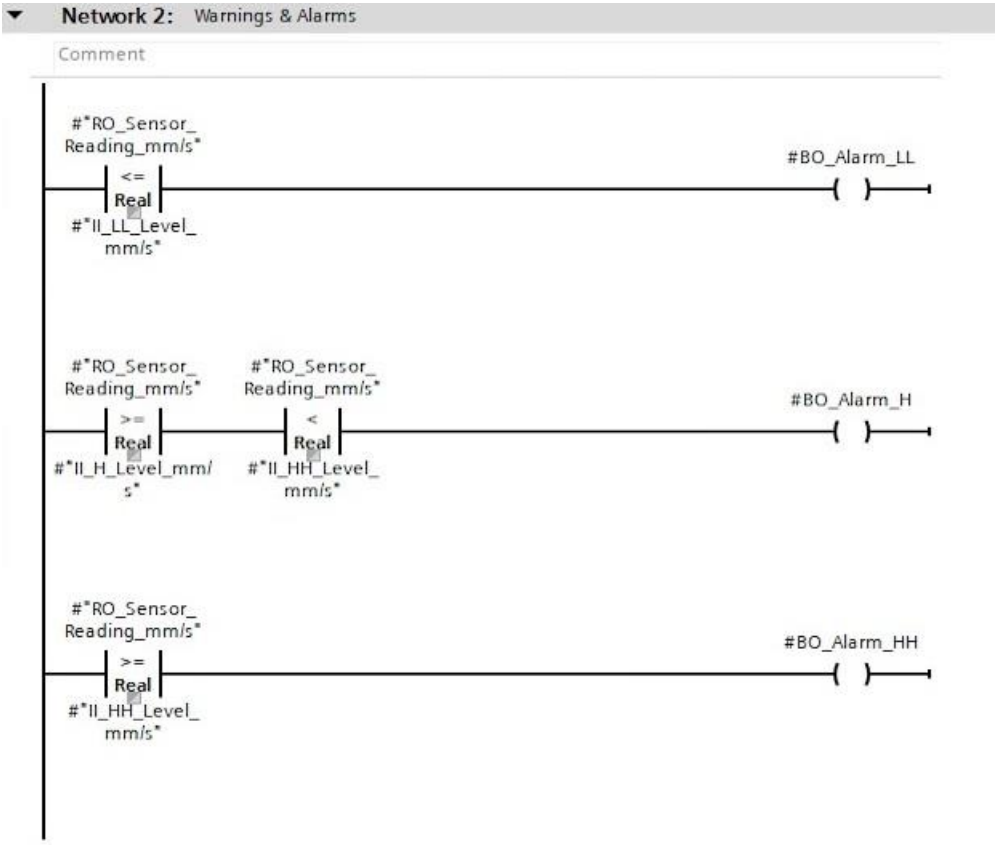


Slika 11. Mreža skaliranja očitanog signala u odgovarajuće jedinice

Program za obradu analognog signala senzora implementiran je tako da precizno skalira vrijednost očitanu sa senzora u odgovarajuće jedinice. U ovom slučaju, ulazni signal se prevodi u milimetre po sekundi (mm/s), što olakšava očitavanje stvarnih razina vibracija. Prva faza u procesu uključuje pretvaranje signala iz senzora u omjer koji prikazuje koliki je postotak maksimalne vrijednosti očitanja senzora postignut. To se postiže dijeljenjem izmjerene vrijednosti senzora s referentnom vrijednošću 27.648, što predstavlja maksimalni raspon signala koji PLC može obraditi u ovom sustavu. Rezultat je vrijednost između 0 i 1 koja omogućuje skaliranje na standardizirani način.

Nakon toga, dobiveni omjer koristi se za izračun stvarne vrijednosti u odgovarajućim jedinicama s pomoću formule za skaliranje. Formula koristi visoku i nisku vrijednost skale kako bi se rezultat prilagodio unutar željenog raspona. Proračun se vrši množenjem omjera s razlikom između visoke i niske vrijednosti skale, te se konačno dodaje niska vrijednost skale, čime se dobiva točan iznos u mm/s (*slika 11*.).

Konačna skalirana vrijednost, pohranjuje se u varijablu *RO_Sensor_Reading_mms*. Ova varijabla omogućuje daljnju obradu podataka o vibracijama u sustavu ili prikazivanje rezultata na korisničkom sučelju.



Slika 12. Mreža upozorenja i alarma

Funkcija upozorenja i alarma dizajnirana je za prepoznavanje situacija koje mogu zahtijevati intervencije ili korektivne akcije, osiguravajući time sigurnost i pouzdanost opreme u radu. Sustav nadzora uključuje nekoliko različitih razina alarma. Prva razina alarma se aktivira kada vrijednost vibracija dosegne ili padne ispod praga za vrlo nisku razinu vibracija. Ova razina upozorava na situaciju u kojoj su vibracije izuzetno niske, što može upućivati na mogući kvar senzora ili neki drugi problem u mjerenju. Aktiviranjem ovog alarma sustav upozorava operatere na potencijalni problem koji zahtijeva provjeru opreme ili senzora. Mreža prikazana na *slici 12.* upravlja generiranjem alarma i upozorenja na temelju skalirane vrijednosti očitanja senzora.

Druga razina, visoke vibracije (H), aktivira se kada vibracije prelaze određeni prag za visoku razinu (H), ali još uvijek ostaju ispod praga za vrlo visoku razinu (HH). Ova razina alarma pokazuje da vibracije premašuju normalne operativne granice, što može signalizirati prve znakove problema u radu opreme, kao što su pojačano trošenje, potencijalna nesigurnost ili preopterećenje sustava. Ova razina alarma zahtijeva pažnju, ali situacija još nije kritična i operateri mogu poduzeti preventivne mjere.

Treća i najviša razina, izrazito visoke vibracije, pokreće se kada vrijednost vibracija dosegne ili premaši prag za vrlo visoku razinu (HH). Ovaj alarm predstavlja kritično stanje koje zahtijeva

hitnu intervenciju, jer izrazito visoke vibracije mogu dovesti ne samo do oštećenja opreme, već do sigurnosnih rizika ili čak nesreća. Aktivacija ovog alarma upozorava operatere na situaciju u kojoj su potrebne trenutačne korektivne mjere kako bi se spriječili daljnji problemi ili oštećenja. Na ovaj način, sustav nadzora vibracija omogućava pravovremeno reagiranje na sve promjene u uvjetima rada, čime se postiže siguran i pouzdan rad cjelokupnog sustava putem mjera za upozorenje i zaštitu.

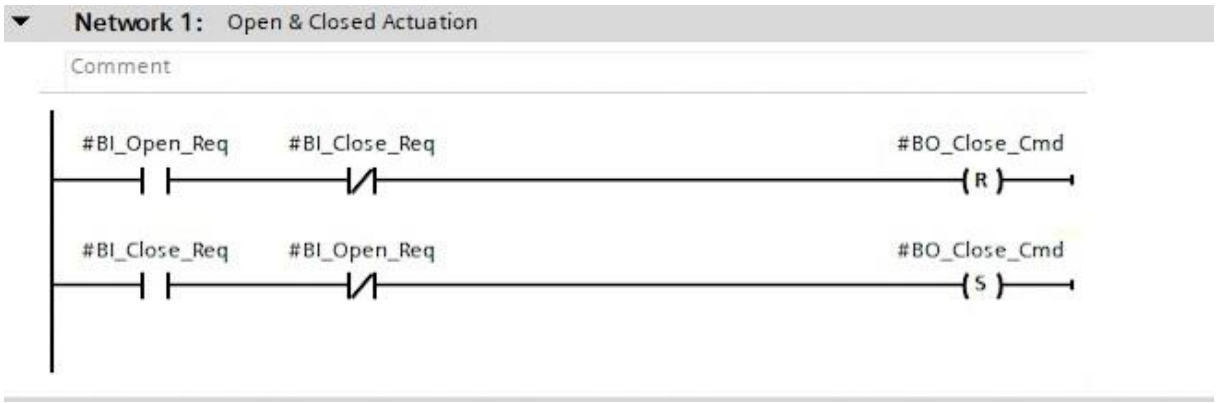
4.3 Upravljanje kočnicom

Poseban funkcijski blok koristi se za upravljanje kočnicom vlaka. Ovim funkcijskim blokom se postiže upravljanje radom kočnice, praćenje njezinog statusa, prepoznavanje i rukovanje greškama te resetiranje grešaka kada je potrebno. Ulazne i izlazne varijable ovog funkcijskog bloka dane su na *slici 13.*

FB_Actuator_Handler

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
1	Input				☐	☐	☐	☐		
2	BI_Open_Req	Bool	false	Non-retain	☑	☑	☑	☐		
3	BI_Close_Req	Bool	false	Non-retain	☑	☑	☑	☐		
4	BI_Open_Feedback	Bool	false	Non-retain	☑	☑	☑	☐		
5	BI_Closed_Feedback	Bool	false	Non-retain	☑	☑	☑	☐		
6	BI_Default_Pos	Bool	false	Non-retain	☑	☑	☑	☐		
7	T_Fault_Delay	Bool	false	Non-retain	☑	☑	☑	☐		
8	BI_Fault_Reset	Bool	false	Non-retain	☑	☑	☑	☐		
9	<Add new>				☐	☐	☐	☐		
10	Output				☐	☐	☐	☐		
11	IO_Status	Int	0	Non-retain	☑	☑	☑	☐		
12	BO_Open_Sts	Bool	false	Non-ret...	☑	☑	☑	☐		
13	BO_Close_Sts	Bool	false	Non-retain	☑	☑	☑	☐		
14	BO_Moving_Sts	Bool	false	Non-retain	☑	☑	☑	☐		
15	BO_Fault_Sts	Bool	false	Non-retain	☑	☑	☑	☐		
16	BO_Close_Cmd	Bool	false	Non-retain	☑	☑	☑	☐		

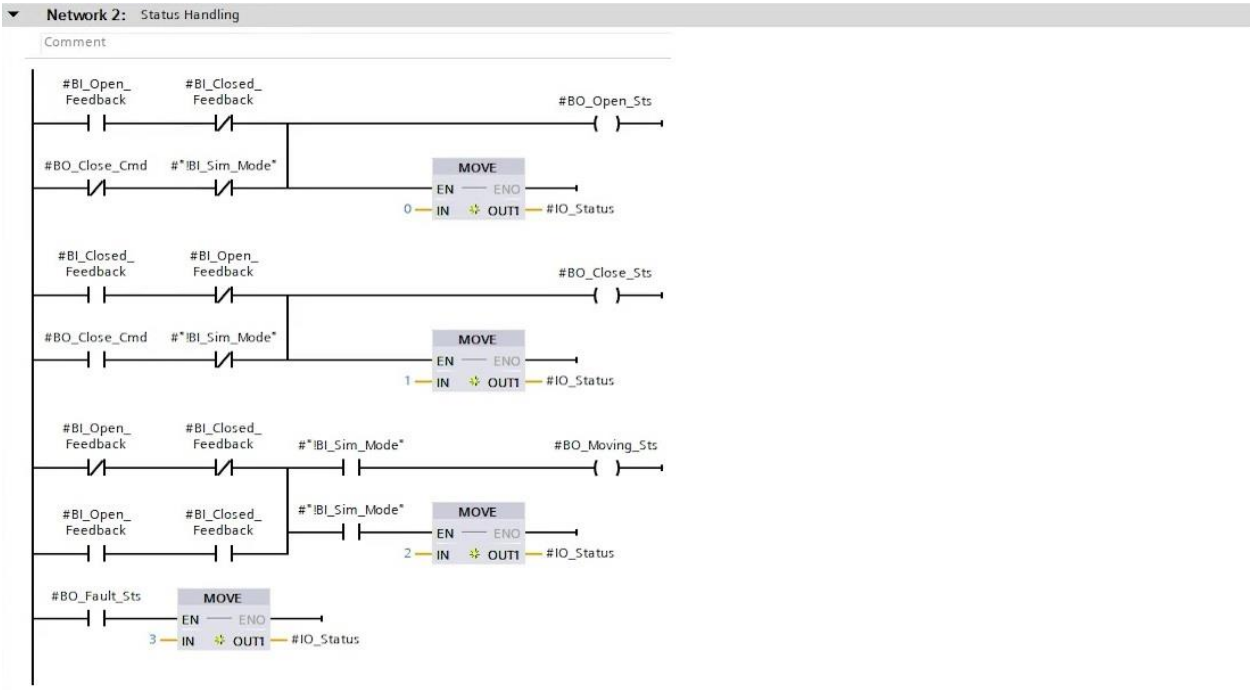
Slika 13. Ulazne i izlazne varijable funkcijskog bloka upravljanja kočnicom



Slika 14. Mreža upravljanja kočnicom, aktiviranje i otpuštanje kočnice

Upravljanje kočnicom, odnosno aktiviranje i otpuštanje kočnice (*slika 14.*) odvija se na temelju zahtjeva. Ako postoji zahtjev za otpuštanje kočnice (BI_Open_Req) i nema zahtjeva za aktiviranjem kočnice (BI_Close_Req), tada se naredba za aktiviranje kočnice (BO_Close_Cmd)

resetira (logička nula), što znači da se vozilo se može kretati slobodno. Kada je potrebno da se vlak zaustavi, mora postojati zahtjev za aktiviranjem kočnice, a pritom ne smije postojati zahtjev za njezinim otpuštanjem. Tada se postavlja potrebna naredba, što znači da će se kočnica aktivirati i vozilo će usporavati do zaustavljanja.



Slika 15. Upravljanje statusom kočnice

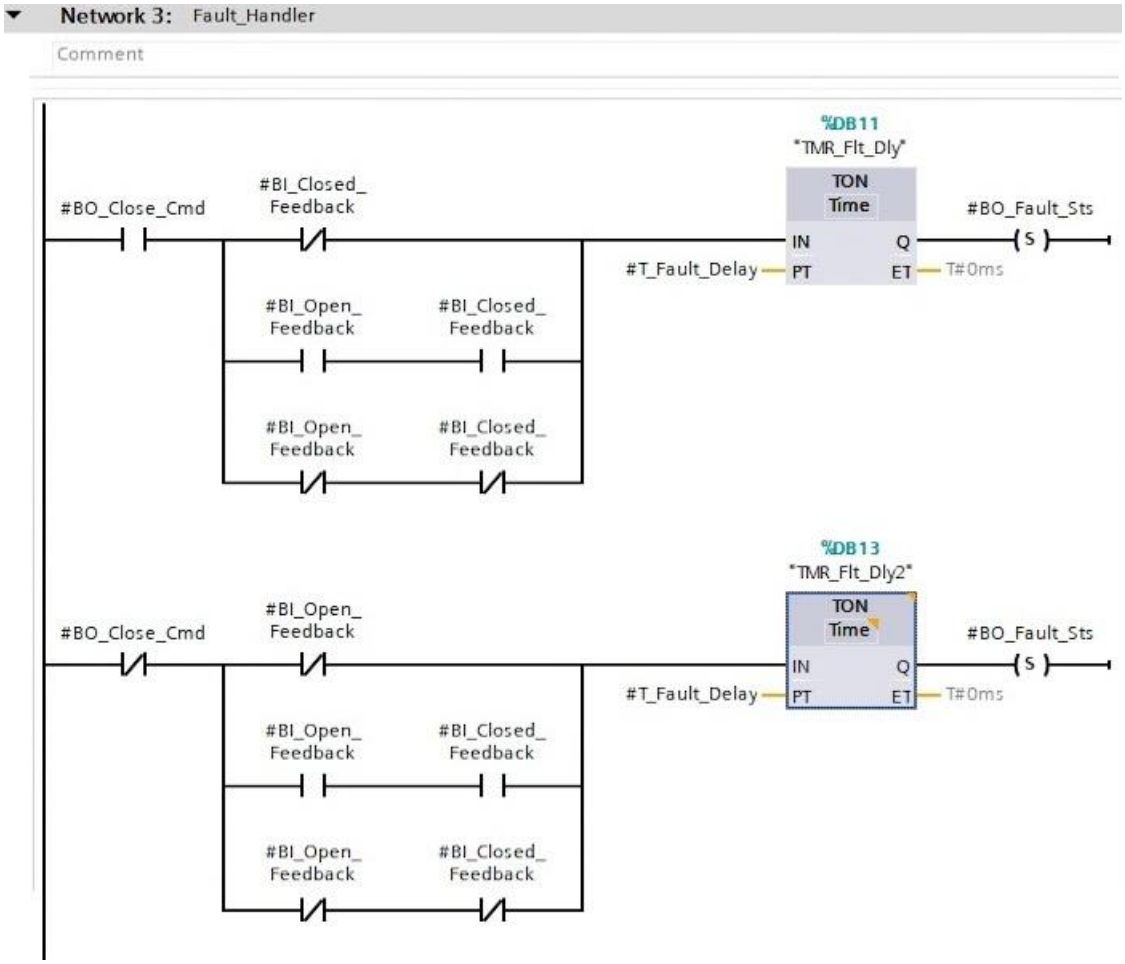
Logika upravljanja kočnicom postavljena je tako da precizno određuje stanje kočnice na temelju povratnih informacija i simulacijskog režima rada. Moguća su četiri različita statusa kočnice: otpušteno, aktivirano, prijelazno i stanje greške, pružajući detaljno praćenje položaja kočnice.

Prvi dio koda na *slici 15.* određuje stanje otpuštene kočnice. Ako postoji povratna informacija da je kočnica otpuštena (signal BI_Open_Feedback je aktivan) i istovremeno nema povratne informacije da je kočnica aktivirana, izlazni status (BO_Open_Sts) postaje aktivan, što označava da je kočnica otpuštena. Uz to, blok *MOVE* postavlja vrijednost statusa kočnice (IO_Status) na 0, što dodatno naglašava da je kočnica u otpuštenom odnosno neaktivnom stanju.

Drugi dio ove logike odnosi se na prikaz stanja aktivirane kočnice. Ako je prisutna povratna informacija da je kočnica aktivirana (BI_Closed_Feedback je aktivan) i nema povratne informacije da je kočnica otpuštena, izlazni status (BO_Close_Sts) postaje aktivan, čime se signalizira da je kočnica aktivirana. U ovom slučaju, blok *MOVE* postavlja vrijednost statusa kočnice na 1, označavajući stanje aktivirane kočnice.

Treći dio prikazane logike, označava prijelazno stanje, odnosno stanje kada je kočnica u pokretu između otpuštenog i aktiviranog položaja. Ako nema povratne informacije da je kočnica otpuštena niti da je aktivirana, ili ako su obje povratne informacije istovremeno aktivne, izlazni status (BO_Moving_Sts) postaje aktivan, što označava da je kočnica u prijelaznom stanju. U tom slučaju, blok *MOVE* postavlja vrijednost statusa na 2, čime dodatno naglašava ovo stanje pokreta. U simulacijskom načinu rada ovo prijelazno stanje može biti zanemareno kako bi se spriječile potencijalne lažne indikacije.

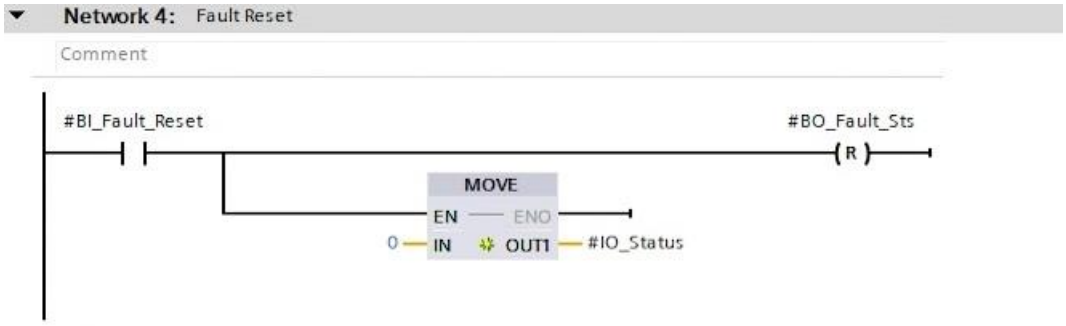
Posljednji dio logike prikazane na *slici 15.* namijenjen je prikazu stanja greške. Ako je prisutan signal greške (BO_Fault_Sts), blok *MOVE* postavlja vrijednost statusa na 3, što označava da je kočnica u stanju greške. Ova indikacija jasno ukazuje na problem s kočnicom koji zahtijeva pažnju ili intervenciju operatera.



Slika 16. Upravljanje greškama kočnice

Mreža prikazana na *slici 16.* služi za upravljanje greškama, posebno kašnjenjem u odgovoru kočnice na zadanu naredbu. Kada se izda naredba za aktivaciju kočnice (BO_Close_Cmd) i nakon toga nema povratne informacije da je kočnica aktivirana (BI_Closed_Feedback) unutar određenog

perioda, vremenski blok počinje s odbrojavanjem. Kada istekne podešeno vrijeme odbrojavanja bez zaprimanja povratne informacije, aktivira se signal greške (BO_Fault_Sts). Slično tome, kada postoji konfliktna situacija gdje oba senzora odašilju povratnu informaciju istovremeno ili nijedan ne pokazuje povratnu informaciju, također se generira greška.

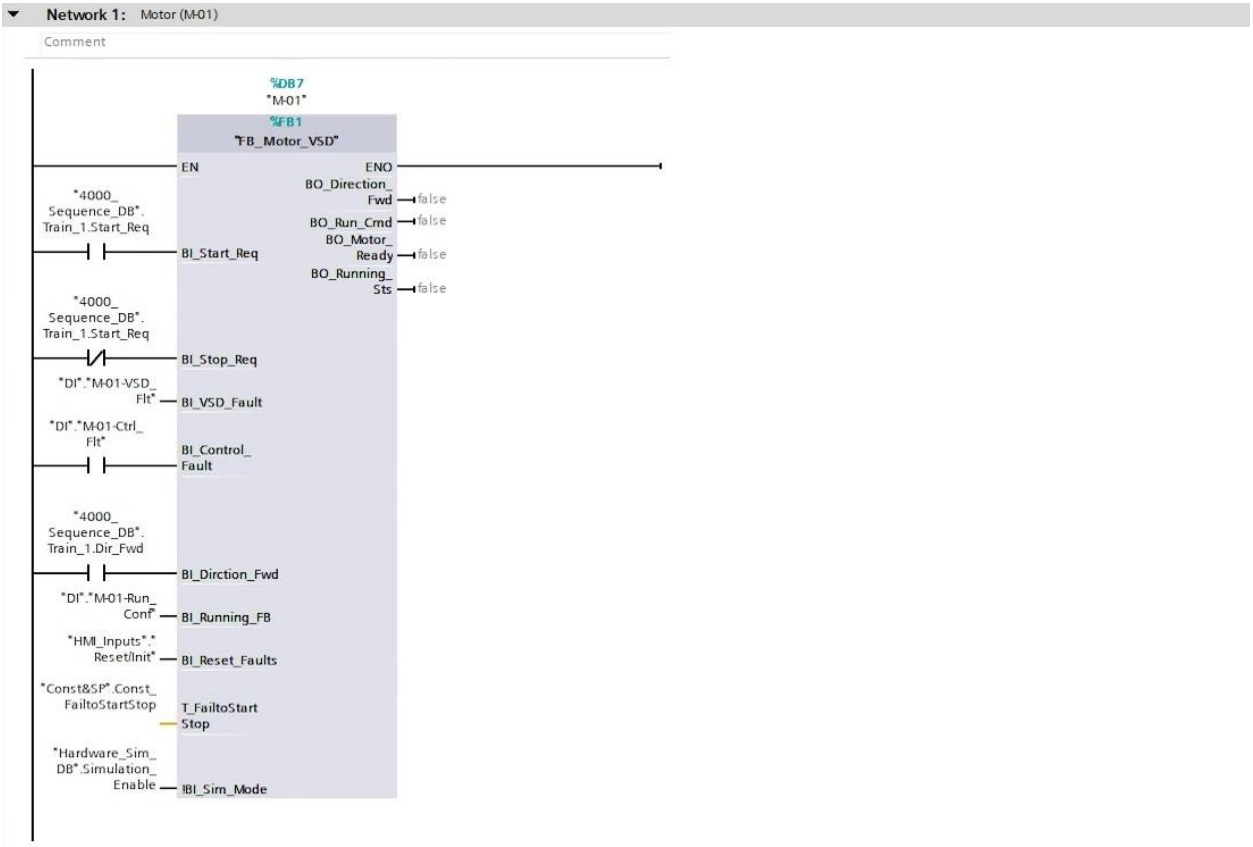


Slika 17. Mreža resetiranja grešaka kočnice

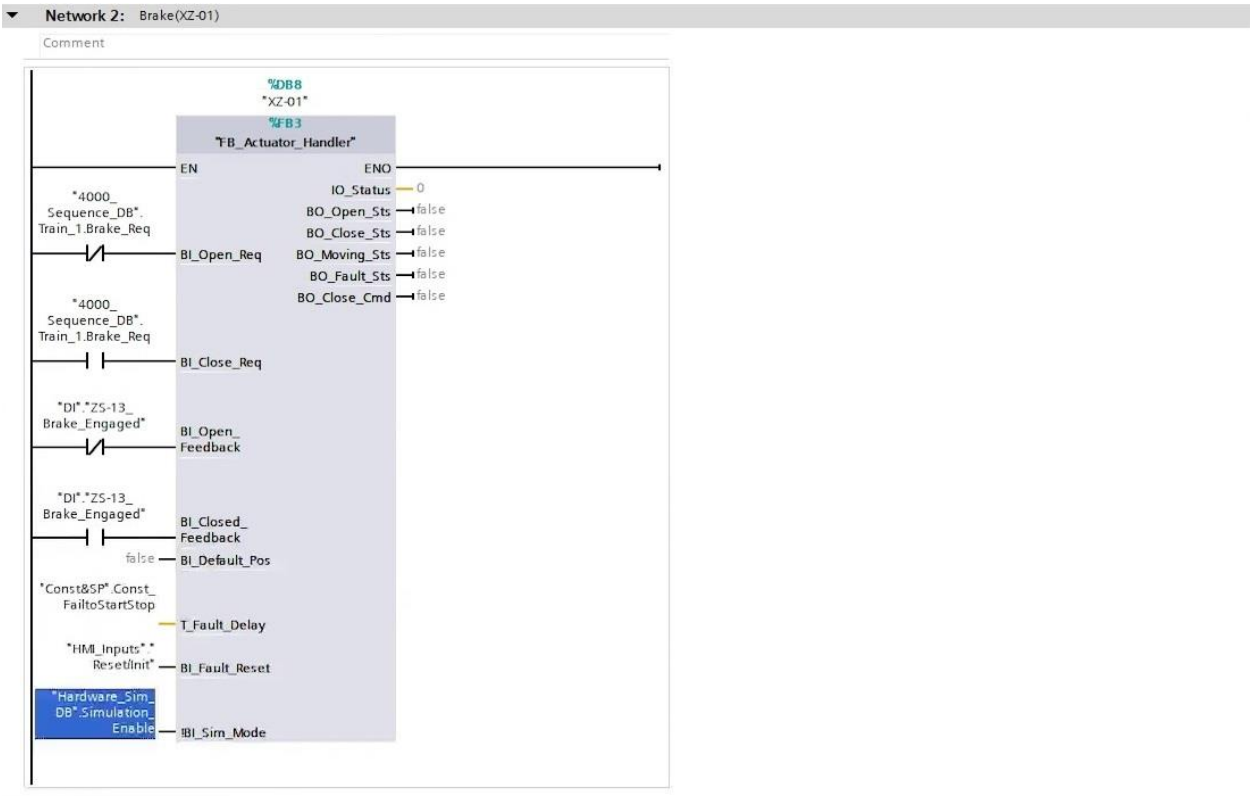
Resetiranje stanja greške kočnice provodi se kako bi nakon prepoznavanja i uklanjanja greške, sustav mogao nesmetano nastaviti s radom (*slika 17.*). Ako postoji zahtjev za resetiranjem greške (BI_Fault_Reset), signal greške (BO_Fault_Sts) se resetira na nulu. Također se postavlja vrijednost statusa (IO_Status) na 0, što označava da nema grešaka u sustavu.

4.4 Pozivanje funkcijskih blokova unutar funkcije upravljanja

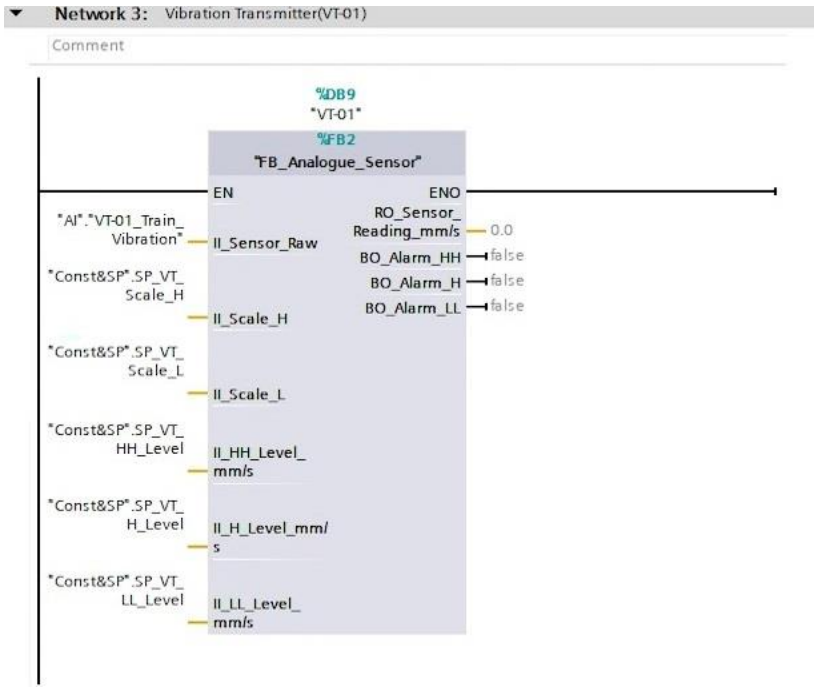
Iako su već razvijeni funkcijski blokovi za upravljanje dijelovima stroja poput kočnica i senzora, sada je potrebno napisati glavni program koji će upravljati cjelokupnim procesom, uključujući ubrzanje, kočenje i zaustavljanje vlaka. U ovom potpoglavlju se pripremaju i povezuju instance funkcijskih blokova koji će se primjenjivati u glavnom programu. Stvara se funkcija (FC) u koju će se pozvati prethodno navedeni funkcijski blokovi. U funkciju ćemo pozvati funkcijske blokove: upravljanja motorom, upravljanja kočnicom i upravljanja analognim senzorima. Na sljedeće tri slike (*slika 18. 19. i 20.*) prikazani su funkcijski blokovi pozvani u FC program upravljanja.



Slika 18. Funkcijski blok upravljanja motorom

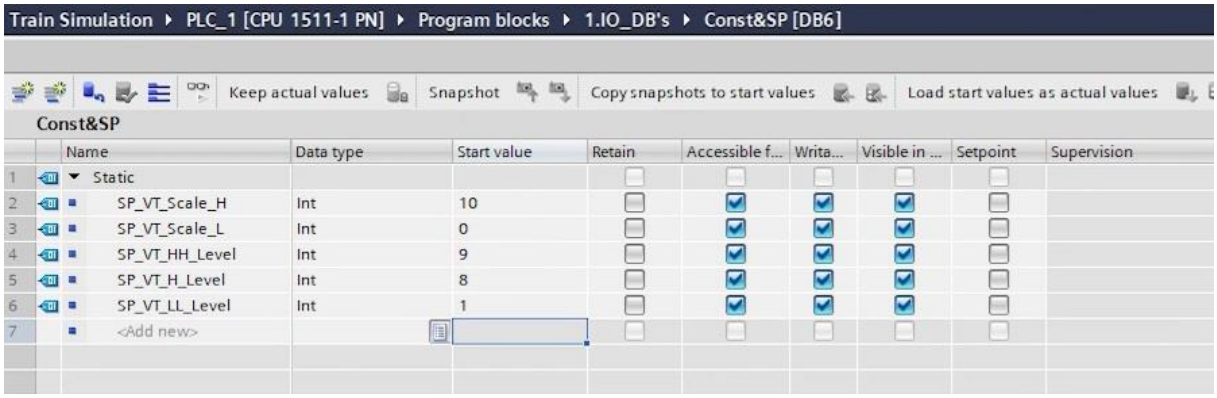


Slika 19. Funkcijski blok upravljanja kočnicom



Slika 20. Funkcijski blok upravljanja sustavom koji mjeri vibracije vlaka

Za potrebe funkcijskog bloka upravljanja vibracijama, moramo odrediti razine vibracije, odnosno visoke i niske skale, putem podatkovnog bloka za konstante i referentne točke (*slika 21*.).



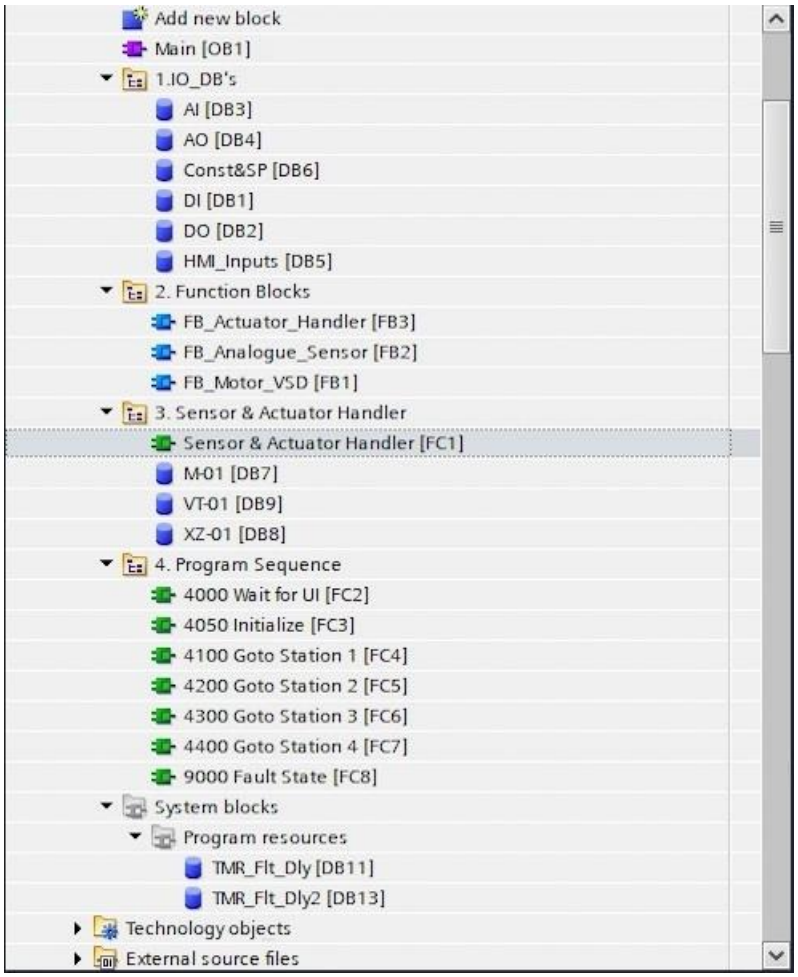
Slika 21. Visoke i niske razine vibracija, definirane u podatkovnom bloku konstanti i referentnih točki

5. GLAVNA PROGRAMSKA SEKVENCA

Programska sekvenca je glavni dio programa koji definira slijed stanja kroz koje sustav prolazi. Svaka funkcija (FC) unutar glavnog programa predstavlja određeno stanje vlaka ili korak u operaciji, kao što su:

- čekanje na korisnički unos (4000),
- inicijalizacija (4050),
- odlazak na stanicu 1 (4100),
- stanje kvara (9000).

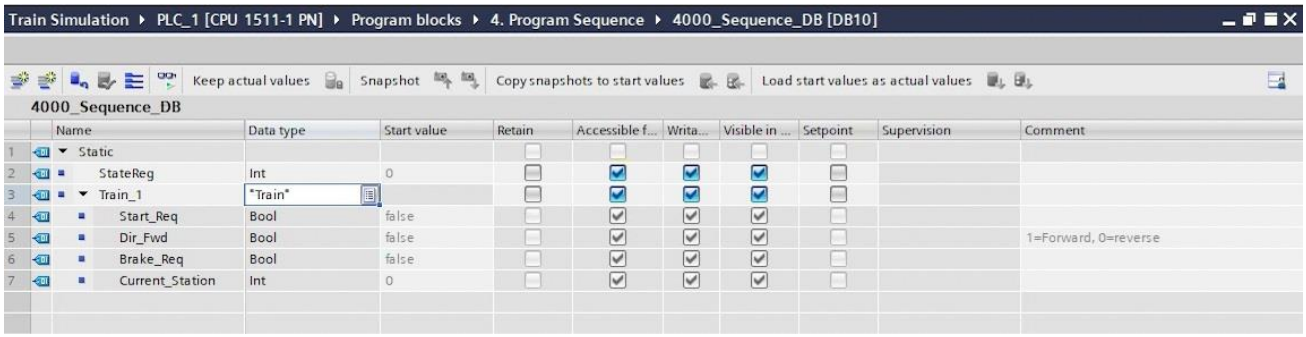
Izbornik s glavnom programskom sekvencom i ostalim dijelovima programa prikazan je na *slici 22.*



Slika 22. Pregled programskog izbornika

Glavna programska sekvenca imat će poseban podatkovni blok za pohranu varijabli pod nazivom 4000_Sequence_DB (*slika 23.*). U ovome programskom bloku *Train_1* odjeljak sadrži varijable

koje upravljaju radom vlaka, poput pokretanja, smjera kretanja (unaprijed ili unazad), zahtjeva za kočenjem te trenutačne stanice na kojoj se vlak nalazi.

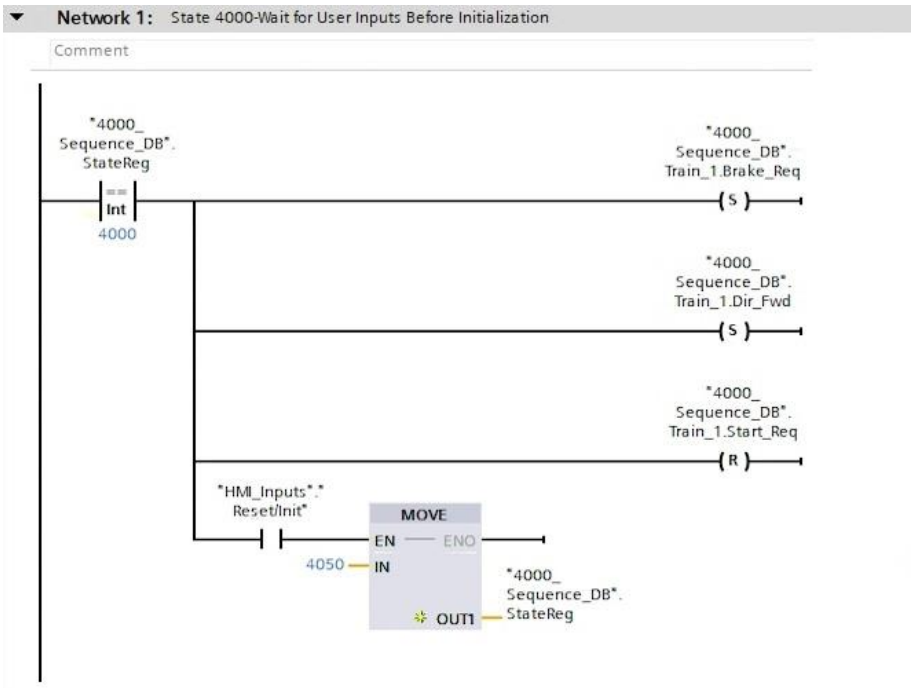


Slika 23. Podatkovni blok za pohranu varijabli glavne sekvence programa

5.1 Čekanje na unos naredbi od strane korisnika

U sustavu se koristi programiranje po sekvencama za upravljanje različitim stanjima vlaka, osiguravajući da vlak reagira na korisničke unose i prelazi u odgovarajuća stanja u skladu s programskim logikom.

Unutar glavne programske sekvence, prva funkcija se odnosi na prvo moguće stanje vlaka, a to je čekanje unosa naredbe korisnika prije inicijalizacije. Pripadajuća mreža dana je na *slici 24.*



Slika 24. Mreža čekanja na unos korisničke naredbe

U prvom dijelu prve logičke grane koja je prikazana na *slici 24.* odvija se usporedba stanja vlaka, na način da se vrijednost u registru stanja uspoređuje s vrijednošću 4000. Ako je ova vrijednost jednaka, sustav prepoznaje da je trenutačno u stanju čekanja. Na ovaj način sustav može voditi računa o tome u kojem se stanju nalazi, što je u ovom slučaju, stanje 4000, odnosno stanje čekanja na unos naredbe korisnika.

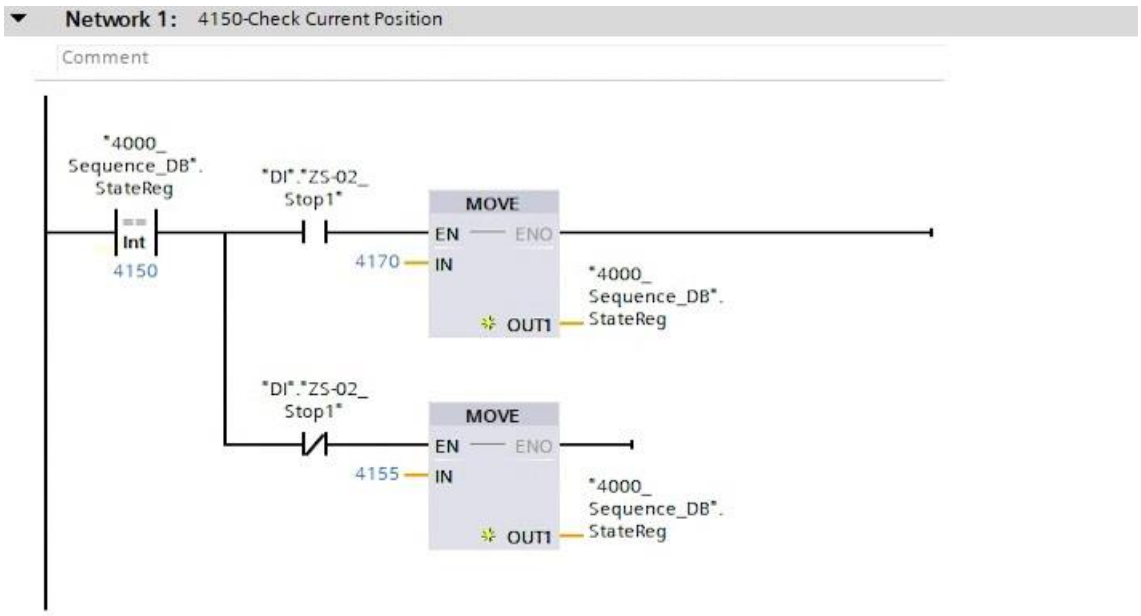
Ako je sustav u stanju 4000, zahtjev za kočenje (Train_1.Brake_Req) se postavlja u stanje 1. To znači da se kočnice aktiviraju i vlak ostaje stacioniran. Aktiviranje kočnica osigurava da vlak ostane sigurno zaustavljen dok sustav čeka na naredbu korisnika. Ovom funkcijom se sprječava neželjeno kretanje vlaka.

U istom stanju (4000), zahtjev za smjerom kretanja unaprijed (Train_1.Dir_Fwd) se resetira na 0, što znači da vlak neće biti spreman za izvršavanje te radnje. Ovim resetiranjem osigurava se da vlak ne započne kretanje prema naprijed dok nije primljena naredba za pokretanje. Također se resetira zahtjev za pokretanjem vlaka (Train_1.Start_Req) na 0, čime se osigurava da motor vlaka nije pokrenut.

U drugoj logičkoj grani se provjerava je li korisnik pritisnuo gumb za resetiranje ili inicijalizaciju putem HMI-ja. Ako je unos za resetiranje/inicijalizaciju aktivan, naredba *MOVE* se koristi za promjenu vrijednosti u registru stanja iz 4000 u stanje inicijalizacije (4050), što označava početak operativnih priprema za pokretanje vlaka.

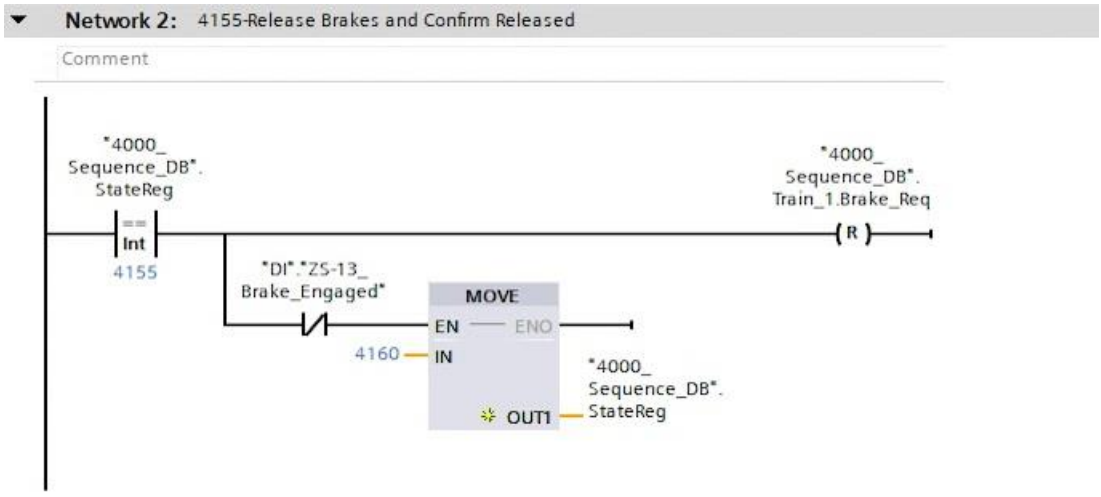
5.2 Glavni program: kretanje prema prvoj stanici

Program funkcije kretanja prema prvoj stanici upravlja kretanjem vlaka između stanica. Svrha ove funkcije (FC) je osigurati da vlak sigurno i učinkovito dolazi na određenu stanicu, uzimajući u obzir njegovu trenutačnu poziciju, otpuštanje kočnica, pokretanje motora, upravljanje brzinom te konačno zaustavljanje na odredišnoj stanici. Svaka mreža unutar funkcije upravlja specifičnim korakom u ovom procesu, prateći povratne informacije sa senzora kako bi se osiguralo ispravno izvršavanje svih akcija prije prelaska na sljedeći korak u sekvenci, kao što je prikazano na sljedećim slikama.



Slika 25. Provjera trenutačne pozicije

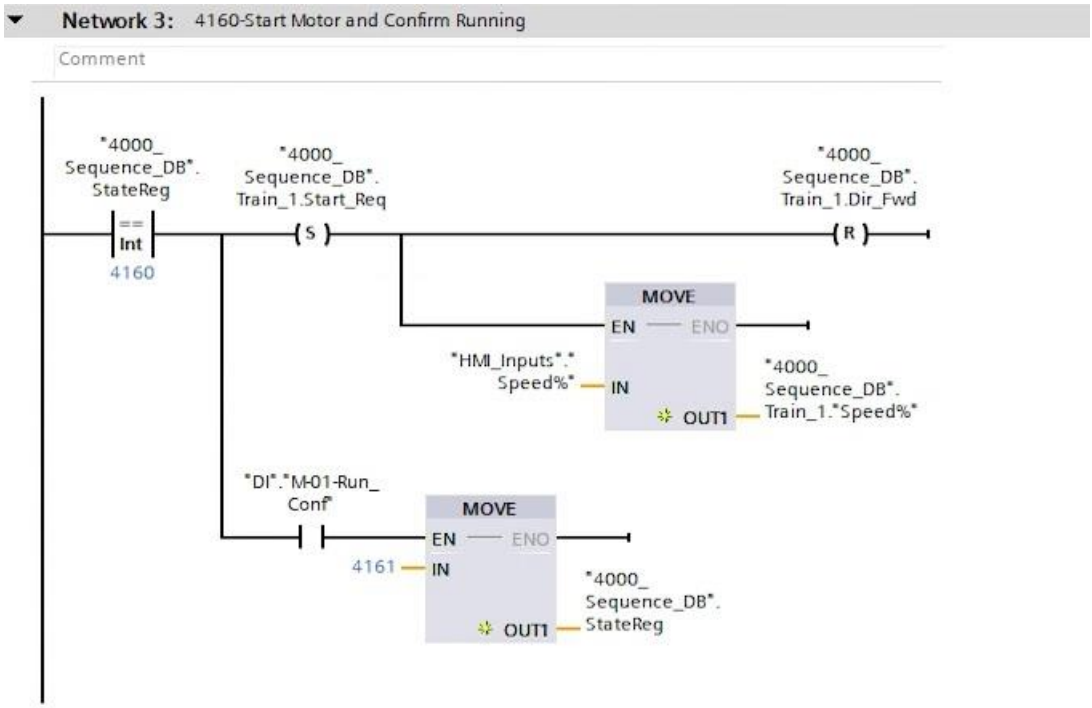
Dio koda koji izvršava provjeru nalazi li se vlak već na odredišnoj stanici – stanici 1, prikazan je na *slici 25.* Ako je trenutačno stanje 4150, sustav provjerava senzore pozicije. Ukoliko se vlak već nalazi na stanici 1 (zaustavni senzor DI_ZS_02_Stop1, smješten na prvoj stanici je aktivan), tada se stanje mijenja na 4170, što znači da se svi koraci za dolazak vlaka na stanicu preskaču jer vlak već stoji na odredištu. U suprotnom, stanje se mijenja na 4155, čime se nastavlja sekvenca za kretanje prema stanici.



Slika 26. Otpuštanje kočnica i potvrda otpuštanja

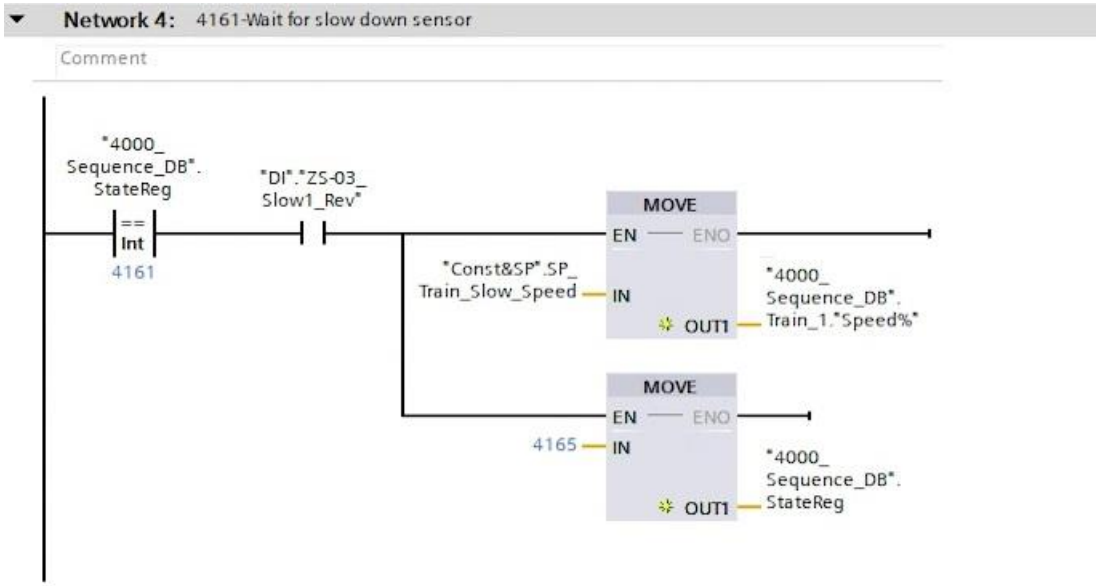
Ako je trenutačno stanje 4155, kočnice se otpuštaju resetiranjem signala za kočenje (Train_1.Brake_Req) kao što je prikazano na *slici 26.* Nakon što senzor

(DI_ZS_13_Brake_Engaged) potvrdi da su kočnice stvarno otpuštene, stanje se mijenja na 4160, čime započinje pokretanje motora.



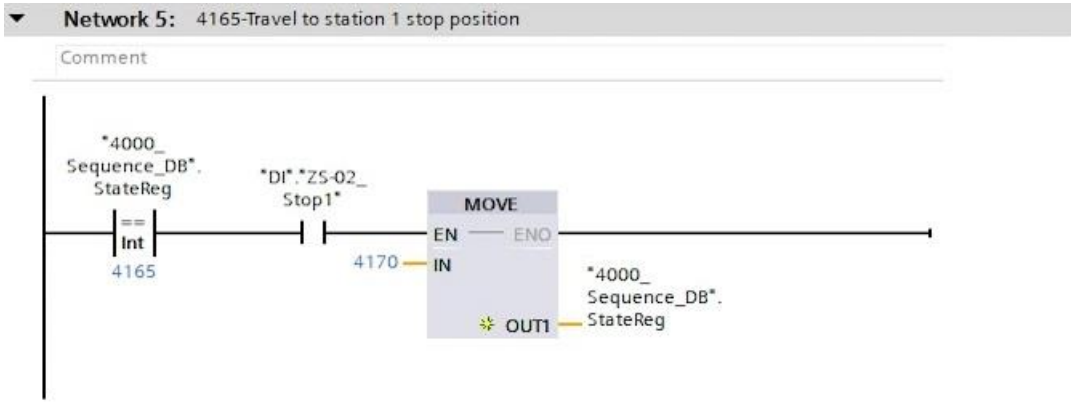
Slika 27. Pokretanje motora i potvrda rada

Kada je sustav u stanju 4160, postavlja se zahtjev za pokretanje motora (Train_1.Start_Req) na vrijednost 1, čime se signalizira da motor treba početi s radom. Smjer kretanja vlaka postavlja se na unatrag resetiranjem smjera unaprijed (Train_1.Dir_Fwd) na 0, što označava da će se motor sada kretati unatrag u smjeru od zadnje prema prvoj stanici. Nakon toga, sustav čeka povratnu informaciju od senzora (DI_M01_Run_Conf) koja potvrđuje da je motor pokrenut i radi. Kada senzor potvrdi rad motora, stanje se mijenja na 4161, čime sustav prelazi na sljedeći korak u sekvenci, a to je čekanje signala za usporavanje. Glavna svrha ovoga dijela programa je osigurati da se motor pokrene u ispravnom smjeru i da se potvrdi njegov rad prije nego što se nastavi s daljnjim koracima upravljanja vlakom.



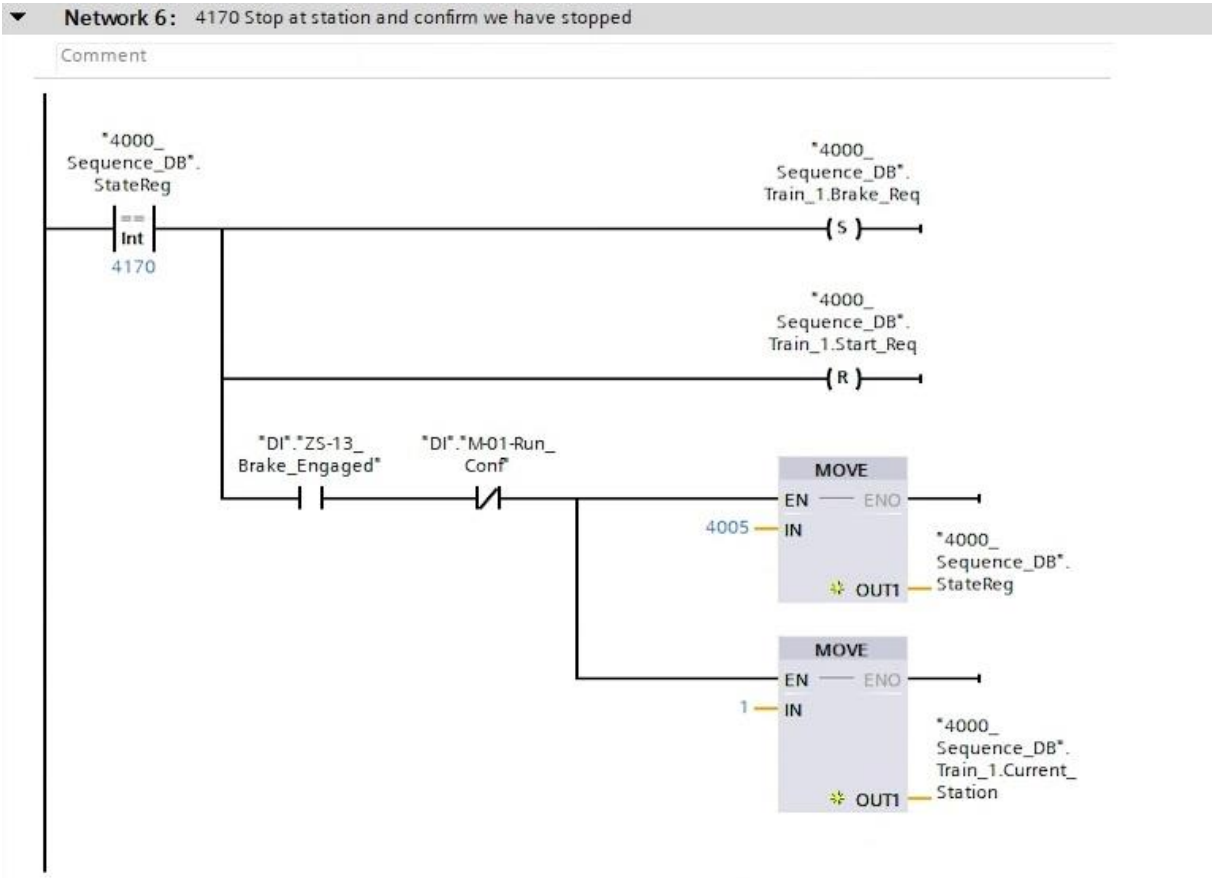
Slika 28. Čekanje signalizacije usporavanja

Ako je trenutačno stanje 4161, vlak se kreće prema stanici i čeka signal aktiviranog senzora za usporavanje (DI_ZS_03_Slow1_Rev). Kada se ovaj senzor aktivira, brzina vlaka se smanjuje na unaprijed definiranu sporiju brzinu (SP_Train_Slow_Speed), nakon čega se stanje mijenja na 4165, što označava dolazak do točke zaustavljanja na stanici (*slika 28.*).



Slika 29. Dolazak vlaka do pozicije zaustavljanja za prvu stanicu

Ako je trenutačno stanje 4165, sustav čeka da vlak dođe do pozicije zaustavljanja na stanici 1 (koju signalizira zaustavni senzor DI_ZS_02_Stop1). Kada vlak stigne na poziciju zaustavljanja, stanje sustava se mijenja na 4170, čime se pokreće sekvenca zaustavljanja vlaka.



Slika 30. Zaustavljanje na stanici i potvrda zaustavljanja

Kada je sustav u stanju 4170, vlaku se aktiviraju kočnice putem zahtjeva za kočenje (Train_1.Brake_Req), te se motor zaustavlja i prestaje s radom, resetiranjem zahtjeva za pokretanje (Train_1.Start_Req). Nakon što se potvrdi da su kočnice uspješno primijenjene (DI_ZS_13_Brake_Engaged) i da je motor zaustavljen (DI_M01_Run_Conf), stanje sustava se mijenja na 4005, što signalizira sustavu čekanje daljnjih naredbi od strane korisnika. Dodatno, program postavlja trenutačni status stanice vlaka na vrijednost 1 (Train_1.Current_Station), označavajući da je vlak stigao na prvu stanicu, kako je i prikazano na *slici 30.* Na ovaj način je realizirano sigurno zaustavljanje vlaka na stanici i priprema sustava za primanje novih naredbi.

5.3 Testiranje sustava

U ovom potpoglavlju testirana je simulacija osmišljenog automata s konačnim stanjima (engl. finite state machine). Najprije se sve funkcije pozivaju u glavni organizacijski blok (*Main OB)*. Nakon što su pozvane, funkcije se zapisuju u PLC. Kada odaberemo kod kretanja prema prvoj stanici i pokrenemo ikonu za „nadzor“, primijetit ćemo da je stanje sustava postavljeno na 0. To

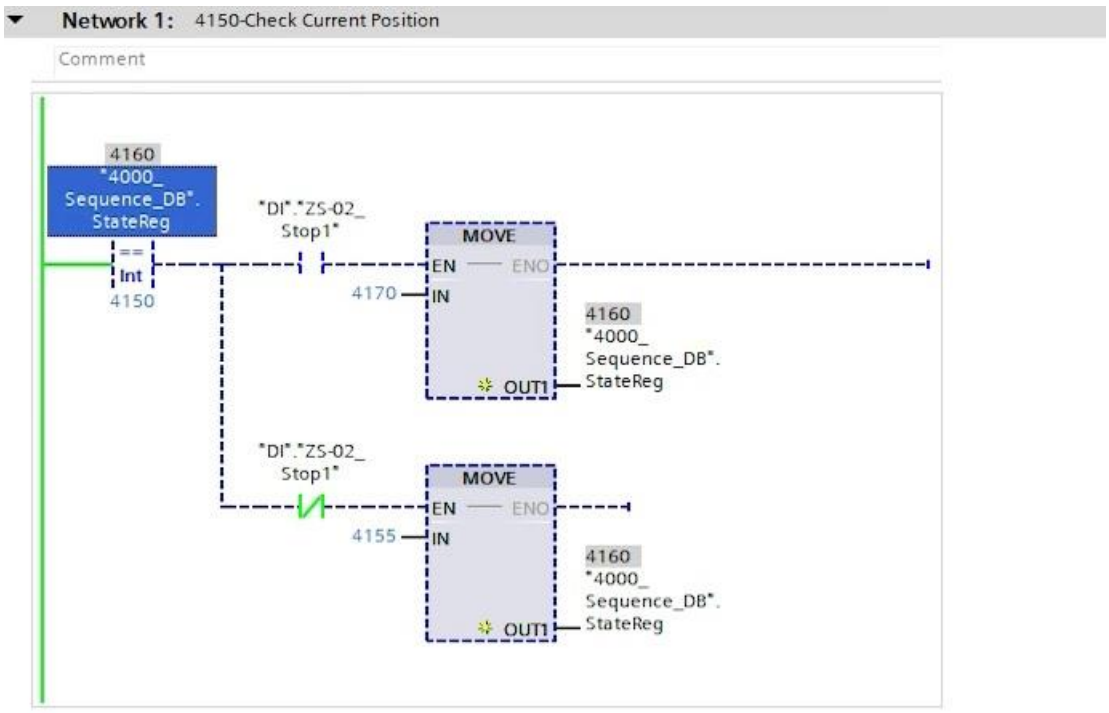
znači da se kod trenutačno ne izvršava jer usporedba stanja provjerava stanje našeg stroja, te stoga kod nije aktivan.

Kako bi se prešlo u odgovarajuće stanje za potrebe testiranja, ručno ćemo modificirati vrijednost stanja na 4150. Kada to učinimo, primjećujemo da se aktiviraju dijelovi koda povezani s tim stanjem. Budući da se vlak već prethodno ne nalazi na prvoj stanici, sustav će prepoznati da zaustavni senzor prve stanice nije aktivan te odmah preskočiti u stanje 4155. Ako simuliramo dalje, vidjet ćemo da sustav prelazi u stanje 4160, gdje se od nas traži pokretanje vlaka, postavljanje smjera kretanja unatrag (prema prvoj stanici) i podešavanje brzine preko HMI unosa.

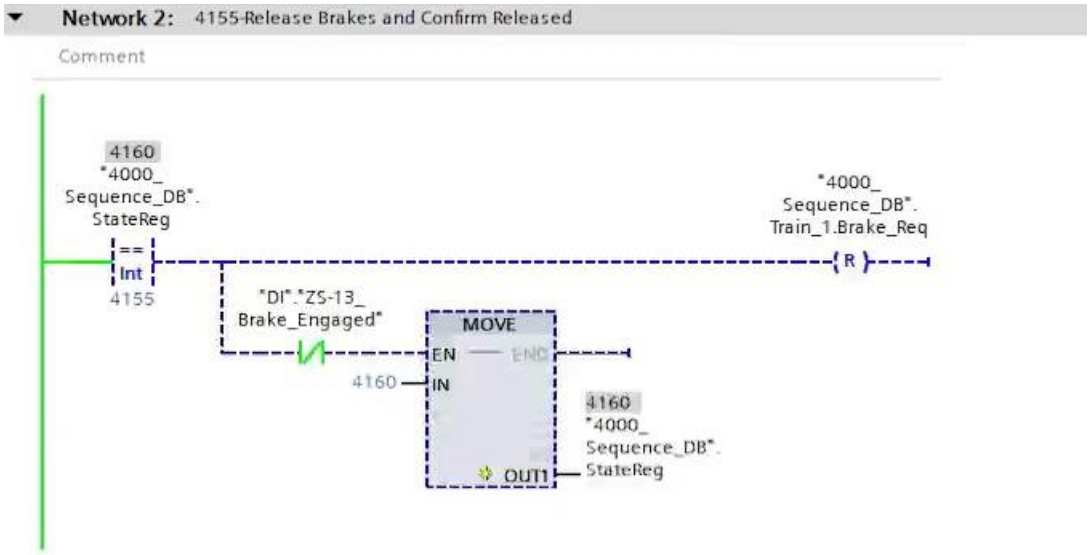
Nakon toga, sustav čeka potvrdu da je motor pokrenut; ako simuliramo da je ta potvrda stigla, stanje se mijenja na 4161, gdje sustav očekuje signal za usporavanje. Ovaj način rada omogućuje precizno upravljanje kodom, jer se aktivira samo onaj dio koda koji je potreban u određenom stanju, dok se ostali dijelovi koda ignoriraju. To čini ovaj pristup izuzetno pouzdanim i jednostavnim za otklanjanje grešaka, jer se uz pomoć informacije o trenutačnom stanju vlaka, može fokusirati na taj specifičan dio koda.

Daljnjom simulacijom, senzor za usporavanje postaje aktivan, te sustav prelazi u stanje 4165 gdje se vlak kreće prema stanici. U nastavku, kada se detektira signal zaustavljanja na stanici, stanje se mijenja na 4170 gdje sustav primjenjuje kočnice i zaustavlja motor. Čeka se potvrda da su kočnice aktivirane i motor zaustavljen, te program prelazi na sljedeće stanje.

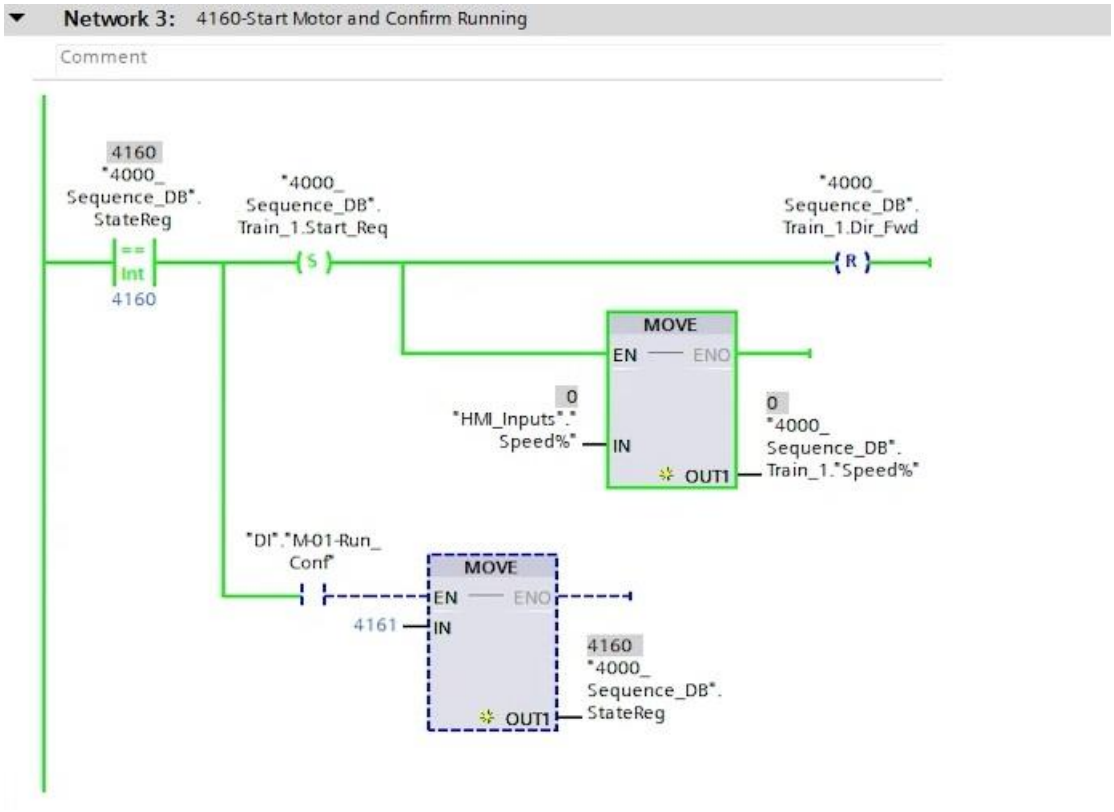
Ovo je kratki pregled našeg automata s konačnim stanjima koji nam pokazuje kako će program funkcionirati kada bude dovršen. Namjera ovog testiranja, prikazanog na *slici 31.* je bila pružiti uvid u to kako se razvija sustav.



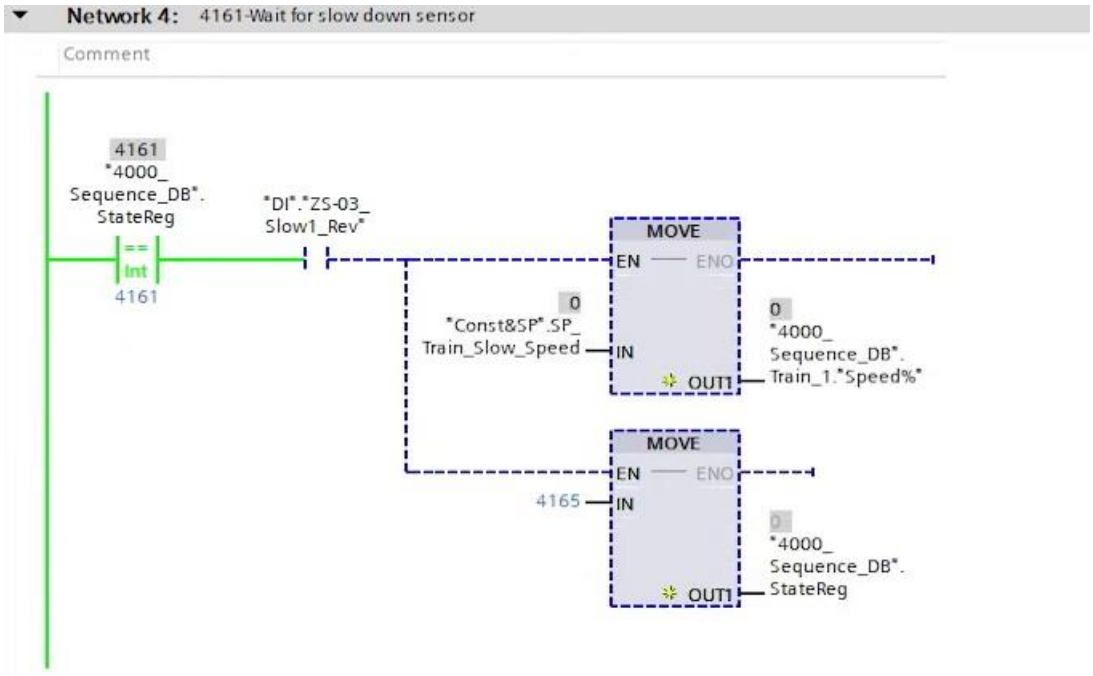
a) Modificiranjem vrijednosti stanja na 4150, sustav provjerava je li aktiviran senzor zaustavljanja na stanici 1. Ako senzor nije aktivan, sustav prelazi u stanje 4155 za izvršavanje daljnjih naredbi, a ako je senzor aktivan, preskaču se potrebni koraci i prelazi se u stanje 4170 jer se vlak već nalazi na zadanoj stanici.



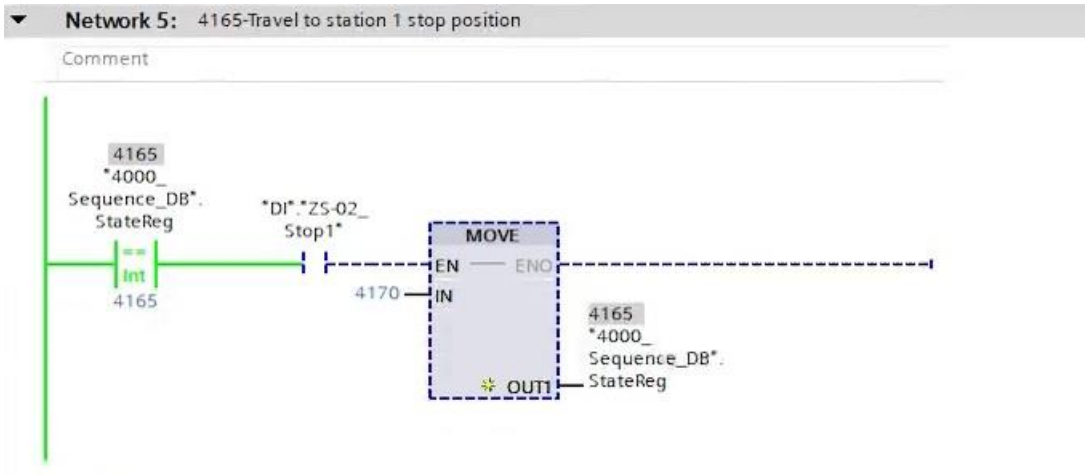
b) U stanju 4155, sustav resetira zahtjev za kočenjem kako bi se otpustile kočnice. Ako senzor potvrdi da su kočnice otpuštene, sustav prelazi u stanje 4160.



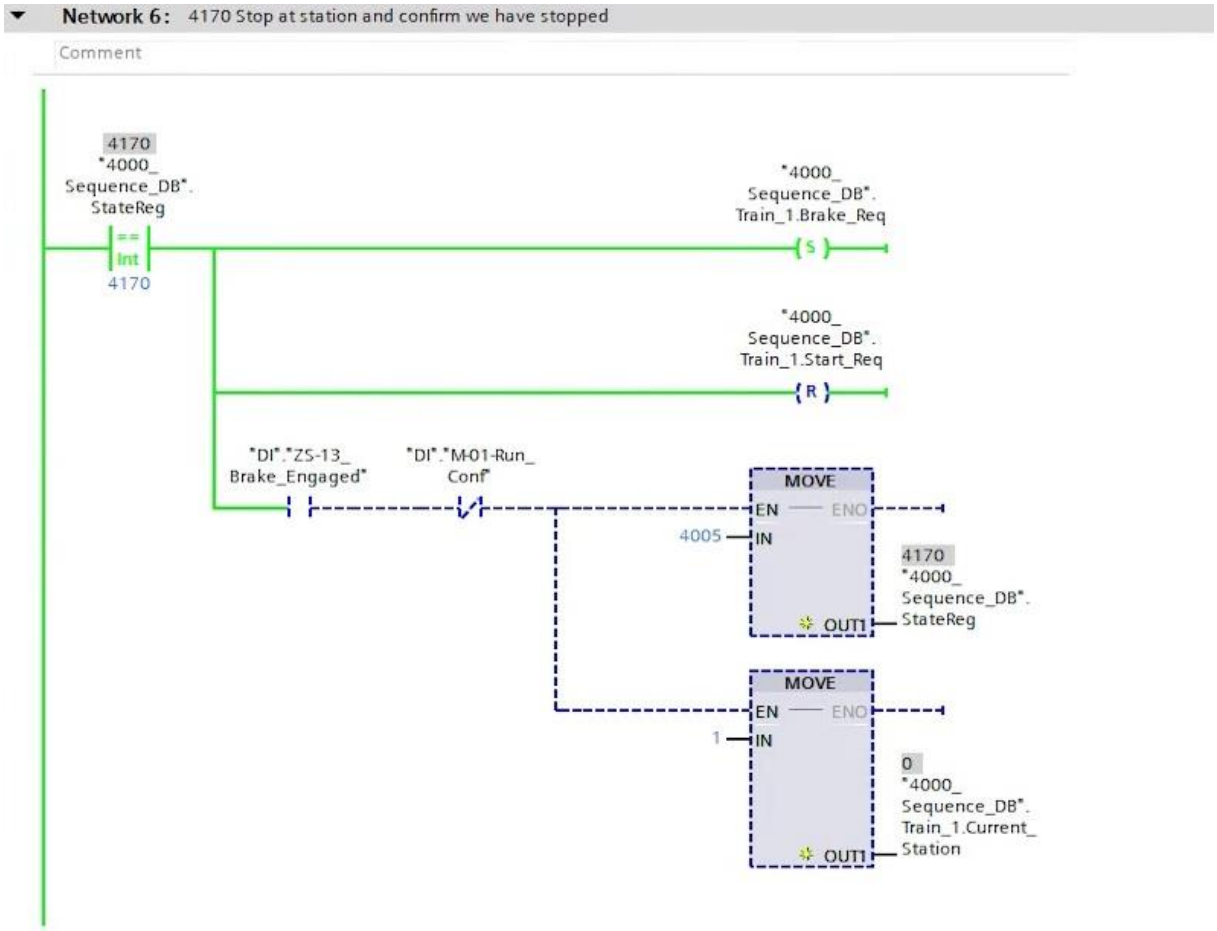
c) U stanju 4160, ručno postavljamo zahtjev za pokretanjem motora i smjer kretanja unatrag. Kada se potvrdi da je motor pokrenut, stanje se mijenja na 4161, te se čeka na signal za usporavanje.



d) U stanju 4161, sustav čeka aktivaciju senzora za usporavanje. Kada se senzor aktivira, brzina kretanja vlaka se smanjuje, a sustav prelazi u stanje 4165.



e) U stanju 4165, sustav prati signal senzora zaustavljanja na prvoj stanici. Kada se senzor aktivira, stanje se mijenja na 4170.



f) U stanju 4170, sustav primjenjuje kočnice i zaustavlja motor. Kada se potvrdi da su se kočnice aktivirale i vlak je zaustavljen, sustav prelazi u stanje 4005, čekajući daljnje naredbe od strane korisnika.

Slika 31. a); b); c); d); e); f); Tijek testiranja sustava

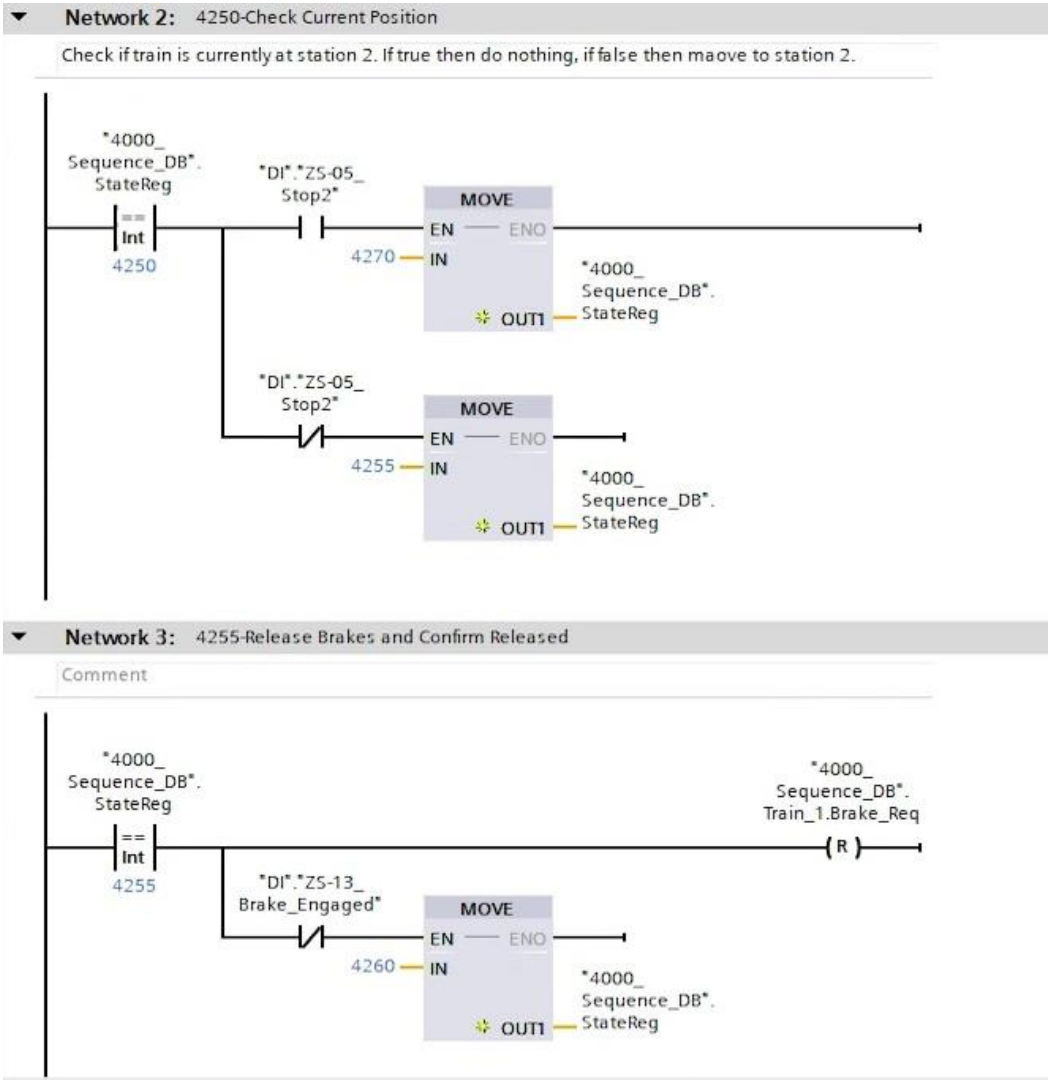
5.4 Program kretanja prema stanicama 2, 3 i 4

U ovome potpoglavlju nastavljamo logiku glavne sekvence, razradom koda kretanja prema ostalim stanicama. Kod za kretanje vlaka između stanica 2, 3 i 4 bit će u odnosu na kod koji izvršava naredbu kretanja prema stanici 1, relativno sličan, uz nekoliko manjih izmjena. Brojevi stanja čine razliku u kodu u ovisnosti o stanicama prema kojima se vlak kreće. Objašnjenje brojeva stanja nalazi se u tablici 5.

Funkcija stanja	**Broj stanja za stanicu 1**	**Broj stanja za stanicu 2**
Provjera trenutačne pozicije	4150	4250
Otpuštanje kočnice i potvrda otpuštanja	4155	4255
Pokretanje motora i potvrda pokretanja	4160	4260
Čekanje na signal usporavanja	4161	4261
Odlazak do zaustavne pozicije za stanicu	4165	4265
Zaustavljanje na stanici i potvrda zaustavljanja	4170	4270

Tablica 4. Opis stanja u kodu glavne sekvence

Kod se izvršava tako da se na samom početku provjerava nalazi li se vlak na drugoj stanici, te ako se već ne nalazi ondje, izvršit će se naredba dolaska vlaka na ovu stanicu. U ovome slučaju relevantan je senzor zaustavljanja na drugoj stanici. Slično kao i kod programa kretanja vlaka prema prvoj stanici, ako početna provjera pokazuje da se vlak već nalazi na drugoj stanici, stanje sustava se mijenja u 4270 čime se preskače petlja dolaska vlaka na stanicu. Ako se vlak ne nalazi na drugoj stanici i tek treba stići na svoje odredište, stanje sustava se mijenja u 4255 te se otpuštanjem kočnice vlak priprema za pokretanje, kao što je prikazano na *slici 32*.



Slika 32. Početni dio koda kretanja vlaka na drugu stanicu

Kada je potvrđeno da je kočnica otpuštena, aktivira se sljedeće po redu stanje; 4260. Ovdje dolazi do razlike u kodu u odnosu na kod kretanja vlaka prema prvoj stanici.

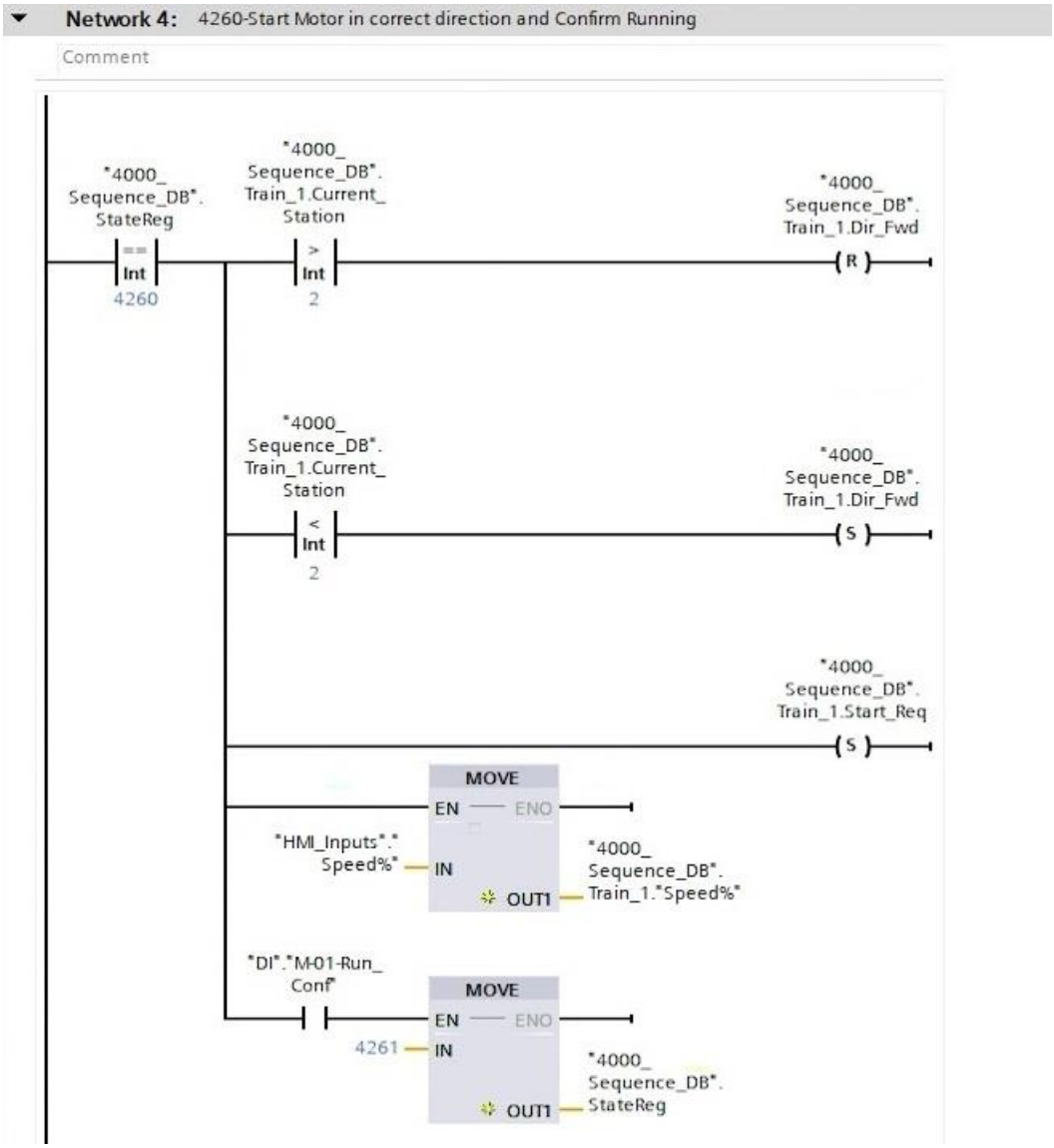
Za premještanje vlaka na stanicu broj 2 potreban je fleksibilniji kod od onoga koji jednostavno usmjerava vlak unatrag prema prvoj stanici. Razlog za to leži u različitim smjerovima kretanja koji su potrebni u ovisnosti o početnoj poziciji vlaka. Ako se vlak nalazi na prvoj stanici, treba se kretati unaprijed da bi stigao do stanice dva. S druge strane, ako je vlak na stanici tri ili četiri, mora se kretati unatrag kako bi stigao na drugu stanicu. Dakle, kod koji automatski šalje vlak unatrag nije odgovarajući, jer ne uzima u obzir trenutačnu poziciju vlaka i specifične zahtjeve smjera kretanja.

Novi kod stoga, uvodi logiku koja prvo provjerava trenutačnu poziciju vlaka i na temelju te pozicije odlučuje o potrebnom smjeru. Kod započinje tako što se očitava trenutačna stanicu vlaka putem varijable koja sadrži broj stanice na kojoj se vlak nalazi. Potom se koristi uvjetna logika koja uspoređuje vrijednost te varijable s brojem druge stanice. Ako je trenutačna stanica veća od

dva, što znači da se vlak nalazi na trećoj ili četvrtoj stanici, vlak će se postaviti tako da ide u smjeru unatrag, jer je to smjer potreban da vlak stigne do druge stanice iz te pozicije. Ovo se postiže resetiranjem bita za smjer naprijed, što sustavu signalizira potrebu za kretanjem unatrag.

Međutim, ako je informacija o trenutačnoj stanici manja od dva, to znači da se vlak nalazi na stanici jedan, te treba krenuti u smjeru unaprijed kako bi stigao na drugu stanicu. U ovom slučaju, kod postavlja bit za smjer unaprijed. Tako se osigurava da vlak uvijek ide u pravom smjeru, neovisno o početnoj poziciji, što ovaj program prilagođava za različite slučajeve, kao što je prikazano na *slici 33*.

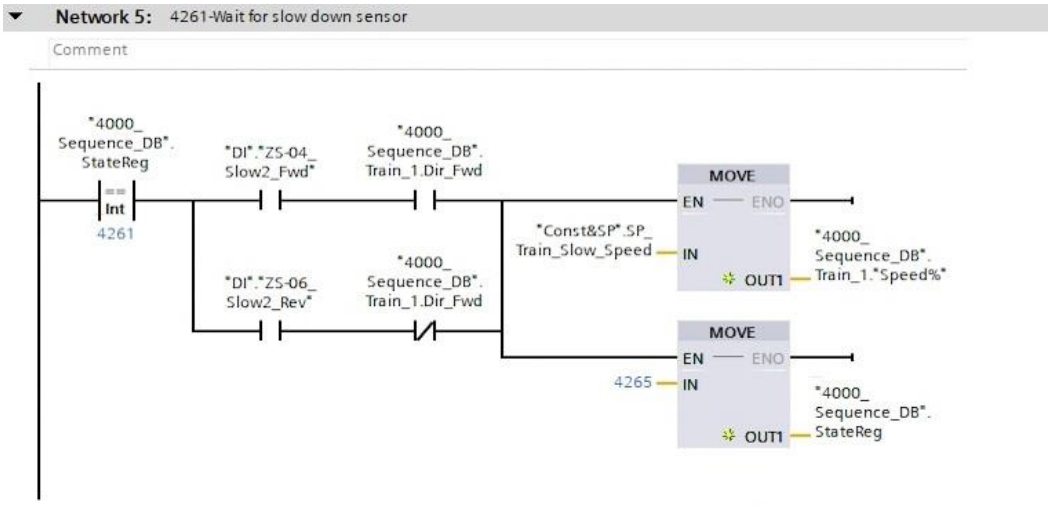
Kad je smjer kretanja postavljen, slijedi pokretanje motora i potvrda o pokretanju. Nakon potvrde, kod prelazi na sljedeći korak. Ova logika pruža automatizaciju koja uzima u obzir trenutačnu poziciju vlaka i odabire optimalan smjer kretanja prema drugoj stanici.



Slika 33. Pokretanje motora i potvrda pokretanja

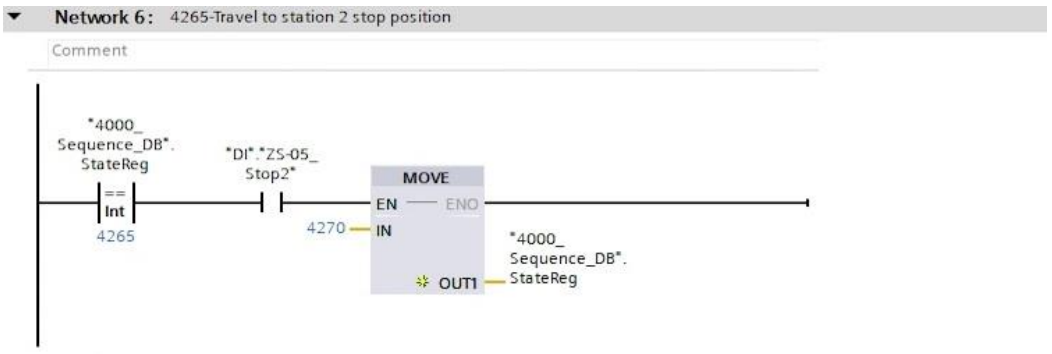
Nakon što motor krene, sljedeći korak je čekanje na aktivaciju senzora za usporavanje, ovisno o smjeru kretanja vlaka kao što je prikazano na *slici 34.* Budući da se vlak može kretati unaprijed ili unatrag, kod koji razvijamo mora prepoznati odgovarajući senzor usporavanja za smjer unaprijed i unatrag.

Upotrebljava se *ILI* logika između senzora, što znači da će sustav prihvatiti signal s bilo kojeg senzora, ovisno o smjeru. Zatim se provjerava smjer kretanja vlaka s pomoću pripadajuće varijable. Kada odgovarajući senzor signalizira usporavanje, smanjuje se brzina vlaka i prelazi se na sljedeći korak sekvence, koji uključuje kretanje vlaka do točne pozicije zaustavljanja na stanici pružajući prilagodbu brzine i precizno zaustavljanje bez obzira na smjer iz kojeg se vlak približava stanici.



Slika 34. Čekanje na aktivaciju senzora za usporavanje

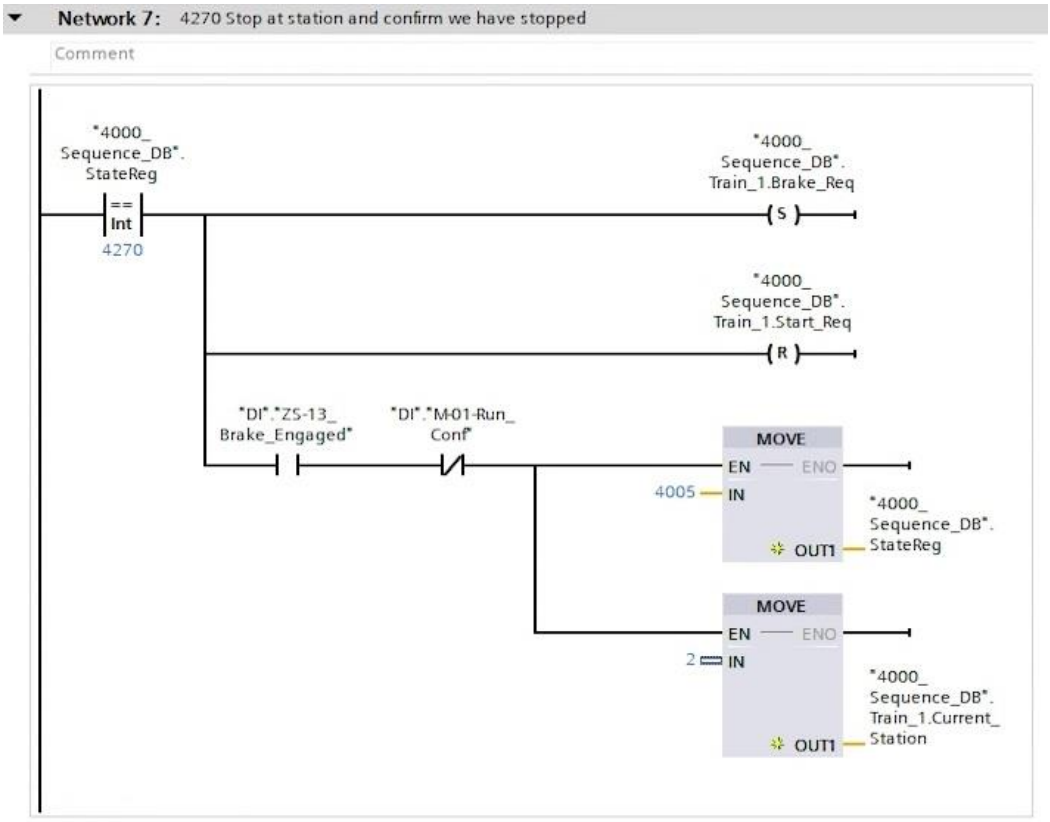
Sljedeći korak u sekvenci jest da vlak stigne do pozicije zaustavljanja na stanici (*slika 35.*). Ako je sustav u stanju 4265, kod je prilagođen tako da prepoznaje aktivaciju senzora na poziciji zaustavljanja na stanici dva. Aktivacijom tog senzora, sekvenca prelazi u završno stanje.



Slika 35. Dolazak vlaka do pozicije zaustavljanja na stanici

Nakon prelaska u završno stanje 4270, pokreće se procedura zaustavljanja vlaka na stanici i potvrđuje se zaustavljanje. Prvo se šalje zahtjev za aktivaciju kočnica kako bi se zaustavio vlak, te se resetira zahtjev za pokretanjem vlaka, čime se signalizira zaustavljanje. Provjerava se jesu li kočnice aktivirane i je li vlak zaista zaustavljen.

Na kraju se postavlja oznaka trenutačne stanice na stanicu broj 2, budući da je vlak upravo stigao. Slijedi vraćanje sustava u stanje 4005, gdje se čeka unos daljnjih naredbi korisnika (*slika 36.*). Ovim koracima razvijen je program za dolazak vlaka na drugu stanicu.



Slika 36. Zaustavljanje na stanici 2 i potvrda zaustavljanja

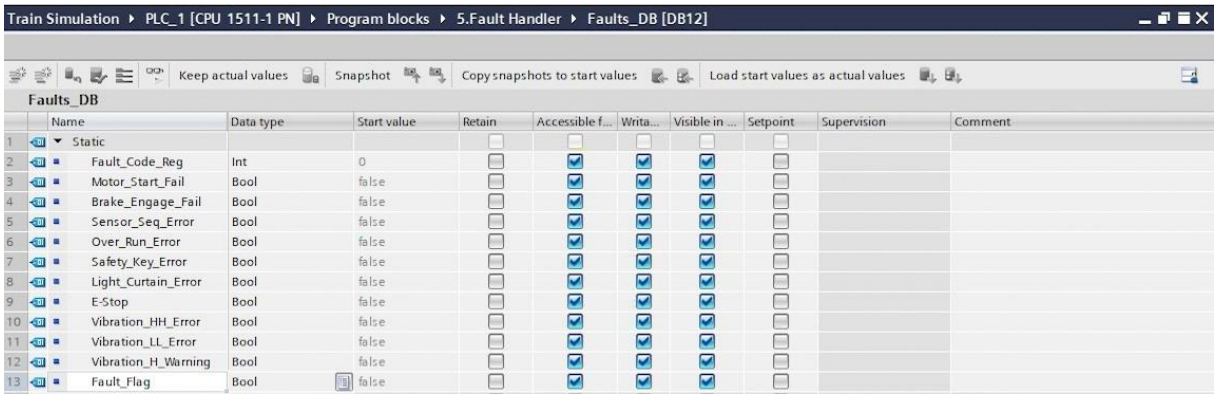
Kod za kretanje vlaka prema trećoj stanici se temelji na kodu za drugu stanicu, dok se kod kretanja prema četvrtoj stanici temelji na kodu kretanja prema prvoj stanici. Razlog tome je u konfiguraciji senzora na stanicama. Druga i treća stanica imaju senzore za usporavanje u oba smjera (unaprijed i unatrag) te zasebne senzore za zaustavljanje. S druge strane, krajnje stanice; 1 i 4, imaju jednostavniju konfiguraciju, s jednim senzorom za usporavanje i jednim za zaustavljanje, budući da ove stanice ne zahtijevaju detekciju smjera prilikom pristupa vlaka, jer im vlak uvijek prilazi iz istog smjera.

6. SUSTAV ZA DETEKCIJU I OBRADU GREŠAKA

Kako bi sustav radio ispravno, važno je obuhvatiti sve potencijalne probleme koji mogu nastati u sustavu u vidu kodova grešaka. U funkcionalnoj specifikaciji dizajna na početku ovoga rada, objašnjeni su kodovi grešaka koji uključuju: neuspješno pokretanje motora, neuspjelu aktivaciju kočnice, nedozvoljene vibracije tijekom rada stroja, prekid sigurnosne svjetlosne zavjese, pritisak gumba za hitno zaustavljanje i pogreške senzora. U ovome poglavlju razvit ćemo kodove grešaka prema funkcionalnim specifikacijama.

6.1 Detekcija grešaka

U ovome potpoglavlju kodove grešaka ćemo realizirati u novoj funkciji koja ima svoj podatkovni blok za pohranu potrebnih varijabli (*slika 37.*). Prvo uvodimo registar grešaka (tipa *Integer*), koji pohranjuje kod specifičan za svaku grešku (npr. kod 10 za *E-stop*, kod 20 za neuspješno pokretanje motora). Zatim se definiraju ostale varijable za pojedinačne greške (npr. neuspješno pokretanje motora, kvar na kočnici, greška senzora). Ove varijable se postavljaju u istinito stanje kada dođe do greške, osiguravajući sustavu praćenje grešaka. Dodatno, općenita oznaka ili zastavica greške (*fault flag*, tipa *Bool)* označava prisutnost bilo koje greške u sustavu, što aktivira odgovarajuću obradu grešaka.



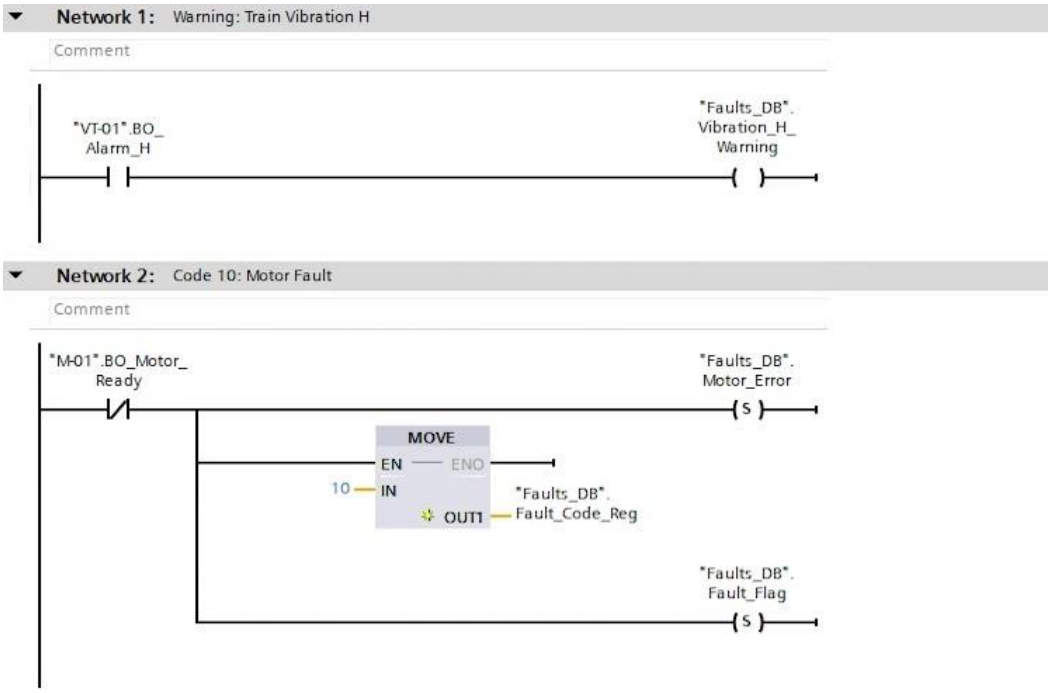
Slika 37. Podatkovni blok varijabli kodova grešaka

Za postavljanje i prikazivanje grešaka, koristi se niz kodova za greške i upozorenja, koji su podijeljeni u intervalima po 10, kako bi se omogućilo lakše dodavanje novih kodova. Primjerice, između koda 10 i 20 može se, prema potrebi dodati novi kod s brojem 15. Kod 10 označava grešku motora, dok kod 20 označava grešku kočnice koja se nije aktivirala. Također, kod 30 predstavlja grešku u redoslijedu aktivacije senzora, dok kod 40 signalizira prekoračenje radnih parametara (engl. overrun). Kod 50 je rezerviran za grešku uklanjanja sigurnosnog ključa, a kod 60 za prekid

sigurnosne svjetlosne zavjese. Izrazito visoka vibracija vlaka ima kod 70 (ranije označena s *HH - high high*), dok niska vibracija ima kod 71 (*LL*). Ovi kodovi pomažu pri diferenciranju razina vibracija, jer su visoka i niska vibracija povezane, pa im se dodjeljuju srodni brojevi. Posljednji kod; 100, rezerviran je kao zaustavni kod (prilikom hitnog zaustavljanja) te signalizira kompletno isključenje sustava. Svi ovi kodovi u nazivu grešaka služe kako bi se mogli prikazati na HMI-u (korisničkom sučelju), a svrha tog prikaza je da korisnici ili tehničari mogu odmah prepoznati o kojem problemu u sustavu je riječ.

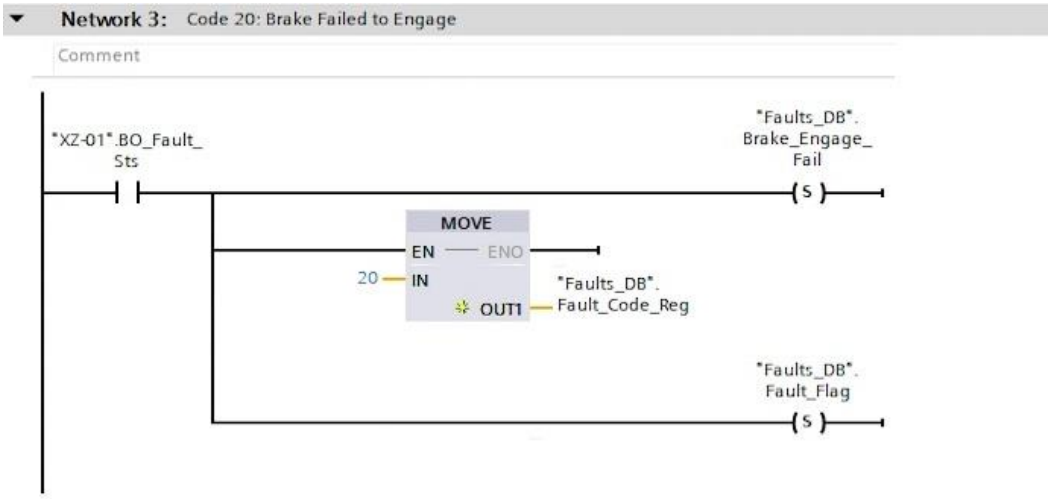
Podaci koji će aktivirati kodove grešaka dobivaju se iz funkcijskih blokova upravljanja kočnicom, motorom i sustava mjerenja vibracija vlaka, koje smo definirali u poglavlju 4. Tako sustav za mjerenje vibracija vlaka kontinuirano mjeri trenutačnu razinu vibracija i šalje te podatke u blok za obradu podataka u programu. Ovaj blok uspoređuje stvarnu razinu vibracija s unaprijed definiranim pragovima. Kada vibracije dosegnu ili premaše te pragove, blok šalje binarni signal koji aktivira odgovarajući kod greške koji je prikazan u prvoj mreži na *slici 38.* Dakle, binarni izlaz (*VT 01 alarm H*) iz funkcijskog bloka koji predstavlja sustav praćenja vibracija vlaka (*slika 20.*), zgodno je iskoristiti u ovom slučaju za aktiviranje koda greške.

Mreža broj 2, prikazana na *slici 38.* upravlja greškom motora provjeravajući status spremnosti motora za pokretanje, putem normalno zatvorenog kontakta. Slično kao i kod upozorenja za visoku vibraciju, podatak o spremnosti motora dolazi s izlaza funkcijskog bloka upravljanja motorom, koji je prikazan na *slici 18.* Ako motor nije spreman za pokretanje, uvjet postaje istinit i pokreće sljedeće radnje. Prvo, aktivira se bit *Motor Error*, čime se bilježi greška motora. Zatim naredba *MOVE* unosi vrijednost 10 u registar grešaka, što osigurava da se na sučelju prikaže specifičan kod greške motora. Također, postavlja se i generička zastavica odnosno općenita oznaka greške, koja označava da je došlo do bilo kakve greške u sustavu.



Slika 38. Upozorenje za visoku vibraciju i kod greške motora

Nastavljajući ovu logiku, kod za grešku neuspjele aktivacije kočnice, koristi status greške iz funkcijskog bloka kočnice (*slika 19.*). Ako dođe do greške kočnice, aktivira se kontakt (*„Fault status“)* koji pokreće postavljanje bita za označavanje problema s kočnicom. Zatim se dodjeljuje kod 20 u registar grešaka, čime se prikazuje specifičan kod za ovu grešku. Na kraju se aktivira općenita oznaka greške, koja označava prisutnost greške u sustavu, kako je i prikazano na *slici 39.*

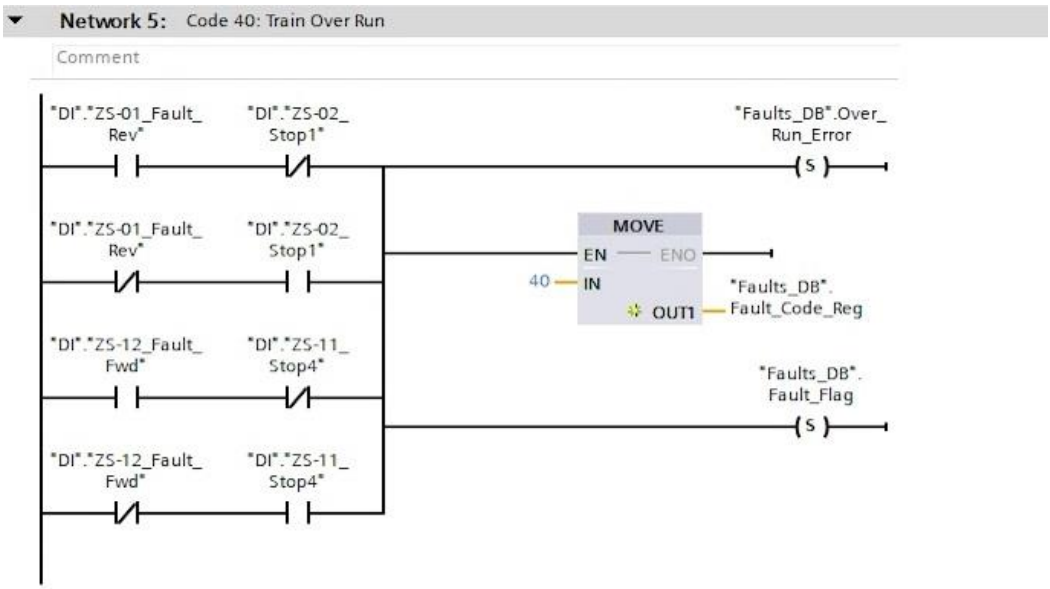


Slika 39. Kod greške kočnice

Greška prelaska preko krajnje pozicije (engl. fault overrun) aktivira se kako bi se spriječilo da vlak pređe krajnju granicu tračnica i potencijalno izazove nesreću. Zato je na prvoj i posljednjoj stanici koje su smještene na početku i na kraju tračnica instaliran još jedan senzor blizine uz postojeće

senzore usporavanja i zaustavljanja koji se nalaze na svakoj stanici. Cilj je provjeriti da oba senzora; i senzor zaustavljanja i dodatni senzor, detektiraju vlak istovremeno u oba smjera. Primjerice, u smjeru unatrag, ako jedan senzor detektira vlak, a drugi ne, ili obrnuto; kod prepoznaje grešku i aktivira oznaku greške. Isti postupak se ponavlja za smjer unaprijed, pri čemu se provjerava da oba senzora istovremeno bilježe prisutnost vlaka, kako je prikazano na *slici 40.*

Ako dođe do neusklađenosti senzora, to ukazuje na potencijalni kvar jednog od senzora, što će označavati grešku. Zatim će se dodijeliti kod 40 u registar grešaka, dok se općenita oznaka greške također aktivira kako bi se označila prisutnost greške u sustavu.

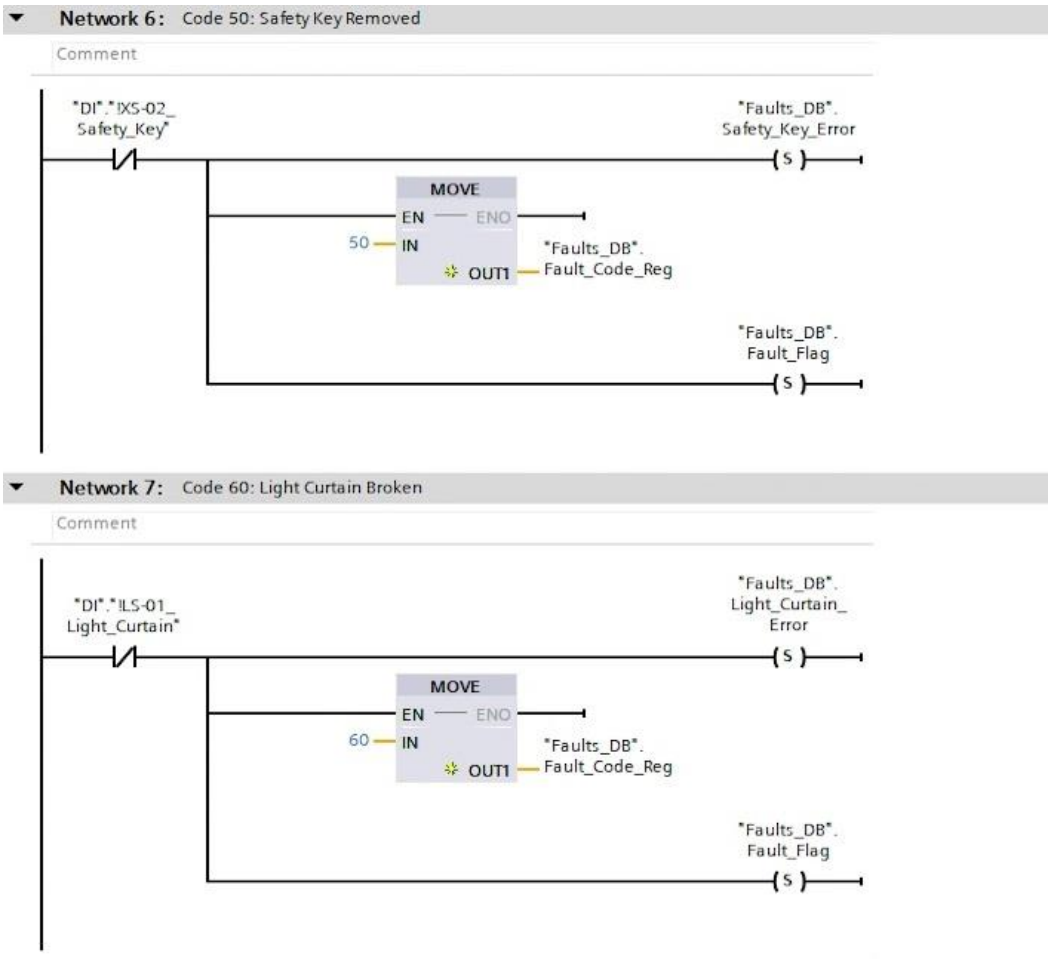


Slika 40. Kod za prepoznavanje prelaska vlaka preko krajnje pozicije

Kako bi se vlak mogao kretati, sigurnosni ključ mora biti smješten u svoje kućište. Greška uklanjanja sigurnosnog ključa aktivira se kada sigurnosni ključ nije na predviđenom mjestu. Status ključa se prati s pomoću invertne logike, što znači da se signal 1 prima kada je ključ uklonjen. Kada je ključ uklonjen, kontakt se zatvara, čime se aktivira bit za označavanje greške. Kod greške 50 zatim se unosi u registar grešaka, a aktivira se i općenita oznaka greške. Cilj je da uklanjanjem sigurnosnog ključa vlak ne može biti pokrenut, jer je netko možda uklonio sigurnosni ključ kako bi na tračnicama obavio održavanje.

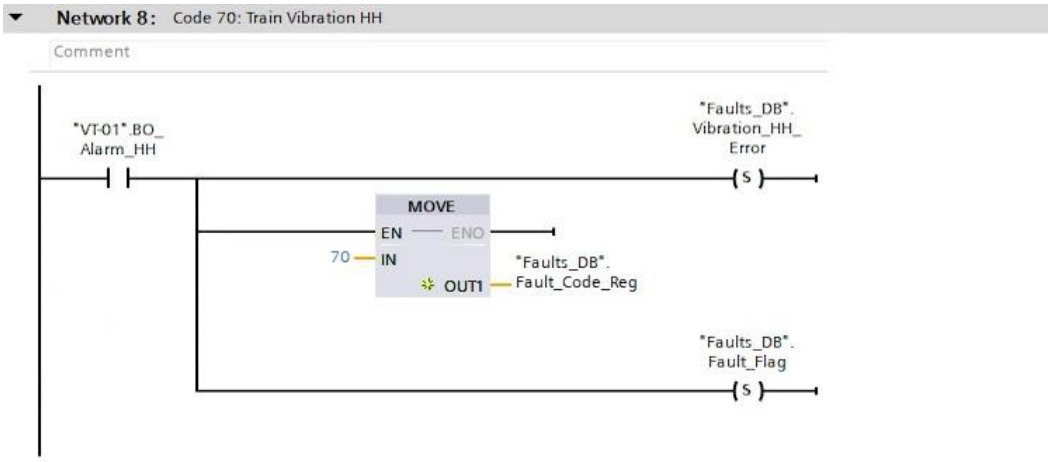
Greška svjetlosne zavjese aktivira se kada laserska sigurnosna svjetlosna zavjesa, prepozna prekid u svom sigurnosnom polju. Laserska sigurnosna svjetlosna zavjesa se koristi kako bi spriječila

pristup područjima gdje se vlak kreće po tračnicama. U normalnim uvjetima, kada nema prepreka, svjetlosna zavjesa odašalje signal 1. Ako netko pređe preko laserske zrake i time je prekine, signal prelazi u stanje 0, što se detektira u sustavu kao stanje greške. U kodu prikazanom na *slici 41.* postavljen je normalno zatvoreni kontakt, što znači da će greška biti aktivirana kada signal pređe u stanje 0. Prekidom svjetlosne zavjese bilježi se greška, a općenita oznaka greške signalizira prisutnost greške u sustavu.



Slika 41. Kod za prepoznavanje greške uklanjanja sigurnosnog ključa i greške prekida sigurnosne laserske svjetlosne zavjese

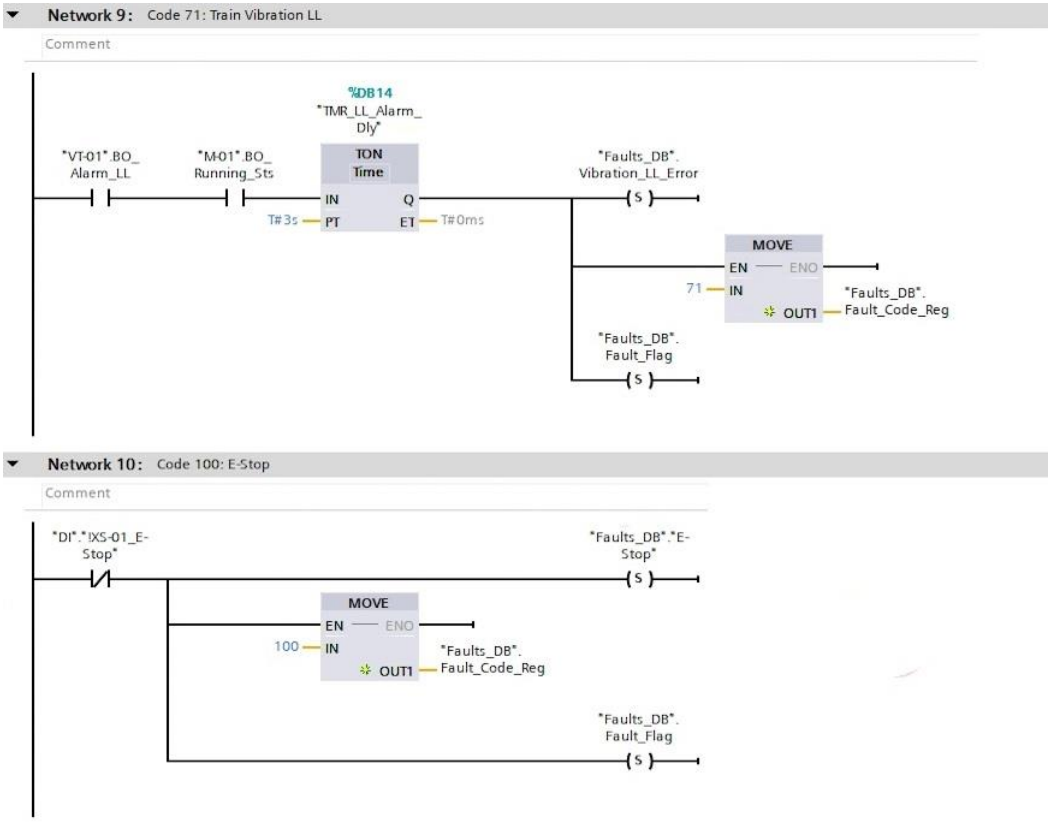
Greška vrlo visoke vibracije vlaka se aktivira kada funkcijski blok koji prati vibracije vlaka detektira vibracije koje prelaze definirani prag. Kod greške 70 unosi se u registar grešaka, a aktivira se i općenita oznaka greške (*slika 42.*).



Slika 42. Kod greške izrazito visoke vibracije

Greška izrazito niskih vibracija vlaka (kod 71) aktivira se kada funkcijski blok zadužen za praćenje vibracija vlaka, detektira vrlo niske vibracije tijekom kretanja vlaka. Za razliku od greške visokih vibracija, ovdje je potrebna dodatna provjera i vremenska odgoda. Predviđen kontakt unutar sustava, stalno prati nisku razinu vibracija, međutim aktivacija greške se događa samo ako je vlak u pokretu, što se provjerava statusom kretanja motora. Osim toga, dodaje se vremenska odgoda od 3 sekunde, kako bi se omogućilo da se vibracije stabiliziraju pri pokretanju vlaka. Ako nakon 3 sekunde nakon pokretanja, vlak još uvijek ima vrlo niske vibracije, kod greške 71 upisuje se u registar grešaka, a opća oznaka greške se aktivira, kao što je prikazano na *slici 43.*

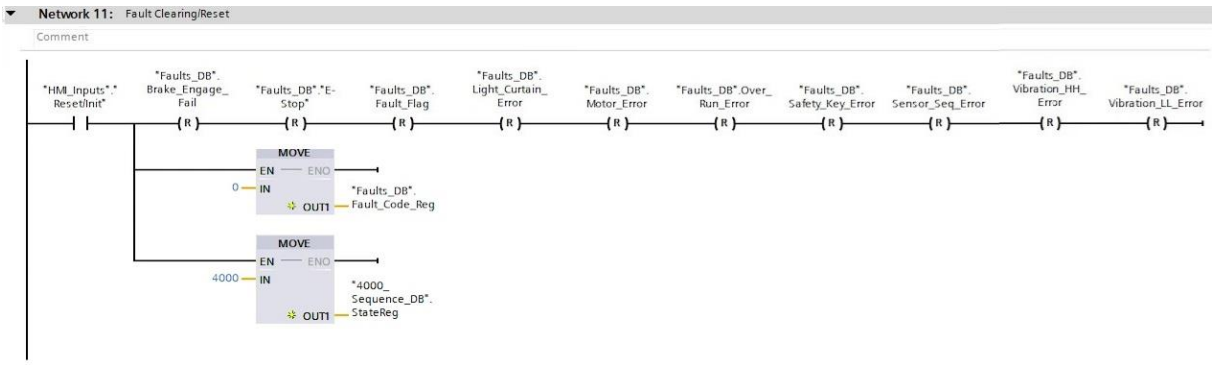
Oznaka hitnog zaustavljanja (engl. Emergency Stop) aktivira se kada je pritisnut gumb za hitno zaustavljanje. U kodu se primjenjuje normalno zatvoreni kontakt kako bi se detektirala promjena stanja na 0 prilikom pritiska gumba za hitno zaustavljanje. U tom slučaju postavlja se oznaka greške, upisuje se kod greške: 100 u registar grešaka i aktivira se općenita oznaka greške, što signalizira da je stroj zaustavljen zbog hitne situacije.



Slika 43. Kodovi grešaka izrazito niske vibracije i hitnog zaustavljanja

Ovime su obuhvaćeni svi kodovi za detekciju grešaka i postavljene su općenite oznake koje ukazuju na prisutnost grešaka kada se pojave problemi u radu stroja. Međutim, operater trenutačno nema mogućnost poništiti te greške, što znači da vlak ostaje trajno zaustavljen nakon pojave grešaka. Kako bi sustav bio funkcionalan, potrebno je dodati kod za resetiranje grešaka, kako bi vlak mogao nastaviti s vožnjom nakon otklanjanja problema.

Za to nam služi kod na *slici 44.* kojim se resetiraju sve oznake grešaka u sustavu. Korisnik može putem gumba na sučelju odabrati *Reset* opciju, čime se sve oznake specifičnih i generičkih grešaka postavljaju na nulu, odnosno poništavaju. Nakon toga se i registar grešaka postavlja na vrijednost nula, što označava da u sustavu više nema aktivnih grešaka. Tako se sustav vraća u početno stanje, spreman za normalan rad nakon otklanjanja poteškoća.

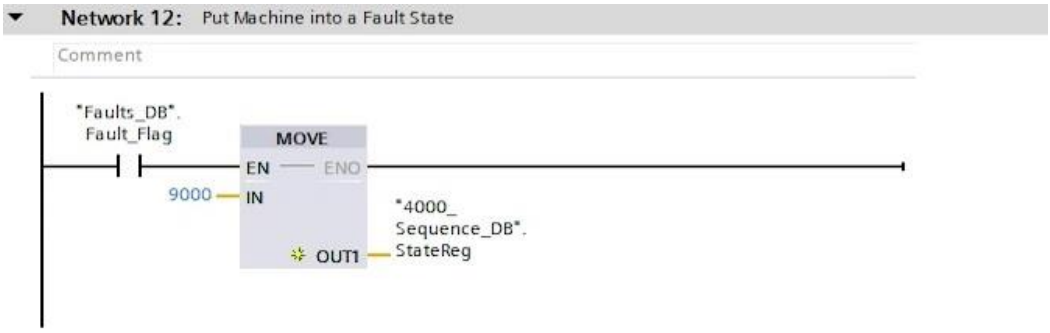


Slika 44. Poništavanje grešaka

Pojavom bilo kakve greške u sustavu, stroj se mora postaviti u stanje greške. Kada se aktivira općenita oznaka greške, upisuje se vrijednost 9000 u registar stanja stroja. Ovaj kod predstavlja stanje greške, u kojem se stroj zaustavlja i ne može nastaviti s radom dok se greška ne otkloni.

Kad se po otklanjanju greške, poništi oznaka greške s pomoću gumba za resetiranje, potrebno je prebaciti stroj iz stanja 9000 u početno stanje, kako bi sustav bio spreman za rad. Zato se prilikom resetiranja stroj vraća u stanje 4000, koje predstavlja čekanje na korisnički unos i omogućava ponovno pokretanje vlaka.

Ovim postupkom, prikazanom na *slikama 45. i 46.* osigurava se da se stroj automatski prebaci u sigurno stanje kad god se otkrije problem u radu sustava, dok se poništavanjem greške vraća u početno stanje spremno za daljnje korištenje.

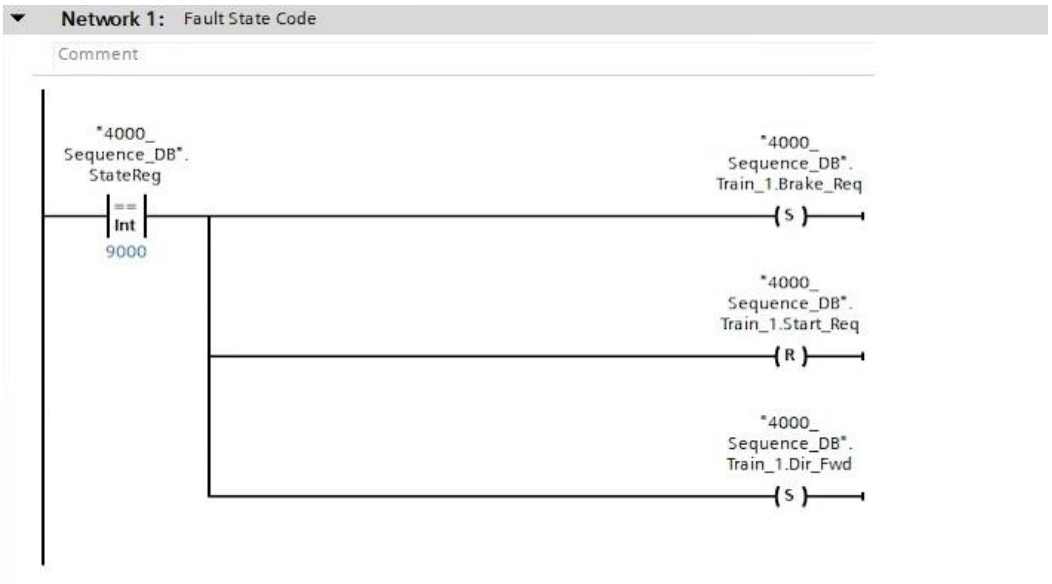


Slika 45. Postavljanje stroja u stanje greške

Stanje 9000 definirano je kao stanje greške u kojem sustav reagira na uočene greške i zaustavlja vlak. Kada registar stanja pokazuje vrijednost 9000, sustav automatski šalje naredbu za kočenje

vlaka, čime se osigurava trenutačno zaustavljanje. Dodatno, smjer vlaka postavlja se prema naprijed radi dosljednosti s ostatkom programa, iako to nije nužno za samu sigurnost.

Dakle, svaki put kad sustav prepozna grešku i postavi se stanje 9000, vlak će automatski aktivirati kočnicu, motor će se zaustaviti, a smjer će biti postavljen unaprijed, što možemo vidjeti na *slici 46.* Nakon što se greška poništi, sustav se vraća u stanje 4000, što predstavlja početno stanje čekanja na korisnički unos, a sve oznake grešaka se brišu kako bi sustav bio spreman za ponovno pokretanje.



Slika 46. Stanje greške

6.2 Detekcija pogreške slijeda senzora – SCL programski jezik

U ovome potpoglavlju objašnjava se način implementacije koda za detekciju grešaka u slijedu senzora, koristeći SCL jezik u PLC programiranju. Analizirat ćemo i usporediti signale senzora kako bi se identificirala bilo kakva nedosljednost u njihovoj aktivaciji. Glavni cilj ovoga koda je detektirati grešku senzorske sekvence i označiti ju kao grešku kada senzori ne slijede očekivani redoslijed aktivacije.

SCL, odnosno *Structured Control Language*, često se koristi za programske zadatke koji su prekompleksni za pisanje u klasičnoj ljestvičastoj logici. SCL omogućuje lakše upravljanje kompleksnim operacijama i logičkim sekvencama jer pruža strukturu sličnu tekstualnim programskim jezicima. U ovom slučaju, SCL se koristi jer omogućuje izradu naprednih funkcija,

poput konverzije bitova u riječ i detekciju sekvencijalnih grešaka senzora, što bi u ljestvičastoj logici bilo mnogo teže za realizirati.

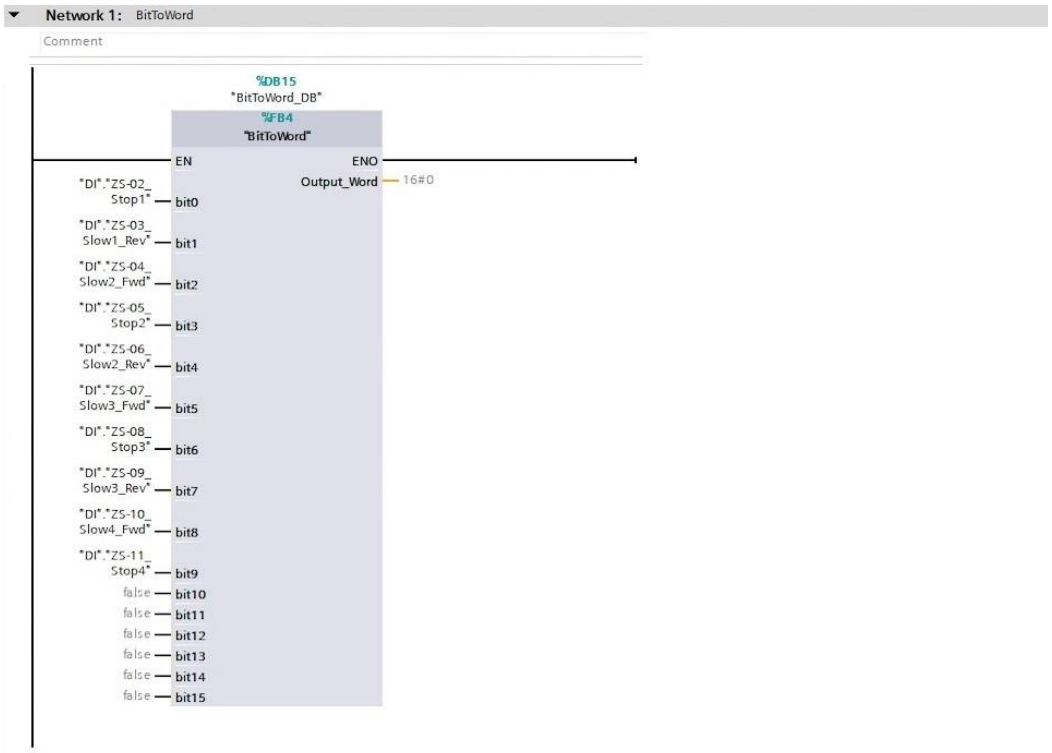
```
1
2
3
4  #Output_Word.%X0 := #bit0;
5  #Output_Word.%X1 := #bit1;
6  #Output_Word.%X2 := #bit2;
7  #Output_Word.%X3 := #bit3;
8  #Output_Word.%X4 := #bit4;
9  #Output_Word.%X5 := #bit5;
10 #Output_Word.%X6 := #bit6;
11 #Output_Word.%X7 := #bit7;
12 #Output_Word.%X8 := #bit8;
13 #Output_Word.%X9 := #bit9;
14 #Output_Word.%X10 := #bit10;
15 #Output_Word.%X11 := #bit11;
16 #Output_Word.%X12 := #bit12;
17 #Output_Word.%X13 := #bit13;
18 #Output_Word.%X14 := #bit14;
19 #Output_Word.%X15 := #bit15;
20
21
22
23
```

Slika 47. SCL kod za konverziju senzorskih ulaza (bit) u jednu varijablu (word)

Prvi je korak u realizaciji detekcije pogreške slijeda senzora, prikazan je na *slici 47.*, a to je pretvaranje pojedinačnih senzorskih ulaza u jednu varijablu tipa „word“ koja sadrži svih 16 bitova, pri čemu svaki bit predstavlja stanje jednog senzora. Korištenjem ovog pristupa možemo upravljati i pratiti stanje svih senzora unutar jedne varijable, kako bi značajno olakšali kasniju obradu podataka.

Svaka linija koda povezuje specifični bit u varijabli *Output_Word* s određenim senzorskim ulazom. Na ovaj način, stanje svakog senzora pretvara se u odgovarajuću binarnu vrijednost unutar varijable *Output_Word.* Primjerice, naredba: #naziv_varijable.%X0 := #bit0; označava dodjeljivanje vrijednosti senzora označenog s *bit0* na prvi bit u varijabli. Na isti način, #Output_Word.%X1 := #bit1; dodjeljuje vrijednost drugog senzora, *bit1,* drugom bitu u Output_Word, i tako sve do *bit15*. Kada je senzor aktivan, vrijednost njegovog bita postavlja se na 1, dok se u slučaju neaktivnosti postavlja na 0. Na primjer, ako su senzori označeni kao bit0, bit3, i bit4 aktivni, odgovarajući bitovi unutar Output_Word varijable će biti postavljeni na 1, dok će ostali bitovi biti 0, čime dobivamo binarnu vrijednost poput 0000 0000 0001 1001. Osnovna prednost ovakvog načina objedinjavanja podataka očituje se prilikom rada s funkcijama poput *Encode,* koja je dizajnirana za rad s 16-bitnim riječima.

Funkcijski blok unutar kojeg je razvijen SCL kod, potrebno je pozvati u program detekcije pogreške slijeda senzora. Možemo vidjeti na *slici 48.* kako pojedine senzore (kao što su "Stop1", "Slow1_Rev" itd.) povezujemo s različitim bitovima u riječi. Na taj način, kada neki senzor postane aktivan, odgovarajući bit u riječi mijenja svoju vrijednost.

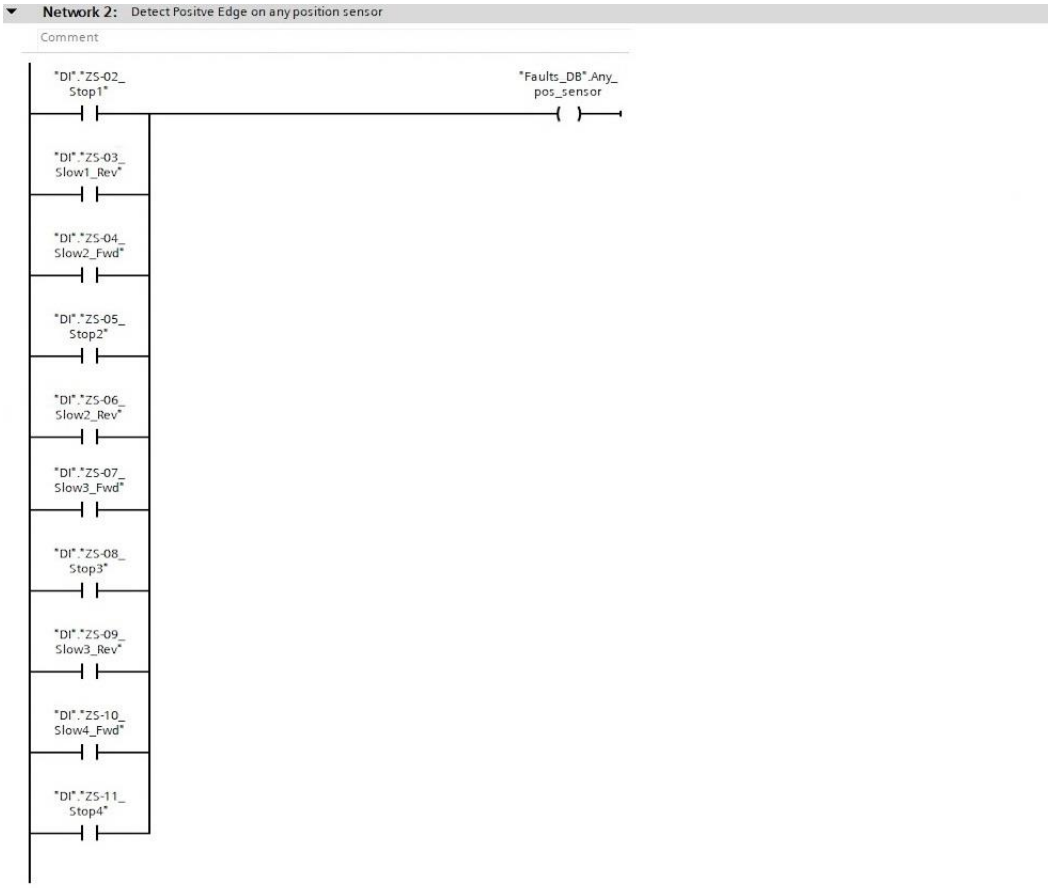


Slika 48. Pozivanje funkcijskog bloka SCL koda u program detekcije pogreške slijeda senzora

Također je važno na koji način program detektira aktivaciju senzora. Sljedeći segment koda prikazan na *slici 49.* koristi paralelni spoj svih senzora. Taj paralelni spoj osigurava da, kada bilo koji senzor postane aktivan, izlaz također postaje aktivan. Drugim riječima, dovoljno je da jedan senzor iz ovog paralelnog spoja promijeni stanje s 0 na 1 kako bi program detektirao aktivaciju. Ovaj signal aktivacije koristi se kao okidač za daljnje procese u programu.

Nakon ovog koraka, primjenom funkcije *Encode* detektira se najmanje značajan bit koji je aktivan unutar riječi. Ova funkcija čita „word“ varijablu i prepoznaje koji je prvi aktivni bit s desna – to jest, koji senzor ima prvu aktivaciju u nizu. *Encode* funkcija tada vraća poziciju tog aktivnog bita kao broj koji predstavlja redni broj senzora, kao što je prikazano u mreži 3, na *slici 50.* Tako svaki

put kada neki senzor postane aktivan, možemo lako evidentirati njegovu poziciju unutar sekvence senzora.



Slika 49. Detekcija aktivacije senzora

Za registraciju trenutka kada senzor prelazi iz neaktivnog (0) u aktivno stanje (1), koristi se funkcija *PTrig* (poznata kao „Scan operand for positive signal edge“). Kada se dogodi pozitivan brid, pokreće se funkcija koja bilježi trenutačni aktivni senzor. Na taj način možemo detektirati samo trenutke kada je senzor postaje aktiviran, ne i kada je stalno aktivan. Time se eliminiraju potencijalni „lažni“ signali i osigurava precizno bilježenje trenutka aktivacije senzora, što pruža stabilno i pouzdano praćenje kretanja vlaka.

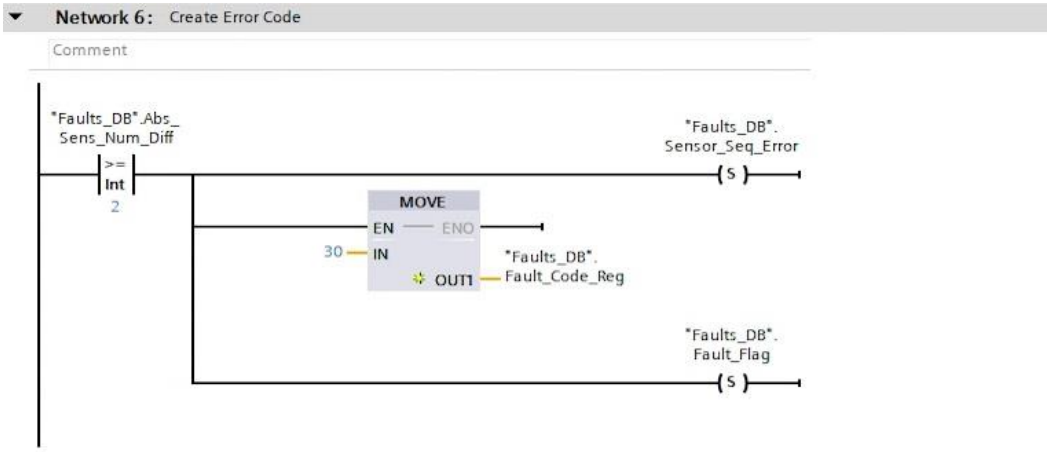
Sljedeći dio programa bilježi trenutačnu i prethodnu poziciju senzora. Kad se vlak kreće pored jednog senzora, a zatim pored drugog, trenutačna pozicija senzora bilježi se kao sadašnja, dok se prethodna pozicija koristi za usporedbu i provjeru redoslijeda aktivacije. Razlika između trenutačne i prethodne pozicije trebala bi uvijek biti jednaka 1, što označava da se vlak kreće kroz

senzorski niz ispravnim redoslijedom. Ako je razlika veća od 1, to ukazuje na grešku, jer objekt „preskače“ jedan ili više senzora, što program prepoznaje kao grešku u sekvenci.

Osim toga se primjenjuje i funkcija apsolutne vrijednosti. Budući da se vlak može kretati unaprijed i unatrag, razlika između trenutačne i prethodne pozicije senzora može biti negativna (pri kretanju unatrag). Kako bi se osigurala konzistentnost podataka, blok apsolutne vrijednosti pretvara sve negativne vrijednosti u pozitivne, osiguravajući da naš program radi ispravno bez obzira na smjer kretanja vlaka. Sve ovo prikazano je na *slici 50.*

Slika 50. Dio programa za detekciju pogreške slijeda senzora

Na kraju programa nalazi se logika za generiranje koda greške. Ako je apsolutna vrijednost razlike senzorskih pozicija veća od 1, program prepoznaje ovo kao grešku u sekvenci i generira kod greške (*slika 51.*). Kod greške postavlja indikator greške i aktivira alarm.



Slika 51. Generiranje koda greške slijeda senzora

Ovime je razvijen program za detekciju pogreške slijeda senzora, pa se ovaj funkcijski blok može pozvati unutar glavnog programa za detekciju grešaka, u kojem smo, u prethodnom odjeljku, definirali kodove za detekciju ostalih grešaka u sustavu.

7. SIMULACIJA KRETANJA VLAKA

U stvarnim uvjetima, završna faza programiranja podrazumijevala bi pregled koda i početak testiranja. Međutim, s obzirom na to da razvijamo cjeloviti program za simulaciju kretanja vlaka koji ne koristi nikakvu fizičku opremu, moramo implementirati dodatni kod za simulaciju stvarnih uvjeta. Ovaj dodatni kod oponašat će fizički vlak, uključujući njegovo kretanje i položaj na pruzi. Također će aktivirati i deaktivirati senzore pozicije, odnosno naše virtualne senzore, dok vlak prolazi duž zamišljene trase koju kreiramo za ovu simulaciju. Stoga za potrebe simuliranja stvarnih uvjeta kreiramo funkcijski blok pozicije, brzine i akceleracije, te funkcijski blok aktivacije senzora (odnosno pozicije vlaka) i pripadajući podatkovni blok. U nastavku ćemo detaljno objasniti kako implementirati ovaj dodatni kod u okviru programskog okruženja kako bismo postigli realističnu simulaciju kretanja vlaka.

7.1 Aktivacija senzora pozicije

Program razvijen u ovom potpoglavlju, primjenjuje se za simulaciju položaja vlaka na tračnicama te automatsku aktivaciju i deaktivaciju senzora ovisno o poziciji vlaka. Budući da se radi o simulaciji, sustav nema informaciju o kretanju stvarnog fizičkog vlaka i senzora koji bi detektirali njegov položaj, već nam za to služi kod opisan u nastavku. Glavna ideja ovoga koda je automatski aktivirati odgovarajuće senzore ovisno o trenutačnoj poziciji vlaka na tračnicama.

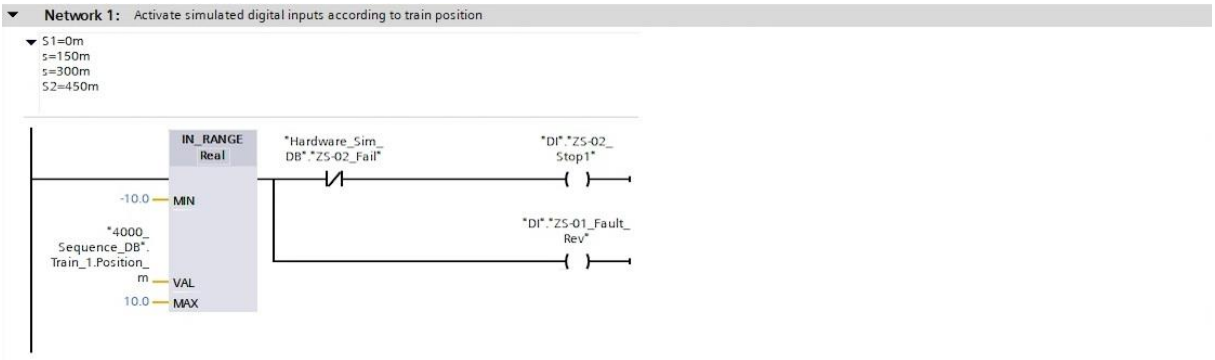
Teoretska pozicija senzora određena je unutar virtualne tračnice na specifičnim pozicijama, koje su udaljene 150 metara jedna od druge. Primjerice, prvi senzor nalazi se na početnoj stanici (0 metara), koja je referentna točka za računanje udaljenosti drugih stanica i senzora. Ostali senzori raspoređeni su tako da je senzor za usporavanje na prvu stanicu u smjeru unatrag smješten na 150 metara, senzor usporavanja na drugu stanicu u smjeru unaprijed je smješten na 300 metara, na 450 metara od početne stanice nalazi se stanica 2 i zaustavni senzor 2 i tako dalje. Dodatne senzore, koje smo u prethodnom poglavlju uveli u program na početnu i krajnju stanicu, koji služe kako ne bi došlo do greške prelaska vlaka preko krajnje pozicije na tračnicama, smješteni su na istoj poziciji kao početna i krajnja stanica, odnosno početni i krajnji zaustavni senzor.

U ovu svrhu koristimo funkciju *IN RAN*GE, koja provjerava nalazi li se trenutačna pozicija vlaka unutar određenog raspona vrijednosti i tako određuje treba li aktivirati ili deaktivirati određeni senzor. *IN RANGE* funkcija uzima tri argumenta: minimalnu vrijednost, maksimalnu vrijednost i stvarnu trenutačnu vrijednost koju provjeravamo. Ako je stvarna vrijednost, odnosno pozicija

vlaka unutar zadanog raspona, izlaz funkcije bit će aktivan (*true*). Ako vrijednost prelazi taj opseg, izlaz funkcije će biti neaktivan (*false*).

Funkcija *IN RANGE* omogućuje da svaki senzor bude aktivan unutar malog raspona od ±10 metara od precizne pozicije senzora. Ova tolerancija od ±10 metara pomaže u simulaciji prirodne preciznosti senzora jer stvarni senzori rijetko reagiraju na strogo definiranoj udaljenosti, već imaju mali opseg unutar kojeg detektiraju prisutnost objekta.

Unutar koda, aktivacija senzora definirana je za svaku poziciju posebno. Svaki od senzora u kodu ima odgovarajući *IN RANGE* blok koji prati poziciju vlaka unutar tog raspona i aktivira određeni senzor. Ovaj se kod zatim ponavlja za sve senzore duž tračnica, i skoro je identičan kodu na *slici 52.*, koji je namijenjen prvom senzoru. Malu iznimku u kodu prvog i posljednjeg senzora čini dodatna grana kojom uzimamo u obzir dodatni zaustavni senzor, postavljen na prvu i zadnju stanicu (ranije spomenut kao zaštita prelaska preko krajnje pozicije). Osim toga, kod je potpuno isti za sve senzore u simulaciji.

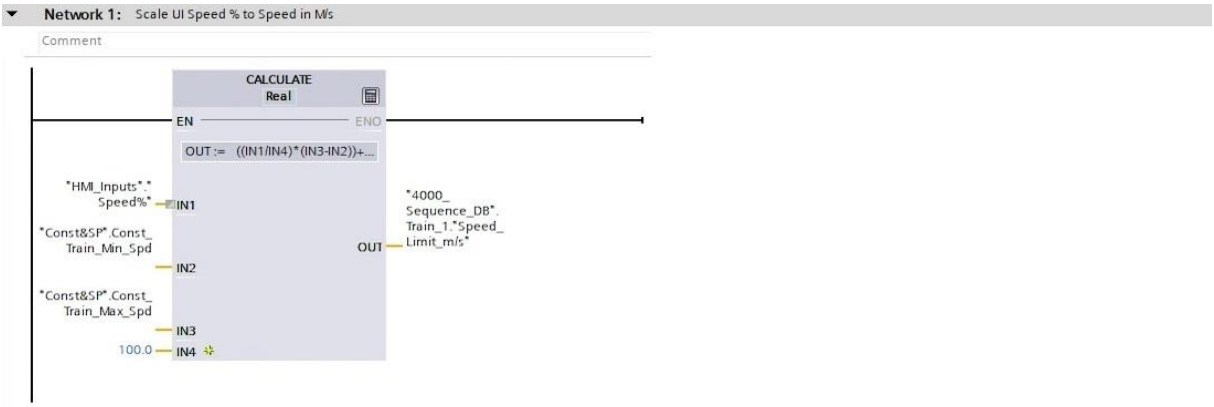


Slika 52. Kod aktivacije prvog senzora (i dodatnog zaustavnog senzora na prvoj stanici)

Uz osnovnu funkcionalnost simulacije položaja, implementirana je i mogućnost simuliranja kvara senzora. Za svaki senzor definiran je zaseban bit pogreške, koji simulira kvar senzora kad je postavljen na 1. Ako je taj bit aktivan, odgovarajući senzor neće reagirati na poziciju vlaka, bez obzira na stvarno stanje na tračnicama. Ovo je korisno kada želimo simulirati kvar u sustavu.

7.2 Pozicija, brzina i ubrzanje

Parametri koje nadziremo i kontroliramo su pozicija, brzina i ubrzanje vlaka. Kako bi korisnici na jednostavan način mogli odabrati željenu brzinu i ubrzanje putem HMI-a, unosi su omogućeni u postocima, čime se ubrzava postavljanje parametara. Međutim, ti postotci zahtijevaju pretvorbu u mjerne jedinice, konkretno metre po sekundi (m/s) za brzinu i metre po sekundi na kvadrat (m/s²) za ubrzanje. Program opisan u nastavku omogućava automatsko skaliranje tih vrijednosti, njihovu preciznu primjenu u simulaciji i kontinuirano praćenje pozicije vlaka na temelju integracije brzine.

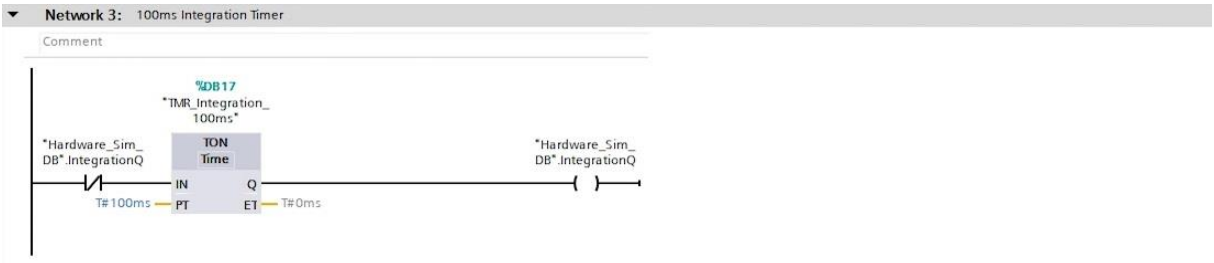


Slika 53. Skaliranje postotka brzine u mjernu jedinicu [m/s]

Prvi korak u simulaciji odnosi se na pretvorbu odabranog postotka brzine u brzinu izraženu u m/s, a za to se koristi blok *CALCULATE*. Na izlazu ovog bloka dobiva se vrijednost brzinskog ograničenja (*Speed Limit m/s*), koja definira najveću dopuštenu brzinu vlaka u simulaciji. Korištenjem ove vrijednosti, sustav omogućuje korisnicima zadavanje željene brzine u postotcima, a da se pritom osigura da vlak neće prekoračiti sigurnosno postavljeni limit. Skaliranje postotka brzine u stvarnu brzinu odvija se prema formuli koja dijeli korisnički unos sa 100 kako bi ga skalirala na vrijednost između 0 i 1. Zatim se ta vrijednost množi s rasponom između maksimalne i minimalne brzine, a na kraju se dodaje minimalna brzina kako bi se osiguralo da brzina nikad ne padne ispod definirane donje granice. Na taj način, korisnik može jednostavno postaviti brzinu u postotcima, dok program automatski pretvara tu vrijednost u stvarnu brzinu u m/s koju će koristiti tijekom simulacije.

Isti postupak koristi se i za skaliranje ubrzanja, uz korištenje odgovarajućih granica za minimalno i maksimalno ubrzanje. Na ovaj način dobivamo vrijednosti brzine i ubrzanja u jedinicama koje sustav može koristiti za precizno simuliranje kretanja vlaka.

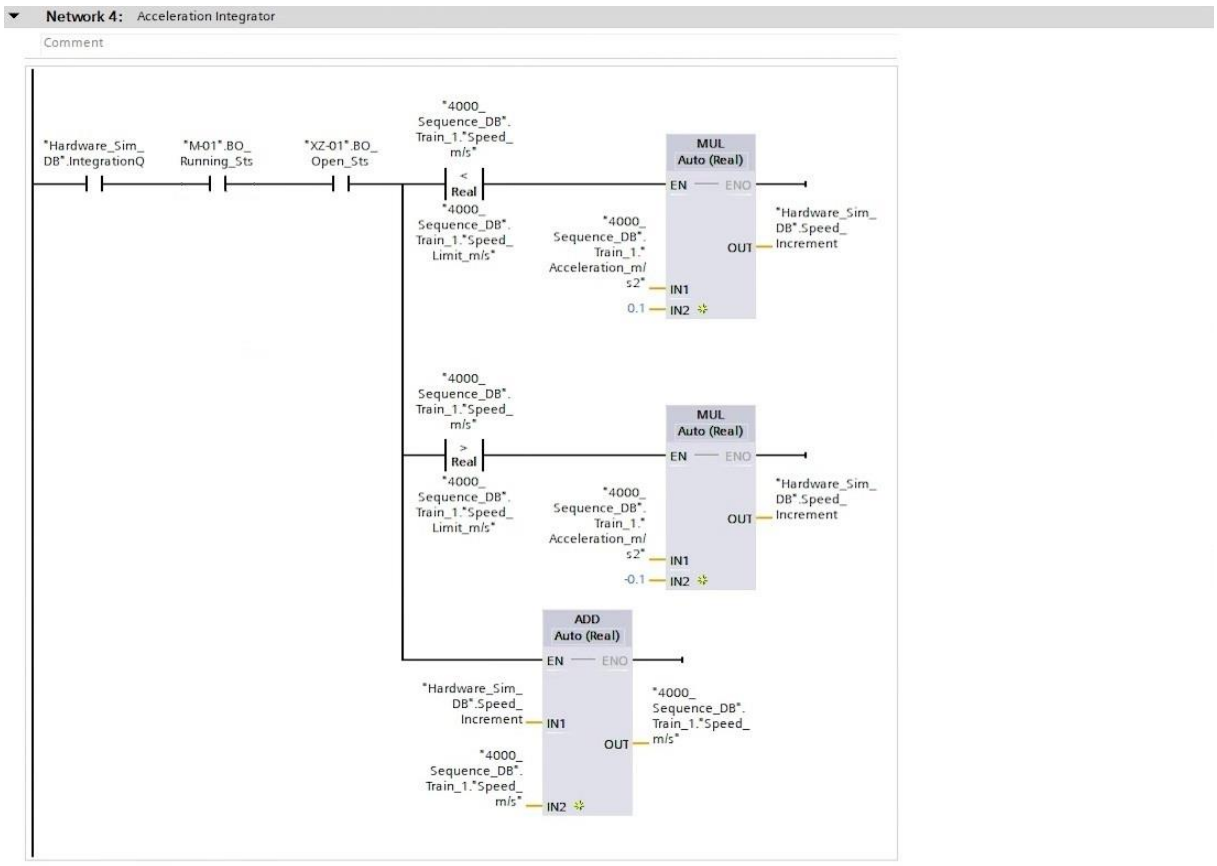
Za praćenje i ažuriranje brzine i pozicije vlaka u redovitim intervalima koristi se tajmer odnosno vremenski relej s odgodom (*slika 54.*). Tajmer se postavlja tako da generira impuls svakih 100 ms, što omogućava programu da kontinuirano preračunava i ažurira vrijednosti brzine i pozicije u kratkim vremenskim intervalima. Budući da je interval dovoljno kratak, možemo računati na preciznu aproksimaciju integracije, što je važno za simulaciju kretanja u stvarnom vremenu. Tajmer radi na način da započinje brojanje od nule i uključuje se kada dosegne 100 ms, postavljajući izlaz na aktivno stanje (*„true"*). Nakon toga automatski se resetira na nulu i započinje ponovno brojanje. Ovaj uvjet s ponavljajućom sekvencom osigurat će točnost i stabilnost simulacije.



Slika 54. Integracijski tajmer

Nakon skaliranja brzine i ubrzanja te postavljanja tajmera, program prelazi na proces integracije ubrzanja kako bi se izračunala trenutačnu brzina vlaka. U ovom koraku, svakih 100 ms trenutačna brzina dobiva ubrzanje, čime se postiže progresivno povećanje ili smanjenje brzine u skladu s korisničkim unosom. Međutim, promjena brzine dopuštena je samo kada su zadovoljeni uvjeti pokretanja: motor vlaka mora biti pokrenut, a kočnica otpuštena. Ako ti uvjeti nisu ispunjeni, brzina ostaje nepromijenjena. Prilikom integracije ubrzanja, dodatno se koristi faktor 0.1 (koji predstavlja omjer 100 ms prema jednoj sekundi) kako bi se prirast brzine prilagodio vremenskom intervalu. Time se trenutačna vrijednost brzine ažurira svakih 100 ms.

S obzirom na to da je moguće prekoračenje sigurnosnog limita brzine, program dodatno provjerava je li trenutačna brzina manja od ograničenja brzine (*Speed Limit m/s*). Ako trenutačna brzina premaši ovo ograničenje, prirast brzine postaje negativan, čime se brzina postupno smanjuje dok se ne vrati unutar dopuštenih granica. Ova logika osigurava prilagodbu sustava za oba smjera kretanja jer blok *ADD* automatski smanjuje brzinu kad ona premaši zadani prag.



Slika 55. Integracija akceleracije

Na isti način odvija se proračun pozicije putem integracije brzine. Kao i kod proračuna brzine, promjena pozicije događa se samo kada su zadovoljeni uvjeti o pokretanju motora i otpuštanju kočnice. Osim toga, smjer kretanja vlaka također utječe na promjenu pozicije. Ako se vlak kreće unaprijed, pozicija se povećava, dok se kod kretanja unatrag pozicija smanjuje. Smjer kretanja određuje se na temelju stanja motora i smjera vožnje, što omogućava realističnu simulaciju kretanja u oba smjera. Isto kao i u slučaju integracije ubrzanja, prirast pozicije prilagođava se vremenskom intervalu od 100 ms množenjem brzine s faktorom 0.1. Dobiveni prirast zatim se dodaje trenutačnoj poziciji vlaka, čime program kontinuirano prati i ažurira poziciju vlaka u simulaciji.

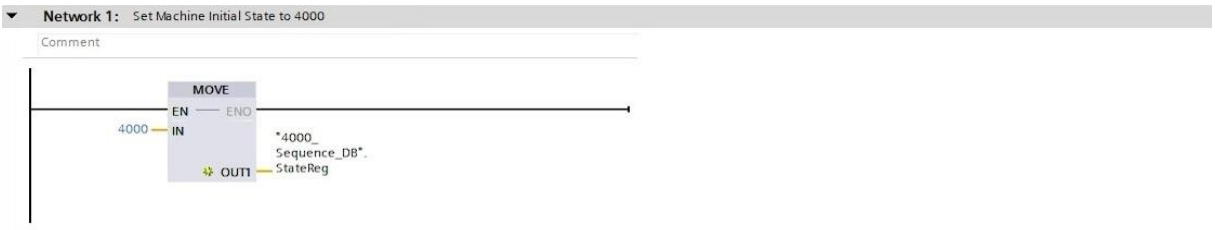
Dakle, glavne pretpostavke koje smo primijenili u ovome dijelu programa odnose se na to da je ubrzanje konstantno, brzina je jednaka integraciji ubrzanja, a pozicija je jednaka integraciji brzine. Također, prikladnim aproksimacijama integracija zaključeno je da se pri vremenskom intervalu od 100 ms odvija zanemarivo malo odstupanje pozicije (2.5 %), koje je dovoljno dobro za potrebe ove simulacije.

8. INICIJALIZACIJA PARAMETARA SUSTAVA

U industrijskim automatiziranim sustavima, inicijalizacija je ključna faza koja osigurava da sustav započne s pravilno postavljenim početnim uvjetima. Proces inicijalizacije omogućava stabilnost i predvidivost u radu stroja ili uređaja, posebno u složenim industrijskim procesima gdje neispravno pokretanje može uzrokovati razne kvarove i smetnje u radu sustava. Zato ćemo u ovome poglavlju objasniti kako se postavlja inicijalizacija sustava u PLC programu, koristeći konkretan primjer postavljanja osnovnih parametara, kao što su početno stanje, brzina, pozicija i ubrzanje vlaka.

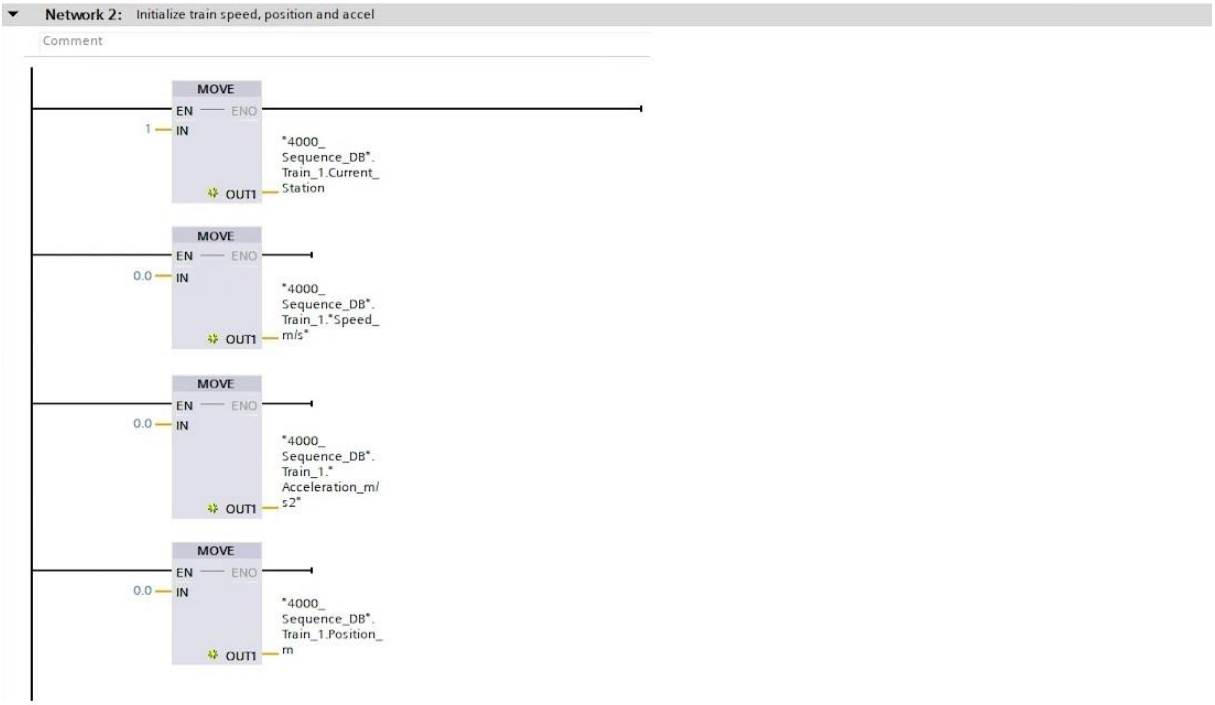
Prvi korak u procesu inicijalizacije jest dodavanje *„startup“* organizacijskog bloka (OB). *„Startup OB“* je posebna vrsta organizacijskog bloka koji se izvršava jednom kada se način rada PLC-a promijeni iz *„stop“* u *„run“*. Drugim riječima, svaki put kada uključimo PLC ili ga resetiramo, ovaj organizacijski blok će se automatski pokrenuti i izvršiti zadane naredbe. Nakon što *„startup OB“* dovrši izvršenje naredbi, prelazi se na glavni ciklus programa, koji se odvija kontinuirano sve dok je PLC u radnom stanju.

U ovome se bloku postavljaju početne vrijednosti za sve značajne varijable u programu, garantirajući sustavu da uvijek započne s predvidivim stanjem. Na primjer, ako ne inicijaliziramo registarske varijable, sustav može započeti s neispravnim ili nedefiniranim vrijednostima. To bi za naš program značio da ako početno stanje ne postavimo na 4000, sustav bi mogao započeti s registrom na nuli, a time bi se onemogućio rad sustava jer se ne bi postiglo početno stanje u kojem sustav očekuje korisničke unose. Stoga je prva vrijednost koju trebamo inicijalizirati početno stanje sustava. Želimo da sustav započne u stanju koje čeka korisnički unos, a to je stanje označeno brojem 4000. Primjenjujemo *MOVE* naredbu koja postavlja vrijednost 4000 u registar stanja unutar baze podataka sekvenci, čime sustav započinje s pravim početnim stanjem. Ova inicijalizacija osigurava da, čim se PLC uključi, sustav bude spreman za rad u ispravnom načinu i čeka unose korisnika.



Slika 56. Inicijalizacija početnog stanja sustava

Drugi važni parametri koje trebamo inicijalizirati uključuju poziciju, brzinu i ubrzanje vlaka. Kao početnu poziciju postavljamo stanicu broj jedan, gdje vlak započinje svoje kretanje. Brzina i ubrzanje postavljeni su na nulu, što osigurava da vlak započinje kretanje iz stanja mirovanja bez ikakvog prethodnog kretanja ili ubrzanja. Ove vrijednosti također postavljamo s pomoću *MOVE* naredbe u istome organizacijskom bloku. Konkretno, postavljamo varijablu *Train_1_Current_Station* na 1, dok *Train_1_Speed* i *Train_1_Acceleration* postavljamo na 0.



Slika 57. Inicijalizacija brzine, pozicije i ubrzanja vlaka

Nakon što smo završili inicijalizaciju, možemo prijeći na dizajn HMI sučelja te pokrenuti simulaciju za testiranje, pri čemu se prilagodbe mogu vršiti prema potrebama sustava. Tijekom testiranja, sigurno će biti potrebno dodatno prilagoditi neke od parametara ili dijelove logike, ali to je uobičajeni proces u razvoju svih automatiziranih sustava, pa tako i razvoja ovog sustava za autonomno upravljanje električnim vlakom.

9. HMI sučelje; implementacija i testiranje

U ovom poglavlju obrađuje se dizajn i implementacija HMI (engl. Human-Machine Interface) sučelja za simulaciju rada razvijenog sustava. HMI sučelje omogućava operaterima jasan pregled kretanja vlaka i statusa ključnih komponenti poput motora, kočnica i senzora, dok kontrolni paneli pružaju mogućnosti upravljanja sustavom, od odabira stanica do simuliranja različitih uvjeta rada. Prvi dio poglavlja posvećen je početnom zaslonu, glavnoj kontrolnoj točki sustava. Uz početni zaslon, sustav uključuje dodatne zaslone za detaljan prikaz grafova brzine i položaja vlaka te popis aktivnih grešaka, pružajući praćenje rada sustava u stvarnom vremenu. Obrađeni su i postupci povezivanja gumba s PLC programom, konfiguriranje klizača, te stvaranje animacije kretanja vlaka. Završno funkcionalno testiranje koje uključuje provjeru kretanja vlaka, zaustavljanja, te reakcije sustava na kvarove, potvrđuje da sustav radi ispravno, ispunjavajući funkcijske specifikacije i sigurnosne zahtjeve.

9.1 Početni zaslon

HMI dizajniramo kao virtualno sučelje gdje operateri mogu pratiti kretanje vlaka i stanje različitih senzora i komponenti sustava. Na početnom zaslonu (*home screen*) prikazan je vlak sa svojim motorom i kočnicama, te senzori koji se nalaze duž pruge kojom vlak putuje. U vizualnom prikazu iskoristit ćemo različite boje i animacije kako bi status svake komponente sustava bio jasan. Osim toga naš početni zaslon sadržavat će i druge komponente koje će biti objašnjene u nastavku.

Prije svega na početni zaslon smještamo senzore koji služe za precizno praćenje pozicije vlaka. Svaki senzor prikazan je simbolom trokuta i nosi jedinstveni identifikacijski broj (oznaku) koji pomaže operaterima u prepoznavanju. Na primjer, senzori su označeni oznakama poput "ZS01", "ZS02" i tako dalje, prema oznakama koje smo definirali u funkcionalnim specifikacijama sustava. Kada vlak dođe u blizinu određenog senzora, taj senzor se aktivira i njegova boja se mijenja u zelenu, što jasno pokazuje da je vlak detektiran na određenoj poziciji. Ako senzor nije aktivan, prikazan je sivom bojom. Ova promjena boje dopušta operaterima da u svakom trenutku jasno vide gdje se vlak nalazi i u kojem smjeru se kreće, a također omogućuje i vizualno otkrivanje mogućih problema u detekciji pozicije vlaka.

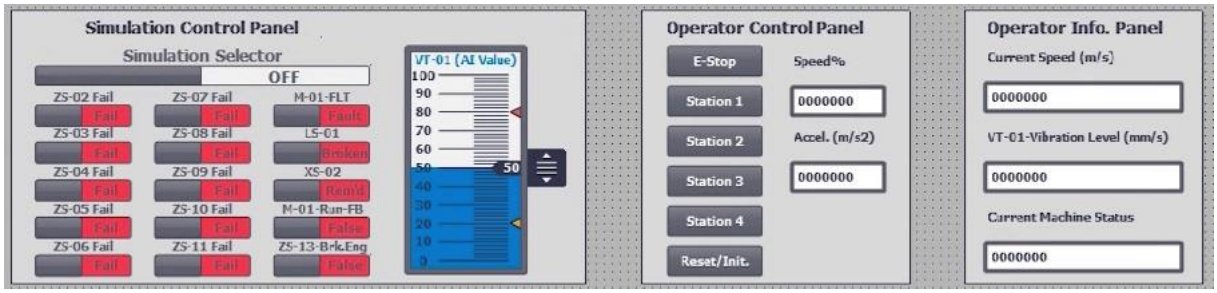
Osim senzora, početni zaslon uključuje vizualni prikaz motora vlaka. Motor ima tri različite boje prikaza koje odražavaju njegovo trenutačno stanje. Zelena boja označava da je motor aktivan i da radi ispravno. Siva boja signalizira da je motor spreman za rad, ali trenutačno nije pokrenut, dok

crvena boja označava kvar ili grešku u radu motora. Kočnice vlaka također su prikazane na sličan način poput motora i imaju tri različite boje koje signaliziraju njihovo stanje. Zelena boja znači da su kočnice aktivirane, siva boja znači da su kočnice otpuštene, dok crvena signalizira grešku u sustavu kočenja.

TIA Portal sadrži grafičku knjižnicu s raznim slikama i ikonama koje se mogu koristiti za prikaz različitih komponenti i stanja na HMI-u. Za motor, možemo primijeniti postojeću ikonu iz TIA Portalove knjižnice. Što se tiče vlaka i kočnica, radi se o specifičnim slikama koje su prilagođene potrebama simulacije i one su učitane iz vanjskih izvora stvaranjem osobnog grafičkog foldera (engl. „My Graphics folder“).

9.1.1 Kontrolni paneli na početnom zaslonu

Na početni zaslon smještena su tri kontrolna panela, kako bi simuliranje bilo preglednije, i kako bi prikaz bio organiziran i jednostavan za praćenje. Simulacijski kontrolni panel (engl. Simulation Control Panel), daje mogućnost odabira između automatskog i simulacijskog načina rada, između kojih se prebacuje odabirom tipki „on“ za simulacijski i „off“ za automatski režim. U simulacijskom načinu rada možemo simulirati različite uvjete i kvarove, testirati funkcionalnost sustava, te ručno aktivirati ili deaktivirati senzore, motor, sigurnosnu svjetlosnu zavjesu. Na primjer, možemo simulirati kvar senzora birajući opciju „neispravan“ na odgovarajućem prekidaču, kako bi se testiralo ponašanje sustava u stanju kvara. Na panelu su smješteni i prekidači za simulaciju kvara motora i svjetlosne zavjese, koja je važan sigurnosni element sustava. Sigurnosna svjetlosna zavjesa ima funkciju zaštite prostora tračnica i odmah signalizira ako je netko prekinuo svjetlosnu barijeru, sprečavajući nesreću. Također je moguće simulirati oba stanja sigurnosnog ključa, koji je ključan za ponovno pokretanje sustava nakon zaustavljanja ili otklanjanja greške. Možemo postaviti da je sigurnosni ključ prisutan u kućištu ili uklonjen. Uklanjanjem ključa simulira se situacija u kojoj se sustav ne može pokrenuti dok se ključ ne vrati na svoje mjesto. Dakle, simulacijski kontrolni panel služi kako bi se simuliralo kretanje vlaka i sve situacije koje se mogu dogoditi u stvarnosti na području kretanja vlaka. Ostala dva panela koja se nalaze na početnom zaslonu namijenjena su operaterima odnosno korisnicima vlaka, gdje je ideja da naš autonomni sustav omogući korisniku koji se vozi vlakom; odabir stanice, te željenu brzinu, odnosno ubrzanje vlaka (u postotcima) i informacijski uvid u trenutačne parametre kretanja vlaka.



Slika 58. Paneli smješteni na početnom zaslonu

Kontrolni panel namijenjen operateru (engl. Operator Control Panel*)*, služi kao upravljačko sučelje za korisnike i omogućuje im upravljanje glavnim funkcijama sustava u stvarnom vremenu, a to su gumb za hitno zaustavljanje, gumbi za odabir stanica te postavke željene brzine i ubrzanja vlaka. Postavljanjem maksimalnih vrijednosti za brzinu i ubrzanje osigurava se da vlak ne premaši zadane granice, neovisno o korisničkom unosu.

Posljednji u nizu na početnom zaslonu, smješten je informacijski panel namijenjen operateru (engl. Operator Information Panel), koji služi za prikaz važnih informacija o trenutačnom statusu vlaka. Na ovom panelu prikazane su: trenutačna brzina vlaka, razine vibracija i status položaja. Brzina vlaka prikazana je u metrima po sekundi (m/s), a operateri mogu u svakom trenutku vidjeti kako se brzina mijenja tijekom simulacije. Pregled razine vibracija, pruža mogućnost praćenja eventualnih odstupanja ili problema u radu. Visoke vibracije mogu ukazivati na mehaničke probleme ili kvar komponenata. Status sustava prikazuje trenutačno stanje vlaka, primjerice, je li vlak zaustavljen, ide li prema određenoj stanici ili se nalazi u stanju čekanja korisničkih naredbi. Ove informacije pružaju operaterima cjeloviti pregled nad radom sustava i osiguravaju donošenje informiranih odluka.

9.2 Dodatni zasloni

Osim početnog zaslona na HMI-u, naš sustav ima još dva dodatna zaslona: Grafikoni (engl. Charts) i Greške (engl. Faults). Zaslon s grafikonima, osmišljen je kako bi korisniku pružio prikaz brzine vlaka i pozicije u realnom vremenu. Za prikaz tih podataka koristi se grafički element *Trend View*, koji se može pronaći u alatnoj traci (*Toolbox*) pod sekcijom *Controls* (Kontrole). Ovaj element omogućava vizualizaciju promjena varijabli vlaka u vidu linijskog grafikona, za preglednije nadgledanje. Grafikoni su korisni za analizu performansi vlaka, procjenu eventualnih promjena u brzini i putanji, te za bolje razumijevanje rada sustava kroz vrijeme.

Drugi dodatni zaslon, namijenjen je prikazu detaljnih informacija o greškama i upozorenjima unutar sustava. Cilj ovog zaslona je informirati operatera o svim trenutačnim problemima u sustavu. Za prikaz informacija o greškama koristi se element *Alarm View*, koji također dolazi iz sekcije *Controls* (Kontrole) unutar alatne trake (*Toolbox*). *Alarm View* omogućava prikaz popisa aktivnih grešaka i upozorenja u sustavu, zajedno s relevantnim detaljima kao što su vrsta greške, vrijeme pojave i razina prioriteta. Na ovaj način, operater ima uvid u sve aktualne probleme te može brzo identificirati i reagirati na kritične situacije. Kada se prikaže određena greška ili upozorenje, operater može poduzeti potrebne mjere, kao što su resetiranje greške ili otklanjanje uzroka problema. Ovaj zaslon pomaže u održavanju optimalnog rada sustava. Za jednostavno prebacivanje između zaslona na HMI-u, na vrhu svakog zaslona postavljena su tri gumba za navigaciju: Početna stranica, Grafikoni i Greške.

9.3 Povezivanje gumba

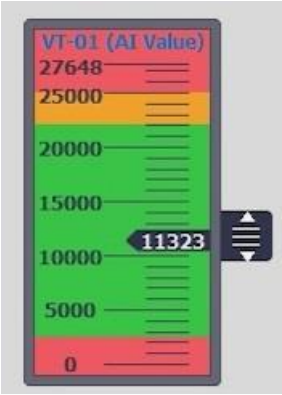
Proces povezivanja gumba u HMI sustavu predstavlja važan korak kojim se interaktivni elementi na sučelju povezuju s PLC programom kako bi se osigurala kontrola i upravljanje sustavom. Ovaj proces započinje odabirom gumba ili prekidača na HMI zaslonu, nakon čega se pristupa postavkama svojstava za taj element. U sekciji *„Events“* odnosno događaji, definiraju se akcije koje će se dogoditi kada korisnik pritisne gumb ili aktivira prekidač, a koje su povezane s određenim oznakama u PLC programu. Primjerice, u slučaju gumba za simulaciju greške senzora, postavlja se radnja *„Set Bit“* kada je prekidač uključen, čime se određeni bit u PLC-u postavlja na vrijednost 1, dok se kod isključenja prekidača koristi *„Reset Bit“* za vraćanje bita na 0. Ova metoda omogućava da se promjene statusa na HMI-ju reflektiraju u stvarnom vremenu u PLC programu, čime se simuliraju različiti uvjeti rada.

Važno je napomenuti da svaki put kad se poveže gumb s odgovarajućom oznakom, automatski se generira HMI oznaka (*tag*) koja povezuje PLC varijablu i HMI komponentu. Osim standardnih gumba, sličan se proces koristi i za druge interaktivne elemente na HMI ekranu, poput klizača za unos analognih vrijednosti ili tipki za navigaciju između ekrana. Svaki element može biti povezan s odgovarajućim funkcijama u PLC programu, što omogućuje potpunu interakciju između HMI sučelja i sustava kojim se upravlja.

9.4 Konfiguracija klizača za simulaciju analognog ulaza

Konfiguracija klizača za simulaciju analognog ulaza (engl. analog slider) omogućava preciznu simulaciju vrijednosti koje dolaze s analognog senzora, u ovom slučaju sustava za mjerenje vibracija. Uz pomoć njega, možemo mijenjati vrijednosti vibracija unutar unaprijed određenog raspona.

Prvi korak u konfiguraciji klizača je povezivanje klizača s odgovarajućom procesnom oznakom. U ovom slučaju, klizač je povezan s oznakom (*tagom*) koja simulira ulazni signal od strane vibracijskog mjernog sustava. Raspon klizača postavlja se prema rasponu Siemensove analogne kartice, što je u ovom slučaju od 0 do 27,648.



Slika 59. Raspon klizača

Kako bi se osigurao prikaz statusa vibracija, definiraju se četiri razine upozorenja i alarma: izrazito niska (u prethodnim oznakama LL), normalna razina, visoka (H), te izrazito visoka (HH) razina. Prema tome, boje na klizaču pokazuju status sustava: zelena označava siguran rad, dok crvena upozorava na kritične razine vibracija, a žuta predstavlja upozorenje na visoku razinu vibracija.

Za određivanje vrijednosti granica, koriste se konstante koje su skalirane prema omjeru između maksimalnog ulaza i fizičkog raspona mjerenja vibracija. Primjerice, za donju graničnu vrijednost uzima se 10 % od maksimalnog ulaza, dakle 2765, dok su gornje vrijednosti postavljene prema definiranim razinama alarma. Konfiguracija klizača omogućava i unos u crvene kritične zone, čime se može simulirati stanje greške, što je korisno za testiranje odgovora sustava u slučajevima kada senzor bilježi opasne razine vibracija.

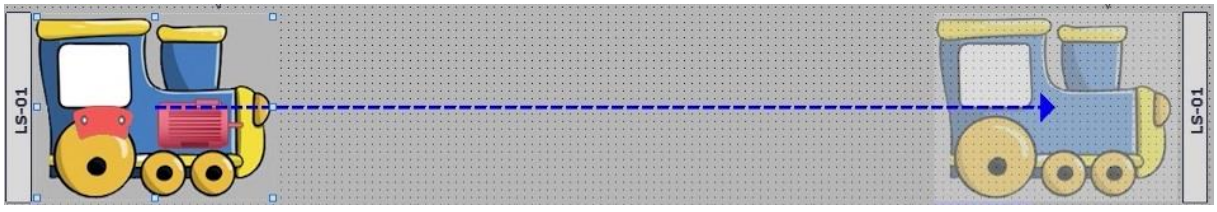
9.5 Postavljanje animacije za kretanje vlaka na zaslonu

Kako bismo osigurali da se na zaslonu prikazuje kretanje vlaka, potrebno je odabrati objekt koji predstavlja vlak, zatim ući u postavke za animacije te odabrati opciju kretanja. Ovdje ćemo stvoriti novu animaciju, birajući da kretanje bude horizontalno. Nakon toga, povezujemo animaciju s odgovarajućom oznakom koja će pratiti poziciju vlaka. U našem slučaju, oznaka je spremljena u PLC programu pod podatkovnim blokom koji se odnosi na simulaciju kretanja vlaka i sadrži zaokružene vrijednosti položaja u metrima. Razlog tome je što kod ovakve animacije TIA Portal ne radi s decimalnim vrijednostima, već prihvaća samo cijele brojeve.

Za precizan prikaz kretanja vlaka, potrebno je odrediti početnu i krajnju točku raspona kretanja, a zatim te vrijednosti treba unijeti u polja *„Range From“ i „Range To“* u postavkama animacije. U razvijenom kodu PLC-a, raspon kretanja vlaka definiran je od -10 do 1360, pri čemu je -10 pozicija koja se nalazi neposredno prije prvoga senzora na stanici 1, dok je 1360 zaustavna pozicija na posljednjoj stanici. Zatim vizualno prilagođavamo početnu poziciju vlaka na zaslonu, tako da se podudara s početnom točkom na ruti, dok za krajnju poziciju postavljamo X koordinatu na kojoj želimo da vlak završi kada dosegne posljednju stanicu u simulaciji. Ova pozicija se podešava testiranjem i prilagođavanjem kako bi simulacija bila što točnija i u skladu s očekivanim kretanjem na zaslonu.

Određeni elementi poput motora i kočnica prikazani su kao zasebni objekti na vlaku. Da bi se osiguralo da svi ti elementi prate kretanje vlaka, potrebno je svaki od njih zasebno animirati koristeći iste postavke kao i za glavni objekt vlaka. To znači dodavanje horizontalne animacije za svaki dodatni objekt (motor i kočnice), povezivanje s istom oznakom za poziciju vlaka te postavljanje istog raspona kretanja (-10 do 1360).

Nakon što su sve animacije postavljene, potrebno je poravnati sve dijelove vlaka tako da budu pozicionirani jedni preko drugih, kako bi vlak, motor i kočnice izgledali kao jedna cjelina. Na kraju, provjeravamo funkcionalnost animacija pokretanjem simulacije kako bismo osigurali da se svi elementi kreću zajedno i da cijela animacija izgleda realistično te da korisniku jasno vizualizira kretanja vlaka na zaslonu.



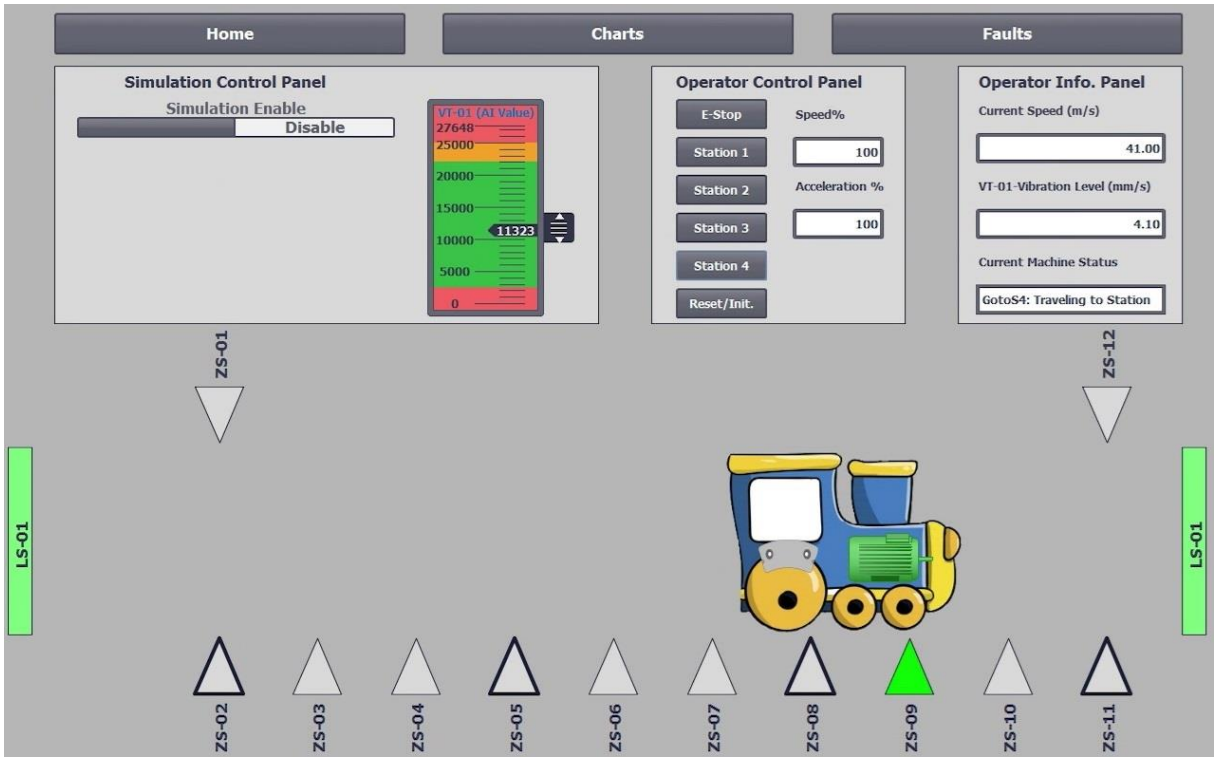
Slika 60. Postavljanje animacije za kretanje vlaka na zaslonu

9.6 Završno funkcionalno testiranje sustava

U ovom potpoglavlju usmjerit ćemo se na simulaciju rada cjelokupnog sustava, što nas dovodi do podešavanja detalja i usklađivanja razvijene logike. Ovo čini završni korak dizajniranja automatiziranih sustava, a sastoji se u testiranju segmenata sustava, otklanjanju grešaka i optimizaciji, kako bi rad sustava bio u potpunosti usklađen s funkcionalnim specifikacijama. U praktičnom okruženju, također bi se precizno provjeravao rad svih sastavnica, od PLC koda do HMI sučelja, prije primjene sustava na terenu. Kada se završe potrebne prilagodbe i postignu željene performanse, provodi se završni funkcionalni test koji potvrđuje da sustav radi ispravno izvršavajući sve predviđene funkcije.

Prvo odabiremo opciju „*Simulation Disable"* i prelazimo u automatski način rada. Test kretanja vlaka započinjemo biranjem stanice prema kojoj želimo da vlak putuje. Nakon odabira stanice, kada se vlak pokrenuo, pratimo ubrzanje vlaka u prozoru koji prikazuje trenutačnu brzinu, te opažamo kako se brzina povećava nakon pokretanja, što je u skladu s očekivanjima. Na polju statusa pozicije ispisuje se odabrana stanica kao trenutačna destinacija vlaka.

Tijekom kretanja vlak aktivira senzore koji registriraju njegov prolazak, svijetleći zelenom bojom. Ikone koje predstavljaju sigurnosnu svjetlosnu zavjesu također svijetle zelenom bojom, što signalizira da je zaštita aktivna. Kada odaberemo dolazak na posljednju stanicu, vlak se zaustavlja na točno predviđenoj poziciji. Po zaustavljanju, motor prelazi u stanje mirovanja, što je prikazano sivom bojom, dok kočnica koja se aktivirala svijetli zelenom bojom. Na isti način testiramo kretanje vlaka unatrag, od četvrte stanice prema prvoj stanici. Ovaj proces potvrđuje da sustav radi ispravno i u skladu s predviđenim specifikacijama.

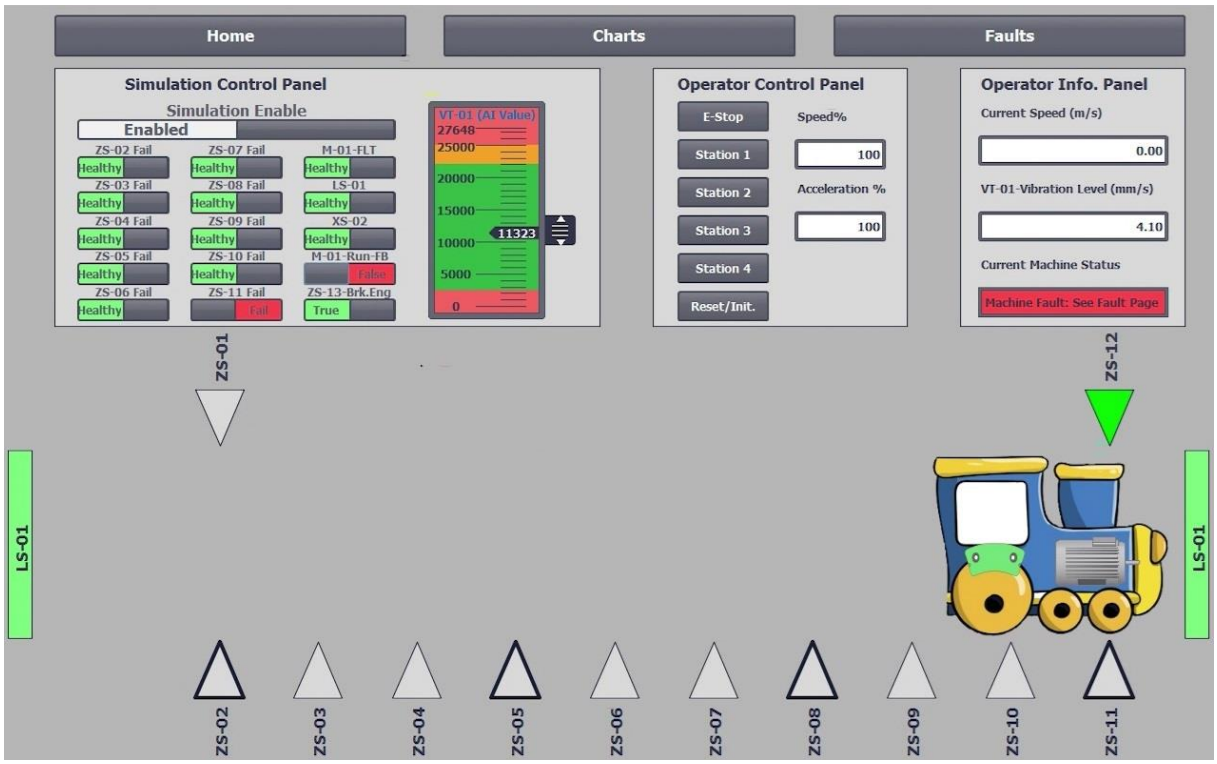


Slika 61. Test kretanja vlaka od prve prema četvrtoj stanici

Prilikom testiranja kako sustav reagira na stanje kvara, sustav je postavljen u simulacijski način rada (*„Simulation Mode Enabled“*). Na simulacijskom kontrolnom panelu prebacivanjem postavki u simulacijski način rada, moguće je testiranje raznih stanja kvarova koja mogu nastati zbog prethodno definiranih grešaka. Ove greške uključuju: kvar motora, neuspjelu aktivaciju kočnice, pogrešku u slijedu aktivacije senzora, prekoračenje putanje vlaka, uklanjanje sigurnosnog ključa, prekid sigurnosne svjetlosne zavjese, izrazito visoke vibracije vlaka, te izrazito niske vibracije vlaka. Osim ovih kvarova, testiramo i upozorenje na visoku vibraciju stroja te provjeravamo funkcionalnost tipke za hitno zaustavljanje (*„Emergency Stop“*) i tipku za resetiranje (*„reset/initialize“*).

Prilikom testiranja provjeravamo sva stanja kvarova u sustavu prema navedenim greškama. Također provjeravamo što se događa kada tijekom kvara prebacimo sustav na automatski način rada, te pritom zadamo naredbu za pokretanje vlaka i odlazak na neku stanicu. U ovom slučaju vlak ostaje u mjestu (motor svijetli sivom bojom, kočnica zelenom bojom), a na ekranu trenutačnog statusa prikazuje se poruka o kvaru stroja: *„Machine Fault: see Fault Page“*. Ova reakcija sustava u skladu je s našim planiranim funkcionalnostima – sustav je dizajniran tako da se, u slučaju kvara, vlak automatski zaustavi i ostane u tom stanju sve dok se problem ne otkloni.

Na zaslonu koji prikazuje povijest grešaka u sustavu nije moguće jednostavnim pritiskom na tipku obrisati zapis o kvaru. Ova je funkcija onemogućena sve dok se ne pritisne tipka za resetiranje, čija je upotreba predviđana nakon otklanjanja kvara. Bitno je napomenuti da se ova tipka može pritisnuti samo kada je motor isključen. Nakon provedenih testova zaključujemo da sustav reagira ispravno i točno prema očekivanjima za slučajeve kvara, što potvrđuje njegovu pouzdanost i sigurnost. Jedan od testova sustava na stanje kvara, prikazan je na *slici 62.*



Slika 62. Test reakcije sustava na stanje kvara

Na *slici 62.* je prikazano testiranje sustava za grešku prelaska preko krajnje pozicije, pri čemu ključnu ulogu imaju senzori na zaslonu. Svi senzori sustava prikazani su trokutastim simbolima s brojčanom oznakom, a senzori koji označavaju pozicije zaustavljanja na stanicama 1, 2, 3 i 4 posebno su istaknuti podebljanim oznakama. Na krajnjim stanicama (početnoj i završnoj), dodatno su postavljena dva senzora (ZS-01 i ZS-12) kao sigurnosna mjera. Njihova je funkcija presudna u slučaju da dođe do kvara osnovnog zaustavnog senzora na krajnjoj poziciji. U takvim slučajevima, dodatni senzori signaliziraju vlaku da se mora zaustaviti, čime se sprječava situacija u kojoj vlak prelazi krajnju poziciju na tračnicama.

Kada svi senzori ispravno rade, vlak prilikom približavanja krajnjoj poziciji aktivira i zaustavni i dodatni senzor u isto vrijeme. Ovo sinkronizirano djelovanje senzora šalje jasan signal sustavu da se vlak treba zaustaviti. Tijekom testiranja greške prelaska preko krajnje pozicije, simulira se kvar zaustavnog senzora ZS-11 na stanici 4. U takvoj situaciji, sustav detektira grešku te unatoč kvaru, prepoznaje potrebu za zaustavljanjem vlaka. Vlak usporava i zaustavlja se, što se prikazuje tako da se simbol motora isključuje i svijetli sivom bojom, a simbol kočnice svijetli zelenom bojom, potvrđujući da su kočnice aktivirane. Na statusnom prozoru sustava ispisuje se poruka o greški, čime sustav obavještava operatera o potrebi provjere i eventualne intervencije. Ovaj test pokazuje kako sustav reagira u scenarijima kada osnovni senzori otkažu, te kako dodatni senzori osiguravaju pravovremeno zaustavljanje vlaka i zaštitu od potencijalno opasne situacije.

Završno funkcionalno testiranje je obuhvatilo provjeru svih ključnih funkcija, uključujući kretanje vlaka između stanica, zaustavljanje na krajnjim pozicijama, te reakciju sustava na različita stanja kvara. Svi senzori, sigurnosne mjere i algoritmi za kontrolu brzine, zaustavljanje i povratne informacije, funkcioniraju u skladu s očekivanjima, osiguravajući nesmetan i ispravan rad sustava.

10. ZAKLJUČAK

Ovaj diplomski rad predstavlja sveobuhvatan pregled razvoja automatiziranog sustava upravljanja autonomnim električnim vlakom kroz kompleksnu simulaciju. Od inicijalnog definiranja funkcionalnih specifikacija, preko odabira potrebnog hardvera i softvera, sve do implementacije složene PLC logike i HMI sučelja, rad prikazuje važnost i izazove primjene suvremene automatizacije u području transporta. Kroz detaljnu razradu svih tehničkih koraka, uključujući programiranje funkcijskih blokova za upravljanje motorom, senzorima i sustavom kočenja, osigurani su svi uvjeti za precizno i sigurno upravljanje autonomnim sustavom.

Poseban naglasak stavljen je na sigurnost i pouzdanost sustava, obuhvaćajući elemente kao što su detekcija grešaka i sigurnosne značajke. Integracija ovih funkcija omogućuje pravovremeno prepoznavanje potencijalnih kvarova, čime se smanjuju rizici i povećava sigurnost sustava. Primjenom HMI sučelja omogućena je intuitivna i jednostavna interakcija korisnika sa sustavom, dok simulacija osigurava uvjete za testiranje i optimizaciju rada.

Rezultati završnog testiranja potvrđuju da razvijeni sustav ispunjava sve zadane specifikacije i zahtjeve, pružajući stabilno i učinkovito rješenje za autonomno upravljanje vlakom. U budućnosti, ovaj bi sustav mogao biti nadograđen implementacijom naprednijih algoritama za prediktivno održavanje i prilagodbu u realnom vremenu, što ostavlja prostora za daljnja istraživanja.

11. LITERATURA

[1] Horowitz, P; Hill; W: „The Art of Electronics“, Third Edition

[2] Scherz, P; Monk, S: „Practical Electronics for Inventors“, Fourth Edition

[3] Wilmshurst, T: „Designing Embedded Systems with PIC Microcontrollers“

[4] Larminie, J; Lowry, J: „Electric Vehicle Technology Explained“

12. SAŽETAK / SUMMARY

U radu je razvijeno konceptualno rješenje sustava automatizacije za autonomno upravljanje električnim vlakom primjenom softverskog paketa Siemens TIA Portal. Obuhvaćena je konfiguracija hardverskih komponenti, izrada funkcijskih blokova za upravljanje motorom, senzorima i kočnicom, te povezivanje i aktiviranje tih blokova u glavnoj programskoj sekvenci. Također su razvijene metode za detekciju i obradu grešaka, uključujući algoritme za prepoznavanje pogrešaka slijeda senzora. Osmišljeno je i implementirano HMI sučelje za kontrolu, praćenje sustava, testiranje putem kontrolnih panela i simuliranje kretanja vlaka. Funkcionalnim testiranjem sustava potvrđena je pouzdanost i ispravnost svih razvijenih funkcija.

Ključne riječi: automatizacija, upravljanje, programabilni logički kontroler (PLC), HMI sučelje, ljestvičasta logika, simulacija, testiranje

This thesis presents a conceptual solution for an automation system designed for the autonomous control of an electric train, developed using the Siemens TIA Portal software package. It includes the configuration of hardware components, development of function blocks for motor, sensor, and brake control, as well as the integration and activation of these blocks within the main program sequence. Methods for fault detection and fault handling were also developed, including algorithms for detecting sensor sequence errors. An HMI interface was designed and implemented for system control, monitoring, and testing through control panels and train movement simulation. Functional testing confirmed the reliability and accuracy of all implemented functions.

Keywords: automation, control, Programmable Logic Controller (PLC), HMI, ladder logic, simulation, testing