Upravljanje procesom punjenja boca primjenom PLC-a

Jakupec, Filip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Upravljanje procesom punjenja boca primjenom PLC-a

Diplomski rad

Filip Jakupec

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 12.07.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Filip Jakupec
Studij, smjer:	DEB - Održiva elektroenergetika
Mat. br. studenta, godina upisa:	D 920, 12.10.2015.
OIB studenta:	31832929446
Mentor:	Izv.prof.dr.sc. Dražen Slišković
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Damir Filko
Član Povjerenstva:	Doc.dr.sc. Emmanuel-Karlo Nyarko
Naslov diplomskog rada:	Upravljanje procesom punjenja boca primjenom PLC-a
Znanstvena grana rada:	**Automatizacija i robotika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	Za raspoloživi stroj za punjenje boca potrebno je projektirati algoritam koji će omogućiti njegov automatski rad. Potrebno je opisati rad stroja za punjenje boca i definirati zahtjeve na sustav automatskog upravljanja. Projektirani upravljački algoritam najprije simulirati u Matlab Stateflow-u i potvrditi njegovu ispravnost, a zatim ga implementirati
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	12.07.2017.

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

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 17.07.2017.

Ime i prezime studenta:	Filip Jakupec
Studij:	DEB - Održiva elektroenergetika
Mat. br. studenta, godina upisa:	D 920, 12.10.2015.
Ephorus podudaranje [%]:	0

Ovom izjavom izjavljujem da je rad pod nazivom**: Upravljanje procesom punjenja boca primjenom PLC-a**

izrađen pod vodstvom mentora Izv.prof.dr.sc. Dražen Slišković

i sumentora

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

Potpis studenta:

SADRŽAJ:

1. UVOD

U današnje vrijeme se teži tome da se proizvodnja u što većoj mjeri ili u potpunosti automatizira. Osim veće i učinkovitije proizvodnje, automatizacijom se povećava kvaliteta proizvoda i smanjuje rizik od ljudske pogreške. Razvojem tehnologije omogućeno je potpuno automatizirano vođenje industrijske proizvodnje, čovjek intervenira samo u slučaju kvarova ili većih poremećaja. Osnovni element za realizaciju sustava za automatizaciju je procesno računalo. Procesno računalo je osnovni element sustava za automatsko vođenje proizvodnog procesa. Najraširenij oblik procesnog računala je PLC (*Programmable Logic Controller*). PLC je krajem 60-ih godina 20. stoljeća zamišljen kao zamjena za relejnu tehniku, tj. elementarni oblik automatizacije [1]. Današnji PLC-ovi imaju čitavo mnoštvo naprednih funkcija i veliku procesorsku moć te značajno nadmašuju ovaj elementarni oblik automatizacije. PLC je danas osnovna komponenta za realizaciju industrijske automatizacije.

U ovom diplomskom radu potrebno je projektirati algoritam za stroj za punjenje boca koji će omogućiti njegov automatski rad. Potrebno je opisati rad stroja za punjenje boca i definirati zahtjeve na sustav automatskog upravljanja. Projektirani upravljački algoritam najprije simulirati u Matlab Stateflow-u i potvrditi njegovu ispravnost, a zatim ga implementirati u PLC (Siemens S7-300), koristeći TIA Portal v13. Ovaj sustav upravljanja treba nadograditi korisničkim sučeljem (HMI) realiziranim na operaterskom panelu Simatic TP700 Comfort.

U poglavlju 2. teorijski je obrađena korištena oprema (PLC, HMI, frekvencijski pretvarač, asinkroni motor i senzori), obrađeni su svi korišteni softverski alati i opisane su korištene web tehnologije. U poglavlju 3. je opisan stroj za punjenje boca. U poglavlju 4. je opisana provedena simulacija upravljačkog algoritma, izrada upravljačkog programa i implementacija u korišteni PLC, izrada korisničkog sučelja HMI-a i izrada web stranice. U poglavlju 5. se nalazi zaključak.

2. RASPOLOŽIVA OPREMA I KORIŠTENE TEHNOLOGIJE

U ovome poglavlju je detaljno opisana potrebna oprema za realizaciju sustava upravljanja stroja za punjenje boca. Opisan je dijagram tijeka koji je korišten za izradu upravljačkog algoritma te je opisan softverski alat u kojem je provedena simulacija upravljačkog algoritma. Također je opisan softverski alat za izradu upravljačkog programa PLC-a i web tehnologije koje su korištene za izradu web sučelja.

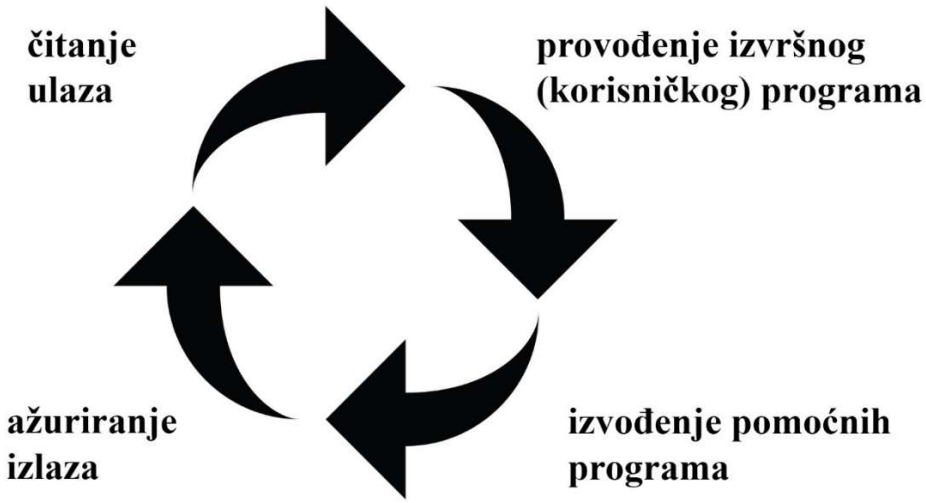
2.1 Oprema za realizaciju sustava upravljanja

Sustav upravljanja stroja za punjenje boca sastoji se od: programibilnog logičkog kontrolera (PLC), HMI panela, frekvencijskog pretvarača, asinkronog motora i senzora. U ovom potpoglavlju je detaljno opisan svaki od navedenih elemenata.

2.1.1 Programibilni logički kontroler (PLC)

Zbog specifičnih ograničenja u brzini i operativnom-informatičkom kapacitetu čovjeka (operatera), u slučaju njihova angažmana u vođenju složenih procesa, pojedine funkcije operatera se poboljšavaju ili zamjenjuju tehničkim uređajima. Ovaj pristup je najprije primijenjen u obradi podataka. Razlog tome je to što se u procese uvodi sve veći broj mjernih mjesta i što se sve više informacija treba obraditi. Primjena računala u vođenju procesa omogućava visok stupanj vizualizacije stanja i tijeka procesa, pri čemu se sinoptički paneli za komunikaciju operater-proces zamjenjuju monitorima [1].

Budući da procesno računalo treba obavljati operativne zadaće (zaštita, upravljanje, regulacije) to računalo mora biti osposobljeno za rad u stvarnom vremenu. Osnovni način rada procesnog računala je ciklično provođenje programa, kao što je prikazano na slici 2.1.



Slika 2.1. Osnovni način rada procesnog računala

Najrašireniji oblik procesnog računala je PLC (*Programmable Logic Controller*). Programibilni logički kontroleri su elektronički uređaji za automatizaciju s funkcijama upravljanja spremljenim, u obliku programa, u memoriji upravljačke jedinice. Prvi PLC-ovi su se pojavili sredinom 70-ih godina prošlog stoljeća. Koristili su se za sekvencijalno upravljanje i signalizaciju, kao zamjena za upravljačke ormare s relejima i sklopnicima. Osnovne prednosti PLC-a su pouzdanost, manje dimenzije i potrošnja, modularna struktura te mogućnost programiranja. Vrlo široku primjenu PLC-ova pratio je i njihov razvoj. Osim tehnološkog usavršavanja, posebno se razvijaju funkcije za regulaciju dinamičkih procesa i komunikacijske funkcije za povezivanje PLC-ova na različite sabirnice i mreže. Komunikacijske funkcije PLC-ova omogućuju njihovo povezivanje u mrežu radi ostvarenja distribuiranih sustava upravljanja složenim postrojenjima i procesima. Neke od komunikacijskih tehnologija su PROFIBUS i Profinet [1].

PROFIBUS (Process Field Bus) je otvoren, digitalni komunikacijski sustav sa širokim spektrom primjene, osobito u proizvodnoj i procesnoj automatizaciji. Prikladan je za brze, vremenski kritične i složene komunikacijske zadaće. ProfiBus se često koristi na najnižim razinama automatizacije, gdje se zahtjeva komunikacija u stvarnom vremenu. Radi se o digitalnoj, bidirekcionalnoj komunikacijskoj mreži sa serijskom sabirnicom koja se upotrebljava za povezivanje izoliranih uređaja kao što su kontroleri, senzori, aktuatori itd. Obično se kao fizički sloj koristi RS 485. Ova sabirnička tehnologija prijenosa je jednostavna i ekonomična i prvenstveno se koristi kad se zahtijeva visoka brzina prijenosa. Koristi se oklopljena upletena

parica. Brzine koje se mogu ostvariti su u području od 9.6 kbit/s do 12 Mbit/s. Maksimalna brzina prijenosa ovisi o duljini linije [2].

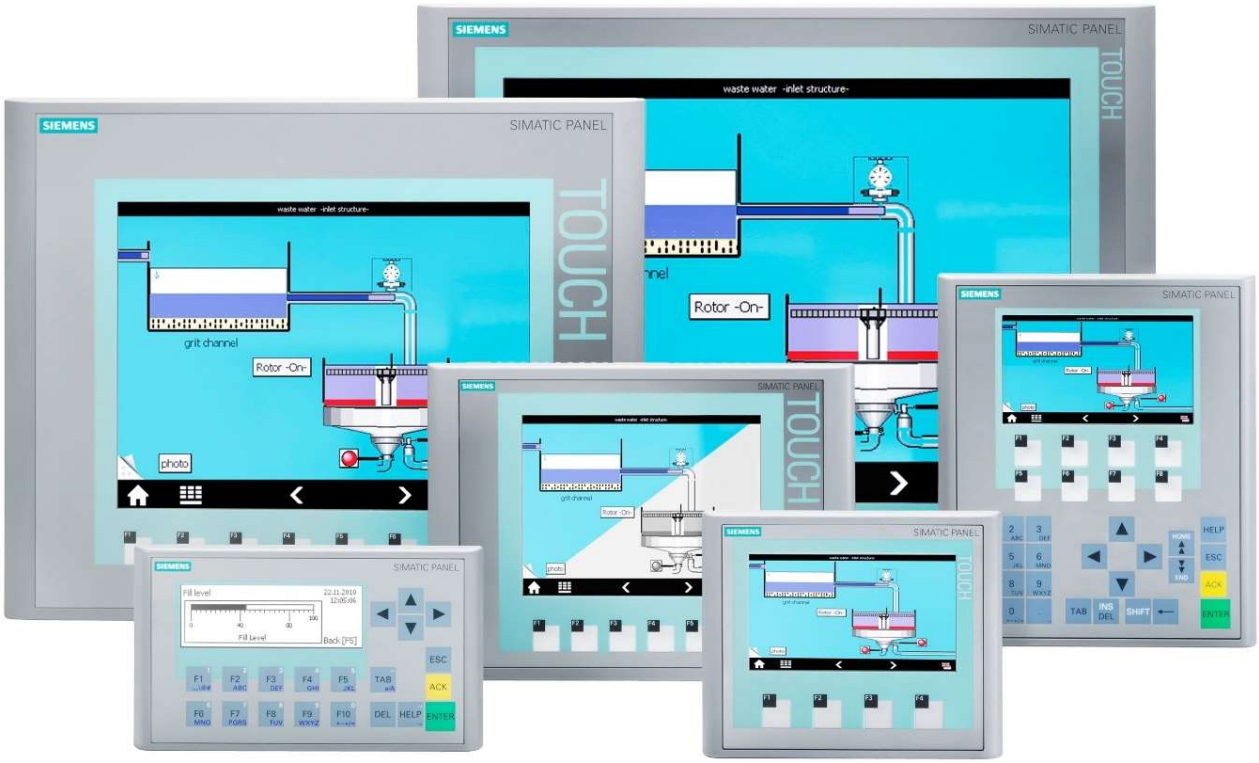
Tvrtka Siemens razvila je tehnologiju *Profinet* (eng. *PROcess FIeld NET*) koja se bazira na mrežnoj tehnologiji i na korištenju *Ethernet* standarda. *Profinet* koristi optimizirani komunikacijski kanal za komunikaciju u stvarnom vremenu, što jamči prijenos vremensko kritičnih podataka između različitih jedinica u mreži, unutar definiranog intervala. *Profinet* omogućuje integraciju različite upravljačke opreme, od razine postrojenja pa sve do razine nadzora i rukovođenja. Koncept *Profinet*-a zadovoljava gotovo sve zahtjeve industrijske automatizacije i može se koristiti za procesnu automatizaciju koja zahtjeva brzo vrijeme reakcije, tj. vremena manja od 100 ms [3].

PLC se sastoji od upravljačke jedinice i periferijskih ulazno/izlaznih (I/O) modula. Upravljačku jedinicu čine procesor i memorija. Dio memorije je nepromjenjivog sadržaja (ROM) i u njoj je pohranjen operacijski sustav koji određuje način upravljanja ulazima i izlazima, raspodjelom memorije te upravljanjem podacima. Korisnički program, kao i sve dinamičke varijable, spremaju se u podatkovnu memoriju (RAM). Slika 2.2 prikazuje različite PLC-ove [1].

Slika 2.2. Siemens Simatic serija PLC-ova

2.1.2 HMI paneli

HMI (Human Machine Interface) je sučelje čovjek-stroj koje operateru daje uvid u stanje procesa te omogućuje interakciju, ako za to postoji potreba. Za razliku od PLC-a koji je namijenjen za upravljanje u stvarnom vremenu te jamči brzo cikličko izvođenje programa, HMI služi samo za vizualizaciju i nadzor proizvodnih procesa pa je potreba za fiksnim i pouzdanim vremenom izvođenja manja. HMI panel omogućava prikaz elemenata korisničkog sučelja na ekranu osjetljivom na dodir te upotrebu viših programskih jezika, ali također i onemogućava rad u stvarnom vremenu. HMI panel putem komunikacijske veze od PLC-a periodički prima podatke koje prikazuje korisniku te šalje podatke koje korisnik unese. Budući da je frekvencija osvježavanja varijabli između PLC-a i HMI panela veličine sekunde i sporije, na HMI panel se ne treba oslanjati za izvođenje vremenski osjetljivih naredbi, naročito onih vezanih uz zaštitu. Nasuprot tome složene proračune vezane samo uz prikaz varijabli, analiza podataka (prikaz statistika), nije preporučljivo provoditi na PLC-u već se mogu izvesti na HMI panelu kako bi se rasteretio PLC. Na slici 2.3 su prikazane razne vrste HMI panela [4].



Slika 2.3. Različite vrste HMI panela

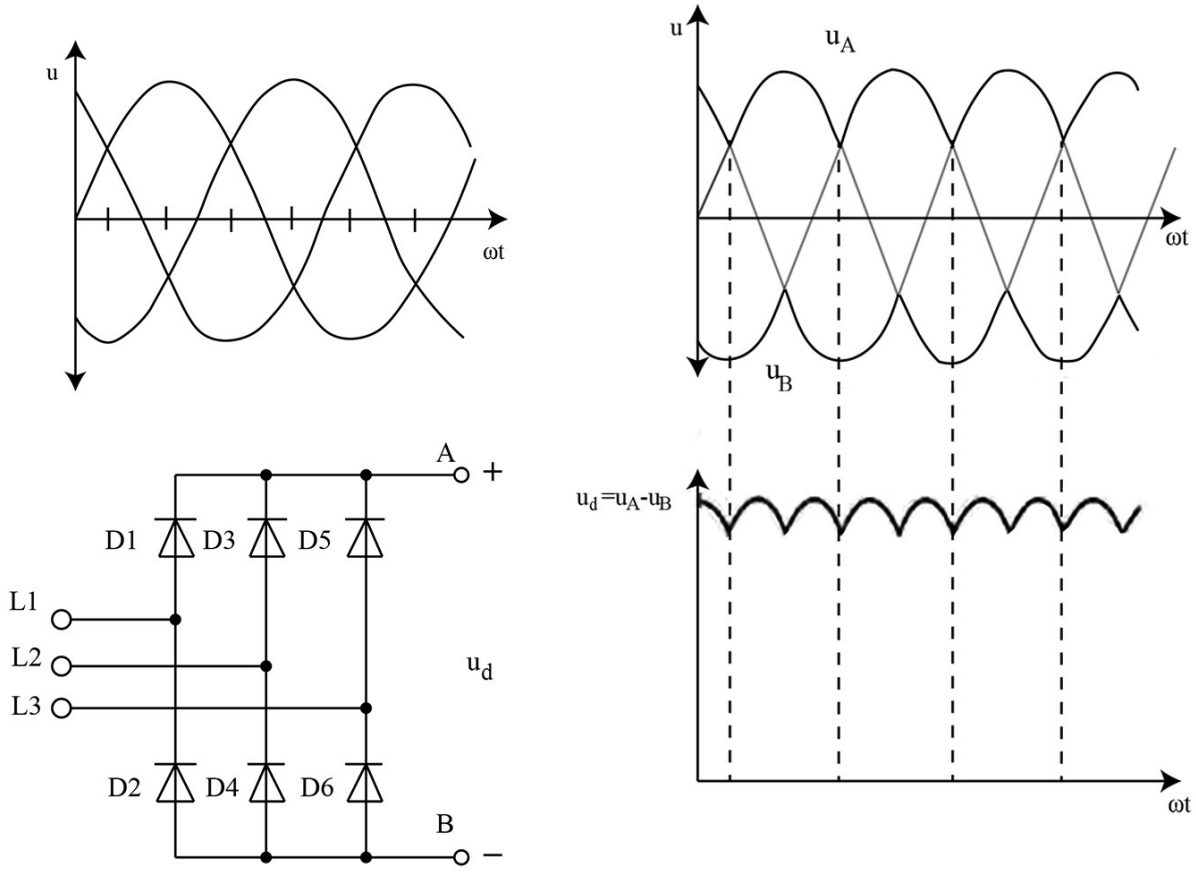
2.1.3 Frekvencijski pretvarač

Frekvencijski pretvarači su uređaji za kontinuiranu promjenu brzine vrtnje izmjeničnih elektromotora. Princip rada se zasniva na činjenici da je brzina vrtnje kaveznog asinkronog motora proporcionalna frekvenciji napona koji se dovodi na stezaljke motora [5]:

$$n = \frac{60 f}{p} \tag{2-1}$$

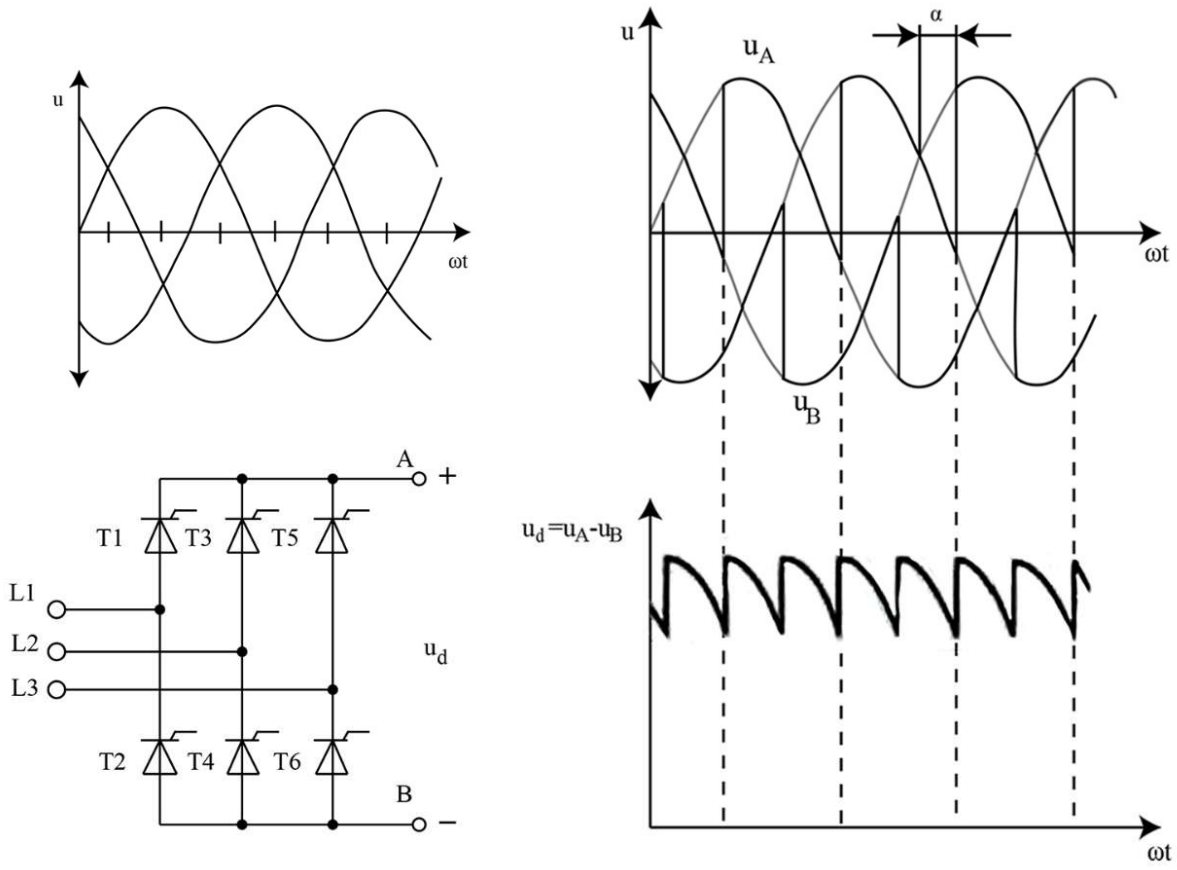
gdje su n - brzina vrtnje motora u min^{-1}, f - frekvencija napona u Hz, p - broj pari polova motora.

Frekvencijski pretvarač, u najužem smislu, se sastoji od tri cjeline: ispravljača, istosmjernog međukruga i izmjenjivača. Ispravljač može biti neupravljiv i upravljiv. Neupravljivi ispravljač čine diode u mosnom spoju koje ispravljaju napon mreže u istosmjerni. Na izlazu ispravljača izlazi pulsirajući istosmjerni napon. Amplituda napona koji se pojavljuje na izlazu ispravljača ovisi o amplitudi napona napajanja i o tome je li napajanje jednofazno ili trofazno. Na slici 2.4 prikazan je trofazni punovalni diodni ispravljač u mosnom spoju sa ulaznim i izlaznim naponom [6].



Slika 2.4. Punovalni neupravljivi ispravljač

Upravljivi ispravljač čine tiristori spojeni u mosnom spoju koji ispravljaju napon mreže u istosmjerni. Na izlazu ispravljača izlazi pulsirajući istosmjerni napon. Amplituda izlaznog napona iz ispravljača za razliku od diodnog ne ovisi samo o amplitudi napona napajanja nego i o kutu paljenja tiristora. Kut paljenja tiristora je trenutak u kojem će tiristor provesti. Na slici 2.5 prikazan je trofazni punovalni tiristorski ispravljač s ulaznim i izlaznim oblikom napona [6].

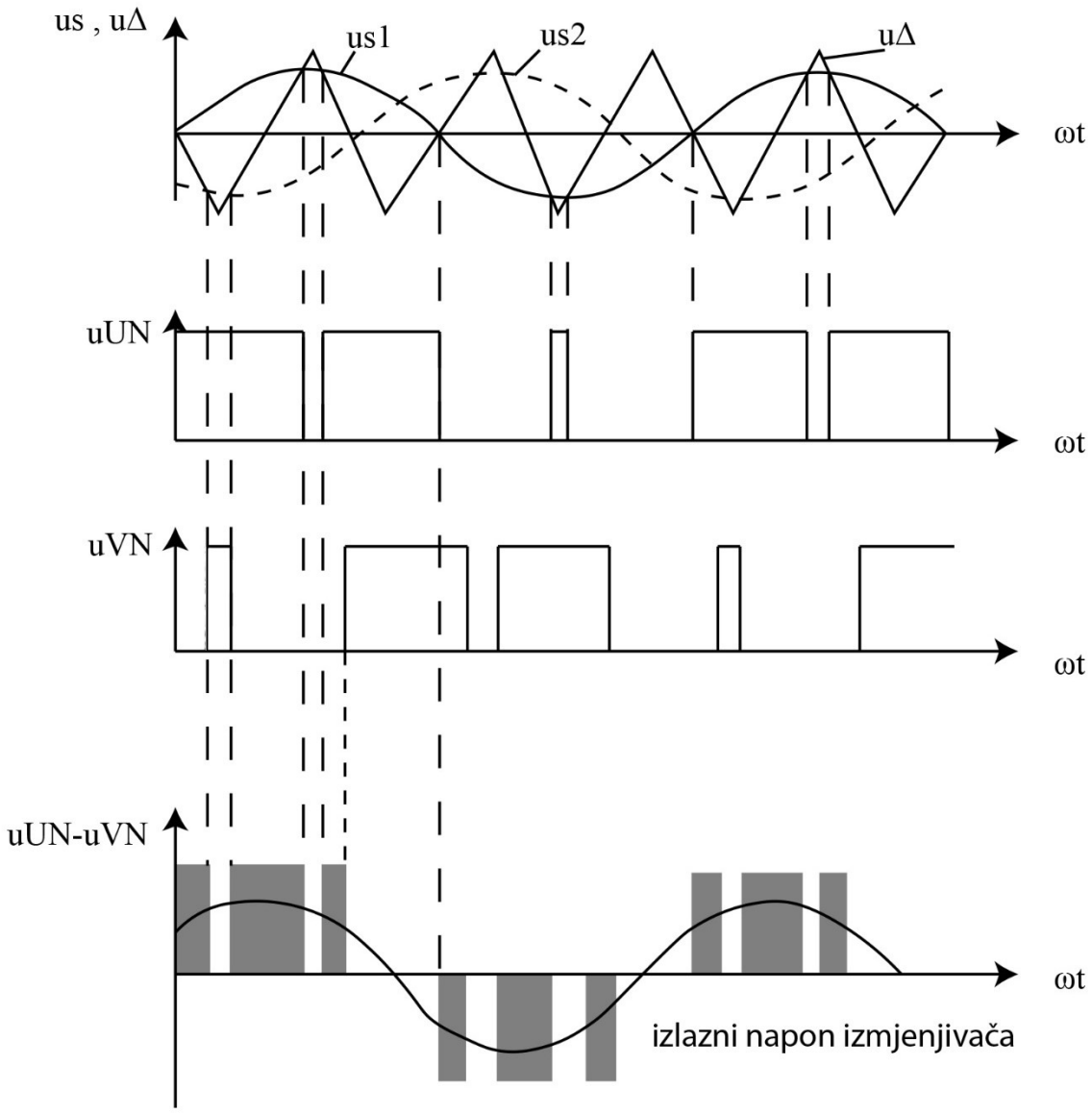


Slika 2.5. Punovalni neupravljivi ispravljač

Istosmjerni međukrug služi za pohranu električne energije. Iz njega motor uzima energiju preko izmjenjivača. Ovisno o rješenju ispravljača i izmjenjivača dva su moguća rješenja za istosmjerni međukrug: strujni istosmjerni međukrug i naponski istosmjerni međukrug. Frekvencijski pretvarač s strujnim međukrugom ima u istosmjerni međukrug ugrađenu zavojnicu velikog induktiviteta, a ispravljač je u tom slučaju uvijek upravljiv. Frekvencijski pretvarači s naponskim istosmjernim međukrugom imaju u istosmjernom međukrugu niskopropusni filtar koji tvore zavojnica i kondenzator, a ispravljač može biti upravljiv ili neupravljiv. U slučaju neupravljivog ispravljača napon na izlazu istosmjernog međukruga je konstantan, a kod upravljivog ispravljača moguće je regulirati napon na izlazu iz istosmjernog međukruga. Istosmjerni međukrug osim funkcije filtriranja napona ima i ulogu odvajanja izmjenjivača od ispravljača te apsorbira strujne udare nastale prilikom pokretanja motora tako da skladišti energiju u prigušnici i kondenzatoru [6].

Gledajući tok energije kroz pretvarač, izmjenjivač je posljednji učinski sklop frekvencijskog pretvarača. On završno prilagođava izlazni napon i frekvenciju frekvencijskog pretvarača. Ako su struja i/ili napon istosmjernog međukruga promjenjivi, izmjenjivač određuje samo frekvenciju izlaznog napona. Ako je napon istosmjernog međukruga konstantan tada izmjenjivač određuje amplitudu i frekvenciju izlaznog napona. Glavni dio izmjenjivača su poluvodički upravljivi ventili koji preklapaju istosmjerni napon iz istosmjernog međukruga u izmjenični napon, za to se koristi PWM (pulsno-širinska modulacija) [6].

PWM je najraširenija metoda za dobivanje sinusnih valnih oblika napona. Srednja vrijednost signala mijenja se tako da se mijenja širina impulsa. Sinusna modulacija zasniva se na sinusnom i trokutastom naponu. Na sjecištima sinusnog i trokutastog signala sklopke izmjenjivača uklapaju ili isklapaju. Sjecišta sinusnog i trokutastog signala detektiraju se tako da se ti signali dovode na ulaz komparatora koji uspoređuje signale. Kada je amplituda sinusnog signala viša od amplitude trokutastog signala tada poluvodička sklopka vodi, a kada je niža tada ne vodi. Na slici 2.6 prikazan je način stvaranja dva PWM fazna napona i linijskog napona između ta dva fazna napona [6].



Slika 2.6. Sinusna modulacija na temelju dva sinusna signala

Današnji frekvencijski pretvarači su složeni uređaji koji sa svojim naprednim funkcijama čine središte reguliranog elektromotornog pogona. Frekvencijski pretvarači omogućuju mjerenje varijabli i dijagnostiku, zaštitu, nadzor, upravljanje i regulaciju elektromotornog pogona, odnosno procesnih veličina. Današnji frekvencijski pretvarači zasnovani su na računalu, koje upravlja radom energetskog dijela pretvarača, izvodi nužne upravljačke funkcije (npr. upravljanje momentom i brzinom vrtnje motora), kao i složeno sučelje s korisnikom [5].

2.1.4 Asinkroni motor

Princip rada asinkronog motora zasniva se na okretnom magnetskom polju. Uvjet za dobivanje okretnog magnetskog polje je da postoje barem dva prostorno pomaknuta namota kroz koje teku fazno pomaknute struje. Trofazni asinkroni motor ima tri prostorno pomaknuta namota (faze) koji se priključuju na trofazni sustav napona koji potjera tri fazno pomaknute struje. Okretno magnetsko polje se vrti sinkronom brzinom n_s koja je proporcionalna frekvenciji f, a obrnuto proporcionalna broju pari polova p statorskog namota i računa se prema relaciji (2-1).

Kod uključivanja motora na napon rotor stoji. Silnice okretnog magnetskog polja presijecaju vodiče rotora u kojima se zbog toga inducira napon, koji kroz njih obzirom da su kratko spojeni (kavezni rotor) potjera struju. Kako na vodiče rotora kroz koje teče struja, a nalaze se u magnetskom polju (statora), djeluje sila, javlja se moment koji pokreče rotor. Asinkroni motor se ne može okretati sinkronom brzinom jer bi se kod sinkrone brzine rotorski vodiči i silnice okretnog magnetskog polja okretali istom brzinom, odnosno jedno naspram drugoga bi mirovali. Drugim riječima, ne bi bilo presijecanja vodiča rotora od strane silnica okretnog magnetskog polja statora, time ni induciranog napona, kao ni struje na rotoru, a prema tome ni elektromagnetske sile. Relativna razlika između brzine rotora i brzine okretnog magnetskog polja (sinkrone brzine) izraženo je postotkom, naziva se klizanje s [7]:

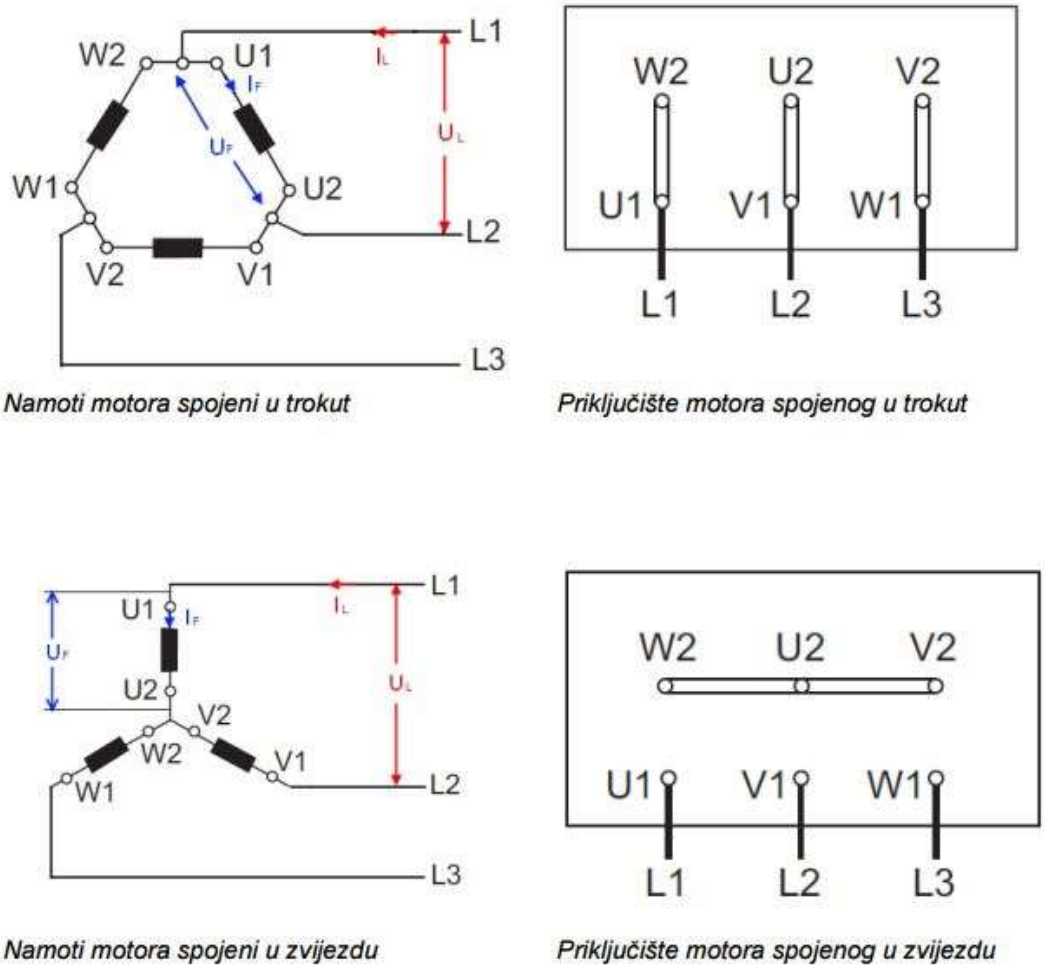
$$s = \frac{n_s - n}{n_s} \tag{2-2}$$

Klizanje pokazuje relativno zaostajanje rotora za statorskim okretnim magnetskim poljem. Kod nazivnog opterećenja iznosi 3-5%. Što je motor opterećeniji radi s većim klizanjem. Klizanje u praznom hodu (kad na osovini motora nema priključenog mehaničkog tereta) je vrlo malo. Što su motori veći (snažniji) to je nazivno klizanje manje.

Smjer vrtnje asinkronog motora se može promijeniti tako da se pri priključku statorskih namota zamijeni redoslijed faza (u priključnoj kutiji zamijene se bilo koja dva vodiča priključnog kabela, ne računajući uzemljenje) [7].

Namoti motora mogu biti spojeni u zvijezdu i u trokut. U spoju zvijezda krajevi namota su spojeni u jednu točku (zvjezdište), a na početke se dovodi trofazni napon. Zvjezdište je na potencijalu 0 V pa svaki namot dobiva faznu vrijednost napona (230 V kod standardne trofazne mreže). Kroz namot teče fazna struja definirana faznim naponom i impedancijom namota, a to je

ujedno i struja koja teče kroz vod, odnosno linijska struja. U spoju trokut namoti su spojeni serijski (kraj prvog s početkom drugog; kraj drugog s početkom trećeg i kraj trećeg s početkom prvog), a na spojne točke dovodi se trofazni napon. Svaki namot spojen je između dvije faze mreže, dakle na linijsku vrijednost napona (400 V kod standardne mreže). Kroz namot teče fazna struja definirana linijskim naponom i impedancijom namota. Struja koja teče kroz vod (linijska struja) je 1,73 puta veća od fazne struje. Shema spojeva namota je prikazana na slici 2.7. [7].



Slika 2.7. Spojevi asinkronog motora

2.1.5 Senzori

Senzor je uređaj koji mjeri fizikalnu veličinu (temperatura, vlažnost zraka, brzina okretaja…) i pretvara je u signal pogodan za daljnju obradu. Senzori mogu na izlazu davati analogni signal

(vrijednost signala može biti u nekom opsegu vrijednosti, npr. od 0 do 10 V) ili digitalni signal (vrijednost može biti ili logička jedinica ili logička nula). Na slici 2.8 su prikazane različite vrste senzora [8].

Slika 2.8. Senzori

Podjela senzora:

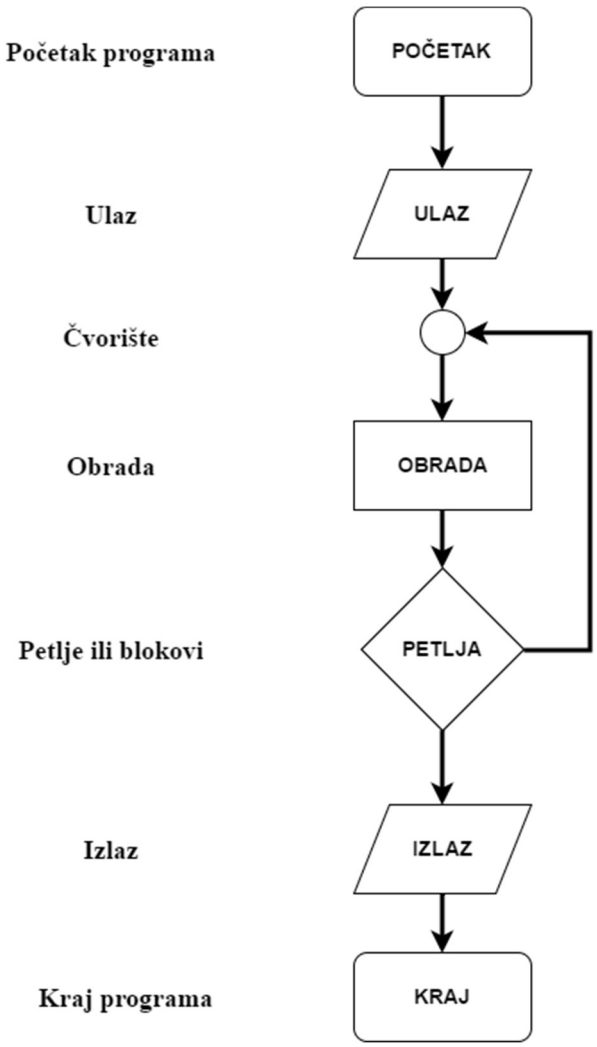
1) Kapacitivni
2) Induktivni
3) Fotoelektrični
4) Ultrazvučni
5) Optički
6) Temperaturni
7) Senzori pritiska
8) Senzori težine
9) Senzori nivoa

2.2 Dijagram tijeka

Stroj za punjenje boca se promatra kao sustav vođen diskretnim događajima. Sustavi koji su vođeni diskretnim događajima, za razliku od sustava vođenih vremenom, imaju ograničen i prebrojiv broj mogućih stanja. Kod sustava vođenih vremenom u svakom trenutku poznato je stanje neke operacije, dok se kod diskretnih sustava prati je li određena operacija započela i je li izvršena [9]. Diskretni procesi se susreću u prirodi i ljudskim aktivnostima. Primjer diskretnog procesa je bankovno ili financijsko poslovanje gdje se obračun novca provodi na kraju nekog vremenskog intervala (dnevno, kvartalno) [10].

Postoji niz postupaka i metoda za analizu diskretnih sustava, dijagram tijeka, pseudokod te petrijeve mreže. U ovom radu se koristi dijagram tijeka. Dijagram tijeka je simbolički algoritam, sastoji se od niza simbola povezanih strelicama koje definiraju tijek i smjer procesa [11].

Dijagram tijeka je grafički prikaz algoritma. Takav način zapisivanja ima nekoliko prednosti nad pseudokodom, koji je tekstualni zapis algoritma. Zapisivanje se provodi međunarodno dogovorenim simbolima i ne ovisi o govornom jeziku osobe koja piše algoritam. Grafički prikaz je jednostavan, pregledan i jednostavno se pronalaze pogreške. Slika 2.9 prikazuje osnovne blokove koji se koriste u dijagramima tijeka [12].



Slika 2.9. Osnovni blokovi dijagrama tijeka

Postoje tri tipa algoritamske strukture: linijska (sekvenca), razgranata (selekcija) i ciklička (petlja). Kod linijskih algoritamskih struktura moguće je samo jedno izvršavanje nekog algoritamskog koraka, tj. nakon izvršavanja jednog algoritamskog koraka izvršavanje se može prenijeti samo na algoritamski korak koji još nije izvršen.

Često je u algoritmima potrebno odlučiti koji od dva ili više algoritamskih koraka treba izvršiti s obzirom na postavljeni uvjet. Jednostavni oblik razgranate algoritamske strukture je izbor jednog od dva algoritamska koraka, ovisno o tome je li postavljeni uvjet istinit. To je poznata „*IF-THEN-ELSE*“ naredba u većini programskih jezika.

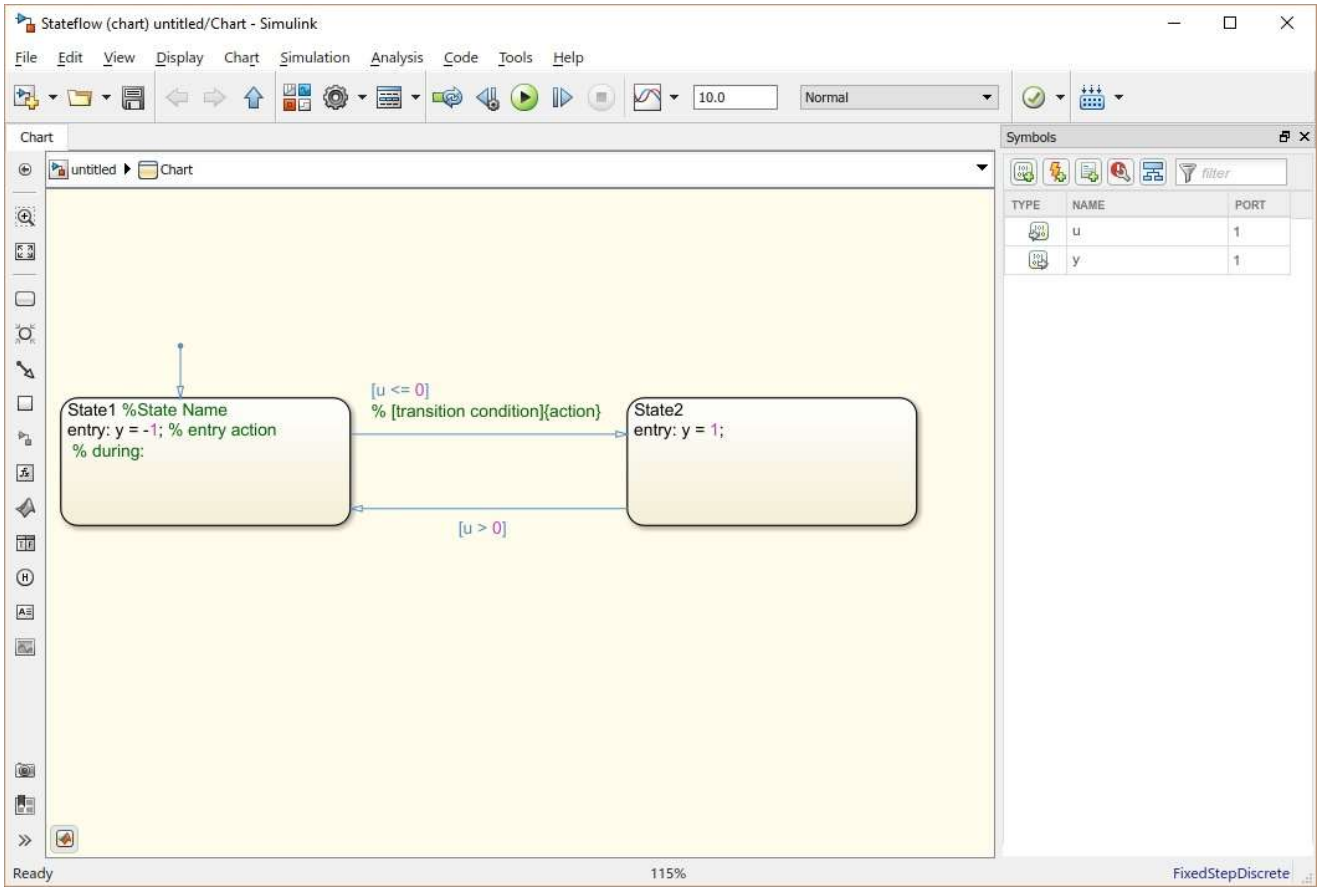
Dio algoritma koji zahtjeva višestruko izvršavanje jednog ili više algoritamskih koraka, pri čemu može doći i do promjene zadanog načina obrade podataka, naziva se petlja. To se ponavljanje

provodi sve dok je neki postavljeni uvjet ispunjen (okončava kad uvjet više nije ispunjen - kad postane neistinit) ili sve dok je određeni uvjet neispunjen (okončava kada je uvjet ispunjen - kad postane istinit) [13].

2.3 Programski alati za simulaciju

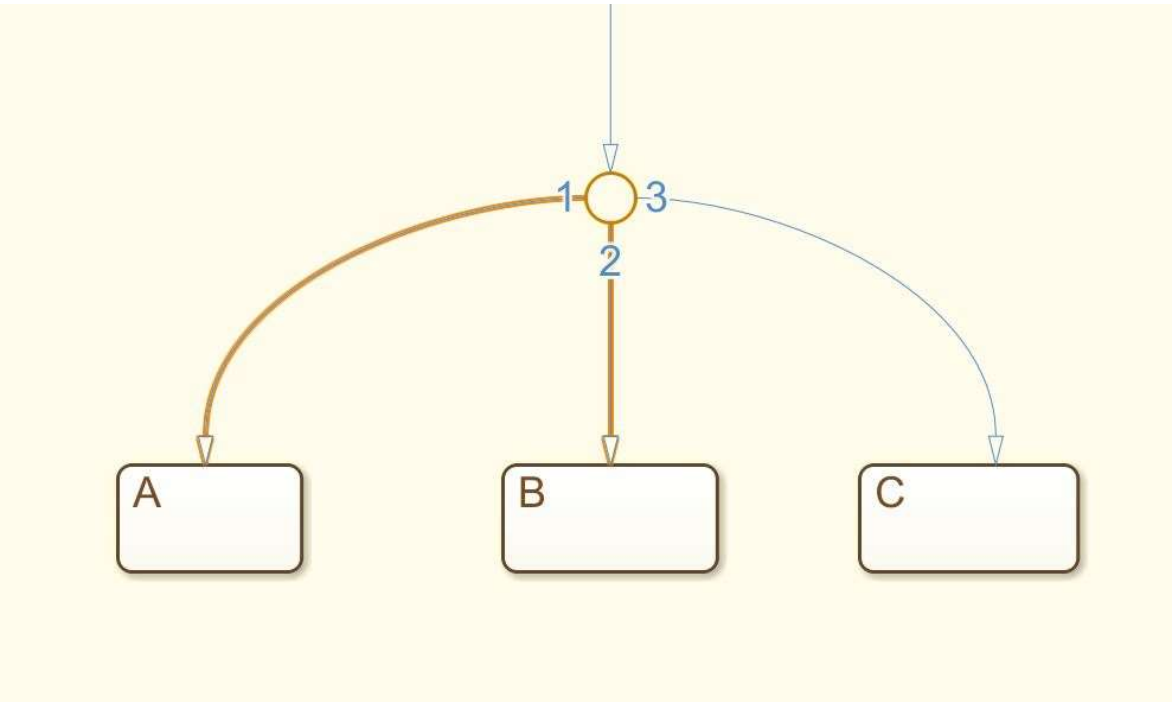
MATLAB je programski jezik visoke razine i interaktivna je okolina za numeričko i matrično računanje te za vizualizaciju i programiranje. Naziv je nastao kao kratica od engleske riječi *MATrix LABoratory*. Prva verzija je napisana krajem 1970. godine. Pomoću MATLAB-a se mogu analizirati podaci, izgraditi algoritmi te stvoriti modeli i aplikacije. Koristi se za niz aplikacija, uključujući obradu signala i komunikacija, obradu sustava, ispitivanja i mjerenja, računalnih financija i računalne biologije [14].

SIMULINK predstavlja grafički alat koji koristi matematičku jezgru MATLAB-a kako bi se provele simulacije sustava. Izrada simulacijskih modela unutar SIMULINK-a obavlja se na jednostavan način korištenjem gotovih grafičkih blokova. Osim postojećih blokova korisnik može napisati i vlastite blokove koristeći m-funkcije ili C/C++ programski jezik [15].



Slika 2.10. Sučelje Stateflow-a

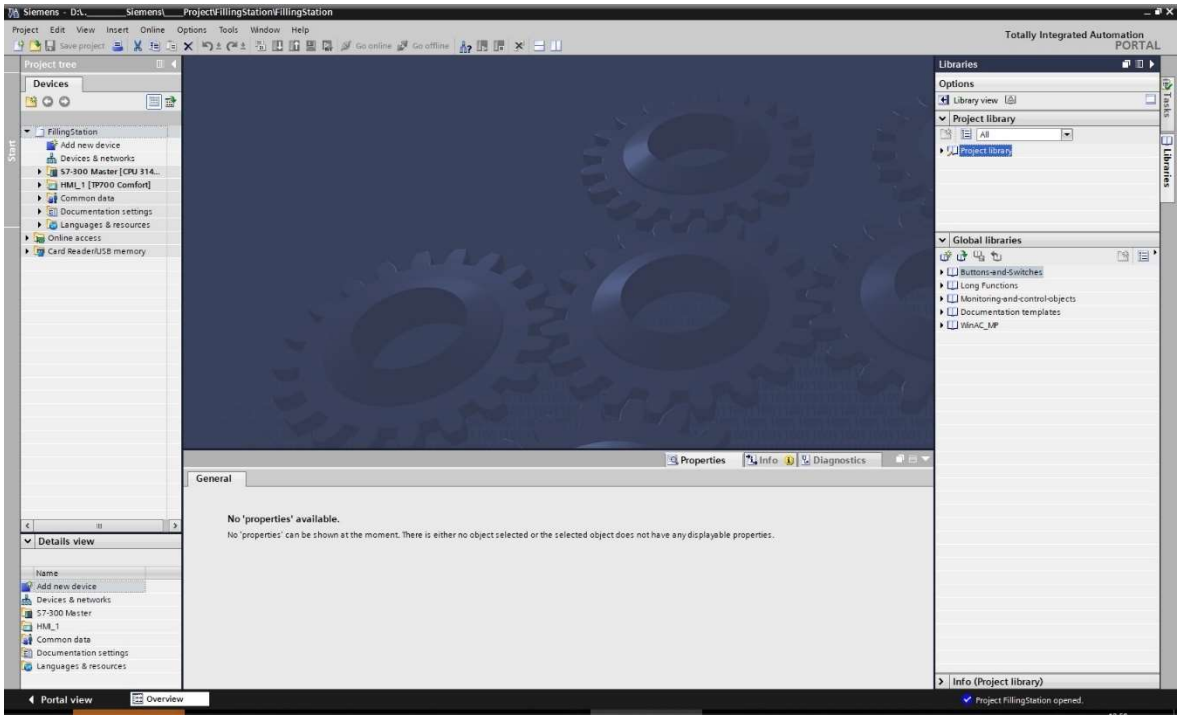
Stateflow je softverski alat unutar SIMULINK-a koji služi za modeliranje i simulaciju sustava. Koristi se za izradu dijagrama tijeka te omogućuje kombiniranje grafičkog i tabličnog prikaza. Na slici 2.10 je prikazano sučelje softverskog alata. Izrada dijagrama tijeka se provodi dodavanjem blokova te njihovim povezivanjem. Nakon svakog bloka je potrebno dodati uvjet, kad je uvjet zadovoljen prelazi se u sljedeći blok. Uvjeti mogu biti vremenski (npr. nakon 10 s program prelazi u sljedeći blok) ili logički (npr. ako je „x“ veći od 5 program prelazi u sljedeći blok). Također postoje i grananja, linija se grana na više linija od kojih svaka linija ima određeni prioritet koji je označen brojem uz liniju (linija s brojem jedan ima najviši prioritet), primjer je prikazan na slici 2.11. U slučaju da uvjet za prelazak u blok A nije zadovoljen prelazi se u blok B, ako je uvjet zadovoljen, a ako nije provjerava se uvjet za blok C. U slučaju da niti jedan od tri uvjeta nisi zadovoljena program ostaje u beskonačnoj petlji. Da bi se to spriječilo poželjno je dodati četvrti blok u koji program može ući bez uvjeta, u tom slučaju četvrti blok služi kao pokazivač na grešku [16].



Slika 2.11. Prikaz grananja u Stateflow-u

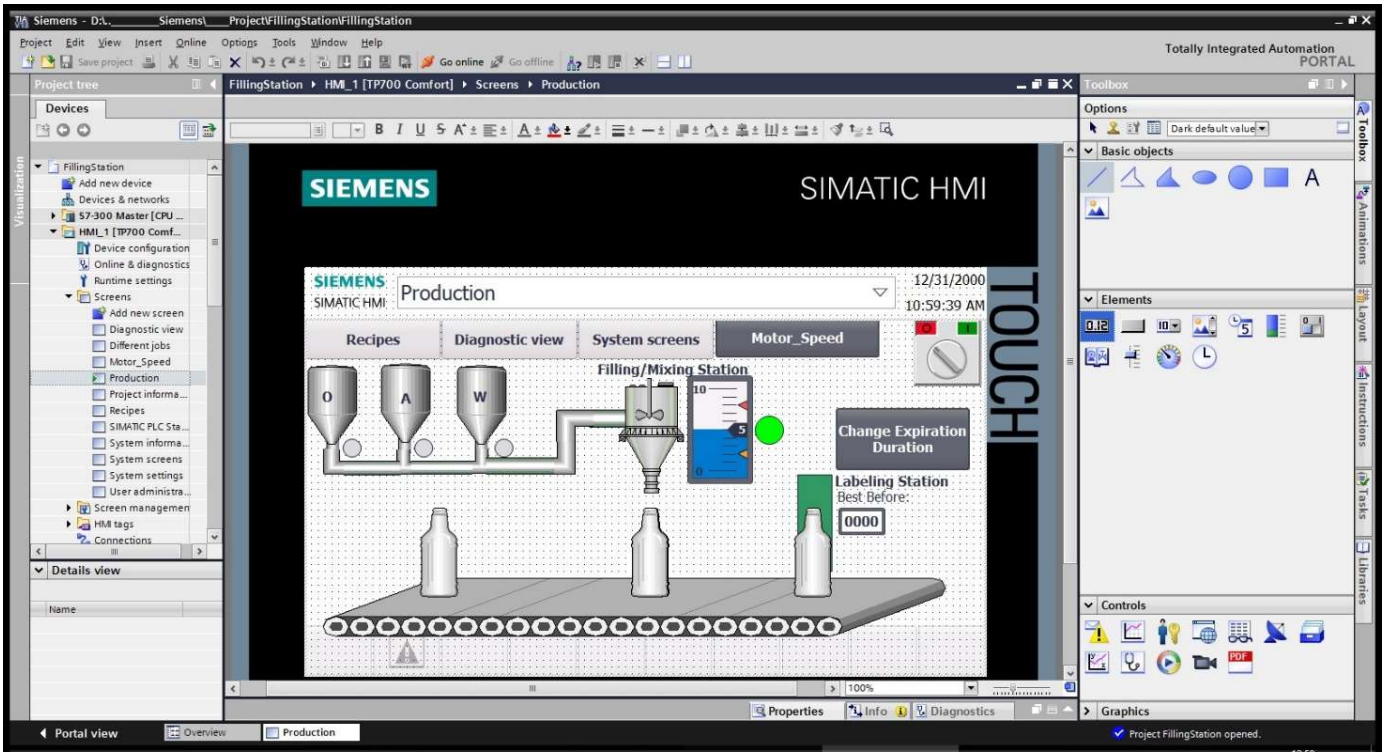
2.4 Programski alati za izradu upravljačkog programa PLC-a

Programsko okruženje koje je korišteno za programiranje PLC-a i HMI-a je Siemens-ov TIA Portal v13. Totally Integrated Automation Portal (TIA Portal) objedinjuje SIMATIC Totally Integrated Automation (TIA) proizvode u jedinstvenu programsku aplikaciju. Svi TIA proizvodi rade zajedno unutar istog programskog okruženja i pružaju podršku korisniku u svim segmentima potrebnim za stvaranje rješenja automatizacije. TIA Portal se koristi za konfiguraciju i programiranje PLC uređaja, ali i za vizualizaciju procesa u jedinstvenoj razvojnoj okolini. Svi podaci se spremaju u zajedničku projektnu datoteku, koristi se zajedničko korisničko sučelje za sve zadaće preko kojeg je moguće pristupiti svim programskim i vizualizacijskim funkcijama u svakom trenutku. Na slici 2.12 je prikazano sučelje TIA Portal-a [17].



Slika 2.12. Sučelje TIA Portal-a

Programski alat WinCC, koji se nalazi u sklopu TIA Portal-a, služi za razvoj i izradu nadzorno-upravljačkih aplikacija za HMI uređaje tvrtke Siemens. Slika 2.13 prikazuje sučelje programskog alata WinCC. Sa slike se može vidjeti da postoje već gotovi grafički elementi koji se samo povuku (*drag and drop*) u glavni ekran i poslože prema želji.



Slika 2.13. Prikaz sučelja WinCC-a

2.4.1 Struktura upravljačkog programa

Upravljački program je organiziran kroz programske blokove koji zajedno tvore programsku cjelinu. Prednosti ovakvog strukturnog načina programiranja su jednostavna organizacija programa, jednostavna izmjena i testiranje koda. Razlikujemo sljedeće blokove [18]:

- **OB**, Organizacijski blokovi (Organization Blocks): obavljaju temeljne funkcije kao što su cikličko izvršavanje programa
- **FB**, Funkcijski blokovi (Function Blocks): to su potprogrami koji se pozivaju iz organizacijskih blokova ili iz drugih funkcijskih blokova
- **SFB**, Sistemski funkcijski blokovi (System Function Bocks): gotovi blokovi napisani od strane proizvođača koji izvode funkcije koje se često koriste u praksi
- **DB**, Podatkovni blok (Dana Block): predstavlja statičku memoriju

2.4.2 Vrste programskih jezika u TIA Portal-u

Programski jezici koriste se za pisanje korisničkih programa. Korisniku je na raspolaganju nekoliko programskih jezika koje bira ovisno o svojim potrebama. TIA Portal podržava LAD, FBD, STL, SCL i GRAPH programske jezike koji su prikazani na slici 2.14.

LAD (*ladder* logika) je po izgledu sličan relejnim shemama. Upravljački program je organiziran kroz programske blokove koji zajedno tvore programsku cjelinu. Prednosti ovakvog strukturnog načina programiranja su jednostavna organizacija programa, jednostavna izmjena i testiranje koda te po potrebi višestruka upotreba pojedinog bloka [19].

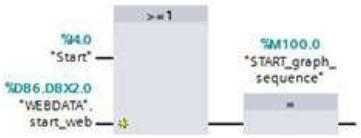
FBD (funkcijski blokovski dijagram) prikazuje sve veze pomoću blokova (logičkih sklopova). Ulazi blokova pridruženi su varijablama čija se stanja i vrijednosti koriste u daljnjem povezivanju signala. Rezultat povezivanja na izlazu bloka pridružuje se nekoj varijabli ili se povezuje s ulazom drugog bloka [19].

STL (lista naredbi) sastoji se od pojedinačnih naredbi navedenih redak po redak. U STL-u su implementirane logičke operacije koje obrađuju stanje signala i vrijednost adresa i varijabli. Osim povezivanja binarnih i digitalnih varijabli mogu se mijenjati sadržaji kompleksnih varijabli, a adrese i varijable se mogu indirektno adresirati [19].

SCL (strukturni upravljački jezik) tekstualni je programski jezik za programiranje složenih algoritama i upravljanje velikom količinom strukturiranih podataka [19].

GRAPH je metoda programiranja koja se koristi za upravljanje slijednim procesima. Program se izvodi korak po korak, a u svakom koraku se izvodi određen broj naredbi. Može se odabrati da se uvjet za prelazak u sljedeći korak prikaže u LAD-i ili FBD-u. GRAPH omogućuje paralelno grananje te time proširuje mogućnost linearnog izvršavanja uzastopnih koraka [19].

LAD



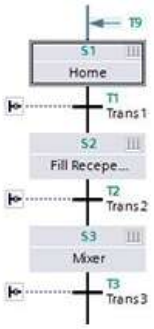
FBD

```
Network 2 : Title:
          LAR1  P##Date_Time
          L     B [AR1,P#0.0]
          T     "Year"
          L     B [AR1,P#1.0]
          T     "Month"
          L     B [AR1,P#2.0]
          T     "Day"
          L     B [AR1,P#3.0]
          T     "Hour"
          L     B [AR1,P#4.0]
          T     "Minutes"
          L     B [AR1,P#5.0]
          T     "Seconds"
```

STL

```
1  #Error := RD_SYS_T(OUT => #System_Time_DB);
2  #Year := BCD16_TO_INT(#System_Time_Array[0]);
3  #Best_before_date_output :=2000 + #Year + #Duration;
4
```

SCL



GRAPH

Slika 2.14. Prikaz različitih programskih jezika

2.5 Web tehnologije

World Wide Web, ili jednostavno web, je klijentsko-serverski model razmjene informacija izgrađen na infrastrukturi interneta te na HTTP protokolu za prijenos podataka. Korisnici web-a pristupaju informacijama putem web preglednika (browser-a) - klijentske aplikacije namijenjene preuzimanju i prezentaciji web dokumenata sa servera, navigaciji među dokumentima te slanju povratnih informacija na server. Neke od web tehnologija su: HTML, JavaScript, CSS (Cascading Style Sheets)… [20].

2.5.1 HTML

HTML dolazi od engleskih riječi HyperText Markup Language. Hypertext je dio teksta koji ima ulogu poveznice (linka), a Markup Language označava način unosa izgleda informacija unutar dokumenta. HTML dokument sadrži oznake elemenata i običan tekst i opisuje web stranicu. HTML dokument naziva se i web stranicom. HTML svojstva koristi se kako bi se promijenio sadržaj HTML elementa a nalaze se unutar početne oznake elementa [21].

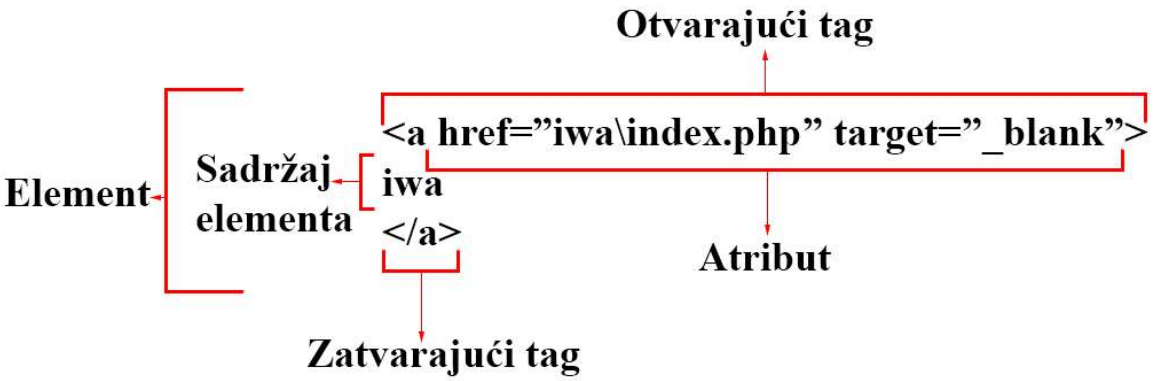
HTML dokument je tekstualna datoteka te ju je moguće kreirati u bilo kojem tekst editoru. Osim tekstualnog sadržaja dokumenta, u HTML datoteci koriste se specijalne sekvence karaktera kojima se opisuje struktura i značenje pojedinih dijelova dokumenta. Tim označnim sekvencama, koji se nazivaju tagovi, omeđuju se dijelovi sadržaja dokumenta čineći tako HTML elemente [20].

Osnovna struktura HTML dokumenta je jednostavna: započinje tagom deklaracije vrste dokumenta <!DOCTYPE>. Iza deklaracije slijedi <html> tag unutar kojeg je smješten cijeli dokument, a završava sa zatvarajućom oznakom </html>. Dokument se sastoji od dva dijela: <head> sekcije, koja sadrži opće, nevizualne informacije, iza koje slijedi <body> sekcija s vizualnim komponentama dokumenta [20].

Elementi HTML stranice ne zahtijevaju razdvajanje nekim posebnim karakterom ili novim retkom. Svaki tag ima definiranu oznaku za svoj početak i kraj pa je sasvim moguće da se datoteka koja definira sadržaj i izgled kompletne web stranice sastoji od jedne dugačke linije, bez ijednog karaktera razmaka. Naravno, u praksi je dobro težiti što boljoj čitkosti koda pa je preporuka, naročito za dulje tagove, izvorni kod stranice vizualno formatirati prateći model standardnog formatiranja blokova kod klasičnih proceduralnih programskih jezika.

Osnovna strukturna jedinica HTML dokumenta jest HTML element, koji se sintaksno opisuje na sljedeći način:

- počinje otvarajućim tagom (opening tag)
- završava zatvarajućim tagom (closing tag)
- sadržaj elementa je sve što se nalazi između otvarajućeg i zatvarajućeg taga
- neki elementi imaju prazan sadržaj
- prazni elementi (bez sadržaja) zatvoreni su u otvarajućem tagu
- većina HTML elemenata može imati atribute koji dodatno opisuju specifičnosti pojedinih tagova

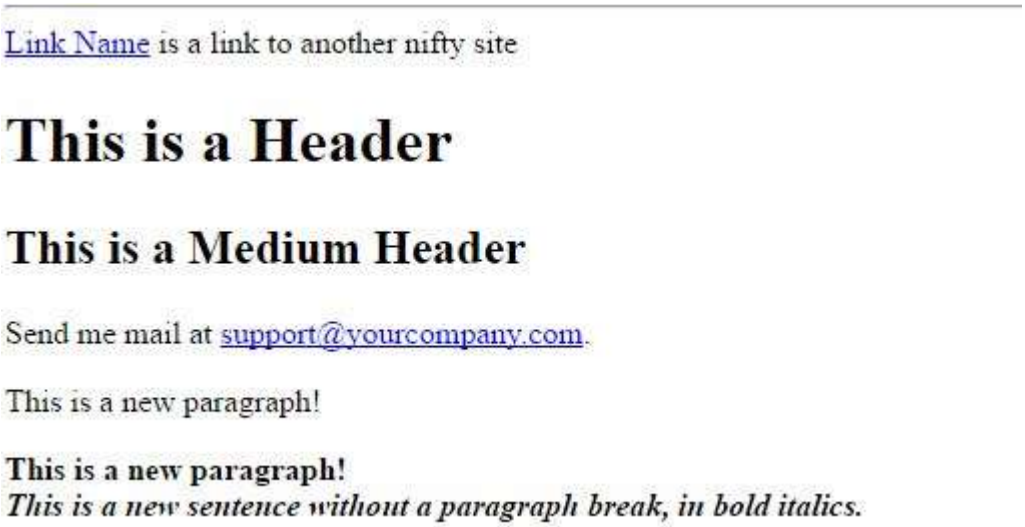


Slika 2.15. Sastavni dijelovi HTML elementa

Slika 2.15 prikazuje sastavne dijelove HTML elementa ili taga. Slika 2.16 prikazuje programski kod jednostavne web stranice, a na slici 2.17 je prikazan sadržaj te stranice kad je otvorena pomoću web preglednika [20].

```
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>Naslov</TITLE>
</HEAD>
<BODY BGCOLOR="FFFFFF">
<CENTER><IMG SRC="clouds.jpg" ALIGN="BOTTOM"> </CENTER>
<HR>
<a href="http://somegreatsite.com">Link Name</a>
is a link to another nifty site
<H1>This is a Header</H1>
<H2>This is a Medium Header</H2>
Send me mail at <a href="mailto:support@yourcompany.com">
support@yourcompany.com</a>.
<P> This is a new paragraph!
<P> <B>This is a new paragraph!</B>
<BR> <B><I>This is a new sentence without a paragraph break, in bold italics.</I></B>
<HR>
</BODY>
</HTML>
```

Slika 2.16. Programski kod jednostavne web stranice



Slika 2.17. Jednostavna web stranica

2.5.2 JavaScript

Povijest JavaScripta seže u 1992. godinu, kada je kompanija Nombas počela s razvojem novog *embedded* skriptnog jezika kojeg su nazvali C-minus-minus (Cmm). Motiv za definiranjem i razvojem potpuno novog skriptnog jezika bio je jednostavan, potreba za skriptnim jezikom dovoljno moćnim da zamijeni makro-jezike koji su osim često ograničenih mogućnosti bili i potpuno nekompatibilni među operacijskim sustavima. Također, poželjna je bila sličnost s nekim od postojećih, rasprostranjenih programskih jezika kako bi se izbjegla potreba za dugotrajnim procesom učenja nove sintakse i interne logike novog alata od strane programera. Kao logičan odabir modela, a naročito sintakse nametnuo se C i C++ zbog široke upotrebe na svim tadašnjim operacijskim sustavima. Naravno, obzirom da C i C++ nemaju mogućnost interpretativnog izvršavanja, u glavnini je preuzeta samo sintaksa. Interna struktura i logika programskog jezika promijenjena je na način da slijedi samo osnovu navedenih jezika koji su poslužili kao modeli, dok je glavnina prilagođena skriptnom izvršavanju [20].

Kako je korištenje interneta dobivalo na popularnosti, postepeno je rasla i potreba za standardnim *client-side* skriptnim jezikom, u najvećoj mjeri zbog jednostavne ali krucijalne potrebe za validacijom web-formi. Bez skriptnog jezika unutar web browser-a, validnost svake forme koju bi korisnik popunio morala bi se provjeriti na serverskoj strani. To znači da bi korisnik nakon što je popunio formu i kliknuo na *submit button*, bilo potreno sljedeće:

- slanje podataka iz forme na server
- provjera valjanosti unesenih podataka
- u slučaju neispravnosti nekog od unesenih podataka, ponovno učitavanje stranice s obavijesti koja od polja u formi nisu korektno popunjena
- ponovno slanje forme na server, itd.

Uzme li se u obzir činjenica da je u to vrijeme najveći broj korisnika pristupao internetu preko modema tipičnom brzinom 28.8 kbps te da je i serverski potencijal tada bio bitno manji nego danas, za samo jedan *round-trip* prilikom slanja forme bilo je potrebno 30-ak sekundi i to za običnu validaciju web-forme. S korisničkog aspekta potpuno je neprihvatljivo čekanje od pola minute da bi, primjerice, za željeni izvještaj dobio obavijest kako neko od polja nije popunjeno, ili kako uneseni datum početka perioda mora biti manji od datuma završetka perioda [20].

Tablica 2.1. Ključne riječi JavaScript-a

abstract	debugger	final	instanceof	public	transient	boolean
default	finally	int	return	true	byte	do
for	long	static	typeof	case	else	goto
new	switch	void	char	enum	if	null
synchronized	volatile	class	export	implements	package	this
while	const	extends	import	private	throw	with
continue	false	in	protected	throws		

Tablica 2.1. prikazuje ključne riječi koje se koriste u JavaScript-u. Neke od njih nisu u upotrebi, već su rezervirane za buduća proširenja. JavaScript koristi sljedeće tipove podataka: numeričke, logičke, stringove, specijalni tip podataka (radi se o dvije predefinirane vrijednosti: null za null vrijednost i undefined za vrijednost varijable koja nije definirana) [20].

3. OPIS STROJA

Razvoj industrije za punjenje boca u zadnjih nekoliko desetljeća doživio je veliki tehnički napredak. To se odnosi na povećanje kapaciteta i produktivnosti u rada za pet do deset puta, u smanjenju troškova punjenja po jedinici proizvoda te u očuvanju kvalitete proizvoda. Kod postrojenja za punjenje i pakiranje boca razlikuje se mokri i suhi dio linije. Stroj za pranje, punjenje, zatvaranje, pasterizaciju i etiketiranje čini „mokri“ dio. Dok, stroj za pakiranje boca, depaletizaciju i paletizaciju, s odgovarajućim transporterima, čini „suhi“ dio linije. U ovome radu je opisan stroj za punjenje boca, koji je dio „mokre“ linije [22].

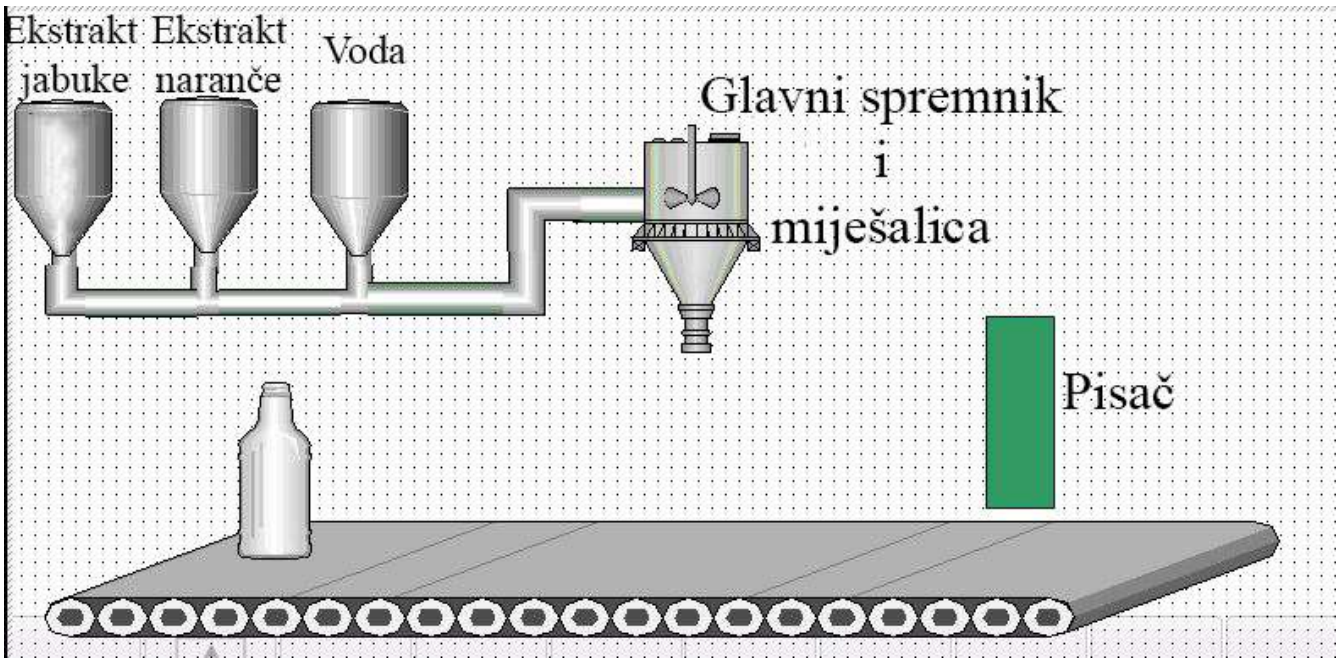
3.1 Stroj za punjenje boca

U ovome diplomskom radu obrađuje se projektiranje upravljačkog programa stroja za punjenje boca koji će se koristiti za punjenje tri vrste napitaka. Mogući odabiri su:

a) Sok od jabuke (20% ekstrakt jabuke i 80% voda)

b) Sok od naranče (20% ekstrakt naranče i 80% voda)

c) Voda



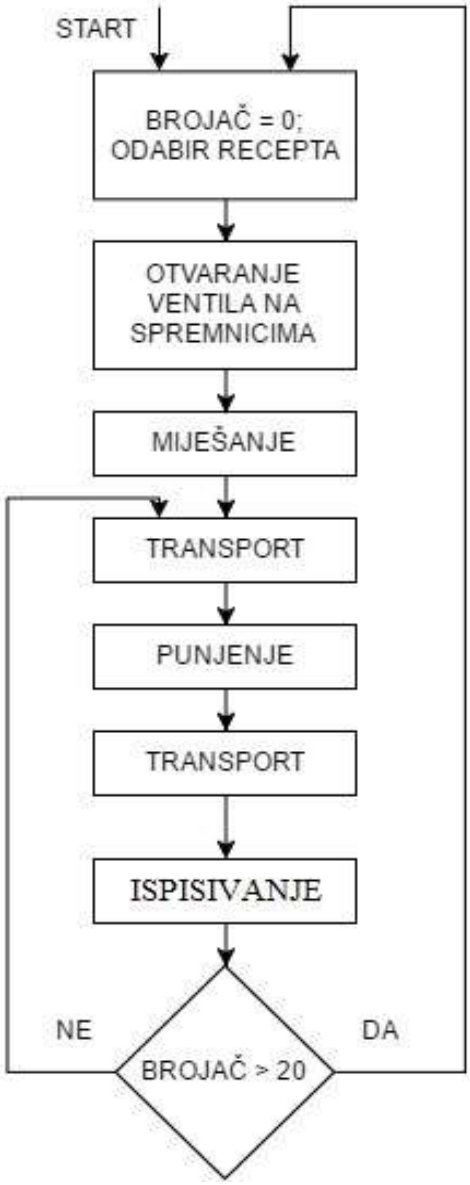
Slika 3.1. Načelna shema stroja za punjenje boca

Slika 3.1 prikazuje pojednostavljeni prikaz stroja koji se sastoji od:

- tri spremnika koji sadrže ekstrakt jabuke, ekstrakt naranče i vodu
- glavnog spremnika u kojem se nalazi miješalica
- po jednog ventila za svaki od spremnika
- transportne trake
- pisača

Zadaća stroja je:

1) Ovisno o odabranom receptu otvoriti odgovarajuće ventile na spremnicima i time napuniti glavni spremnik
2) Promiješati sadržaj glavnog spremnika
3) Pokretnom trakom transportirati praznu bocu do mjesta za punjenje
4) Otvoriti glavni ventil i napuniti bocu
5) Napunjenu bocu transportirati do mjesta za ispisivanje
6) Na mjestu za ispisivanje na bocu se ispisuje rok trajanja



Slika 3.2. Tijek procesa

Slika 3.2 prikazuje tijek procesa. Na početku rada stroja je potrebno odabrati recept (vrstu napitka). Kad je recept odabran brojač se postavlja u nulu i otvaraju se odgovarajući ventili na spremnicima te se puni glavni spremnik. Punjenje glavnog spremnika, tj. vrijeme koje su ventili na spremnicima otvoreni, je unaprijed određeno. Nakon što je glavni spremnik napunjen potrebno je njegov sadržaj promiješati, miješanje traje pet sekundi. Transportnom trakom prazna boca se transportira do mjesta za punjenje. Kad dođe do mjesta za punjenje traka se zaustavlja, glavni ventil se otvara i puni bocu dvije sekunde. Zatim se napunjena boca transportira do pisača, gdje se transportna traka zaustavlja i na bocu se ispisuje rok trajanja, brojač se povećava za jedan. Proces punjenja se ponavlja dvadeset puta, kao što se vidi na slici 3.2. Kada brojač dođe na vrijednost

dvadeset, tj. kad je napunjeno dvadeset boca, ponovno se prema prvotno odabranom receptu puni glavni spremnik i cijeli postupak se ponavlja.

Stroj za punjenje boca se promatra kao automatizirani proizvodni sustav jer se proces punjena odvija bez sudjelovanja čovjeka. Riječ je o sustavu s diskretnim događajima koji za razliku od kontinuiranog sustava ima ograničeni broj stanja. Za izradu modela sustava potrebno je znati sve korake koji se moraju izvršiti za ostvarenje zadatka te koji uvjeti moraju biti zadovoljeni kako bi proces prešao iz jednog stanja u drugo [23].

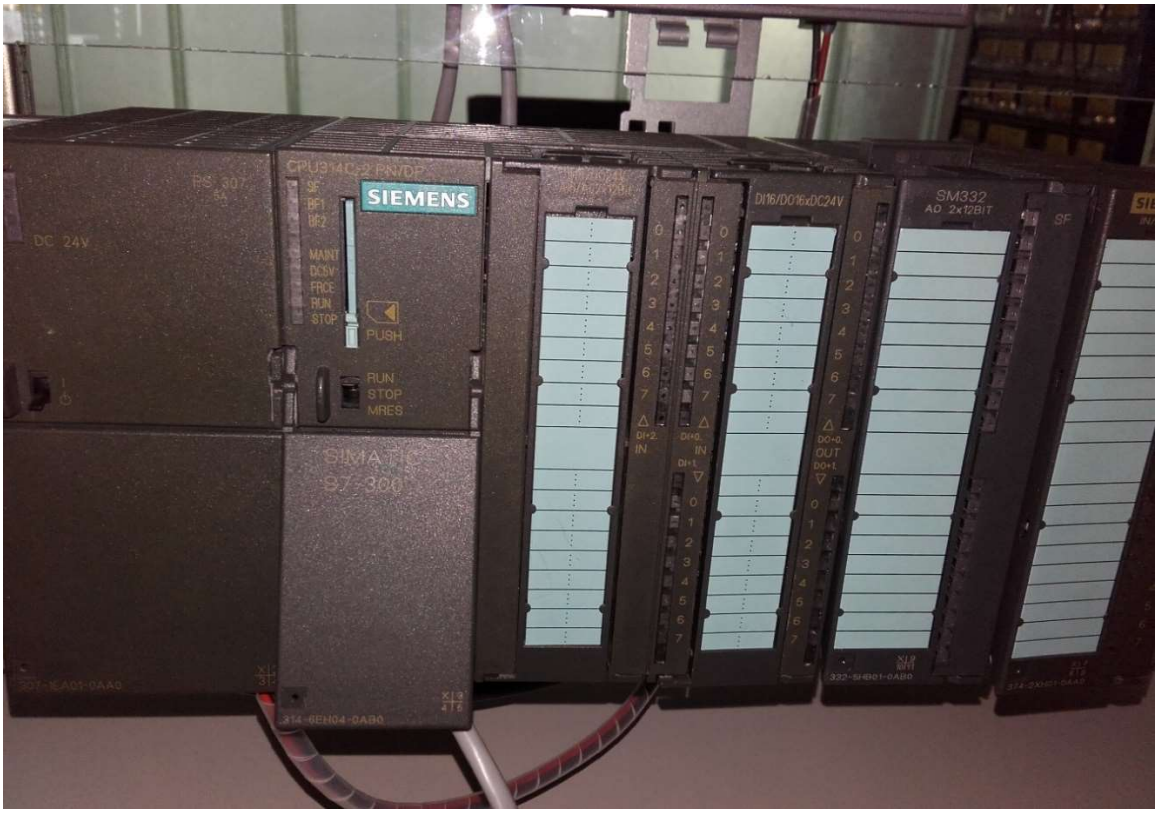
3.2 Korištena oprema za realizaciju sustava upravljanja stroja za punjenje boca

Za realizaciju sustava upravljanja stroja za punjenje boca potrebni su:

1) PLC

Korišten je Siemens Simatic S7 300 (CPU 314C-2 PN/DP). Ovo je jedan od najkorištenijih PLC-a u industriji. PLC je prikazan na slici 3.3. Ovaj PLC se sastoji od pet modula:

-napajanje
-modul s 5 AI i 2 AO
-modul s 16 digitalnih ulaza i 16 digitalnih izlaza
- modul s 2 AO
- testni modul koji simulira ulaze ili izlaze



Slika 3.3. Siemens Simatic S7 300

2) HMI panel

Korišten je Simatic HMI TP700 Confort. To je 7“ panel s ekranom osjetljivim na dodir koji prikazuje 16 mil. boja. Omogućuje povezivanje s PLC-om preko profibus/MPI sučelja ili preko ethernet-a. Opremljen je s 12 MB korisničke memorije, prikazan je na slici 3.4.

Slika 3.4. Simatic HMI TP700 Confort

3) Frekvencijski pretvarač

Korišteni je frekvencijski pretvarač Emerson Commander SKA 1200075. Ovo je frekvencijski pretvarač snage 0,75 kW koji se spaja na jednofazni napon od 230 V. Pogodan je za upravljanje manjih trofazni asinkronih motora. Moguće ga je upravljati preko PLC-a naponski (0-10 V) ili strujno (0-20 mA ili 4-20 mA). U ovom radu je korišteno upravljanje koristeći naponske signale. Slika 3.5 prikazuje ovaj frekvencijski pretvarač.

Slika 3.5. Emerson Commander SKA 1200075

4) Asinkroni motor

Motor korišten u ovom radu je prikazan na slici 3.6, a njegove karakteristike se nalaze u tablici 3.1. Motor je korišten u spoju trokut, što znači da mu je potreban napon od 220 V. Snaga motora iznosi 0,55 kW.

Slika 3.6. Korišteni asinkroni motor

Tablica 3.1. Podaci s natpisne pločice motora

cos φ	0,8
n	2800 o/min
napon u spoju trokut	220 V
struja u spoju trokut	2,6 A
napon u spoju zvijezda	380 V
struja u spoju zvijezda	1,5 A

5) Senzori

U ovom radu korišteni su fotoelektrični senzori SICK WT12-2B523, ovo je logički senzor, tj. izlaz senzora može biti „1“ ili „0“. Ovi senzori se sastoje od dva glavna djela: lasera i fotodiode. Laser je stalno uključen i odašilje svijetlost. Ako se svijetlost reflektira od obližnjeg objekta i osvijetli fotodiodu, koja se nalazi nekoliko milimetara iznad lasera, izlaz senzora će biti „1“. Kada fotodioda nije osvijetljena izlaz senzora je „0“. Ova vrsta senzora se obično

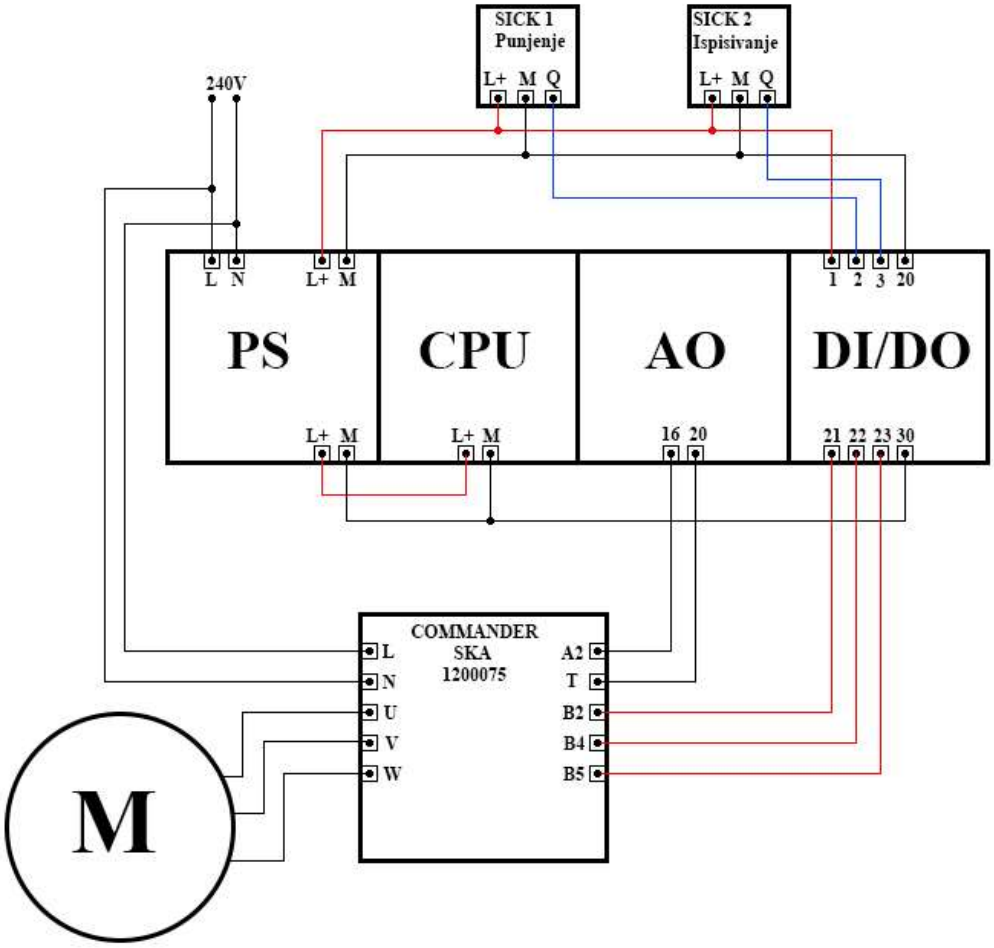
postavlja na transportnu traku. Kad proizvod, koji se transportira transportnom trakom, prolazi pokraj senzora od proizvoda se reflektira svijetlost i senzor na svojem izlazu daje „1“ i time aktivira neku daljnju radnju. Na slici 3.7 je prikazan korišteni senzor.

Slika 3.7. Senzor SICK WT12-2B523

3.3 Shema spajanja

Prethodno opisani uređaji su spojeni prema shemi na slici 3.8. Frekvencijski pretvarač i napajanje PLC-a (PS) su spojeni na gradsku mrežu 230 V frekvencije 50 Hz. Napajanje PLC-a na svojem izlazu daje standardni industrijski napon od 24 V istosmjerno i njime se napaja PLC (CPU), senzori (SICK 1 i 2) i modul s digitalnim izlazima. L+ označava 24 V, dok je s M označena masa.

Brzina vrtnje motora koji pokreće transportnu traku je kontrolirana preko analognog izlaza PLC-a (AO kontakti 16 i 20) koji na svojem izlazu daje napon iznosa 0-10 V što odgovara frekvenciji 0-50 Hz frekvencijskog pretvarača. Da bi se motor pokrenuo određenom brzinom, postavljenom preko AO, frekvencijski pretvarač mora od PLC-a dobiti „odobrenje“, tj. kontakti B2 (*enable*) i B4 (*run*) moraju biti u logičkoj jedinici.



Slika 3.8. Shema spajanja sustava upravljanja stroja za punjenje boca

4. REALIZACIJA PROJEKTA

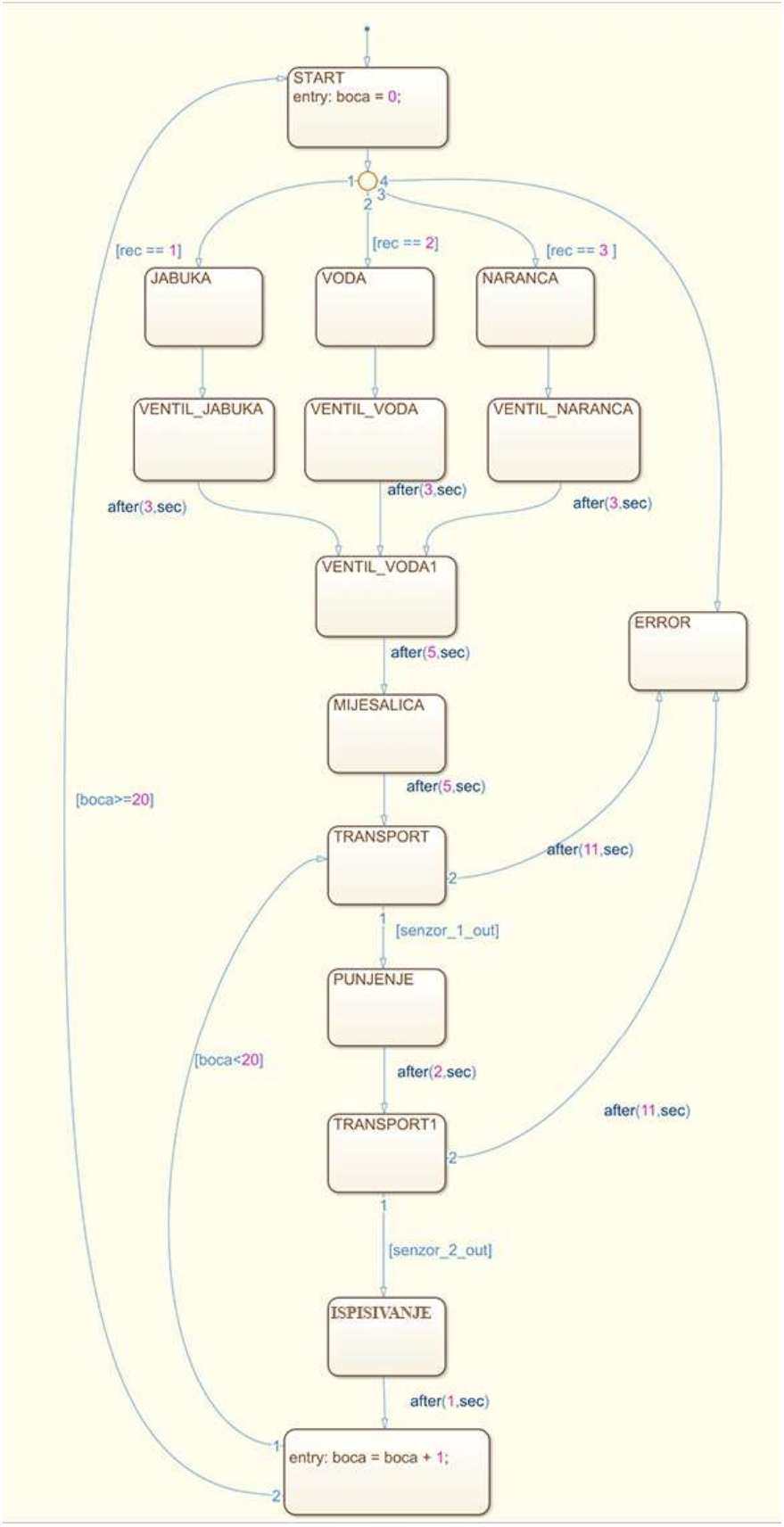
U ovom poglavlju je opisan način izrade i simulacije upravljačkog algoritma u MATLAB Stateflow-u te njegova implementacija u odabrani PLC. Opisana je izrada HMI sučelja i web stranice koja omogućuje nadzor i upravljanje nad procesom. U potpoglavlju 4.5 je opisan način testiranja upravljačkog algoritma.

4.1 Simulacija upravljačkog algoritma

Budući da stroj za punjenje boca promatramo kao sustav s diskretnim događajima za njegovu simulaciju je korišten dijagram tijeka. Koristeći osnovne blokove dijagrama tijeka, opisane u poglavlju 2.2, izrađen je upravljački algoritam stroja. Dijagram tijeka je izrađen u softverskom alatu MATLAB Stateflow koji je opisan u potpoglavlju 2.3.

Slika 4.1 prikazuje dijagram tijeka stroja za punjenje boca u Stateflow-u. Kao što se vidi na slici na početku programa varijabla „boca“ se postavlja u nulu te dolazi do grananja. Kod grananja program provjerava vrijednost varijable „rec“, vrijednost ove varijable se postavlja prije pokretanja programa.

Ako varijabla iznosi jedan, znači da je odabran sok od jabuke te se otvara ventil spremnika s ekstraktom jabuke na tri sekunde. Ukoliko vrijednost varijable iznosi dva, odabrana je voda te se na tri sekunde otvara spremnik vode. Ako je varijabla postavljena na tri, odabran je sok od naranče te se otvara ventil spremnika s ekstraktom naranče na tri sekunde. Ukoliko vrijednost varijable nije jedan, dva ili tri program odlazi u blok „ERROR“ te je prije ponovnog pokretanja programa potrebno postaviti vrijednost varijable „rec“.



Slika 4.1. Upravljački algoritam stroja za punjenje boca prikazan dijagramom tijeka u MATLAB Stateflow-u

Neovisno o odabiru napitka ventil spremnika s vodom se otvara na pet sekundi te je time glavni spremnik napunjen. Potrebno ga je promiješati, a miješanje traje pet sekundi. Nakon miješanja sve je spremno za punjenje boca.

Pokreće se transportna traka te transportira praznu bocu do mjesta za punjenje. Transportna traka radi sve dok boca ne aktivira senzor („senzor_1_out“), kad je senzor aktiviran transportna traka se zaustavlja te se boca može puniti. Ukoliko prođe jedanaest sekundi od početka rada transportne trake, a senzor nije aktiviran program odlazi u blok „ERROR“, što u ovom slučaju znači da nije bilo prazne boce na traci.

Punjenje boce traje dvije sekunde, tj. glavni ventil je otvoren dvije sekunde. Nakon punjenja boca se transportira do pisača. Transport boce radi na isti način kao i ranije, transportna traka radi sve dok senzor („senzor_2_out“) nije aktiviran ili u slučaju da nije aktiviran u jedanaest sekundi od početka rada trake program odlazi u blok „ERROR“ i to znači da je boca nestala s transportne trake između punjenja i pisača.

Kad se boca zaustavi na mjestu za ispisivanje na nju se ispisuje datum do kojeg je najkasnije moguće iskoristiti proizvod, tj. rok trajanja. Ovdje se boca zaustavlja na jednu sekundu. Program povećava varijablu „boca“ za jedan te ponavlja postupak (transportiranje, punjenje, transportiranje, etiketiranje). Kada varijabla dosegne vrijednost dvadeset program se izvršava iz početka, programskog bloka „START“.

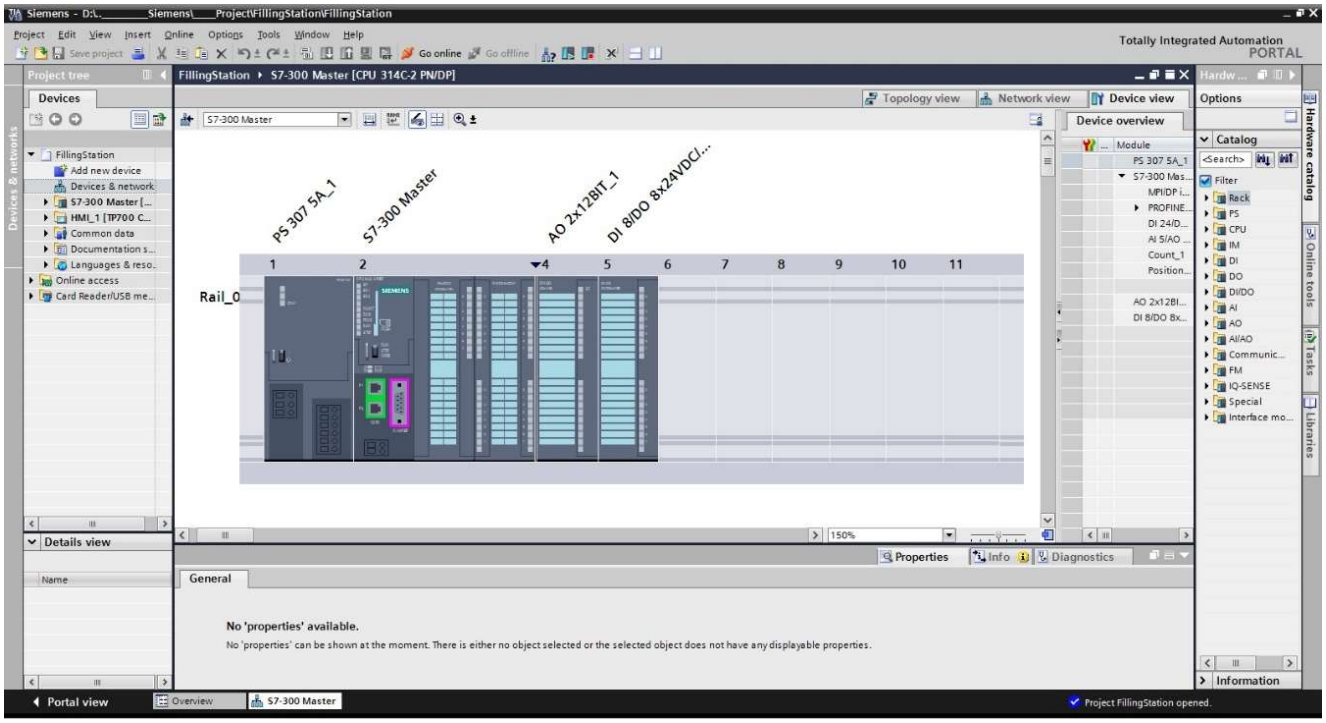
Simulacijom je utvrđeno da ne postoje logičke pogreške i beskonačne petlje te je moguće implementirati ovaj algoritam u PLC.

4.2 Implementacija upravljačkog algoritma u PLC

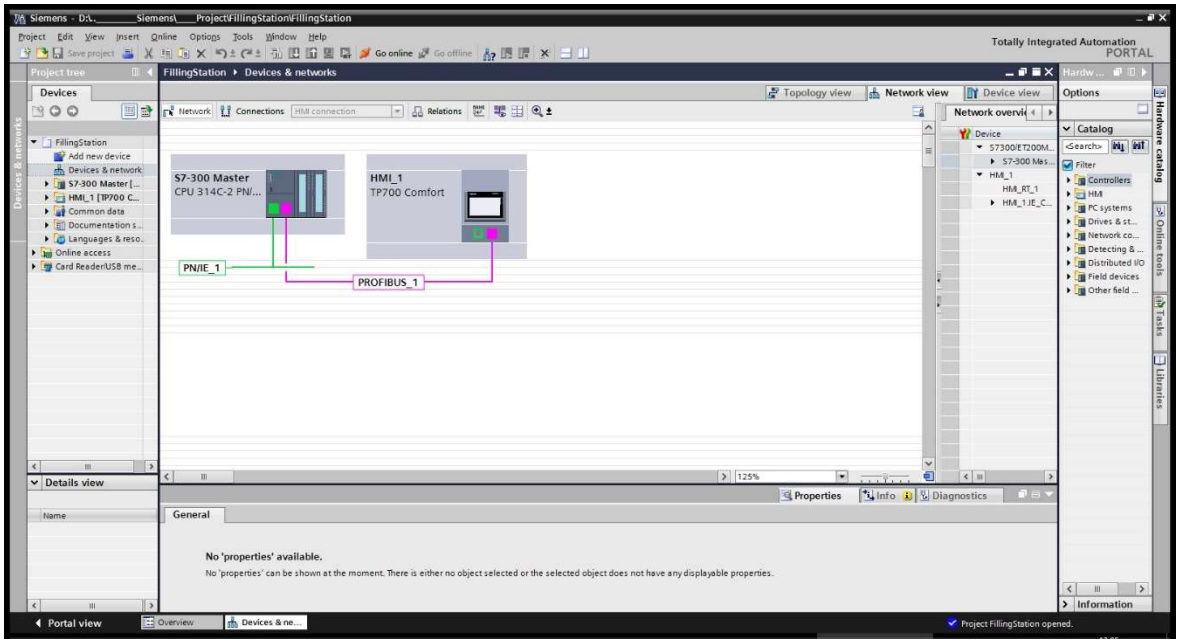
Za implementaciju upravljačkog algoritma, čija je ispravnost prethodno potvrđena simulacijom u potpoglavlju 4.1, koristi se softverski alat Step 7, koji je dio TIA Portal-a v13, koji je ranije opisan. Prvi korak u radu u TIA Portal-u je definiranje sklopovske konfiguracije PLC uređaja. Potrebno je s PLC-a očitati nazive pojedinih modula i te module dodati u TIA Portal-u. Dodani moduli su:

- Napajanje PS 307 5A (6ES7 307-1EA01-0AA0)
- CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0)
- AO 2x12BIT (6ES7 332-5HB01-0AB0)
- DI 8/DO 8x24VDC/0.5A (6ES7 323-1BH01-0AA0) (testni modul)

Ovaj korak je prikazan na slici 4.2.



Slika 4.2. Dodavanje modula PLC-a



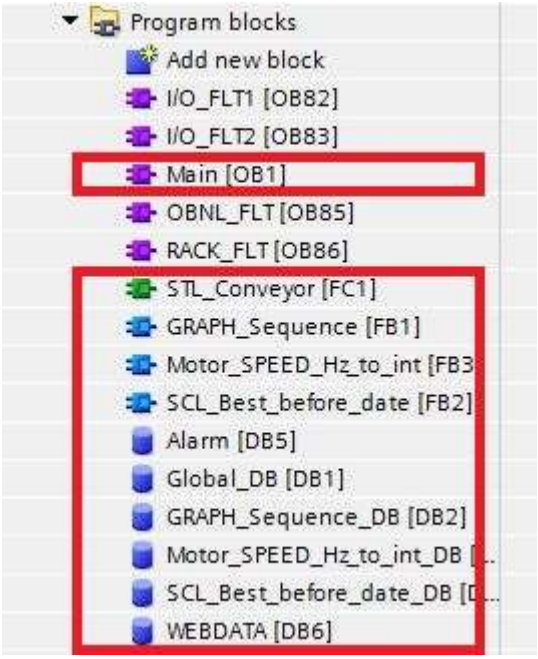
Slika 4.3. Povezivanje PLC-a i HMI-a

Sljedeći korak je dodavanje HMI panela i njegovo povezivanje s PLC-om. Korišten je panel Simatic HMI TP700 Confort kao što je ranije spomenuto. Kao što je prikazano na slici 4.3, HMI je povezan s panelom preko *profibus*-a, a PLC je preko *profinet*-a povezan s lokalnom mrežom preko koje se na njega može spojiti računalom u svrhu njegovog programiranja i omogućuje PLC-u držanje web servera.

4.2.1 Korišteni programski blokovi

Kao što je opisano u poglavlju 2.4.1, program PLC-a je podijeljen u blokove radi bolje organizacije i preglednosti. U programiranju stroja za punjenje boca korišteni su sljedeći blokovi:

- Main [OB1]
- STL_Conveyor [FC1]
- GRAPH_Sequence [FB1]
- Motor_SPEED_Hz_to_int [FB3]
- SCL_Best_before_date [FB2]
- podatkovni blokovi (DB) za spremanje varijabli

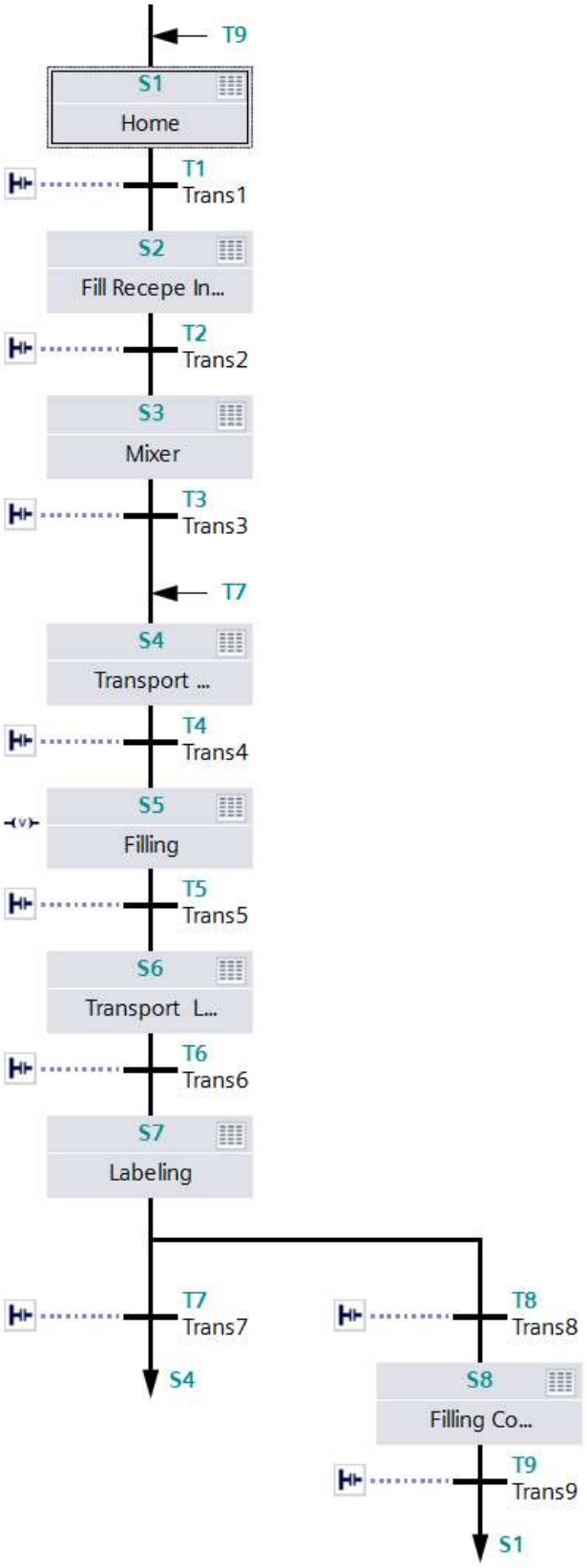


Slika 4.4. Korišteni blokovi

Na slici 4.4 su prikazani korišteni blokovi. Blokovi koji se nalaze unutar crvenih pravokutnika su dodani ručno, a ostale softver dodaje automatski.

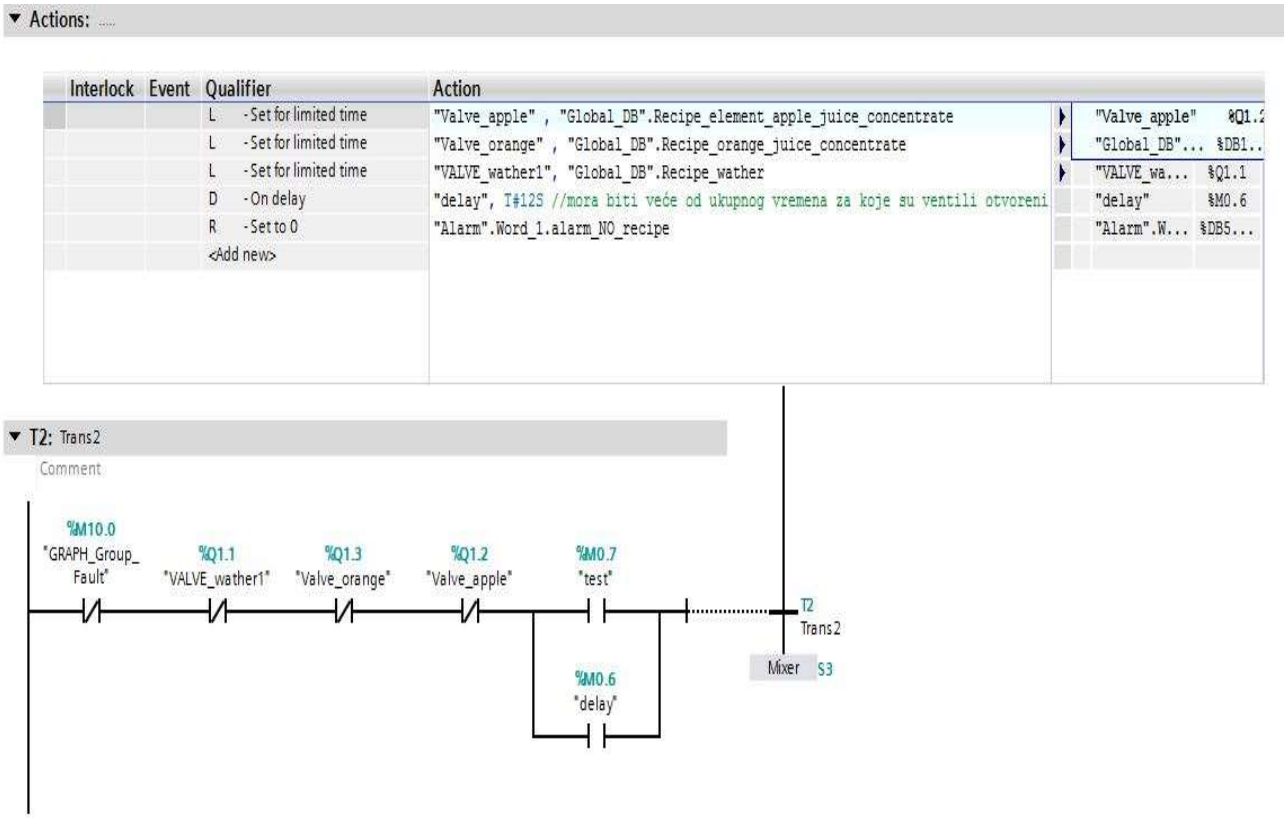
4.2.2 GRAPH_Sequence [FB1]

Kako bi se simulirani model stroja iz MATLAB Stateflow-a prenio u PLC korišten je GRAPH programski jezik zbog sličnosti s dijagramom tijeka u kojem je izvedena simulacija.



Slika 4.5. Korišteni blokovi u GRAPH-u

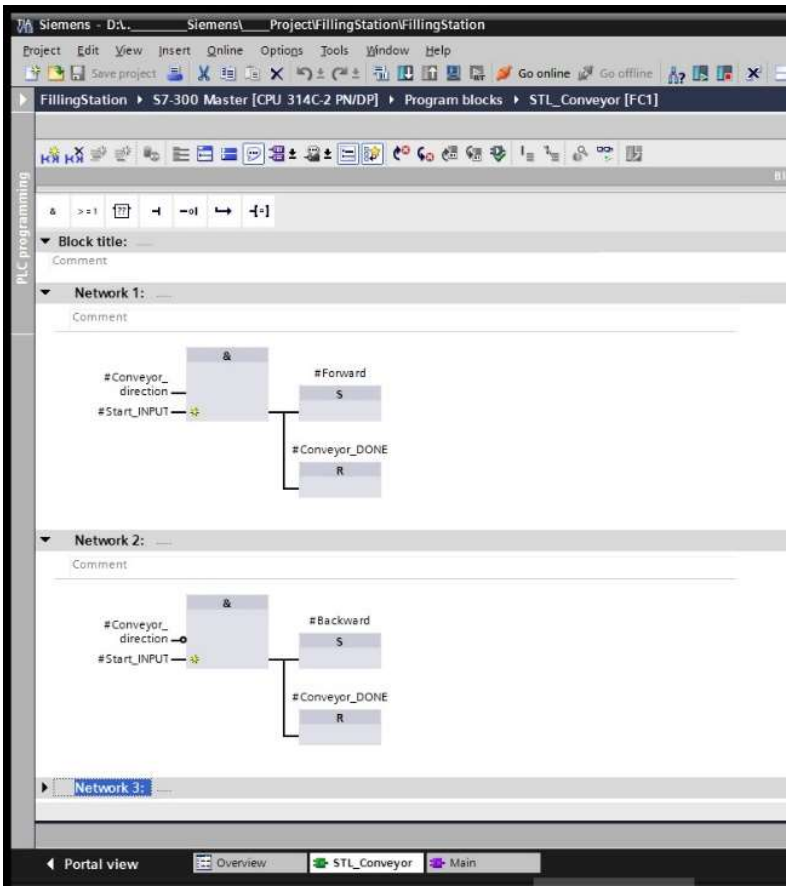
Prilikom programiranja u GRAPH programskom jeziku dodaju se blokovi, u ovom slučaju je korišteno osam blokova kao što se vidi na slici 4.5. Za prelazak u sljedeći blok moraju biti ispunjeni određeni uvjeti. Uvjeti za prijelaz npr. iz drugog (S2) bloka u treći (S3) su prikazani na slici 4.6. Kao što se vidi na slici koristi se LAD programski jezik. Potrebno je razlikovati normalno otvorene prekidače (NO) i normalno zatvorene prekidače (NC). *GRAPH_Group_Fault*, *VALVE_wather1*, *Valve_orange* i *Valve_apple* su NC prekidači, a *test* i *delay* su NO prekidači. NC prekidači se otvaraju kada je memorijska adresa za koju su definirani u „1“, npr. ako je *GRAPH_Group_Fault* u „1“, što znači da je došlo do neke greške u sustavu, kontakt prekidača će se otvoriti i „struja“ neće moći do *Trans2*, tj. prijeći u sljedeći blok (LAD sheme se čitaju s lijeva na desno). NO prekidači ovdje služe za detekciju pogreške. U slučaju da se niti jedan ventil ne otvori (to se događa ako nije odabrana vrsta napitka koju stroj puni) ovi prekidači zaustavljaju program i javljaju grešku. Da nema ovih prekidača program bi prešao u sljedeći blok (S3), tj. stroj bi krenuo miješati glavni spremnik bez da je on napunjen.



Slika 4.6. Uvjeti za prelazak u sljedeći blok

4.2.3 STL_Conveyor

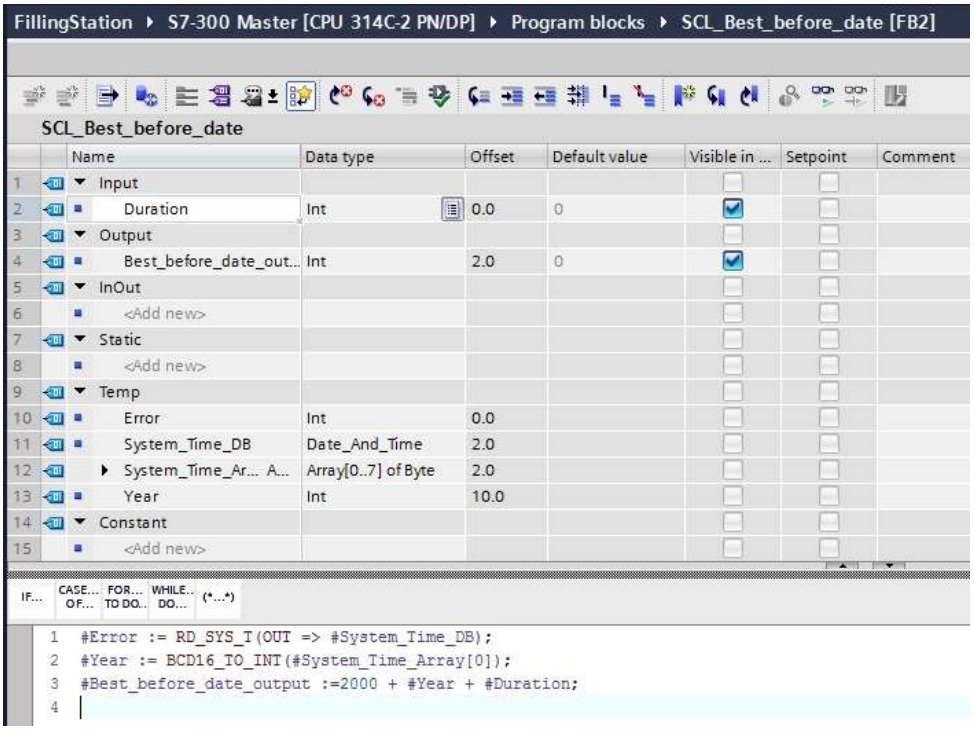
Ovaj blok se sastoji od 21 programske mreže. Programske mreže su pisane u FBD programskom jeziku, kao što je prikazano na slici 4.7. Ovaj blok je zadužen za: komunikaciju s frekvencijskim pretvaračem, signalizaciju (upravlja sa svjetlećim diodama na testnom modulu; diode prikazuju u kojem bloku u GRAPH_Sequence se program nalazi) i radnje vezane za alarme.



Slika 4.7. Blok STL_Conveyor

4.2.4 SCL_BBD

Ovaj blok koristi SCL programski jezik, tj. Phyton, kao što se vidi na slici 4.8. Ovaj blok služi za ispisivanje godine roka trajanja proizvoda na bocu. PLC uzima trenutno sistemsko vrijeme (vrijeme koje je postavljeno u PLC-u), tj. godinu, te mu dodaje vrijeme trajanja proizvoda, koje se postavlja na HMI-u. Vrijeme trajanja proizvoda ručno postavlja operater stroja, ono može biti od nula do pet. Npr. ako je vrijeme trajanja postavljeno na tri, PLC zbraja tri s trenutnom godinom, tj. 3 + 2017 i na bocu se ispisuje 2020.

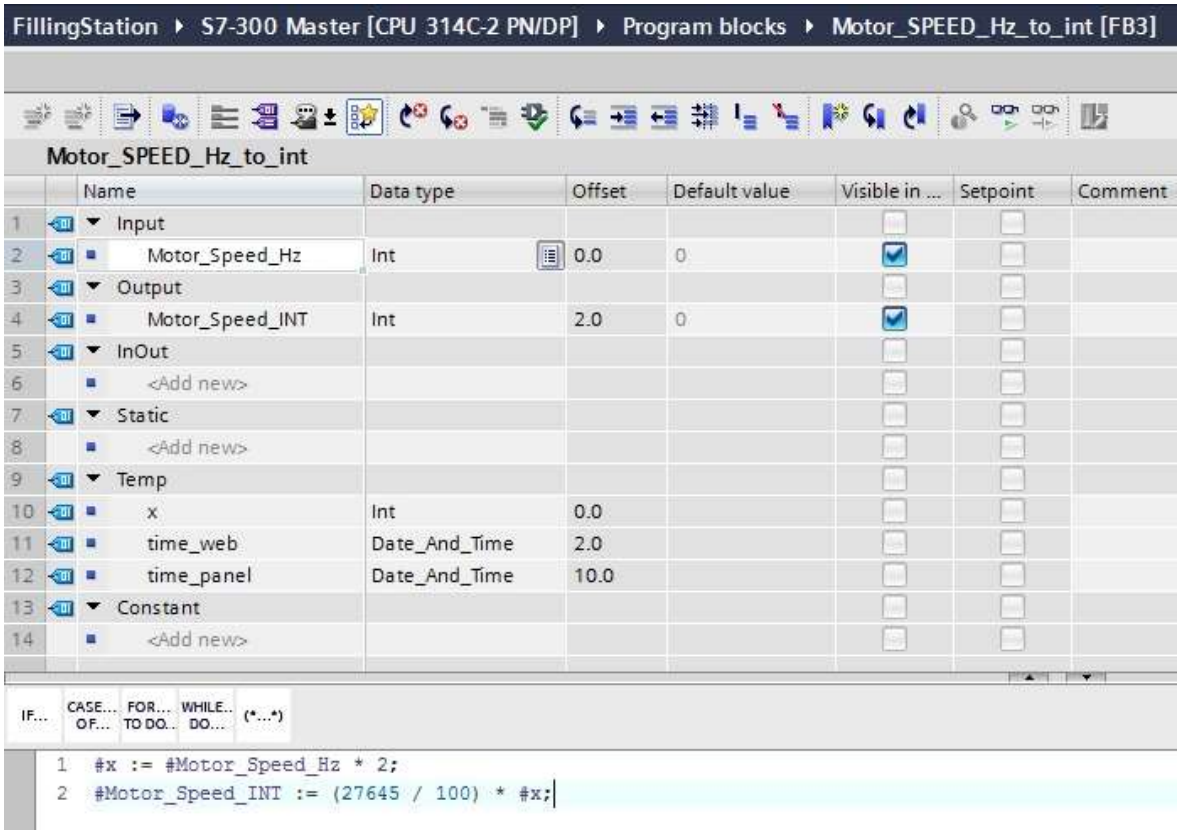


Slika 4.8. Blok SCL

4.2.5 SCL2

Ovaj blok služi za konverziju Hz u cjelobrojni tip podataka (*Int*). Brzina vrtnje motora je određena s frekvencijom frekvencijskog pretvarača, ona može biti od nula do pedeset Hz što rezultira brzinom vrtnje od 0 do 3000 o/min prema relaciji (2-1).

Na HMI panelu operater može odabrati željenu frekvenciju od 0 do 50 Hz, međutim PLC-u je za daljnju obradu potrebna ta vrijednost u *Int* rasponu vrijednosti, od 0 do 27 645. Kad PLC raspolaže s *Int* vrijednosti frekvencije on tu vrijednost automatski pretvara u napon vrijednosti 0 do 10 V na analognom izlazu na koji je spojen frekvencijski pretvarač. Slika 4.9 prikazuje SCL kod koji se koristi za potrebnu pretvorbu vrijednosti.

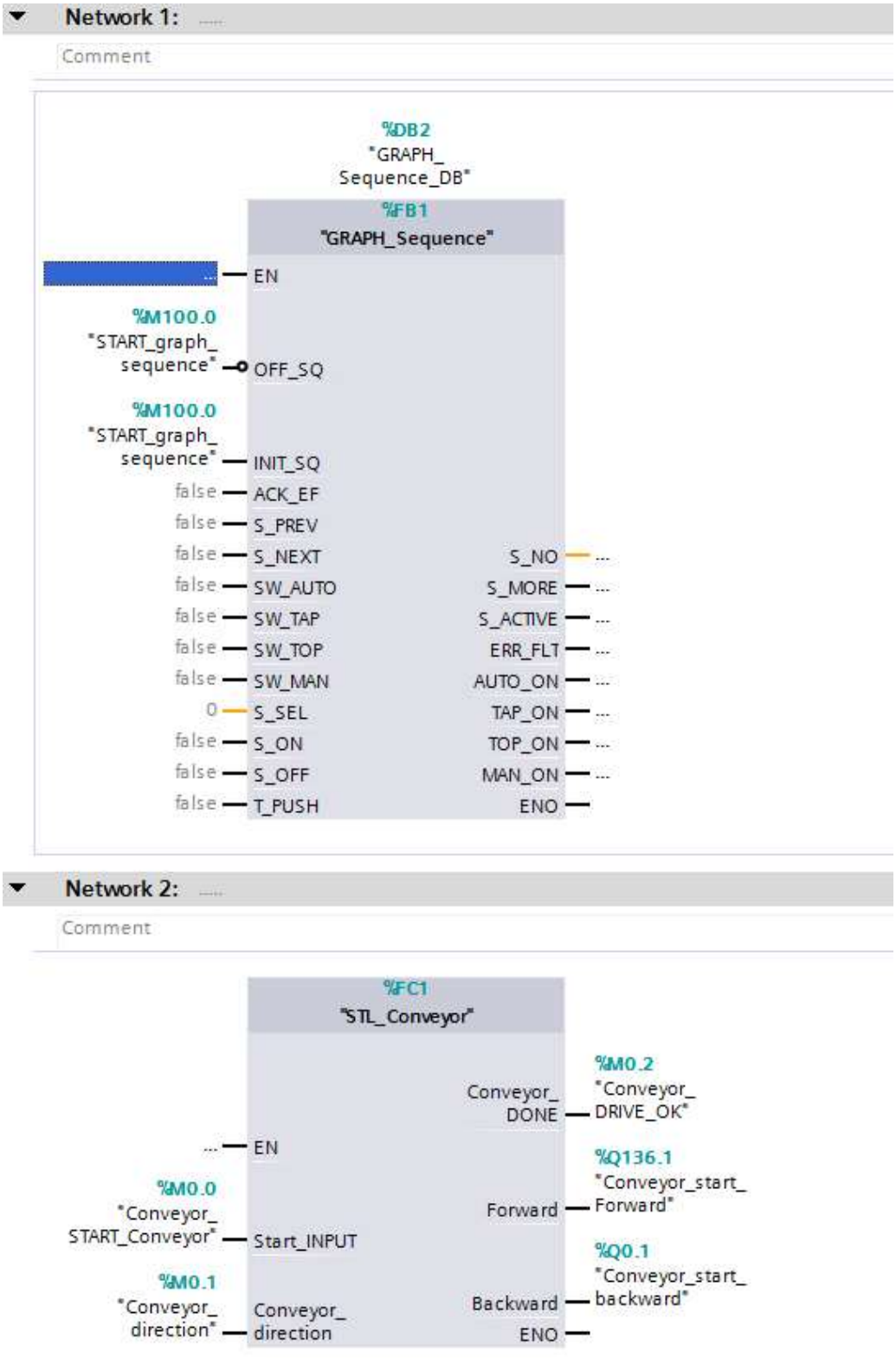


Slika 4.9. Blok Motor_SPEED_Hz_to_int

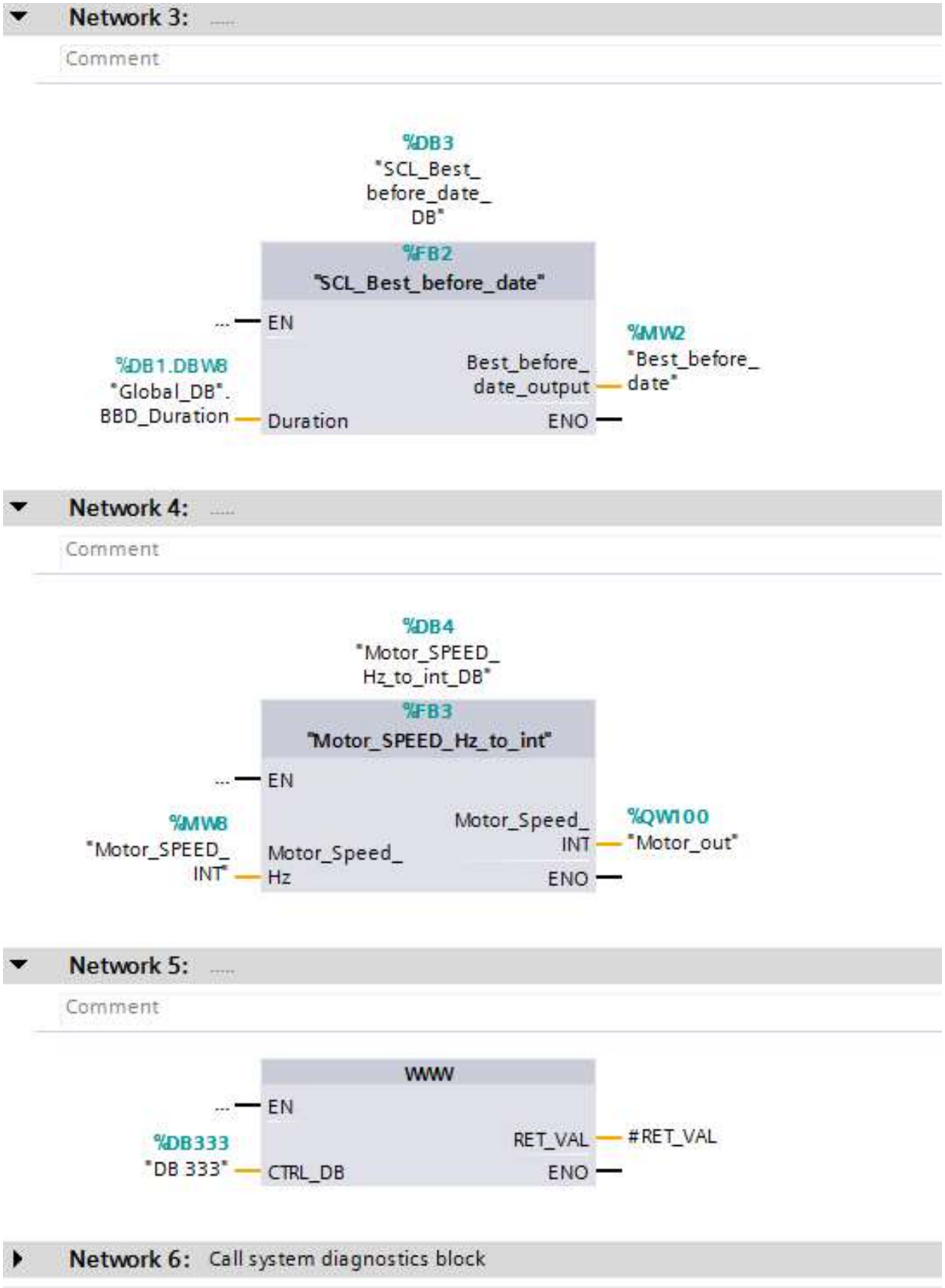
4.2.6 OB1

OB1 je glavni blok. Kao što se vidi na slici 4.10 i slici 4.11, svi prethodno opisani blokovi se nalaze u ovom blok. Ovaj blok se izvršava ciklički, maksimalno vrijeme jednog ciklusa je tvornički postavljeno na 150 ms, no ono se može promijeniti.

Programska mreža pet se sastoji od bloka WWW koji služi držanje web servera što je detaljnije objašnjeno u potpoglavlju 4.4.



Slika 4.10. Blok OB1 (1/2)



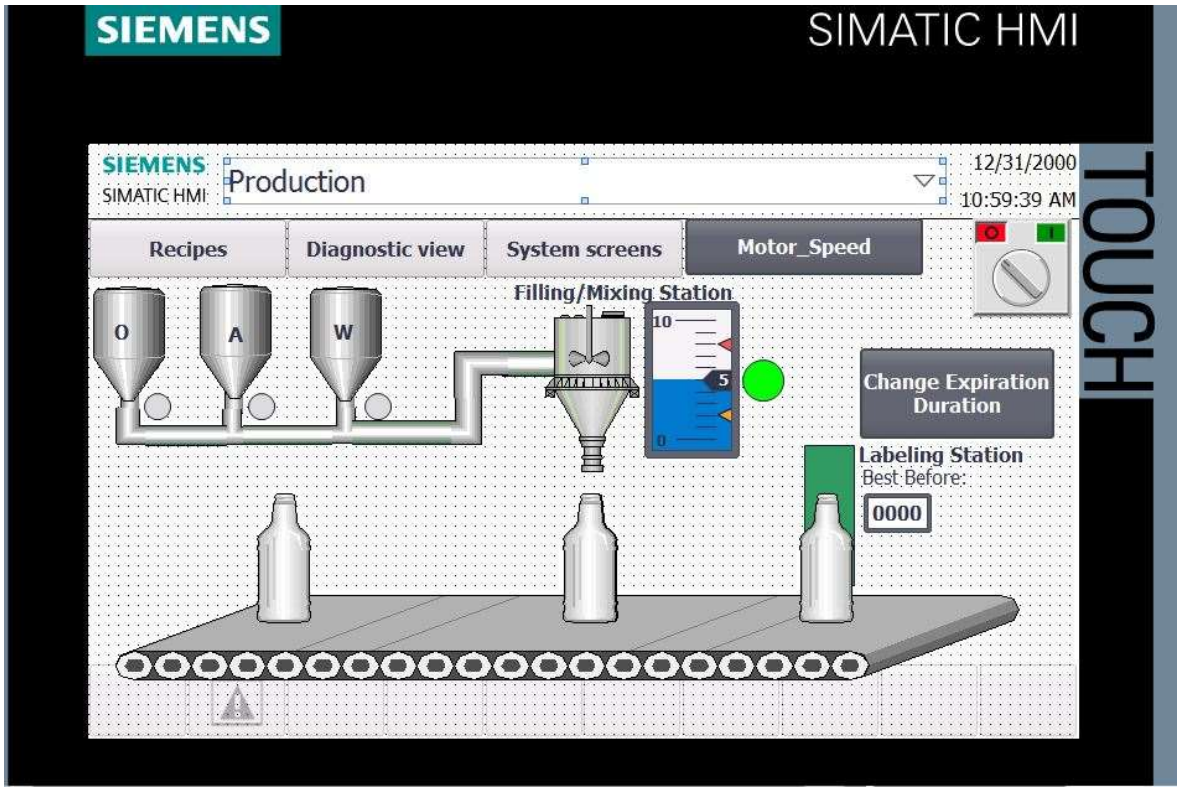
Slika 4.11. Blok OB1 (2/2)

4.3 Izrada korisničkog sučelja HMI-a

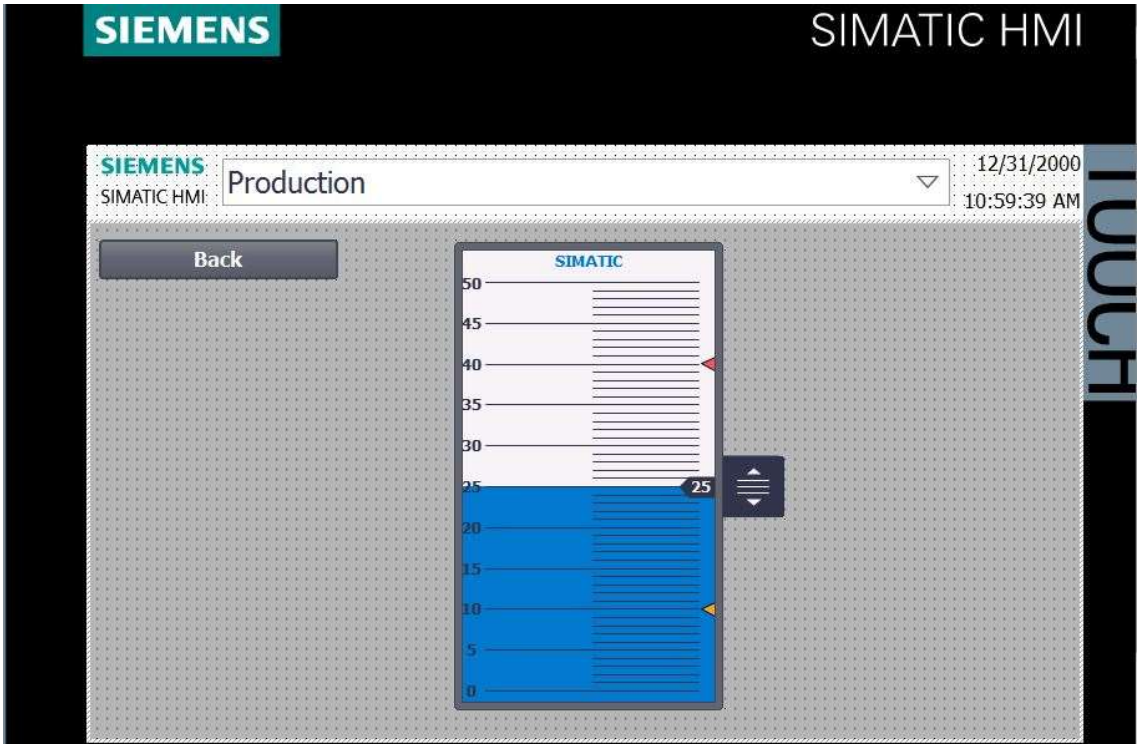
Koristeći WinCC, koji je dio TIA Portala, izrađeno je korisničko sučelje HMI-a. U ovome radu su izrađena tri „ekrana“ koji su prikazani na slikama 4.12 do 4.14. Ekran „Production“ je glavni ekran. On prikazuje tijek procesa, sadrži indikatore koji prikazuje koji je ventil otvoren, prikazuje brojač koji broji koliko je boca napunjeno, omogućuje pokretanje i zaustavljanje procesa i s njega se može prijeći u druge ekrane.

Ekran „Motor_SPEED“ služi za regulaciju brzine motora. Ovaj ekran sadrži klizač kojim se odabire frekvencija od 0 do 50 Hz te se time određuje brzina motora kao što je objašnjeno prethodno u tekstu.

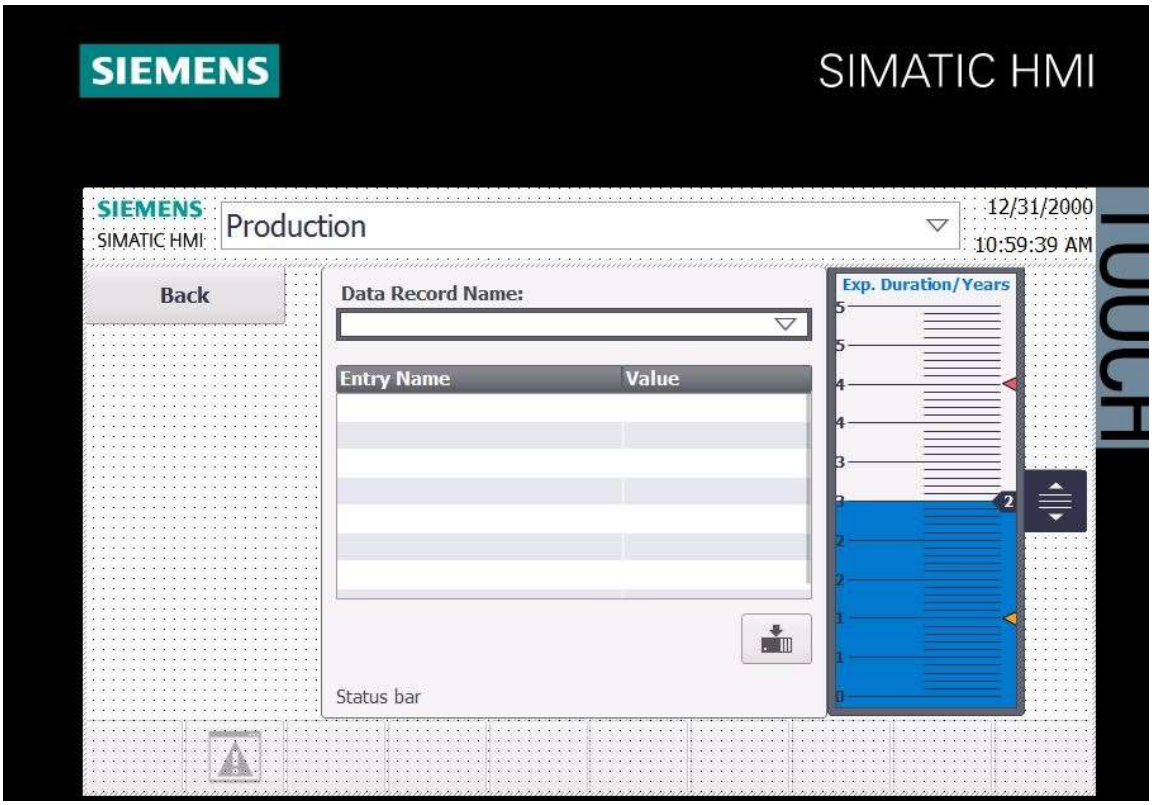
Ekran „Recipes“ služi za odabir napitka koji stroj puni i za odabir roka trajanja proizvoda.



Slika 4.12. Ekran Poduction



Slika 4.13. Ekran Motor_SPEED



Slika 4.14. Ekran Recept

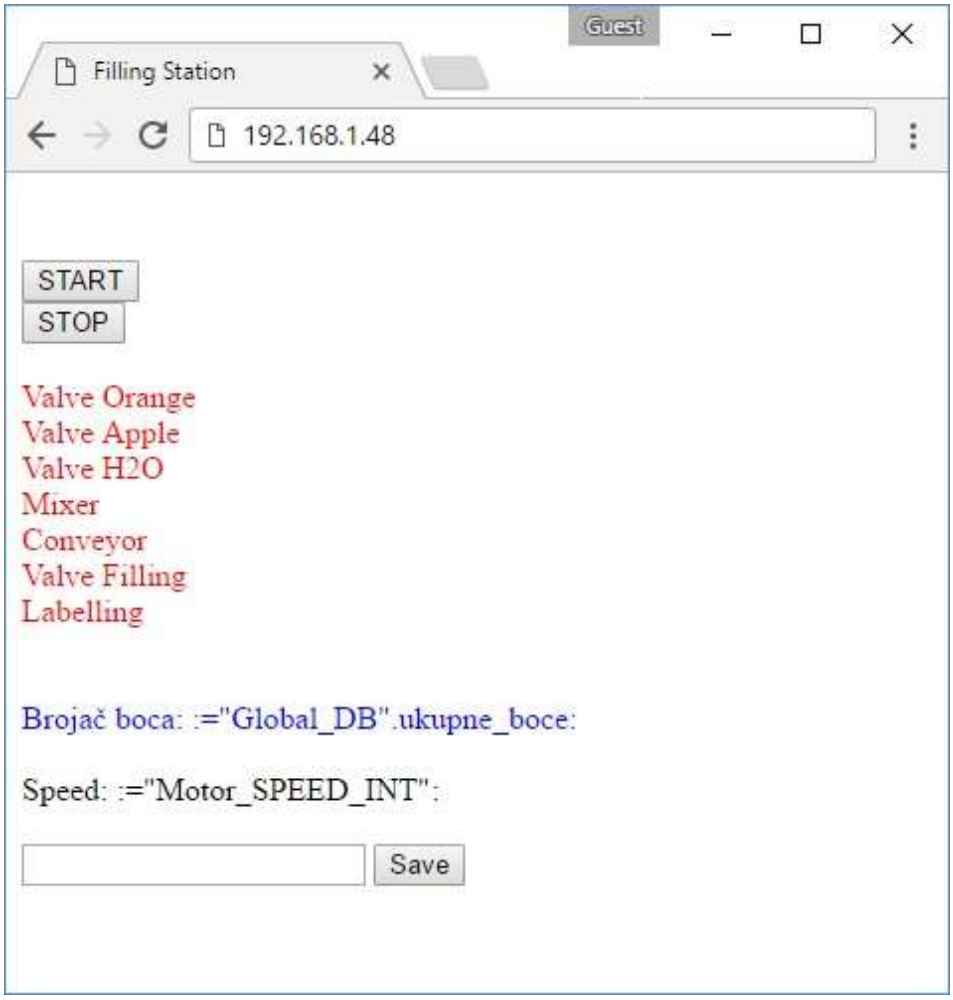
4.4 Web server

Siemens Simatic S7 300 ima mogućnost držanja web servera te je moguće izraditi jednostavnu web stranicu koja sadrži određene podatke o procesu. Ova web stranica je dohvatljiva samo računalima koji su spojeni na lokalnu mrežu u kojoj se nalazi PLC. Web stranici se pristupa upisivanjem IP adrese PLC-a u internet preglednik na računalu. Web stranica je izrađena korištenjem HTML-a i JavaScript-a, koji su opisani u potpoglavlju 2.5.

HTML je korišten za dizajn web stranice, a JavaScript služi za razmjenu podataka s PLC-om. Korištena je biblioteka jQuery koja smanjuje količinu koda potrebnu za izvršavanje pojedinih zadataka. Cijeli programski kod web stranice je prikazan u prilogu.

Slika 4.15 prikazuje gotovu web stranicu. Web stranica ima jednostavan dizajn iz razloga što je PLC ograničen s memorijom. Stranica se osvježava svake sekunde, tj. preuzima podatke s PLC-a. Izrađena web stranica sadrži gotovo sve funkcije HMI panela. Nedostatak web stranice je nemogućnost odabira vrste napitka koji stroj puni. Funkcije web stranice:

1) Omogućuje pokretanje i zaustavljanje procesa
2) Prikazuje u kojem je stanju proces
3) Prikazuje broj boca koje su napunjene
4) Omogućuje promjenu brzine motora



Slika 4.15. Web stranica

4.5 Puštanje u pogon i testiranje upravljačkog programa

Nakon što je upravljački sustav spoja spojen prema shemi sa slike 3.8 i u PLC je implementiran upravljački program, pristupa se njegovom testiranju. Testiranje treba utvrditi ispravan rad upravljačkog sustava ili ukazati na greške u njegovom radu. Testiranje se provodi u četiri faze:

1) **Testiranje za vrijeme normalnog pogona**

Ova faza testiranja simulira uvjete rada u normalnom pogonu. Na HMI panelu je odabrana vrsta pića koju će stroj puniti i proces je pokrenut. Utvrđeno je da se proces izvršava normalno te da ne dolazi do pogrešaka.

2) Testiranje za vrijeme poremećaja

Testiranje za vrijeme poremećaja provodi se tako da se simulira neki od mogućih poremećaja. Testirane se četiri vrste poremećaja:

a) Kada recept nije odabran
b) Kada je brzina vrtnje motora postavljena na 0
c) Kada senzor broj 1 nije aktiviran (vidi sliku 4.1)
d) Kada senzor broj 2 nije aktiviran (vidi sliku 4.1)

U svakom od navedenih slučajeva proces je automatski zastavljen i na HMI panelu je prikazana pogreška i razlog zaustavljanja procesa.

3) Testiranje dodatnih funkcija

Pod dodatnim funkcijama se smatra promjena brzina vrtnje motora. Ove funkcija se smatra dodatnom zato što se postavi jednom i vrlo je mala vjerojatnost da će se kasnije mijenjati. Testirana je na način da je na HMI panelu, na ekranu „Motor_SPEED“, klizačem mijenjana vrijednost od 0 do 50, ova vrijednost predstavlja frekvenciju kojom frekvencijski pretvarač napaja motor. Promjenom vrijednosti klizača mijenja se brzina vrtnje motora.

4) Testiranje web sučelja

Testiranje web sučelja je izvršeno tako da je u web pregledniku za adresu upisana IP adresa PLC-a, time je otvorena prethodno izrađena web stranica. Testiranje je provedeno za normalan pogon (dio procesa koji je u tijeku je zelene boje, vidi sliku 4.15.) i za vrijeme poremećaja (kod poremećaja u radu na ekranu se pojavljuje novi prozor, *pop up*, koji obavještava korisnika o pogrešci). Upisivanjem željene frekvencije i pritiskom na gumb „Save“ (slika 4.15.) mijenja se brzina vrtnje motora, a pritiskom na gumb „START“ ili „STOP“ pokreče se ili zaustavlja proces. Testiranjem je utvrđeno da su sve funkcije web sučelja ispravne.

5. ZAKLJUČAK

U ovom radu opisan je upravljački sustav stroja za punjenje boca koji se sastoji od PLC-a, HMI-a, frekvencijskog pretvarača, asinkronog motora i senzora te su objašnjeni osnovni principi rada za svaki od korištenih uređaja.

U softverskom alatu MATLAB Stateflow je izrađen upravljački algoritam po kojem će stroj raditi. Algoritam je predstavljen pomoću dijagrama tijeka. Simulacijom je utvrđeno da algoritam ne sadrži logičke pogreške niti beskonačne petlje te se pristupilo njegovoj implementaciji u raspoloživi PLC.

Za implementaciju razvijenog upravljačkog algoritma u PLC Siemens Simatic S7 300 korišten je u programski alat TIA Portal v13. Program je pisan korištenjem LAD, FBD, SCL i GRAPH programskih jezika. U istom programskom alatu je izrađeno HMI korisničko sučelje koje služi za interakciju operatera i PLC-a. Budući da PLC ima mogućnost realizacije web servera, izrađena je web stranica koristeći HTML i JavaScript. Web stranica omogućuje praćenje procesa preko lokalne mreže i upravljanje nekim njegovim osnovnim funkcijama.

Upravljački sustav je testiran u četiri faze (testiranje za vrijeme normalnog pogona, testiranje za vrijeme poremećaja, testiranje dodatnih funkcija i testiranje web sučelja) te je utvrđeno da se upravljački algoritam izvršava bez pogrešaka.

LITERATURA

1. D. Slišković: Procesna automatizacija, Fakultetska skripta, ETF Osijek, Osijek, 2009.
2. S. Perković: PROFIBUS, Zagreb, 2007.
3. G. Malčić, D. Maršić, J. Čučuk: Konfihuriranje Profinet I/O mreže u radu sa energetskim pretvaračem, Zagreb, 2014.
4. https://www.fer.unizg.hr/_download/repository/esa_lab_5.pdf , Izrada rješenja za upravljanje automobilskim klima uređajem s uključenim HMI-om, 24.4.2017.
5. http://www.aip.com.hr/cesta-pitanja/4/ , Frekvencijski pretvarači, 23.3.2017.
6. I. Šantalab: Izrada neizravnog frekvencijskog pretvarača pomoću arduina, Varaždin, 2015.
7. D. Vučetić: Brodski električni strojevi i sustavi, Rijeka, 2011.
8. https://hr.wikipedia.org/wiki/Senzori , Senzori, 23.3.2017.
9. I. Sindičić: Planiranje slijeda zadataka u sustavima s diskretnim događajima, http://www.fer.unizg.hr/_download/repository/kvalifikacijski_final.pdf , 23.3.2017.
10. H. Babić: Signali i sustavi, Zagreb, 1996.
11. https://hr.wikipedia.org/wiki/Dijagram_tijeka , 23.3.2017.
12. http://sstehnickast.skole.hr/upload/sstehnickast/images/static3/2326/File/Izrada_dijagrama_toka.pdf , Izrada dijagrama toka, 23.3.2017.
13. www.pbf.unizg.hr/content/download/22631/86898/version/2/file/pap4.pdf , Algoritmi, dijagrami toka, pseudoprogrami, 28.4.2017.
14. https://hr.wikipedia.org/wiki/MATLAB , MATLAB, 23.3.2017.
15. https://www.fer.unizg.hr/_download/repository/SIMULINK_SKRIPTA.pdf , Uvod u Simulink, 23.3.2017.
16. https://www.mathworks.com/products/stateflow.html , Stateflow, 23.3.2017.
17. D. Maršić, G. Malčić, I. Vlašić: Izrada programskih komponenti u TIA Portal programskom okruženju, Zagreb
18. I. Mejaš: Regulacija hidrauličnog sustava u realnom vremenu pomoću industrijskog PLC-a, Zagreb, 2013.
19. H.Berger: SIMATIC automatizacijski sustavi, Graphis, Zagreb, 2013.
20. http://www.irenaholjevac.com/dipl/ , Klijentske web tehnologije, 25.4.2017.
21. http://www.mojwebdizajn.net/skriptni-jezici/vodic/html/html-javascript.aspx , JavaScript i web dizajn, 23.3.2017.
22. http://www.tehnologijahrane.com/knjiga/tehnologija-industrijskog-punjenja-i-pakovanja-pica , 24.4.2017.
23. M. Jurić: Projektiranje radnih operacija na proizvodnoj liniji u automobilskoj industriji, Osijek, 2015.

SAŽETAK

Ovaj rad opisuje upravljački sustav stroja za punjenje boca, koji se sastoji od PLC-a, HMI-a, frekvencijskog pretvarača, asinkronog motora i senzora. Napravljen je upravljački algoritam koji je simuliran u MATLAB Stateflow-u. Algoritam je implementiran u PLC pomoću softverskog paketa TIA Portal te je izrađena nadzorno-upravljačkih aplikacija za HMI. Izrađeno je web sučelje korištenjem HTML-a i JavaScript-a preko kojeg se može nadzirati i upravljati strojem.

Ključne riječi: stroj za punjenje boca, MATLAB Stateflow, PLC, HMI, TIA Portal

ABSTRACT

Bottle filling process control using PLC

This paper describes control system of a filling machine, which consists of PLC, HMI, frequency converter, induction motor and sensors. Control algorithm of the machine was created and simulated in MATLAB Stateflow. Algorithm was implemented in to the PLC using TIA Portal in which is also created supervision application or the HMI. Web interface was created using HTML and JavaScript through which can be monitored and control pocess of the filling machine.

Keywords: bottle filling machine, MATLAB Stateflow, PLC, HMI, TIA Portal

ŽIVOTOPIS

Rođen je 20.12.1993. godine u Koprivnici. Osnovno obrazovanje je završio u O.Š. Braće Radić u Koprivnici 2008. godine. Iste godine upisuje Obrtničku školu u Koprivnici, smjer elektrotehničar. Godine 2012. završava srednju školu s izvrsnim uspjehom. Iste godine upisuje preddiplomski studij elektrotehnike na Elektrotehničkom fakultetu u Osijeku koji uspješno završava 2015. godine i upisuje diplomski studij elektrotehnike smjer elektroenergetika. Izvrsno se služi računalom i dobro poznaje računalne sustave. Ima dobro znanje engleskog jezika.

_____

Potpis

PRILOG – Programski kod web stranice PLC-a

```
<!-- AWP_In_Variable Name="'Motor_SPEED_INT'" -->
<!-- AWP_In_Variable Name="'WEBDATA".start_web' -->
<!-- AWP_In_Variable Name="'WEBDATA".fault' -->
<!-- AWP_In_Variable Name="'Valve_orange'" -->
<!-- AWP_In_Variable Name="'Global_DB".ukupne_boce' -->
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <title>Filling Station</title>
      <script src="jquery-2.0.2.min.js"></script>
   </head>

   <body>
   </br>
   </br>
<!--START i STOP.....-->
     <form method="post" action="">
        <input type="hidden"  name="'WEBDATA".start_web' value="1"  >
        <input type="submit" value="START">
     </form>
     <form method="post" action="">
        <input type="hidden"  name="'WEBDATA".fault' value="1"   >
        <input type="hidden" name="'WEBDATA".start_web' value="0" >
        <input type="submit" value="STOP">
       </form>
  </br>
  <p1 id="paragraforange" style="color:red" >Valve Orange</p1>
  </br>
  <p2 id="paragrafaplle" style="color:red">Valve Apple</p2>
  </br>
  <p3 id="paragrafvoda" style="color:red">Valve H2O</p3>
```

```
  </br>
  <p4 id="paragrafmixer" style="color:red">Mixer</p4>
  </br>
  <p5 id="paragrafconveyor" style="color:red">Conveyor</p5>
  </br>
  <p6 id="paragraffilling" style="color:red">Valve Filling<p6>
  </br>
  <p7 id="paragraflabelling" style="color:red">Labelling</p7>
</br>
</br>
</br>
<!--Brojac boca.....-->
<p9 id="brojacpar" style="color:blue">Brojač boca:</p9>
<label id="brojac" style="color:blue">:="Global_DB".ukupne_boce:</label>
</br>
</br>
<!--Label prikazuje brzinu motora, 1/2s.....-->
  <p8 style="color:black">Speed:</p8>
  <label id="speed" style="color:black">:="Motor_SPEED_INT":</label>
</br>
</br>
<!--Form za promjenu brzine motora.....-->
  <form method="post">
    <input name='"Motor_SPEED_INT"'type="text" />
    <button type="submit">Save</button>
  </form>
</body>
<!--…….....-->
<script type="text/javascript">
    $(document).ready(function(){
        $.ajaxSetup({ cache: false });
    setInterval(function() {
<!--orang valve.....-->
      $('#oranga').empty();
      $.get("valveorange.html", function(result){
```

```
    $("#oranga").append(result.trim());
    var a = $('#oranga').text();
    if(a=='1'){
     document.getElementById('paragraforange').style.color = "green";
    }
    else {
     document.getElementById('paragraforange').style.color = "red";
    }
    });
<!--aplle valve.....-->
    $('#applelbl').empty();
    $.get("valveaplle.html", function(result){
    $("#applelbl").append(result.trim());
    var b = $('#applelbl').text();
    if(b=='1'){
     document.getElementById('paragrafaplle').style.color = "green";
    }
    else {
     document.getElementById('paragrafaplle').style.color = "red";
    }
    });
<!--voda.....-->
    $('#vodalbl').empty();
    $.get("valvevoda.html", function(result){
    $("#vodalbl").append(result.trim());
    var c = $('#vodalbl').text();
    if(c=='1'){
     document.getElementById('paragrafvoda').style.color = "green";
    }
    else {
     document.getElementById('paragrafvoda').style.color = "red";
    }
    });
<!--mixer.....-->
    $('#mixerlbl').empty();
```

```
      $.get("mixer.html", function(result){
      $("#mixerlbl").append(result.trim());
      var d = $('#mixerlbl').text();
      if(d=='1'){
        document.getElementById('paragrafmixer').style.color = "green";
      }
      else {
        document.getElementById('paragrafmixer').style.color = "red";
      }
      });
<!--Conveyor.....-->
    $('#conveyorlbl').empty();
    $.get("conveyor.html", function(result){
      $("#conveyorlbl").append(result.trim());
      var e = $('#conveyorlbl').text();
      if(e=='1'){
        document.getElementById('paragrafconveyor').style.color = "green";
      }
      else {
        document.getElementById('paragrafconveyor').style.color = "red";
      }
    });
<!--filling.....-->
    $('#fillinglbl').empty();
    $.get("fill.html", function(result){
      $("#fillinglbl").append(result.trim());
      var f = $('#fillinglbl').text();
      if(f=='1'){
        document.getElementById('paragraffilling').style.color = "green";
      }
      else {
        document.getElementById('paragraffilling').style.color = "red";
      }
    });
<!--label.....-->
```

```
        $('#labellinglbl').empty();
        $.get("label.html", function(result){
        $("#labellinglbl").append(result.trim());
        var varija = $('#labellinglbl').text();
        if(varija=='1'){
          document.getElementById('paragraflabelling').style.color = "green";
        }
        else {
          document.getElementById('paragraflabelling').style.color = "red";
        }
        });
      },50);
<!--Uzima brzinu motora.....-->
      setInterval(function() {
          //$('#speed').empty();
          $.get("motor.html", function(result){
            $('#speed').text(result);
          });
        },500);

<!--alarm punjenje.....-->
setInterval(function() {
      $('#labelalarmfill').empty();
      $.get("alarmfill.html", function(result){
        $('#labelalarmfill').append(result.trim());
        var afil = $('#labelalarmfill').text();
        if(afil == '1'){
          alert("Greška! Nema boce na punjenju!");
        }
      });
    },500);
    <!--alarm labell.....-->
    setInterval(function() {
          $('#labelalarmlab').empty();
          $.get("alarmlab.html", function(result){
```

```
        $('#labelalarmlab').append(result.trim());
        var alab = $('#labelalarmlab').text();
        if(alab == '1'){
          alert("Greška! Boca je nestala na putu od punjenja do etiketiranja!");
        }
       });
      },500);

   <!--alarm recept.....-->
   setInterval(function() {
       $('#labelalarmrecept').empty();
       $.get("alarmrecept.html", function(result){
         $('#labelalarmrecept').append(result.trim());
         var arec = $('#labelalarmrecept').text();
         if(arec == '1'){
           alert("Recept nije odabran!");
         }
       });
      },500);
      <!--alarm motor brzina .....-->
      setInterval(function() {
           $('#labelalarmmot').empty();
           $.get("alarmmot.html", function(result){
             $('#labelalarmmot').append(result.trim());
             var amot = $('#labelalarmmot').text();
             if(amot == '1'){
               alert("Greška! Brzina motora je 0!");
             }
           });
          },500);
<!--ukupne boce.....-->
setInterval(function() {
     //$('#brojac').empty();
     $.get("ukupneboce.html", function(result){
       $('#brojac').text(result.trim());
```

```
    });
  },1000);
    });
</script>
<!--labeli skriveni, skripte za promjenu boje paragrafa ih koriste.....-->
<label id="oranga" style="display:none"></label>
<label id="applelbl" style="display:none"></label>
<label id="vodalbl" style="display:none"></label>
<label id="mixerlbl" style="display:none"></label>
<label id="conveyorlbl" style="display:none"></label>
<label id="fillinglbl" style="display:none"></label>
<label id="labellinglbl" style="display:none"></label>
<label id="labelalarmfill" style="display:none"></label>
<label id="labelalarmlab" style="display:none"></label>
<label id="labelalarmrecept" style="display:none"></label>
<label id="labelalarmmot" style="display:none"></label>
</html>
```