UNIVERZA V NOVI GORICI
POSLOVNO-TEHNIŠKA FAKULTETA

NAČRTOVANJE SEKVENČNEGA KRMILJA IN REGULACIJE Z IZBRANIM KRMILNIKOM

DIPLOMSKO DELO

Jure Jezeršek

Mentor: prof. dr. Juš Kocijan

Nova Gorica, 2010

ZAHVALA

Velikokrat si želimo v življenju nekaj doseči, a zaradi pomanjkanja iniciative velikokrat tudi obupamo. Toda obstajajo trenutki, ki dopuščajo pot do uspeha. Eden takih trenutkov je bila zame potrditev mentorja prof. dr. Juša Kocijana glede mojega praktičnega usposabljanja in diplomske naloge. Kot prvemu bi se zato zahvalil profesorju, da mi je bil vedno na voljo za pogosta vprašanja in mi je v težavnih trenutkih priskočil na pomoč. Prav tako bi se zahvalil somentorici dr. Maji Bračič Lotrič ter Univerzi v Novi Gorici, ki mi je omogočila opravljanje praktičnega usposabljanja, na osnovi katerega je bila tudi napisana diplomska naloga. Želim se zahvaliti tudi vsem sodelavcem Poslovno-tehniške fakultete za prijetno vzdušje v času praktičnega usposabljanja ter v preteklih študijskih letih. Zahvalil bi se rad tudi Damjanu iz podjetja Synatec d.o.o. Idrija, ki mi je bil pripravljen pomagati in me je večkrat sprejel v podjetju, kjer smo reševali težje praktične probleme, ter prijateljem in dekletu, ki me je vseskozi spodbujala. Nenazadnje pa bi se rad zahvalil staršem, saj so me vsa ta leta študija neprestano spodbujali in mi stali ob strani. Zato hvala vsem.

NASLOV

Načrtovanje sekvenčnega krmilja in regulacije z izbranim krmilnikom

IZVLEČEK

Diplomsko delo prikazuje načrtovanje vodenja za programirljivi logični krmilnik proizvajalca Moeller. Z njim lahko izvajamo sekvenčno in zvezno vodenje.

V prvem delu, ki obravnava sekvenčno vodenje, so opisane vse potrebne komponente krmilnika, instalacija ter uporaba okolja XSoft. Prav tako je opisana tudi sama modelna naprava za poučevanje sekvenčnega vodenja, ki je v našem primeru simulator pralnega stroja znamke Feedback. Na koncu prvega dela je opisan postopek načrtovanja vodenja. V prilogi je dodan celovit program, ki smo ga napisali na podlagi zahtev za vodenje. Programiran je v strukturiranem tekstu.

Drugi del predstavlja postavitev za zvezno vodenje. Opisan je zvezni regulator izveden na krmilniku proizvajalca Moeller, pri katerem smo v tem delu delali z analognimi signali in ne več z digitalnimi. Nato je opisano okolje XSoft in povezava računalnika s procesom. Zatem je opisana modelna naprava enosmerni električni motor DCMT (DC Motor trainer) za poučevanje regulacije. Poleg tega je opisan še postopek načrtovanja zveznega vodenja. Na koncu pa je po danih zahtevah izvedeno načrtovanje regulacije ter preizkus delovanja s pomočjo izrisovalnika grafov v istem programu XSoft.

Menimo, da so dobro oblikovani praktični primeri zelo uporabni v vsakdanjem življenju, saj lahko pomagajo pri reševanju podobnih primerov avtomatskega vodenja.

KLJUČNE BESEDE

avtomatsko vodenje, modeliranje, načrtovanje regulacije, modelne naprave

TITLE

The design of sequential and continuous control with selected controller

ABSTRACT

The bachelor thesis presents a control design for a programmable logic controller made by Moeller. With this device one can conduct a sequential and continuous control.

The first part of the thesis, in which all the necessary components of the controller, the installation and the use of the XSoft environment are described, considers a sequential control. The first part also describes a sequential control test rig which, in our case, is a simulator of the *Feedback* washing machine. The end of the first part includes the design procedure of the control. The entire program, written on the basis of the control requirements, is added in the supplement and is programmed in the structured text.

The second part presents a continuous control design. We described continuous control implementation on PLC produced by Moeller. In this part, we did not deal with digital signals any more. Instead, we operated with continuous ones. A description of the XSoft environment and a connection between a computer and procedure is followed by a description of the DC motor trainer test rig. Besides, a planning procedure of the continuous control was also included. At the end, the control design and the control validation with the help of a graph renderer in the XSoft program were carried out, according to the given requirements.

We believe that these well designed examples are very useful in an everyday life, as they help to solve similar cases of the automatic control.

KEYWORDS

automatic control, modeling, control design, test rigs

KAZALO

KAZALO SLIK

KAZALO TABEL

1 UVOD

Avtomatizacija je področje tehnike, ki zahteva veliko predznanja. V vsakdanjem življenju se ves čas srečujemo z avtomatsko vodenimi napravami, pa se tega sploh ne zavedamo ali pa nam preprosto ni mar, saj je pomembno le, da naprave delujejo. Take naprave so npr. hladilnik, semafor, bančni avtomat, avtomat za kavo, igralni avtomat itd. Pri izvedbi načrtovanja vodenja se zato srečujemo z bolj ali manj težavnimi primeri. V diplomskem delu smo se odločili prikazati praktična primera načrtovanja vodenja s krmilnikom. Krmilnik lahko uporabljamo tako za sekvenčno vodenje kot tudi za regulacijo. Stremeli smo k temu, da smo čimbolj razumljivo opisali sekvenčno vodenje in regulacijo na praktičnih primerih. Poleg samega programa smo opisali vse potrebne informacije in navodila, ki so potrebna za ponovitev obravnavanih primerov.

Zamisel za diplomsko nalogo izvira iz snovi predmeta Projektiranje in avtomatizacija tehnoloških sistemov, ki je obvezen v drugem letniku študija na Poslovno-tehniški fakulteti Univerze v Novi Gorici. Vsebina tega predmeta je avtomatsko vodenje. V našem primeru smo se odločili dati poudarek načrtovanju, simulaciji ter izvedbi računalniškega vodenja s krmilniki. Pri tem smo sodelovali s podjetjem Kolektor Synatec, ki je eno izmed vodilnih slovenskih podjetij na področju storitev za potrebe avtomatizacije v industriji in gospodarstvu ter na področju elektronske in elektrotehnične opreme.

Namen diplomske naloge je bil predstavitev načrtovanja vodenja s programirljivimi krmilniki. Ključni vprašanji pri tem sta bili, kako krmilnik sploh deluje in kateri krmilnik potrebujemo. Prvi del naloge je predstavil sekvenčno vodenje na osnovi diagrama poteka. Ponazorili smo, kako načrtujemo program za vodenje, ga na podlagi diagrama poteka prenesemo v krmilnik ter opravimo poskus na modelni napravi. Naredili smo popolno dokumentacijo o tem, kako se krmilnik programira, in opisali nekatere funkcije, ki jih je krmilnik zmožen izvajati. Opisali smo navodila za priključitev osebnega računalnika na krmilnik in navodila za namestitev samega programa XSoft.

Drugi del diplomske naloge je predstavitev zveznega vodenja na osnovi regulacijskih knjižic. Ključna vprašanja so bila, kako povežemo modelno napravo s krmilnikom in računalnikom, kako se knjižice za regulacijo prenesejo v krmilnik ter kako se uporabljajo. Opisali smo, kako načrtujemo regulacijo ter kako jo preizkušamo na modelni napravi.

Za prikaz uporabe izbranega krmilnika Moeller in načrtovanje vodenja smo uporabili modelni napravi. Za sekvenčno vodenje je to bil elektronski simulator pralnega stroja proizvajalca Feedback. V drugem delu diplomske naloge pa smo se posvetili regulaciji hitrosti enosmernega električnega motorja znamke Quanser.

2 AVTOMATSKO VODENJE PROCESOV

Namen sistemov za vodenje je ta, da »pomagajo človeku voditi procese ali pa ga pri tem celo nadomeščajo«. (Strmčnik, (ur.), 1998, str. 78) Učinkovito delovanje tovrstnih sistemov se kaže skozi različne učinke, kot so povečanje proizvodnje, racionalizacija energije, zmanjševanje onesnaževanja okolja, racionalizacija surovin in izboljšanje kvalitete samega produkta.

V naslednjih podpoglavjih bomo predstavili zgodovino avtomatizacije skozi faze njenega razvoja. Opisali bomo, kaj je sistem in kaj proces ter katere zakonitosti se uporabljajo za njihov prikaz. Povedali bomo, kakšen je namen sistemov za vodenje ter kakšne so njihove funkcije. Navedli bomo, iz česa so narejeni in kako ti sistemi delujejo. Opisali bomo torej mehanizme vodenja ter prikazali njihovo strukturo.

2.1 Splošno o nastanku avtomatizacije oziroma računalniškega vodenja procesov

Razvoj avtomatizacije je tesno povezan z odkritjem parnega stoja (J. Watt, leta 1769) ter z mehanizacijo tkalnega stroja (Cartweight, leta 1784–86), odkritjem parne lokomotive (Stephenson, leta 1814), z odkritjem principa elektrodinamike (Siemens, leta 1866) in osnovnimi znanji iz kemijske tehnologije ter z odkritjem Ottovega (leta 1876) in Dieselovega motorja (leta 1893-97) ter s tem povezano avtomobilsko industrijo in predelavo nafte. Tem štirim ciklom strokovnjaki dodajajo še petega z odkritjem tranzistorja (Shockley, leta 1948), čemur je sledil hiter razvoj miniaturizacije elektronskih naprav ter informatike. Pojav in uporaba digitalnih računalnikov sredi dvajsetega stoletja pa je avtomatsko vodenje vodila k pravemu razcvetu in njegovem pojavljanju na skoraj vseh področjih človeške dejavnosti.

Prvi primer regulatorja je bil uporabljen prav pri parnem stroju, in sicer kot centrifugalni regulator. Avtomatizacija proizvodnih procesov se je začela intenzivneje razvijati v 30-ih letih 20. stoletja, zelo važno prelomnico v razvoju konceptov in uporabe avtomatskih sistemov za vodenje pa je predstavljala druga svetovna vojna. V tem času so razvili enostavne elektromehanske regulacijske in krmilne elemente. Vse večje potrebe po kvalitetni regulaciji, po avtomatskem zagonu in ustavitvi ter regulaciji na konstantno referenčno vrednost pa so zahtevale stalna izpopolnjevanja in izboljšave do razvoja elektronk, transduktorjev in

tranzistorskih ojačevalnikov. Končno pa je v petdesetih letih konstantni razvoj pripomogel k nastanku elektronskih regulacijskih krmilnih elementov za specialna področja. Z razvojem računalniške tehnologije so se sredi šestdesetih let pričeli uporabljati procesni računalniki kot univerzalni regulacijski in krmilni sistemi. Od prvotne centralizacije regulacijskih in krmilnih funkcij v enem samem računalniku zaradi zmanjševanja stroškov so kaj kmalu ugotovili, da je višjo fleksibilnost delovanja in varnost obratovanja možno doseči le z decentralizacijo avtomatskih nalog. Razvoj vezij visoke integracije je omogočil razvoj mikroračunalnikov, ki so temeljito spremenili strukturo sistemov računalniškega vodenja procesov.

Avtomatizacija je uporaba sistemov za vodenje (kot so numerično vodeni sistemi, programirljivi logični krmilniki ter industrijski regulatorji), ki z uporabo računalnikov ter druge informacijske tehnologije omogoča njihovo uporabo. Z avtomatskim vodenjem zmanjšamo potrebo po človeškem posredovanju.

2.2 Sistemi in sistemska teorija

»Izvor besede sistem izhaja iz grščine, njen pomen pa bi lahko povzeli kot neko urejeno tvorbo, ki predstavlja nasprotje naključnosti ali kaosa. Najbolj splošna definicija pravi, da je sistem množica elementov, ki imajo medsebojne relacije in relacije z okoljem. Te relacije pa se izražajo kot neka izmenjava snovi, energije in informacije. Bistvo sistema je, da predstavlja več kot vsoto posameznih komponent, kar pomeni, da sistem dobi neko novo kvaliteto z različnimi povezavami njegovih sestavnih delov ali komponent, ki rezultirajo v sinergiji, prav ta sinergija pa predstavlja neko dodano vrednost sistema.« (Strmčnik, (ur.) 1998)

Beseda proces pa izhaja iz latinske besede »procedere« (napredovati) in pomeni nek potek, postopek ali dogajanje. Strmčnik pravi, da »tak potek, postopek oz. dogajanje povzroči neko spremembo stanja v sistemu, pri tem pa je ključnega pomena čas, preko katerega lahko opazimo spremembo stanja. Če se torej pojavi v sistemu neka izmenjava snovi, energije in informacije v različnih časovnih intervalih, lahko zaključimo, da znotraj sistema poteka nek proces. Sistem je torej nek okvir, v katerem potekajo procesi, ali pa sredstvo, ki omogoča potek procesa. Proces pa je vsebina sistema ali njegovo življenje.« (Strmčnik, (ur.) 1998, str. 17) Shematsko je proces prikazan na sliki 2.1.

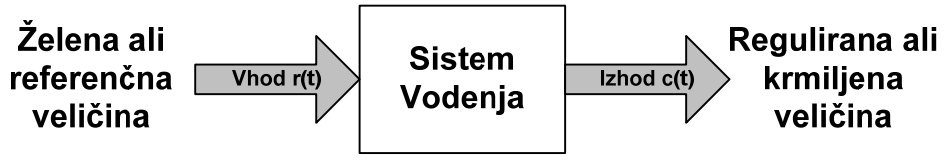
Slika 2.1: Shematski prikaz procesa

Sistemov in procesov je več vrst, najbolj pa nas v diplomski nalogi zanimajo tehnični procesi, ki so skupek soodvisnih potekov v nekem sistemu, ki rezultirajo v transformaciji, transportu ali skladiščenju materije, energije ali informacije.

»Sistemska teorija se torej ukvarja s takšnimi lastnostmi sistemov, ki so skupne za vse sisteme. Lastnost sistemov, ki je očitno skupna, je vodenje in vsak sistem ima neko obliko vodenja, ki teži k ohranjanju integritete sistema. V tem smislu je vodenje torej pogoj za obstoj nekega sistema.« (Strmčnik, (ur.), 1998, str. 25)

2.3 Vodenje

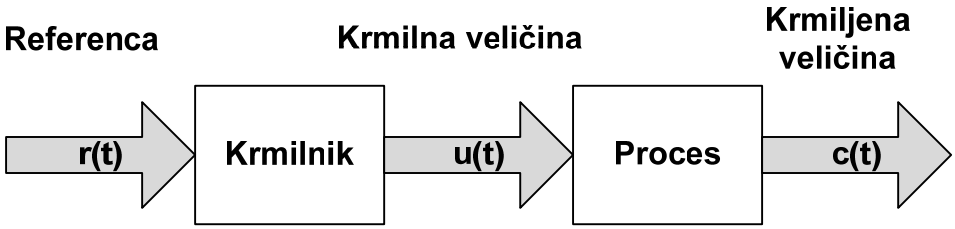
Diplomska naloga obravnava predvsem sisteme z notranjimi procesi (aktivnostmi), pri katerih je sedanje stanje odvisno od preteklih stanj. Definicija vodenja po Strmčniku je torej: »Vodenje je proces, s katerim vplivamo na delovanje (obnašanje) sistema z namenom, da dosežemo nek zastavljen cilj.« (Strmčnik, (ur.), 1998, str. 25) Sistem vodenja lahko ponazorimo z blokom, kot ga prikazuje slika 2.2.



Slika 2.2: Shematski prikaz sistema vodenja

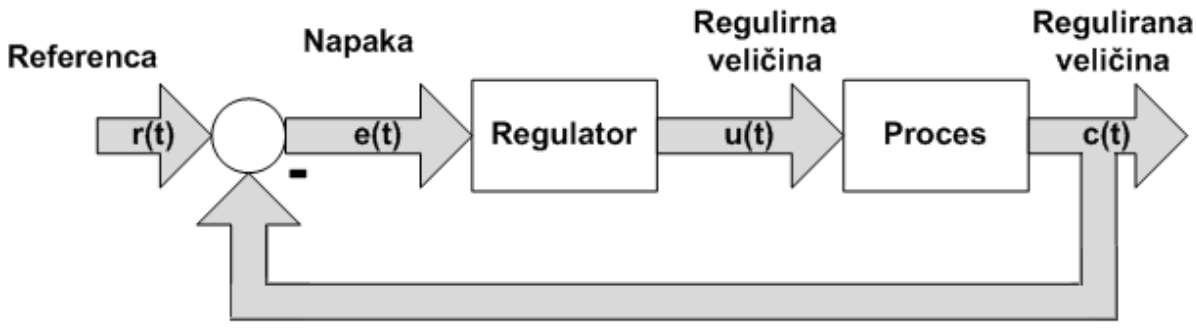
Izhodu sistema pravimo glede na vrsto sistema regulirana ali krmiljena veličina, vhodu pa želena ali referenčna veličina. Zahteva sistema vodenja je ta, da se regulirana ali krmiljena veličina čimbolj ujema z referenčno oziroma želeno veličino ne glede na motnje, ki vplivajo na sistem vodenja.

Poznamo dva principa vodenja, to sta odprtozančno ter zaprtozančno vodenje. Pri prvem gre za princip vodenja, pri katerem dejansko ne preverjamo, če se sistem obnaša tako, kot smo predvideli. Torej tak sistem, kot ga prikazuje slika 2.3, nima povratne zanke, s katero bi krmilnik sistema dobil povratno informacijo, na osnovi katere bi vplival na krmiljeno veličino. V tem primeru referenca *r(t)* (referenčna veličina, želena veličina) deluje na krmilnik, ki s pomočjo veličine *u(t)* zagotavlja, da je izhod procesa oz. krmiljena veličina v določeni korelaciji z referenčno veličino.



Slika 2.3: Bločni diagram odprtozančnega vodenja

Na drugi strani pa je zaprtozančni sistem vodenja, kot ga prikazuje slika 2.4, tak, da ima povratno zanko iz izhodne proti vhodni ali referenčni veličini. »Bistvo tega sistema je v zagotavljanju, da se informacija o dejanskem stanju sistema (informacija izhoda) primerja z informacijo o želenem stanju sistema (informacija vhoda), razlika pa povzroči akcijo, ki popravlja dejansko stanje.« (Strmčnik, (ur), 1998) »Če želimo dobiti točnejšo regulirano veličino *c(t)*, jo moramo primerjati z referenco *r(t)*, regulator pa definira ustrezno regulirano veličino na osnovi napake tako, da ta zmanjšuje napako.« (Zupančič, 1992).



Slika 2.4: Bločni diagram zaprtozančnega regulacijskega sistema

2.3.1 Mehanizmi sistemov za vodenje

Mehanizme sistemov za vodenje lahko razdelimo na naslednji način:

a) Mehanizmi opazovanja:

- Zajemanje, pretvorba in prenos podatkov. Osnovo tega segmenta predstavljajo različni merilni principi, katerih bistvo je pretvorba fizikalne (kemijske) veličine v (najpogosteje) električni signal, ki ga je nato mogoče nadalje obdelovati in prenašati. Primeri so: senzor premika, tlaka, temperature, pretoka itd.
- Obdelava podatkov in ugotavljanje stanja procesa, opreme in proizvoda, pri čemer imamo v mislih računalniško obdelavo podatkov, uporabo različnih matematičnih modelov ter formul za izračun izpeljanih veličin.
- Prikazovanje, protokoliranje, arhiviranje in posredovanje podatkov. V tem segmentu so prevladujoči mehanizmi, ki se nanašajo na strukturo, organizacijo in manipulacijo s podatki.

b) Mehanizmi razmišljanja in odločanja:

- Primerjava dejanskega in želenega obratovanja.
- Priprava ukrepov.
- Odločanje.

c) Mehanizmi ukrepanja

Tu so najpomembnejši principi, ki omogočajo pretvorbo informacije o potrebnem ukrepu v ustrezno fizično akcijo, kjer gre najpogosteje za pretvorbo električnega signala v ustrezen mehanski premik v okviru izvršnega sistema. Najpogosteje so uporabljeni elektrohidravlični, elektropnevmatski in elektromotorni mehanizmi.

2.3.2 Struktura sistemov za vodenje

»Strukturo definiramo iz gradnikov sistema in povezav, torej iz česa je sistem narejen in kako je sestavljen.« (Strmčnik, (ur.) 1998, str. 97) Sistemi za vodenje so sestavljeni iz petih ključnih podsistemov:

- Merilni sistemi (senzorji) predstavljajo čutila sistema za vodenje. Njihova funkcija je, da pretvorijo neko fizikalno ali kemijsko veličino v nek (najpogosteje električni) signal. Najpogostejši so merilniki tlaka, premikov, temperature itd.
- Izvršni sistemi (aktuatorji) so gradniki, s katerimi je mogoče v procesu povzročiti spremembo v pretoku energije ali snovi. Njihova funkcija je, da pretvorijo neko informacijo v ustrezno spremembo na procesu.
- Komunikacijski sistemi predstavljajo sistem za prenos informacij med različnimi gradniki v sistemu vodenja. Pri tem gre bodisi za komunikacijo med stroji oz. napravami samimi ali za komunikacijo med strojem in človekom z uporabo raznih vmesnikov, kot je npr. zaslon ali tipkovnica.
- Računalniške in druge naprave, ki prevzamejo večino funkcij »opazovanja in razmišljanja«. Sem spadajo gradniki, kot so programirljivi logični krmilniki (angl. programmable logic controller (PLC)) za sekvenčno vodenje, regulatorji za regulacijo ter procesni računalniški sistemi.
- Človek, ki še vedno predstavlja ključni del sistema za vodenje procesov in lahko občasno prevzame vlogo ostalih štirih navedenih podsistemov.

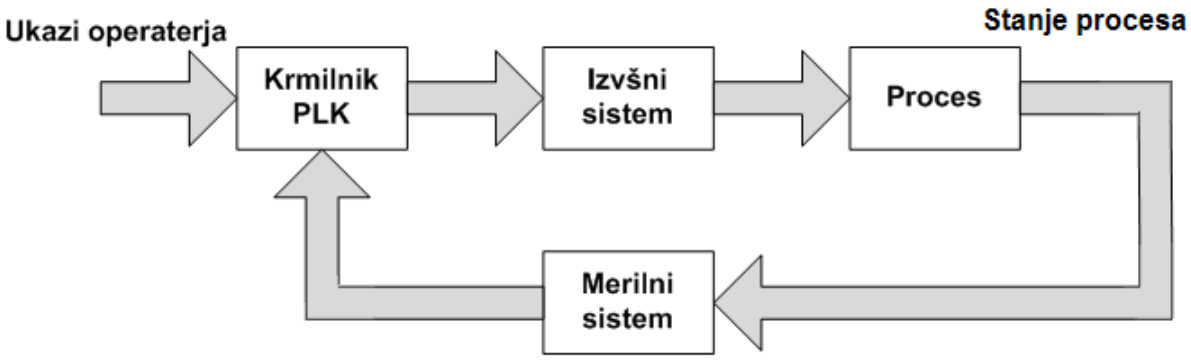
2.4 Metode in postopki za izvedbo funkcij vodenja

Pri izvedbi metod in postopkov za izvedbo funkcij vodenja najpogosteje uporabljamo dve različni možnosti funkcij vodenja, in sicer sekvenčno vodenje ter regulacijo.

2.4.1 Sekvenčno vodenje

Predstavlja ga predpisano zaporedje delovnih operacij, ki naj se izvršijo namesto predpisanih vrednosti posameznih veličin. Cilj sistema za avtomatsko vodenje na tem mestu je zagotavljanje pravilnega zaporedja operacij – od tod tudi ime sekvenčno vodenje. »Sekvenčno vodenje je definirano kot vodenje procesa skozi zaporedje diskretnih stanj, s čimer dosežemo določen cilj vodenja. Diskretno stanje procesa predstavlja množica stanj vseh operativnih enot procesa. To so npr. končna stikala, tipke, ventili, motorji. Stanja posameznih enot lahko zavzamejo le dve vrednosti. Opisujejo jih izrazi kot: stikalo S1 je sklenjeno, tipka T1 je pritisnjena, ventil V5 je zaprt, motor M2 je vključen itd. Izvajanje določene operacije pomeni ustrezno kombinacijo stanj posameznih enot in ustreza enemu ali zaporedju več diskretnih

stanj.« (Strmčnik, (ur.), 1998, str. 241) Iz definicije je mogoče sklepati, da si pri tej obliki vodenja številne operacije sledijo v določenem zaporedju, kjer se naslednja operacija izvrši šele, ko je predhodna že končana. Sistem sekvenčnega vodenja je prikazan na sliki 2.5.



Slika 2.5: Shema sekvenčnega vodenja

Stanje procesa se spreminja ob nastopu diskretnih dogodkov, npr. stikalo je pritisnjeno, sproži se nek postopek, ki omogoča vrtenje elektromotorja, dokler ne doseže neke mejne vrednosti, ki jo senzor zazna in izklopi električno napajanje motorju, posledično se motor ustavi. Sistem vodenja mora zagotavljati, da si dogodki in stanja sledijo v predpisanem zaporedju, s čimer dobimo predpisano zaporedje delovnih operacij, in predstavljajo nek postopek. Nekatere operacije si sledijo zaporedno ali pa tečejo vzporedno.

Glavni element sistema vodenja predstavlja krmilnik, s katerim vodimo proces. Krmilnik zagotavlja pravilno zaporedje stanj in dogodkov. Možna je tudi programska izvedba ukazov, ki se nato prenesejo v krmilnik in jim ta zna slediti. Izhodi krmilnika preko krmilnih signalov prožijo aktuatorje v procesu in omogočajo nastop naslednjega dogodka.

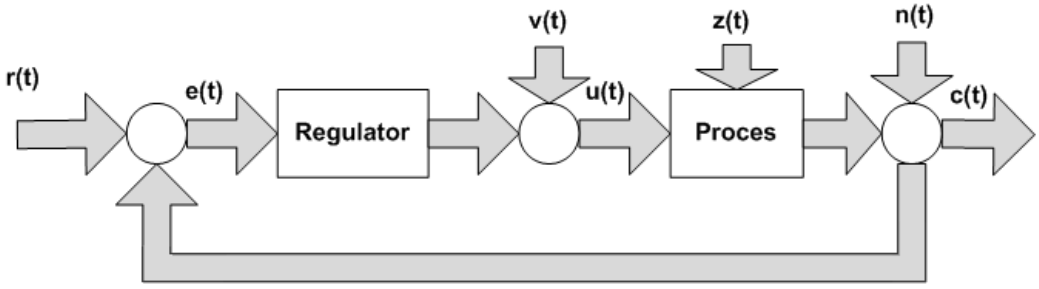
S stališča krmilnika diskretno stanje procesa predstavlja množica vrednosti vhodnih in izhodnih spremenljivk. Število stanj procesa, ki jih krmilnik lahko zaznava, je odvisno od njihovega števila. V primeru, da ima krmilnik tri vhodne in tri izhodne spremenljivke, dobimo, z ozirom da so spremenljivke binarne (angl. Boolean), 64 možnih diskretnih stanj.

Vendar pa običajno v procesu ne nastopajo vsa možna stanja. Poleg stanj procesa ima lahko krmilnik še dodatna notranja stanja. Število teh je odvisno od kompleksnosti vodenja.

Izvedb krmilnikov je več vrst, dandanes pa se najpogosteje uporabljajo programirljivi logični krmilniki, ki so izrinili relejska logična vezja in polprevodniška logična vezja, uporabljena v preteklosti.

2.4.2 Regulacija oziroma zvezno vodenje

»Regulacija je povratnozančno vodenje procesa, pri katerem vplivamo na proces tako, da se veličine, ki jih reguliramo, čimbolj ujemajo z želenimi (referenčnimi) veličinami ne glede na motnje, katerim je regulacijski sistem podvržen.« (Strmčnik, (ur.), 1998, str. 255) Bločno shemo regulacijskega sistema za zvezno vodenje prikazuje slika 2.6, kjer sta izvršni in merilni sistem vključena v regulator oz. proces in je *r(t)* referenca, *e(t)* pogrešek, *u(t)* je regulirna, *c(t)* pa regulirana veličina. *v(t)* predstavlja motnjo na vhodu procesa, *n(t)* pa motnjo na izhodu, medtem ko motnja *z(t)* deluje v procesu samem.



Slika 2.6: Bločna shema regulacijskega sistema v prisotnosti motenj

»Delovanje regulacijskega sistema za vodenje stremi k temu, da regulacijski sistem odpravlja motnje oziroma, da le-te čim manj vplivajo na regulirano veličino. Dober regulator izdatno zaduši vpliv motnje in ga v čimkrajšem času izniči. Vendar je ta čas omejen z dinamiko procesa, saj se motnja *v(t)* prenaša na regulirano veličino preko dinamike procesa. Pri načrtovanju regulacijskega delovanja je torej manj pomembno, kako regulacijski sistem sledi morebitnim spremembam reference.« (Zupančič, 1992, str. 6)

Razlika med bločnim diagramom zaprtozančnega regulacijskega sistema (slika 2.4) in bločno shemo regulacijskega sistema v prisotnosti motenj (slika 2.6) je samo v tem, da so na slednji

prikazane motnja na vhodu procesa (*v(t)*), motnja v samem procesu (*z(t)*) in motnja na izhodu procesa (*n(t)*).

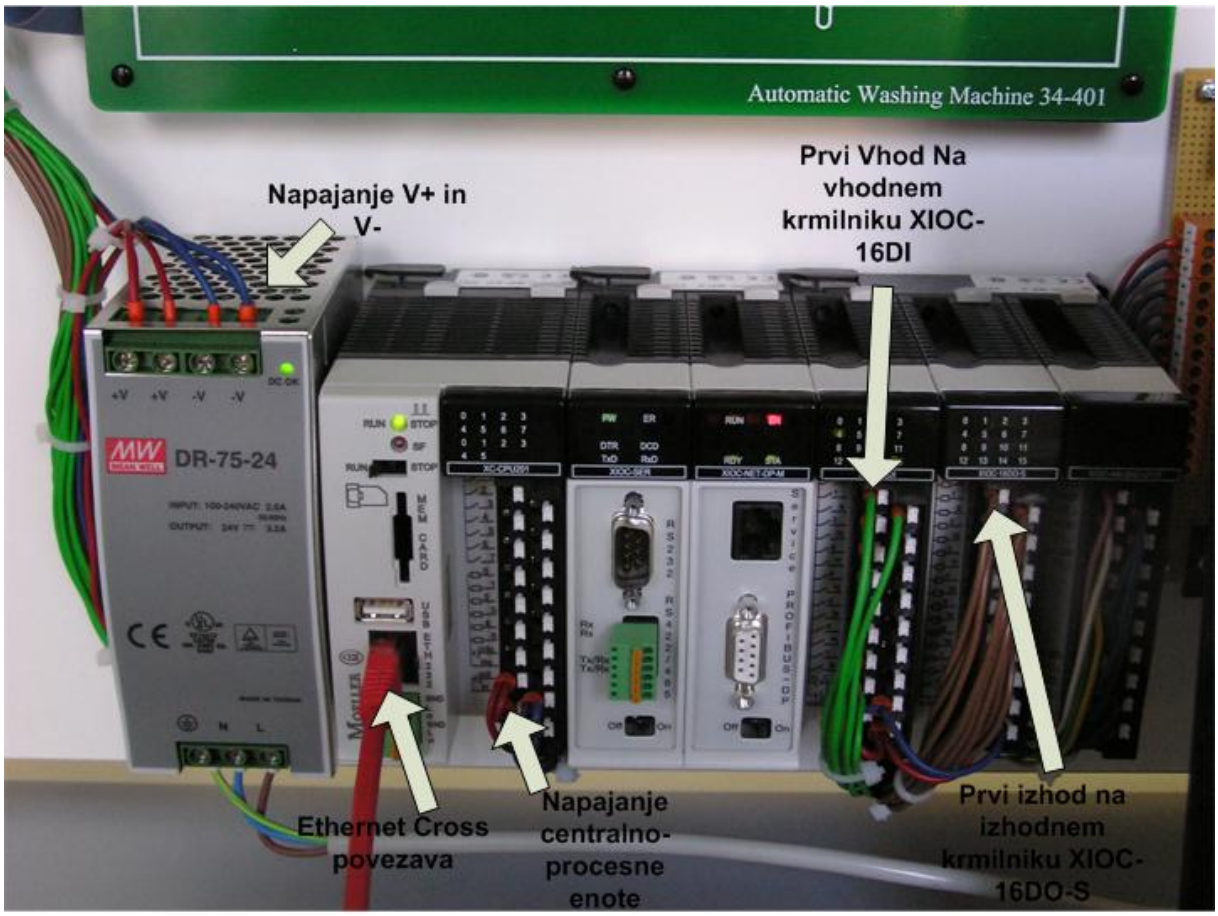
3 OPIS KOMPONENT KRMILNIKA

»Programirljivi logični krmilnik je osnovno orodje, ki je danes na razpolago za avtomatizacijo industrijskih procesov in naprav. PLC je na mikroprocesorju zasnovana krmilna naprava, ki je bila razvita kot nadomestek relejne logike. V osnovi je namenjena za realizacijo sekvenčnega vodenja, omogoča pa tudi reševanje enostavnejših regulacijskih nalog.« (Strmčnik, 1998, str. 561 in 562) Osnovna funkcija PLC-ja je krmiljenje izhodov na osnovi stanja vhodov.

- PLC je dejansko mikroračunalnik, katerega shema in programska oprema je prilagojena za delo v industrijskem okolju. Sestavljen je iz štirih ključnih elementov (Strmčnik, (ur), 1998). To so:
- Centralna procesna enota (CPU). To so »možgani krmilnika«. Vsebuje enega ali več mikroprocesorjev, ki krmilijo delovanje PLC-ja in izvajajo uporabniški program. CPU krmili tudi komunikacije in povezave z drugimi enotami sistema. Sistemski program je shranjen v stalni pomnilnik (ROM), spremenljivke v dinamični pomnilnik (RAM), uporabniški program pa v dinamični pomnilnik ali programsko nastavljivi pomnilnik (EEPROM).
- Napajalnik. PLC običajno potrebuje napajanje z 220V izmenične napetosti ali 24V enosmerne napetosti. Napajalnik skrbi za potrebne nižje enosmerne napetosti za delovanje CPU-ja in vhodno-izhodnih enot.
- Vhodne enote: skrbijo za sprejem zunanjih signalov (digitalnih in analognih), pretvorbo signalov v nivoje, potrebne za nadaljnjo obdelavo in zaščito CPU pred motnjami iz okolja. Stanje digitalnih vhodov se običajno prikazuje s svetlobnimi diodami za diagnostiko delovanja PLC-ja.
- Izhodne enote: skrbijo za prenos rezultatov izvedbe programa krmilnika na izhodne signale (digitalne in analogne) in za zaščito CPU-ja pred motnjami iz okolja. Stanje digitalnih izhodov se običajno prikazuje s svetlobnimi diodami.

Za izvedbo sekvenčnega vodenja potrebujemo proces, opremljen z merilnikom in izvršilnim sistemom, krmilnik in vmesnik. Potrebujemo pa tudi programsko opremo za izvedbo programa. Programska oprema je običajno dobavljena skupaj z nabavljenim krmilnikom. Ta program omogoča načrtovanje vodenja in prenos programa na PLC.

Za izvedbo našega praktičnega dela smo uporabili krmilno konfiguracijo s krmilnikom proizvajalca Moeller, model XC200, ki je krmilnik višjega zmogljivostnega razreda. Krmilnik, ki ga prikazuje slika 3.1, je opremljen z ustreznim naborom vhodno-izhodnih enot za binarne (digitalni vhodi in izhodi) in zvezne (analogni vhodi in izhodi) signale skupaj s programsko opremo Xsoft za programiranje krmilnikov po standardu IEC 61131-3 in knjižnicami za regulacijo.



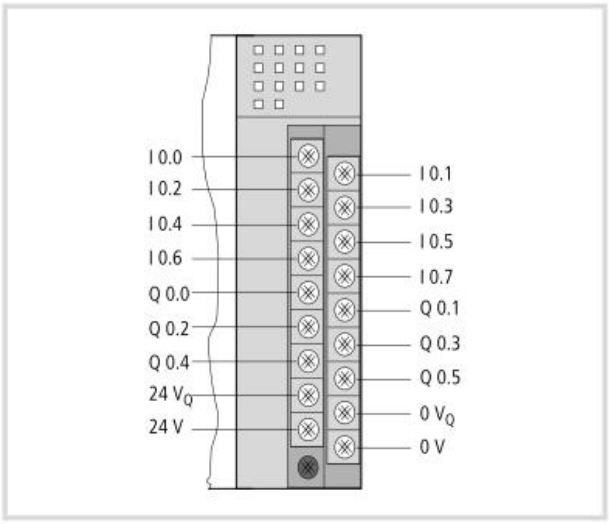
Slika 3.1: PLC proizvajalca Moeller

3.1 Krmilnik proizvajalca Moeller s programskim okoljem XSoft za sekvenčno vodenje

3.1.1 Opis krmilnika Moeller XC200

Modularni krmilnik serije XC200 nudi visoko procesorsko kapaciteto in je zasnovan za uporabo na industrijskih strojih in izgradnjo sistemov za nadzor. Krmilnik je sestavljen iz procesorske enote in modulov za razširitev. Za delovanje mora imeti krmilnik tudi napajalnik (slika 3.1), ki pa ni del krmilnika. Na sliki 3.2 je prikazana zunanjost procesorske enote XC-CPU201. Levo zgoraj na sliki 3.2 je glavno stikalo, s katerim zaženemo program, ki je shranjen v pomnilniku. Desno zgoraj so indikatorji LED, ki prikazujejo, kateri izhodi so med izvajanjem programa v krmilniku aktivni. Kratica ETH/232 pa pove, da lahko modul XC-CPU201 povežemo z osebnim računalnikom preko vmesnika RS-232 ali pa prek Ethernet Cross kabla. Pod plastičnim pokrovom na desni strani je šest digitalnih izhodov in osem digitalnih vhodov ter priključki za priklop napajanja (natančneje prikazano na sliki 3.3).

Slika 3.2: Zunanjost centralno procesne enote modula XC-CPU201

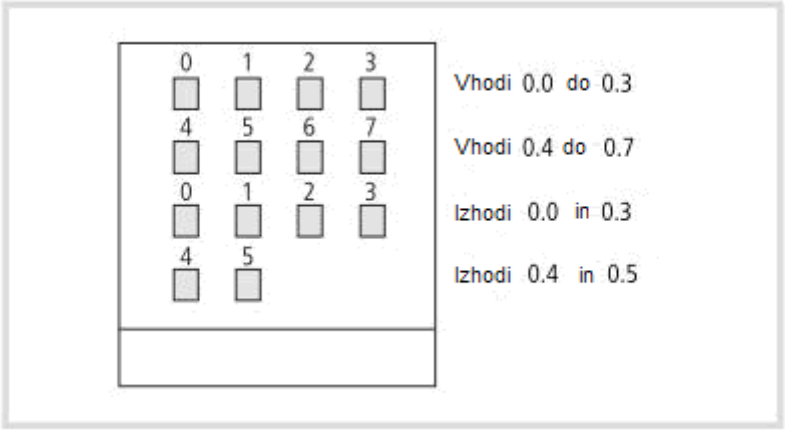


Slika 3.3: Priključne sponke za napajanje (24VQ, 24V, 0VQ, 0V), digitalni vhodi (I0.0 – I0.7) in digitalni izhodi (Q0.0 – Q0.5)

Na sliki 3.3 je skica vhodov, izhodov ter napajanja za Moellerjev krmilnik družine XC200. 0VQ/+24VQ je napajalna napetost za lokalne vhode/izhode, 0V/+24V pa napajalna napetost za procesorsko enoto.

3.1.2 Indikatorji LED

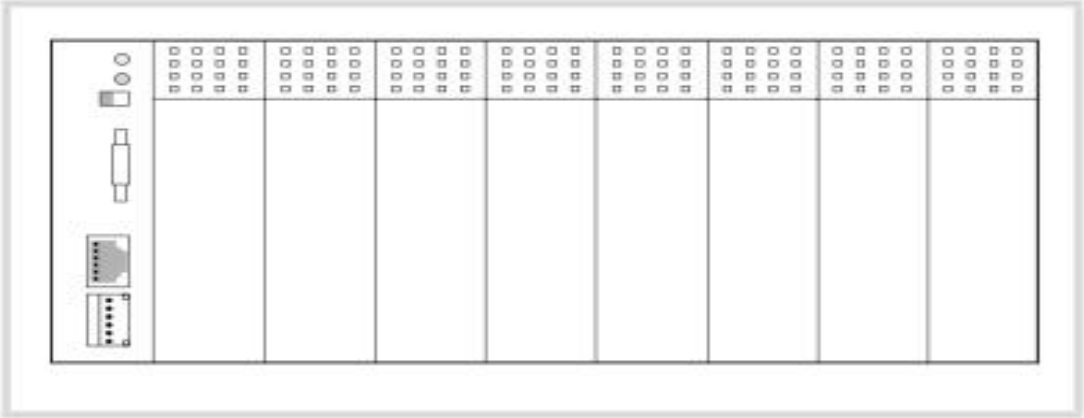
Na sliki 3.4 vidimo shemo indikatorjev LED pri procesorski enoti ter pri vhodno-izhodnih modulih.



Slika 3.4: Indikatorji LED pri CPU201 ter pri vhodno-izhodnih modulih

Na sliki 3.4 so prikazani indikatorji LED, ki označujejo vhode z od 0.0 do 0.7 ter izhode z od 0.0 do 0.5. Ko posamezna dioda sveti, pomeni, da je tisti vhod/izhod aktiven (H- level). K sami procesorski enoti (CPU201) dodajamo vhodno-izhodne module.

Na napravo lahko priključimo največ 15 XI/OC vhodno-izhodnih modulov (slika 3.5). Vhodni modul uporabimo za nadzor tipk in stikal, izhodni modul pa za indikatorje LED. V našem primeru imamo za sekvenčno vodenje v konfiguraciji vhodni modul (XIOC-16DI) in izhodni modul (XIOC-16DO). Za regulacijo zveznih sistemov pa lahko priključimo modul XIOC-4AI-2AO-U1-I1, ki je analogni vhodno-izhodni modul, ki lahko sprejema ali oddaja različno jakost električnega toka (med 4 in 20 mA) ali napetosti (med 0 in 10 V). Tipično zunanjo postavitev XC-CPU201 z XIOC razširitvenimi moduli lahko vidimo na sliki 3.5. Na sliki 3.1 pa lahko vidimo našo konfiguracijo razširitvenih modulov ter povezavo med njimi.



Slika 3.5: Tipična postavitev procesorske enote krmilnika XC200 z vhodno-izhodnimi razširitvenimi moduli

4 INSTALACIJA PROGRAMA IN OKOLJE XSoft

XSoft je razvojno okolje za naš PLC. Omogoča preprost pristop programerja k IEC (*International Electrotechnical Commission*) programiranju. Uporaba urejevalnikov (***Editor***) ter razhroščevalnih funkcij (***Rebuilt all***) temelji na dokazanih razvojnih programskih okoljih naprednih programskih jezikov, kot so C++ in Visual. Program lahko simuliramo ter po korakih ali v celoti pregledujemo sintakso. Dodatna funkcija programa XSoft je na primer možnost nastavitev vhodno-izhodnih spremenljivk na določene vrednosti. Celoten projekt lahko kadarkoli izvozimo v tekstovno datoteko.

Začnemo z vstavitvijo zgoščenke, ki vsebuje program *XSoft-Professional*. Sledimo navodilom pri instalaciji. Najbolje, da ne spreminjamo priporočenega mesta instalacije, saj bo v tem primeru vedno potrebno na novo določiti izvorno mesto datotek, ki jih program potrebuje za delovanje. Sedimo sprotnim navodilom do konca nameščanja programa. Po končanem nameščanju lahko program zaženemo.

4.1 Preizkušanje in nastavljanje raznih parametrov v ukazni vrstici PLC – Browser

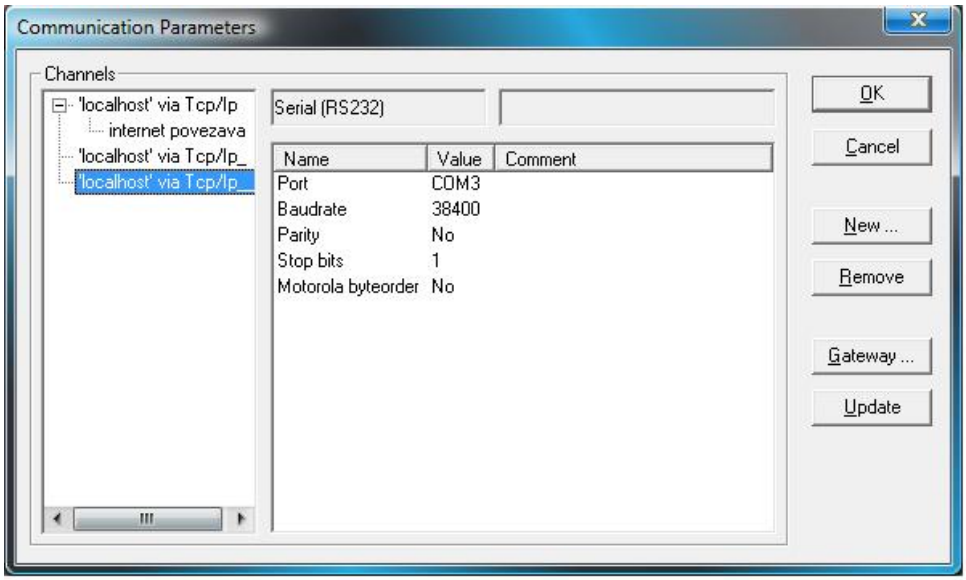
V kolikor smo povezani s PLC-jem prek računalnika, lahko z naslednjim terminalom programiramo in nadziramo sistem PLC-ja. Gremo pod jeziček ***Resources*** in dvakrat kliknemo na ***PLC – Browser***. Odpre se terminalno okno, v katerega lahko pišemo ukaze. Z osnovnim ukazom ***?*** XSoft izpiše spisek ukazov, ki jih PLC lahko izvede in poizvede iz PLC-ja. Tako lahko na primer z ukazom ***memdisk_sys*** vidimo, koliko je še prostega prostora na disku PLC-ja. Z ukazom ***setpwd*** lahko nastavimo geslo, ki ga bomo potrebovali za dostop do PLC-ja. Ukaz ***filedir*** bo izpisal vse datoteke, ki so na samem PLC-ju, ***z getipconfig*** pa bo program izpisal trenutne IP-nastavitve. To so samo nekateri od ukazov.

4.2 Povezava med osebnim računalnikom in krmilnikom

Da bi omogočili krmilniku izvajanje želenega programa, moramo programsko kodo prenesti iz osebnega računalnika na pomnilnik krmilnika. To storimo tako, da računalnik povežemo s krmilnikom prek Ethernet Cross kabla ali kabla RS-232.

4.2.1 Povezava krmilnika z računalnikom preko vmesnika RS-232

V telekomunikacijah je RS-232 priporočen standard za serijski prenos binarnih podatkov. Za priključitev se običajno uporablja serijska vrata (COM 1). Standard podpira tako sinhrono kot asinhrono pošiljanje podatkov. Omogočen je tudi obojestranski tok podatkov. Če želimo vmesnik RS-232 priklopiti med našim osebnim računalnikom ter krmilnikom, naredimo naslednje: v XSoftu gremo na ***Online→Communication Parameters*** ter kliknemo ***New***. Namesto ***TCP/IP (Level 2 Route)*** kliknemo ***Serial (RS-232)*** in povezavi izberemo ime (slika 4.1). Nato izberemo ***COM1*** ali ***COM2***, odvisno od mesta priključitve. Če imamo na računalniku samo ena COM vrata, bo program sam izbral COM1 vrata.



Slika 4.1: V Xsoft določimo povezavo prek vmesnika RS-232

4.2.2 Povezava z Ethernet Cross kablom

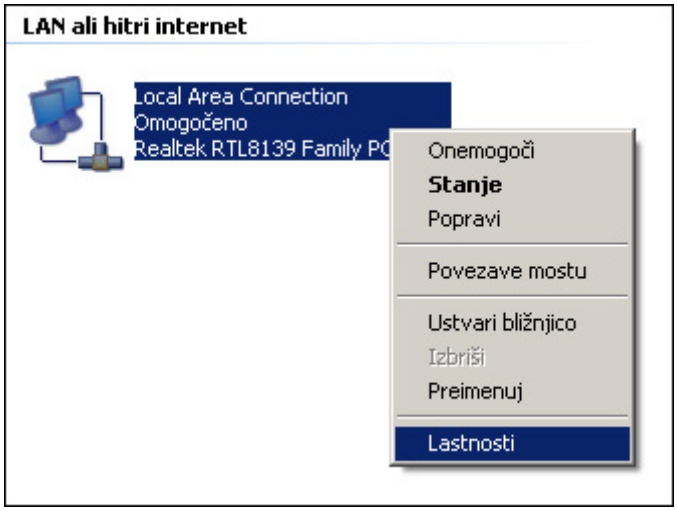
Ob prevzetju PLC-ja je krmilniku priložen kabel RS-232, ki ga uporabljamo za komuniciranje med PLC-jem in osebnim računalnikom. V kolikor na računalniku nimamo vmesnika RS-232, imamo pa mrežni vmesnik, lahko uporabimo Ethernet Cross kabel (XT-CAT5-X-2 ali XT-CAT5-X-5). Da bi ugotovili IP-naslov (kratica IP označuje Internet Protocol in natančno določa računalnik v omrežju), na katerem je dosegljiv naš krmilnik, si lahko naložimo brezplačen program Netscan. Program je brezplačen in ga lahko najdemo na svetovnem spletu (SoftPerfect, 2009) . S tem programom ugotovimo dejanski IP-naslov PLC-ja, saj ga je možno programsko spremeniti (več v podpoglavju 4.2). Da bi zagotovili povezavo z osebnim

računalnikom, bomo morali določiti IP-naslov osebnega računalnika v enako družino kot IP-naslov PLC-ja.

4.2.3 Nastavitev sorodnega IP-ja na računalniku in krmilniku

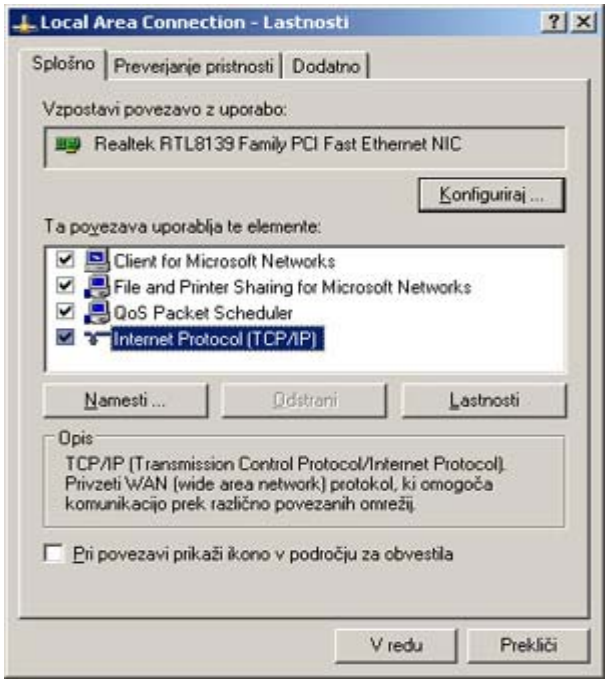
Postopki nastavljanja, ki so opisani v tem dokumentu, se lahko razlikujejo glede na operacijski sistem. Tako je besedilo pripravljeno za operacijski sistem Windows 98, slike pa so za lažje razumevanje uporabnikov ostalih operacijskih sistemov povzete iz operacijskega sistema Windows XP. V glavnem se te nastavitve ne razlikujejo in se konfigurirajo po enakem principu.

V opravilni vrstici kliknemo gumb ***Start***, pokažemo na ***Settings*** (Nastavitve) in kliknemo ***Control Panel*** (Nadzorna plošča). Na nadzorni plošči dvokliknemo ikono ***Network*** (Omrežje) (slika 4.2).



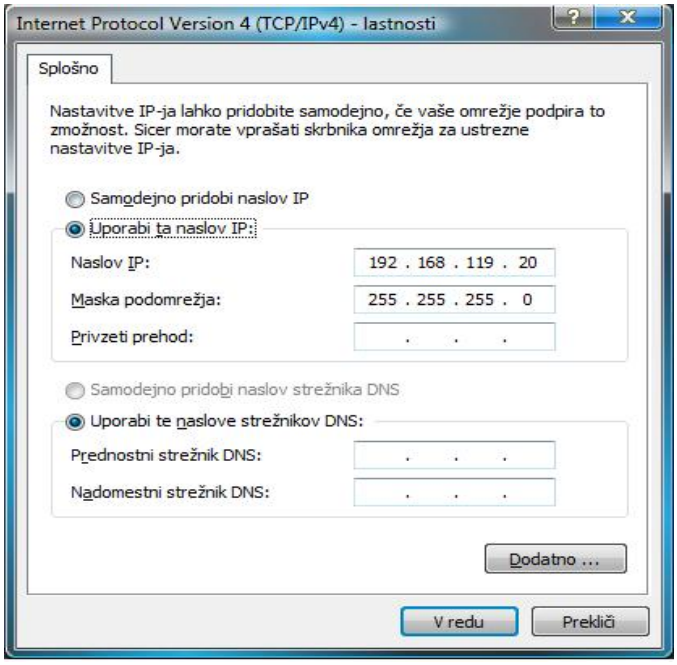
Slika 4.2: Local Area Connection (Omrežna soseščina)

V pogovornem oknu ***Network*** (Omrežje) morata biti na kartici ***Configuration*** (Konfiguracija) nastavljena računalnikova omrežna kartica in protokol TCP/IP, ki mora biti vezan na računalnikovo omrežno kartico (slika 4.3).



Slika 4.3: Internet Protocol (TCP-IP) in konfiguracija

Kliknemo protokol ***TCP/IP*** in nato kliknemo gumb ***Properties*** (Lastnosti). V pogovornem oknu ***TCP/IP Properties*** (TCP/IP - lastnosti) najprej kliknemo jeziček ***IP Address*** (IP naslov). Kliknemo ***Specify an IP address*** (Navedite IP naslov) in nato v polje ***IP Address*** (IP naslov) vtipkamo svoj IP-naslov (v našem primeru je to 192.168.119.20). V polje ***Subnet Mask*** (Maska podomrežja) vtipkamo ***255.255.255.0*** (slika 4.4).

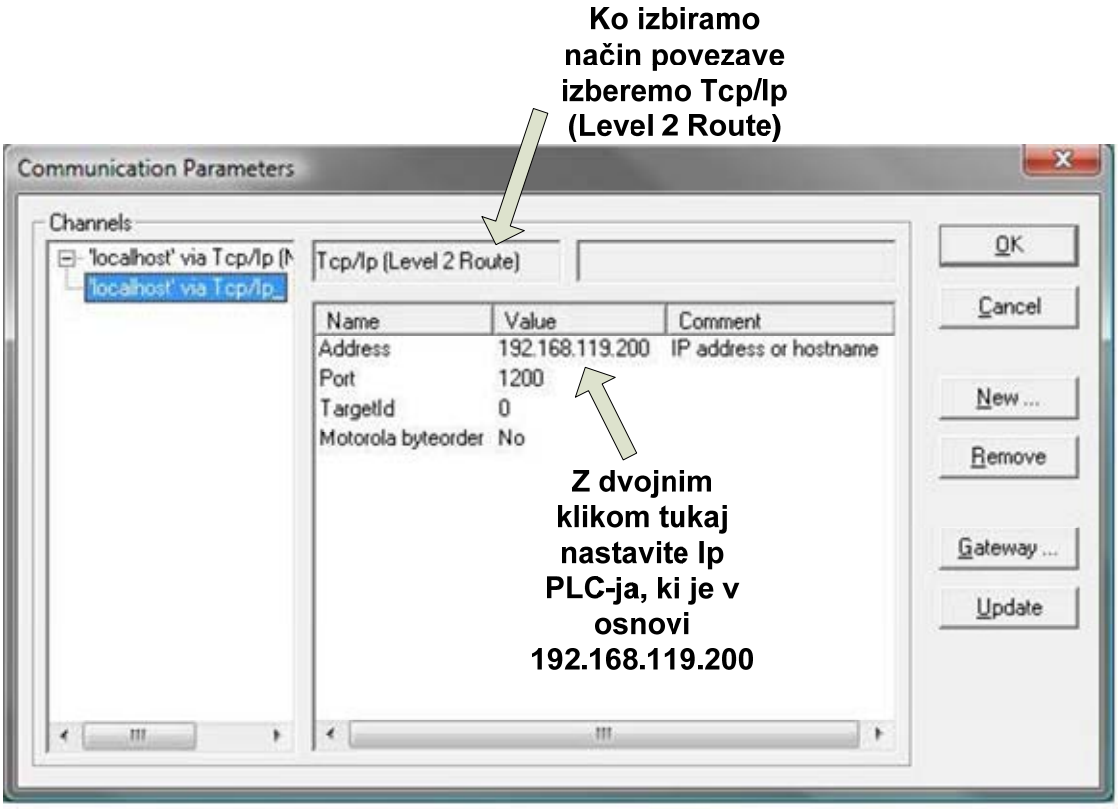


Slika 4.4: Nastavitev IP-naslova na osebnem računalniku

Nastavitev IP-naslova v programu XSoft

V XSoftu gremo na ukazno vrstico ***Online→Communication Parameters***. Odpre se novo okno. Izberemo ***New***. Nato spremenimo ime povezave in IP-naslov (Slika 4.5).

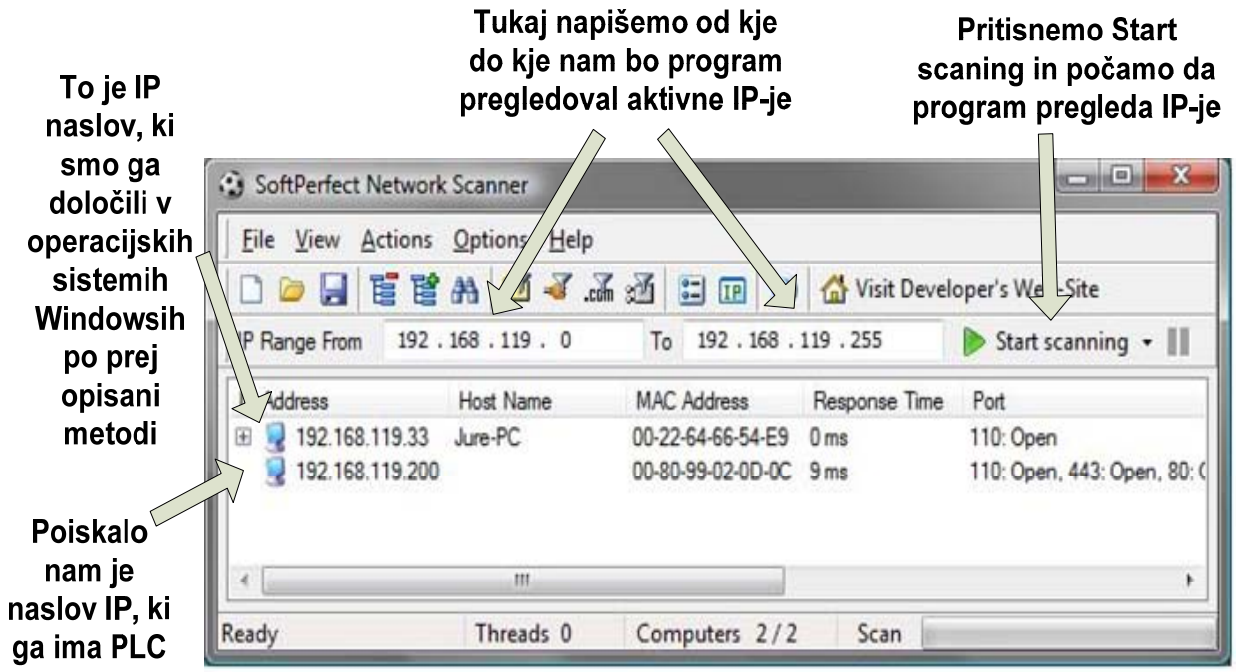
Tukaj je pomembno, da damo računalnik v isto družino kot PLC ali obratno, saj se drugače ne bosta videla. V osnovi je PLC-jev IP nastavljen na ***192.168.119.200***. Če ga ne spreminjamo, moramo tudi računalniku določiti IP iz iste družine, kar pomeni, da mora imeti na začetku enake številke, zadnjo pa lahko poljubno določimo (primer 192.168.119.xxx). V našem primeru smo računalniku nastavili IP-naslov na ***192.168.119.20*** (slika 4.4). Na sliki 4.6 lahko vidimo vmesnik programa Netscan.



Slika 4.5: Določitev IP-naslova v XSoftu

4.3 Uporaba brezplačnega programa Netscan

V primeru, da IP krmilnika ni nastavljen na osnovno (prednastavljeno) vrednost, moramo najprej poiskati dejanski naslov, ki ga ima krmilnik. Iz priložene zgoščenke ali iz interneta naložimo brezplačni program Netscan (glej sliko 4.6). Ko program zaženemo, določimo od kje do kje bo program pregledal IP-vrata. V našem primeru bomo pregledali vrata od ***192.168.119.0*** do ***192.168.119.255***. Tako smo poiskali nastavitve, da smo se lahko prek Ethernet Cross kabla povezali na PLC.



Slika 4.6: Vmesnik programa Netscan

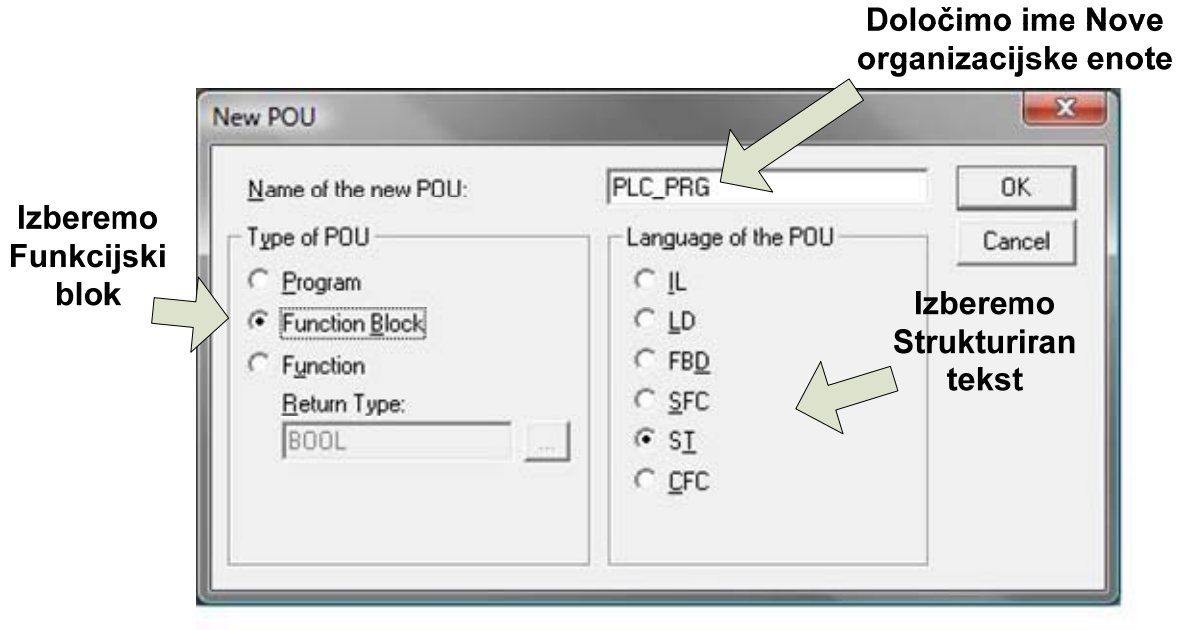
4.4 Nastavitev novega (praznega) projekta

V glavnem oknu programa XSoft izberemo ***File→New***. V oknu ***Target Settings*** v vrstici ***configuration*** izberemo model procesorja (CPU), ki ga bomo uporabljali. V našem primeru je to XC-CPU201_EC256K-8DI-6DO. Oznaka pove, da gre za procesor, ki ima 256 kilobajtov notranjega pomnilnika, 8 digitalnih vhodov in 6 digitalnih izhodov. Ko smo izbrali uporabljen procesor kliknemo ***OK.*** V oknu, ki se odpre pod ***Type of the new POU***, izberemo ***Program***, ime pa pustimo ***PLC_PRG*** in pod ***language of the POU*** izberemo ***ST (Structured language)***.

POU pomeni programska organizacijska enota, angl. Program Organization Unit. S temi nastavitvami smo izbrali, da bomo program pisali v strukturiranem jeziku (ST).

Do sedaj smo izbrali krmilnik, ki ga bomo uporabljali, določili novo organizacijsko enoto program ter določili programski jezik *strukturiran tekst*. V glavnem programu (PLC_PRG) nato začnemo s programiranjem.

Program krmilnika za svoje delovanje potrebuje globalne in lokalne spremenljivke. Ko npr. naredimo nov ***POU***, v katerem bo del celotnega programa (manjša celota), bo program zahteval, da deklariramo vse spremenljivke. Spremenljivke bo računalnik sicer želel deklarirati sam, programer pa mora pri tem določiti, za katere vrste podatka gre. Spremenljivko lahko deklariramo kot ***ARRAY***, s katerimi gradimo 1-, 2- ali 3-dimenzionalna polja. Binarni spremenljivki lahko dodelimo vrednost ***TRUE*** ali ***FALSE*** in zavzame 8 bitov. Spremenljivki ***BYTE*** je dodeljeno 8 bitov, z uporabo le-te imamo lahko 255 različnih stanj te spremenljivke. Spremenljivke ***DATE*** se uporablja za datum (npr. ***DATE#1996-05-06***). ***INT*** dodeli spremenljivki 16 bitov, s katerimi lahko prikažemo ± 32768 različnih celoštevilčnih vrednosti. ***REAL*** se uporablja za števila s tako imenovano plavajočo vejico, ki se uporablja za predstavitev realnih števil. Spremenljivke ***STRING*** lahko vsebujejo katerikoli niz znakov. V deklaraciji spremenljivke ***STRING*** določimo najdaljšo dolžino niza znakov. ***TIME*** je spremenljivka, v katero lahko shranimo vrednost časa (npr. ***TIME1 := t#12h34m15s***;). ***TON*** je spremenljivka, ki ima v sebi lastnost, da postane ***TRUE***, ko preteče čas, ki smo ga nastavili. Čas začne teči, ko je izpolnjen nek želen pogoj. ***TON*** pomeni Time On.

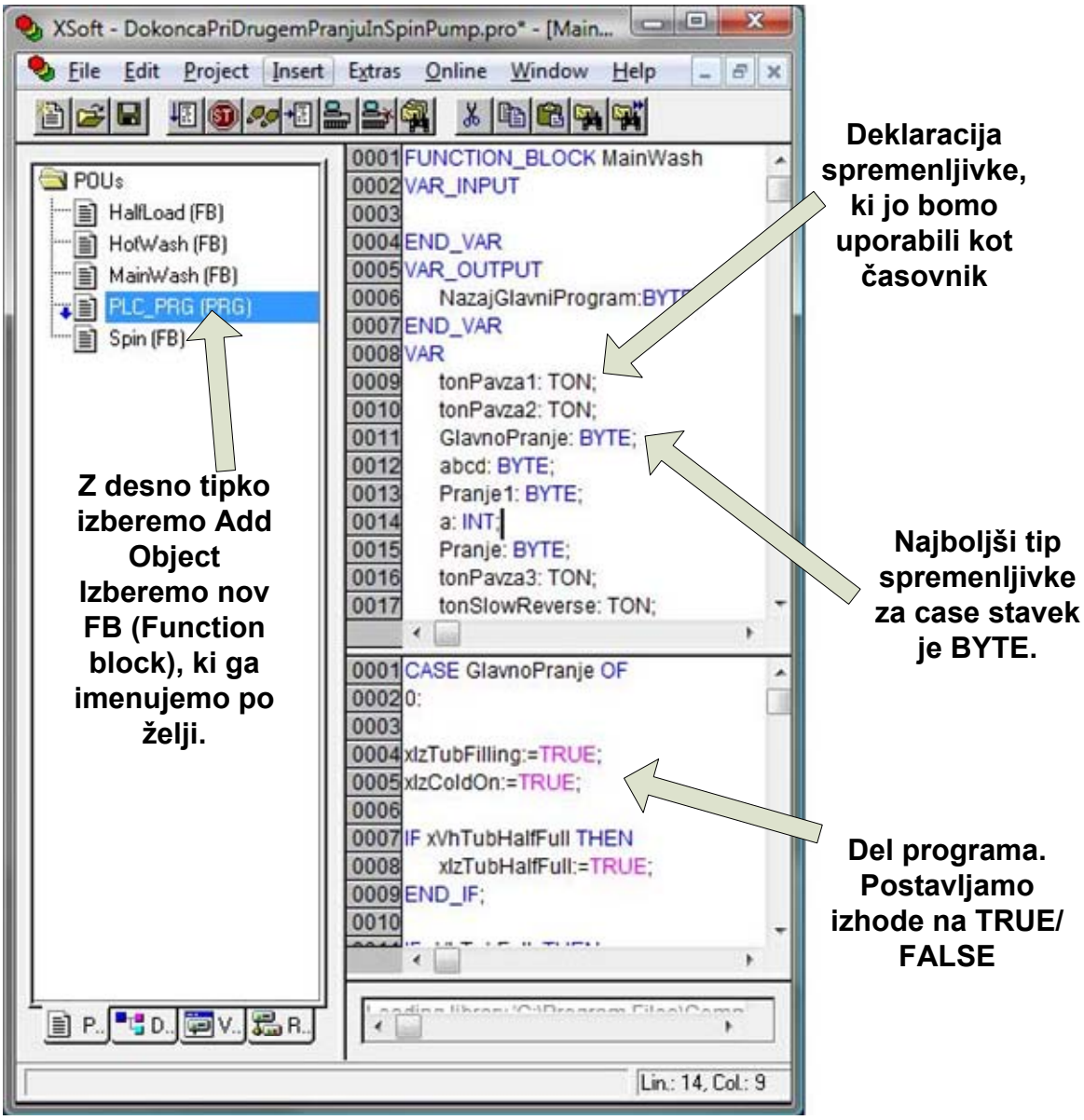


Slika 4.7: Okno programa XSoft s katerim odpremo nov projekt (POU)

4.5 Funkcijski blok (FB) v obliki POU (programska organizacijska enota)

Nov POU deklariramo tako, da z miško kliknemo z desno tipko na levo okno, kot prikazuje slika 4.8, in izberemo ***Add Object***. V našem primeru (slika 4.7) smo želeli ***Function Block*** in ***ST*** (Structured text).

Da bi ustvarili novo programsko organizacijsko enoto ***funkcijski blok*** za pisanje programa v strukturiranem tekstu, moramo storiti naslednje. V levem oknu kliknemo desno tipko na miški in izberemo ***Add Object***, kot prikazuje slika 4.8. Prikaže se novo okno (slika 4.7). V tem oknu izberemo ime objekta, določimo, da bo to funkcijski blok, ter izberemo strukturiran tekst. V našem primeru smo nov POU definirali kot ***FB_MainWash : MainWash*** (slika 4.8).



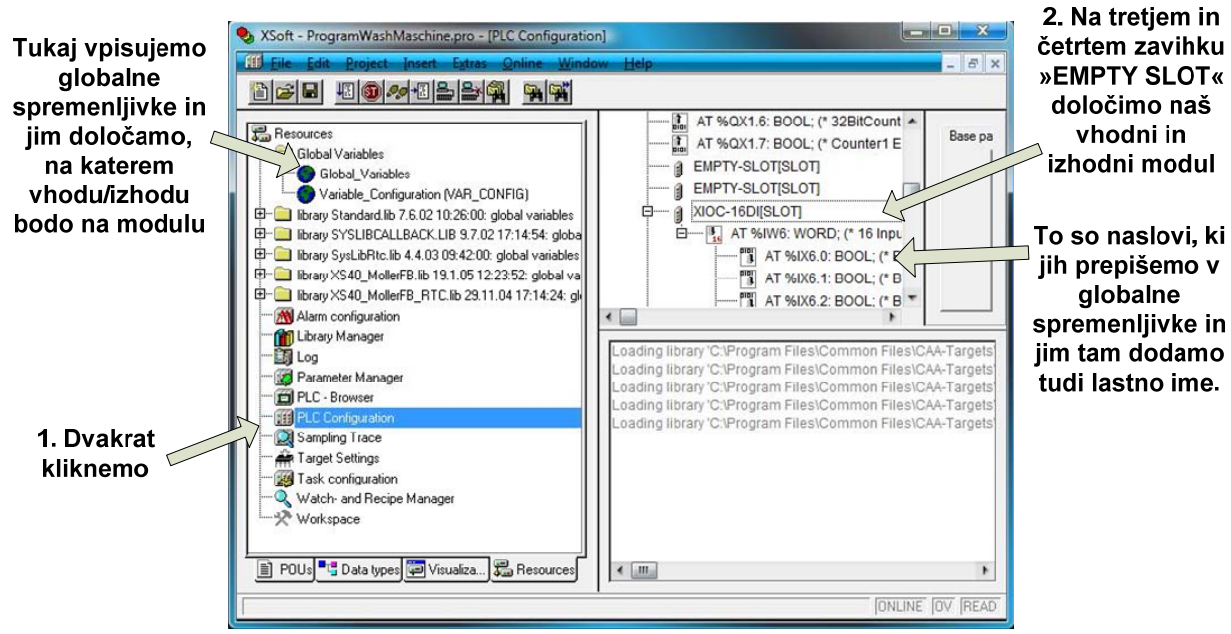
Slika 4.8: Program, spremenljivke, funkcijski bloki

To novo programsko enoto nato v programu (spodnje okno) lahko kličemo. V našem primeru smo jo klicali z ukazom FB_HotWash();. Tako smo naredili nov objekt. Sedaj moramo v tem objektu definirati izhodno spremenljivko. To naredimo med vrstico VAR_OUTPUT in END_VAR. V našem primeru smo si zbrali spremenljivko in mu določili tip: NazajGlavniProgram : BYTE;. V samem programu v spodnjem oknu pa se vrnemo nazaj v ***PRG_PLC(PRG)*** tako, da napišemo naslednji ukaz NazajGlavniProgram:=2***;***. Zaradi tega ukaza bo program, ko bo prišel do tega stavka, skočil na ***PRG_PLC*** v drugi korak prvega CASE stavka. Zadnji stavek, ki ga moramo napisati, da bo naš nov POU deloval, pa je:

IzbiraPranja:=FB_HotWash.NazajGlavniProgram;. Ta stavek napišemo v spodnjem oknu glavnega programa in to na mestu, kjer želimo, da se program nadaljuje.

4.6 Določitev vhodnih in izhodnih modulov

Na sliki 4.9 je prikazano, kam vse lahko dostopamo iz glavnega okna programa XSoft. Pod jezičkom ***Resources*** najprej izberemo module, ki jih uporabljamo v našem primeru. Če dvakrat kliknemo na ***PLC Configuration***, se odpre novo okno. Nato razširimo pogled in poiščemo mesto, kjer piše ***EMPTY-SLOT[SLOT].*** Pri tem moramo paziti na to, da nastavimo module natančno. To se naredi tako, da pogledamo na dejanski PLC in štejemo od procesorja proti desni. Prvi modul desno od procesorja je ***XIOC-SER***, drugi je ***XIOC-NET-DP-M***, tretji je nato vhodni ***XIOC-16DI*** in četrti ***XIOC-16DO-S***. Zadnja dva nas bolj zanimata, saj jih bomo rabili za izvedbo programa. Prva dva pa sta za serijsko komunikacijo in za povezavo preko interneta. Na sliki 4.9 je razvidno, da smo namesto ***EMPTY_SLOT[SLOT]*** vstavili ***XIOC-16DI[SLOT]***, kar ustreza našemu vhodnemu modulu. To smo naredili tako, da smo z desno tipko miške kliknili na ***EMPTY_SLOT[SLOT]*** ter nato kliknili ***Replace element***, kar omogoča izbiro ustreznega modula.



Slika 4.9: PLC Configuration

4.7 Dodajanje knjižic

Nato moramo dodati še nekatere datoteke, ki omogočajo uporabo določenih knjižic, kot so ***SysLibRtc.lib***, ***XS40-MoellerFB_RTC.lib***, ***Standard.lib***. To so knjižice, ki jih uporabljamo za upravljanje s časom, izdelavo funkcijskih blokov. Omogočajo tudi uporabo drugih standardnih funkcij (npr. if stavki, case stavki itd.). To naredimo tako, da gremo na jeziček ***Resources→Library Manager***. Nato v okencu levo zgoraj pritisnemo desno tipko na miški in kliknemo na ***Aditional Library*** ali pa pritisnemo tipko ***insert*** (vstavi). V raziskovalcu gremo nivo višje in pod mapo ***Lib_Common*** izberemo datoteko ***XS40_MoellerFB_RTC.lib.*** S tem smo pridobili dostop do dodatnih funkcij v samem programu.

4.8 Določanje globalnih spremenljivk

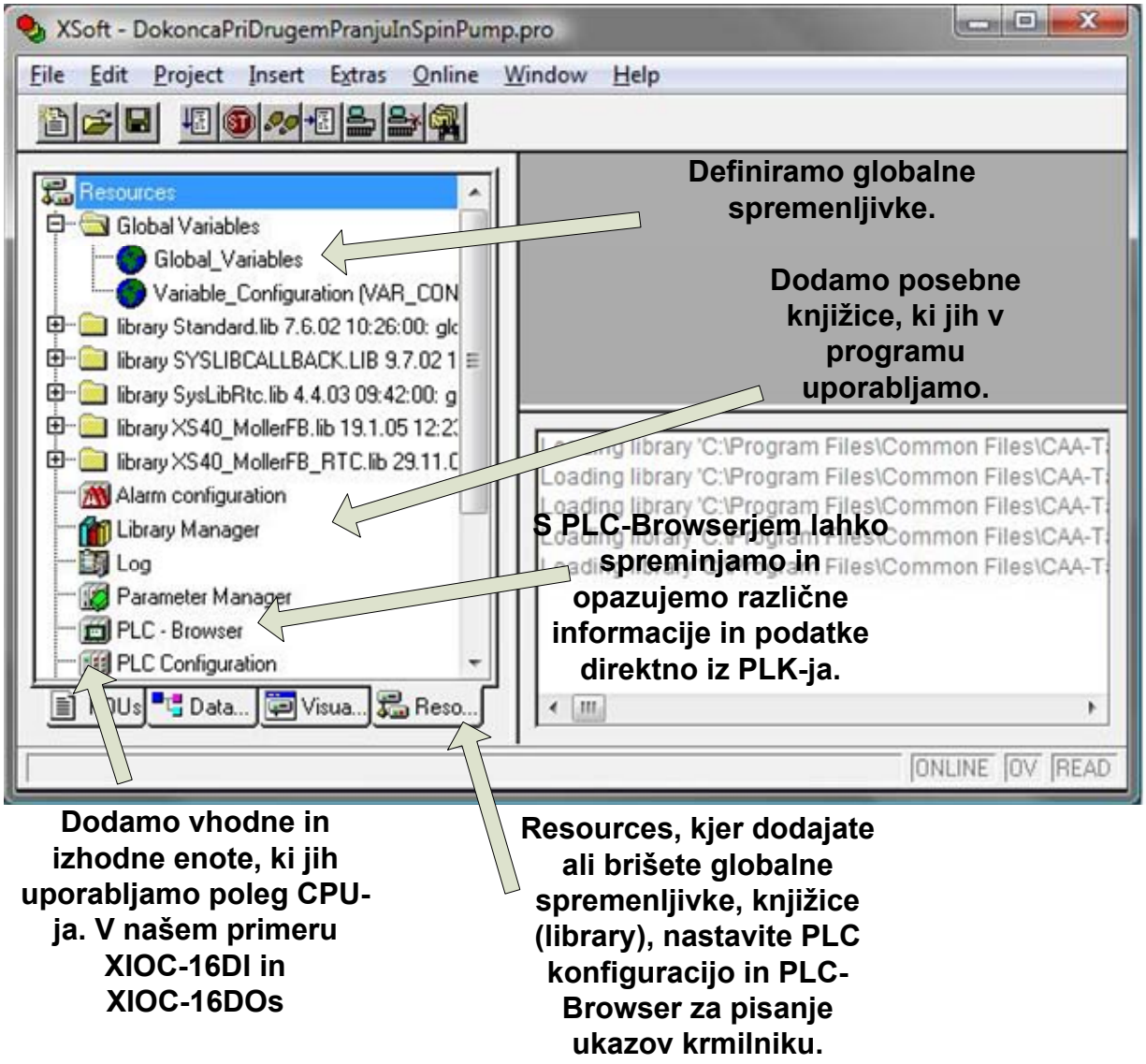
Pod jezičkom ***Resources→Global Variables*** nato določimo globalne spremenljivke. To so spremenljivke, ki bodo predstavljale vhode (tipke in stikala) in izhode (lučke). Iz prej določenega vhodnega in izhodnega modula prepišemo naslove na modulu in jim določimo imena za lažje razumevanje pri samem programiranju. Primer deklaracije ene vhodne in ene izhodne spremenljivke:

```
VAR_GLOBAL
        (* Digitalni vhod *)
        xVhHalfLoad AT %IX6.0:BOOL;
        (* Digitalni izhod *)
        xIzFastForwardSpin AT %QX2.0 : BOOL;
END_VAR
```

V zgornjem primeru vidimo, kako se deklarira globalne spremenljivke. Spremenljivke pišemo med VAR_GLOBAL in END_VAR. ***X*** na začetku ni obvezen, vendar pove, da gre za spremenljivko tipa boolean (dvojiški sistem), ki ima samo dve možnosti. To je **FALSE** in/ali **TRUE**. ***AT*** pomeni, na kateri nožici bo, v našem primeru bo **xVhHalfLoad** na **%IX6.0**. ***IX6.0*** pove, da gre za vhod, izhod bi imel npr**. QX2.0**. ***BOOL*** pomeni, kot smo že prej povedali, da gre za binarno spremenljivko. V priloženem programu so določene vse vhodne in izhodne spremenljivke, ki smo jih nastavili po zgornjem opisu. Fizične naslove vhodnih

spremenljivk lahko preslikamo iz prej nastavljenega ***PLC Configuration*** (glej sliko 4.9) pod jezičkom ***XIOC-16DI[SLOT]*** in izhodne spremenljivke pod jezičkom ***XIOC-16DO-S***.

Tako smo nastavili vse potrebne nastavitve. Sedaj lahko začnemo pisati program za avtomatsko vodenje pralnega stroja s sekvenčnimi ukazi.



Slika 4.10: Dodajanje knjižic Library Manager, ukazna vrstica PLC-Browser in konfiguriranje vhodno-izhodnih modulov PLC Configuration

4.9 Pisanje programa

Ko smo nastavili krmilnik, lahko začnemo s pisanjem programa. V programu XSoft lahko pišemo v različnih programskih jezikih. V našem primeru smo izbrali strukturiran tekst (ang. Structured text – ST), ki omogoča podobno programiranje kot program C++. Poleg strukturiranega teksta lahko v XSoftu programiramo še s seznamom ukazov (angl. Instruction list – IL), s sekvenčnim funkcijskim diagramom (angl. Sequential function chart – SFC), z diagramom funkcijskih blokov (angl. Function block diagram – FBD), z lestvičnim diagramom (angl. Ladder Diagram – LD) ter z zveznim funkcijskim diagramom (angl. Continuous function chart – CFC).

V XSoftu so v programskem oknu (slika 4.8) zgoraj desno deklarirane vse spremenljivke, spodaj desno pa je primer programa.

4.10 Osnove strukturiranega teksta

Ta programski jezik je podoben jeziku C++. Podpira ukaze in zanke, kot so ***for***, ***case***, ***while*** itd. (slika 4.8). V nadaljevanju bomo prikazali primere uporabe teh stavkov in drugih ukazov. Strukturiran tekst je sestavljen iz niza navodil, ki so določeni v visokonivojskem programiranju (npr. ***IF***…***THEN***…***ELSE***), ali zank (npr. ***WHILE***…***DO***). To omogoča večjo preglednost programa ter manjšo verjetnost napake. Spodaj je prikazana primerjava strukturiranega teksta (angl. Structured text - ST) in seznama ukazov (angl. Instructions list - IL):

Primer zanke za izračun eksponenta z osnovo 2 v seznamu navodil - IL:

```
LD      Counter
JMPC end
LD      Var1
MUL   2
ST      Var1
LD      Counter
SUB   1
ST      Counter
JMP   Loop
End:
LD      Var1
```

ST ERG

Primer zanke za izračun eksponenta z osnovo 2 v strukturiranem tekstu - ST:

```
WHILE counter<>0 DO
Var1:=Var1*2;
Counter:=counter-1;
END_WHILE
Erg:=Var1;
```

4.10.1 Stavki IF

Z IF stavkom lahko program preveri stanje in odvisno od pogoja izvršuje navodila.

Sintaksa:

```
IF <Boolean_ izraz 1> THEN
<IF_ukazi>
{ELSIF <Boolean_ izraz 2> THEN
<ELSIF_iukazi 1>
…
ELSIF <Boolean_izraz n> THEN
<ELSIF_ukazi n-1>
ELSE
<ELSE_ukazi>}
END_IF
```

Primer:

Naredi IF stavek, ki naredi naslednje: če imata spremenljivka *xVhDoorClosed* in hkrati tudi spremenljivka *xVhStart* vrednost ***TRUE***, potem se postavita spremenljivki *xIzDoorLocked* in *xIzStart* na vrednost ***TRUE***.

```
IF xVhDoorClosed AND  xVhStart THEN
 xIzDoorLocked:=TRUE;
 xIzStart:=TRUE;
 END_IF;
```

4.10.2 Stavki CASE

S CASE stavkom lahko poljubno kontrolno spremenljivko (»<Var1>«) z več različnimi navodili skonstruiramo v en element. To naredimo tako, da s stavkom CASE pregledno zapišemo večsmerno odločitev, ki jo sicer lahko dosežemo tudi z več gnezdenimi stavki IF.

Syntaksa:

CASE <Var1> OF

<vrednost1>: <navodilo 1>

<vrednost 2>: < navodilo 2>

<vrednost 3, vrednost 4, vrednost 5>: < navodilo 3>

<vrednost 6 .. vrednost 10>: < navodilo 4>

...

<vrednost n>: < navodilo n>

ELSE <ELSE navodilo >

END_CASE;

Primer:

Naredi CASE stavek s kontrolno spremenljivko, ki se imenuje ***ImeCasea***, ki bo imel 3 vrednosti in tri navodila. Pri prvi vrednosti (0) naj bo navodilo ***if stavek*** s pogojem, da gre program v funkcijski blok ***HOTWASH***, če je vhod ***XVhHotWash TRUE***. Nato naj ***ImeCasea***

dobi vrednost (1). Drugo navodilo naj bo kratek komentar. Nato naj dobi ***ImeCasea*** vrednost (2). V tretji vrednosti napišite pogoj, da program ne javi napake. Zaključite CASE stavek.

```
CASE ImeCasea OF

0:
        IF xVhHotWash THEN
        FB_HOTWASH();
        END_IF
        ImeCasea:=1;

1:      (** Med temi znaki lahko v programu pišemo komentarje, ki jih program ne bo
bral kot program. V CASE stavku lahko gremo od vrednosti 0 do 254, v vsaki
vrednosti pa mora biti vsaj znak ; sicer bo program javil napako. Znak ; je minimalen
pogoj, da program ne javi napake. Ta CASE stavek bi lahko imel samo eno vrednost,
ker sta ostali dve vrednosti prazni.   **)

        ImeCaesa:=2;

2:

        ;

        END_CASE
```

4.10.3 Odštevalnik

Funkcijski blok odštevalnik izvede nastavljeno operacijo po nastavljeni časovni zamudi.

Deklaracijski del: TONInst : TON ;

Način klica: Pavza.Q

Deklaracija vhodne spremenljivke ***IN (VarBOOL1)***, zaradi katere se začne odštevati čas ***PT***. Ko preteče nastavljeni čas (5 sekund), spremenljivka ***TONInst*** dobi vrednost ***TRUE***:

TONInst(IN := VarBOOL1, PT:= T#5s);

Primer:

Definirajmo spremenljivko ***»Pavza« tipa TON*** (Time on). Napišimo IF stavek, ki bo postavil ***xVhTubFull*** na ***TRUE***, ko preteče čas v odštevalniku ***Pavza***, ki se začne odštevati, ko je CASE polnjenje v vrednosti 1 (IN:=Polnjenje = 1). Odštevalni čas določimo na 7s ***(PT:=t#7s)***.

Deklarativni del:

```
Pavza : TON ;
```

Programski del:

```
IF Pavza.Q THEN xVhTubFull:=TRUE;
END_IF
Pavza(IN:=  Polnjenje=1 , PT:=t#7s );
```

4.10.4 Zanka za poljubno število ponavljanj

Recimo, da smo v drugem koraku CASE stavka ***Pranje***. Radi bi, da bi se neka sekvenca ponovila dvakrat. To naredimo z naslednjim zaporedjem ukazov. Zanka se bo ponavljala tako, da dokler bo spremenljivka i manjša kot 2, se bo progam vračal na korak 0, kjer se bo spremenljivka i povečala za ena, in bo zopet preveril, ali je spremenljivka i manjša od 2. Ko bo i=2, bo program skočil na korak 1 in tako zaključil CASE stavek. Po vsakem stavku (IF, CASE) moramo stavek zaključiti. Pri CASE stavku se to naredi z ukazom ***END_CASE***.

```
CASE Pranje OF
        0:
        i:=i+1;
    IF i<2 THEN Pranje:=0;
    ELSE
        i:=0;
        Pranje:=1;
    END_IF
        1: (**i je 2 ali več**)
END_CASE
```

4.10.5 Komentarji

V XSoftu lahko pišemo tudi komentarje za lažje razumevanje programa. Ti stavki se ne bodo obravnavali kot programska koda, ampak bodo v pomoč pri razumevanju programa. Primer komentarja : (* TO JE SAMO KOMENTAR *)

4.10.6 Enostavni primeri sekvenčnega vodenja

1. Če pritisnemo stikalo ***Door Closed*** in gumb ***Start***, se prižgeta indikatorja ***xVhDoorLocked*** in ***xVhStart***.

```
IF xVhDoorClosed AND xVhStart THEN
        xIzStart:=TRUE;
        xIzDoorLocked:=TRUE;
END_IF
```

2. Vključimo indikator ***xIzSlowReverse***. Čez 5 sekund ***(Pavza.Q)*** ga ugasnemo in prižgemo indikator ***xIzSlowForward***. Čez 5 sekund ***(Pavza1.Q)*** ga ugasnemo.

```
xIzSlowReverse:=TRUE;
        IF Pavza.Q THEN
                xIzSlowReverse:=FALSE;
                xIzSlowForward:=TRUE;
        END_IF
        IF Pavza1.Q THEN
                xIzSlowForward:=FALSE;
        END_IF

Pavza(IN:= xIzSlowReverse, PT:=t#5s , Q=> , ET=> );
Pavza1(IN:= xIzSlowForward, PT:=t#5s , Q=> , ET=> );
```

3. Kot naloga 2, ampak naj se ***xIzSlowReverse*** in ***xIzSlowForward*** ponovita dvakrat v zanki. Lahko si pomagamo s CASE stavkom.

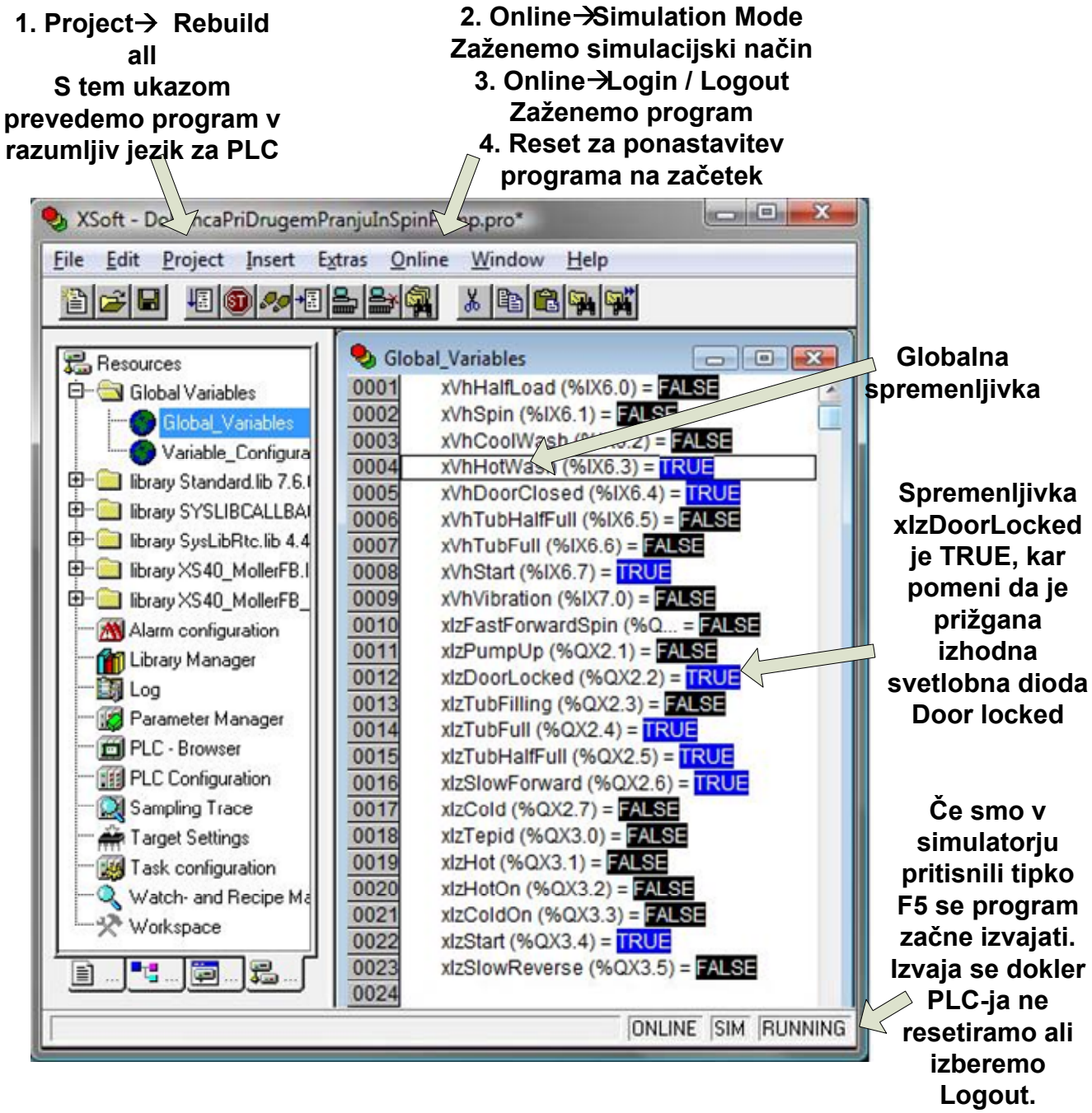
```
CASE ponavljanje OF
0:
xIzSlowReverse:=TRUE;
        IF Pavza.Q THEN
                xIzSlowReverse:=FALSE;
                xIzSlowForward:=TRUE;
        END_IF
        IF Pavza1.Q THEN
                xIzSlowForward:=FALSE;
                ponavljanje:=1;
        END_IF
1:
        i:=i+1;
        IF i<2 THEN Ponavljanje:=0;
        ELSE
        i:=0;
        ponavljanje:=2;
        END_IF
2:
END_CASE
```

4.11 Simulacija programa v XSoftu

Ko smo napisali vsaj en stavek v strukturiranem tekstu, lahko to simuliramo in na ta način preverimo delovanje programa. Najprej moramo projekt prevesti v kodo, ki je razumljiva PLC-ju. To naredimo tako, da v XSoftu v zgornji orodni vrstici kliknemo ***Project→Rebuild all***. Nato izberemo ***Online→Simulation Mode***. S tem smo programu povedali, da želimo simulirati naš program. Program smo tako prevedli v kodo, razumljivo za PLC, nato gremo na ***Online→Login***. S tem ukazom smo zagnali simulacijo. Sedaj bodo dvojiške spremenljivke vidne kot črn kvadratek ***FALSE***, ki ponazarja izklopljeno stikalo, kot je to prikazano na sliki 4.5. Na začetku programa so vsa stikala nastavljena na ***FALSE***.

4.11.1 Uporaba simulatorja

Ko smo v simulatorju (slika 4.11), lahko spreminjamo vrednosti spremenljivk. To naredimo tako, da dvakrat kliknemo na spremenljivke za posamezne vhode in izhode in pritisnemo ***Ctrl+F7***. Seveda se nič ne zgodi, ker program še ni pognan. Da bi ga zagnali, moramo pritisniti ***F5***. Ko to naredimo, se v spodnji vrstici okna prikaže indikator ***Running*** (slika 4.11). Na isti sliki lahko vidimo tudi vse globalne spremenljivke v zavihku ***Global_Variables***.



Slika 4.11: Simulacija v Xsoft prikaz vrednosti globalnih spremenljivk

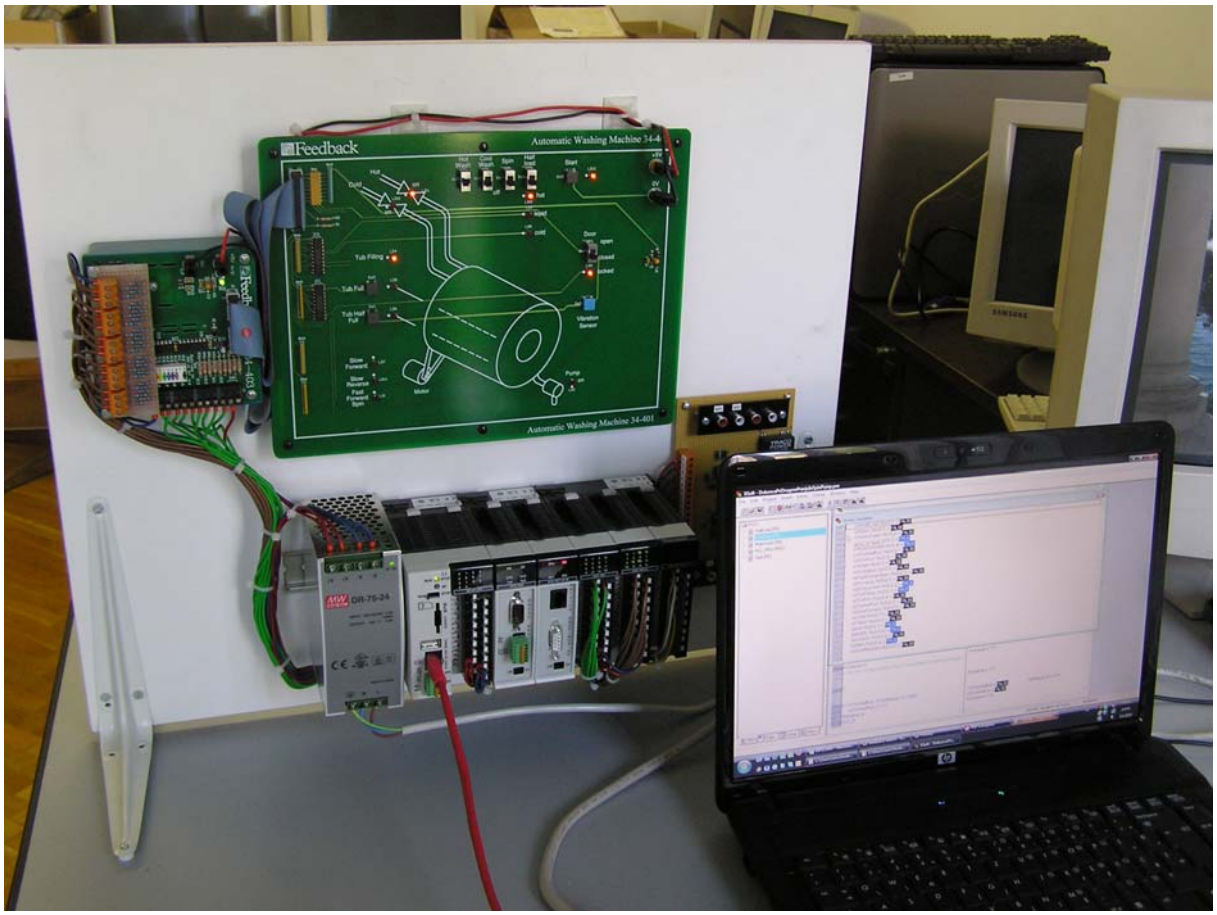
4.12 Prenos programa iz osebnega računalnika na krmilnik

Najprej računalnik povežemo s PLC-jem s standardnim cross Ethernet kablom, kot je opisano v prejšnjem poglavju. Ker smo predhodno nastavili IP-naslove računalniku in PLC-ju, lahko sedaj vklopimo napajanje krmilnika. Stikalo krmilnika, ki določa način delovanja, naj bo nastavljeno na ***STOP***. Zaženemo XSoft, nato izberemo ***Project→Rebuild all.*** Nato pod jezičkom ***Online*** izklopimo ***Simulation mode***, ki smo ga uporabljali za simulacijo. Nato zopet pritisnemo ***Online→***. Nato se odpre okno, kjer piše sporočilo, da je bil program spremenjen in ali želimo naložiti spremembe (slika 4.12). Če kliknemo ***da***, se bo program iz osebnega računalnika prenesel na PLC. Na modulu krmilnika procesorske enote sedaj preklopimo na ***RUN***. Sedaj lahko iz programske opreme XSoft na osebnem računalniku resetiramo, ustavimo in zaženemo program. Program na krmilniku pa lahko s stikalom zaženemo tudi neposredno na procesorski enoti krmilnika.

Slika 4.12: Potrditveno okno za kopiranje programa XSoft v krmilnik

5 MODELNA NAPRAVA ZA POUČEVANJE SEKVENČNEGA VODENJA

5.1 Simulator pralnega stroja s programirljivimi krmilniki proizvajalca Moeller



Slika 5.1: Modelna naprava za poučevanje sekvenčnega vodenja

Pranje perila s pralnim strojem je tipičen primer sekvenčnega postopka, kjer si posamezni dogodki ali faze sledijo druga za drugo. Je dober primer za poučevanje sekvenčnega vodenja. Zaradi tega bomo z elektronskim simulatorjem avtomatskega pralnega stroja Feedback Automatic Washing Maschine 34-401 in krmilniki proizvajalca Moeller prikazali sekvenčno vodenje pralnega stroja. Simulator pralnega stroja s krmilniki se nahaja v kabinetu Poslovno-tehniške fakultete. Kabinet je namenjen preučevanju modelnih naprav, ki so v lasti Univerze v Novi Gorici.

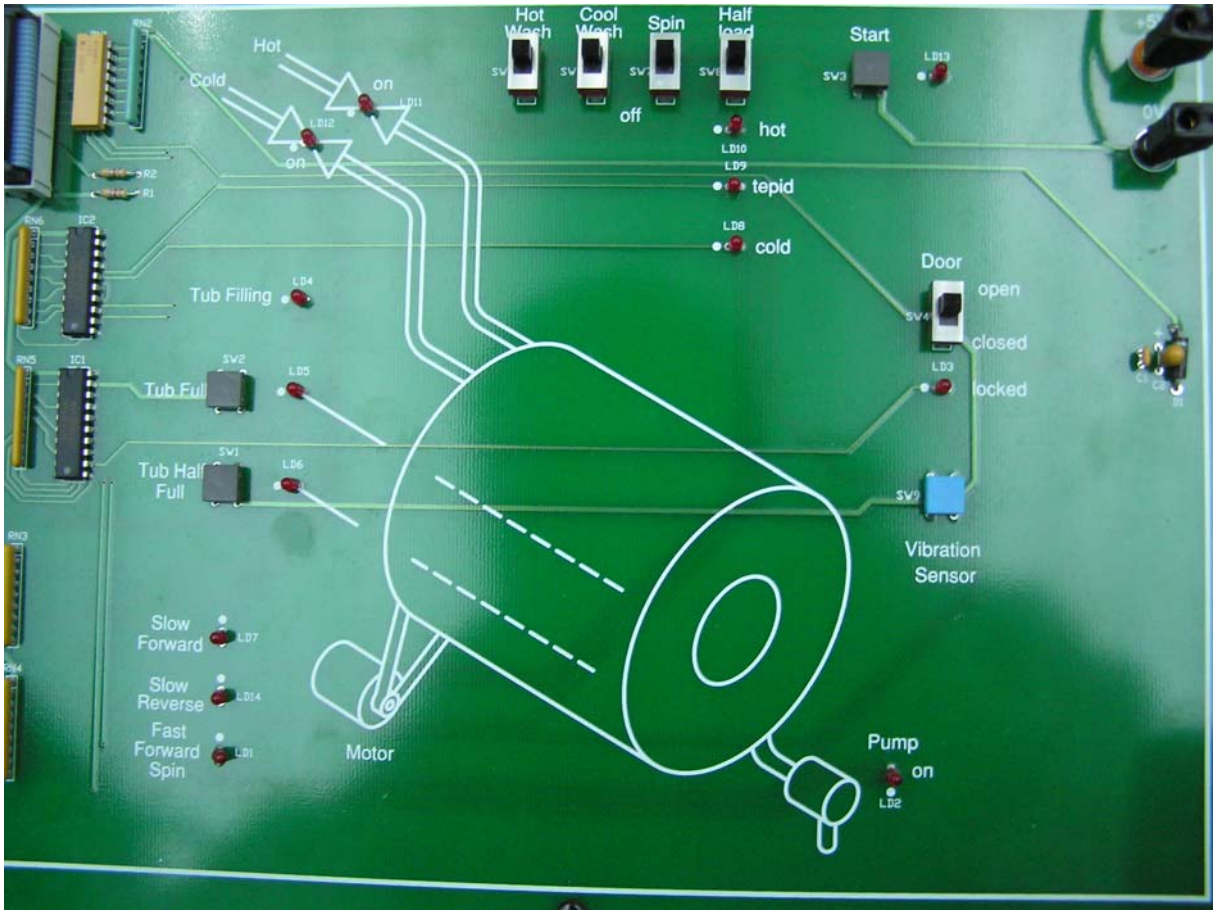
5.1.1 Opis modelne naprave Feedback

Modul avtomatski pralni stroj (angl. Automatic Washing Maschine) je interaktivni elektronski simulator gospodinjskega pralnega stroja. Lučke ponazarjajo različna stanja v samem stroju, tipke in stikala pa so namenjena za interakcijo s samim procesom pranja. Naprava je povezana (slika 5.1) s PLC-vmesnikom preko trakastega kabla (angl. Ribbon Cable), ki služi za prenos podatkov iz PLC-ja na simulator. Naprava je narejena tako, da z njo lahko prikažemo sekvenčno vodenje.

Opis sestavnih delov modelne naprave pralnega stroja:

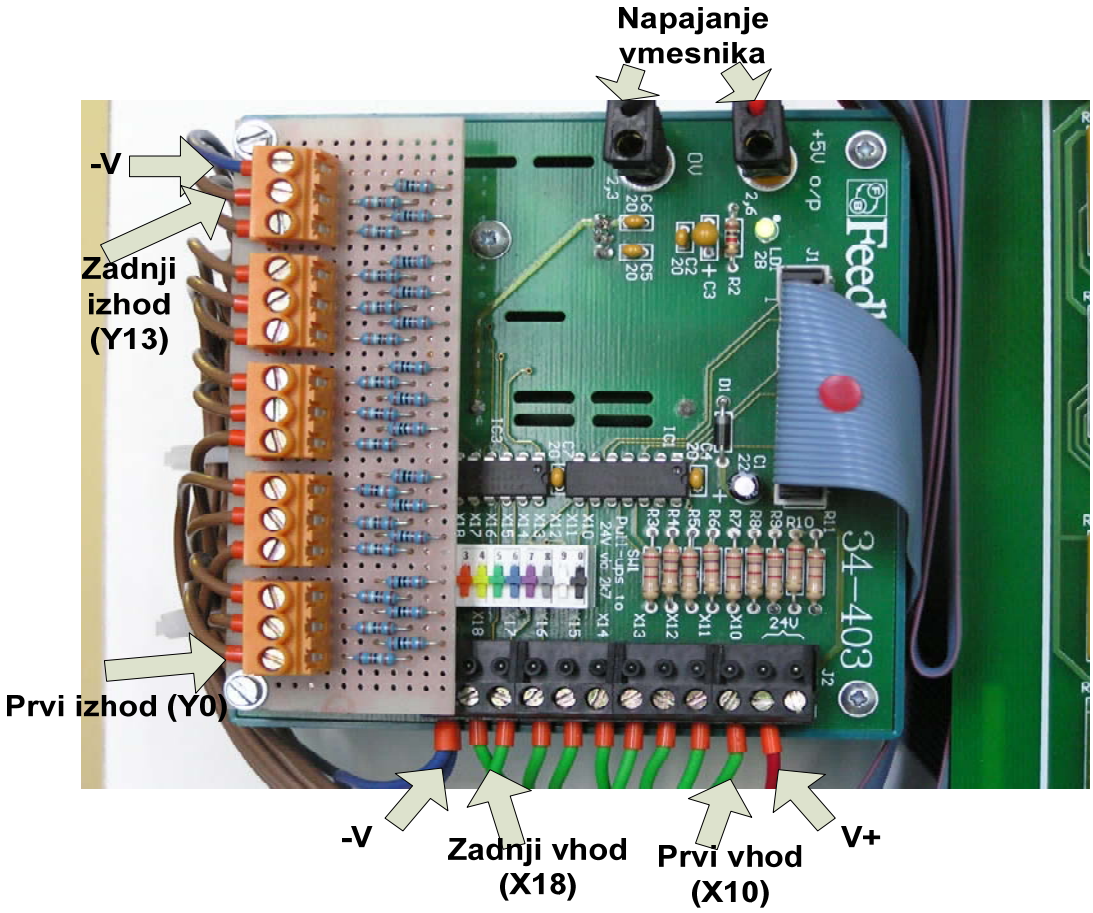
a) Elektronski simulator pralnega stroja je sestavljen iz elektronskega vezja s pripadajočimi svetlobnimi indikatorji in stikali. Pogled nanj je prikazan na sliki 5.2, kjer so vidni ključni elementi, ki so:

- večsmerni konektor na vrhu podlage, na katerega priklopimo povezovalni večžilni kabel in ga povežemo na Feedbackov PLC-vmesnik (PLC Interface 34-403) (slika 5.3), ki zagotavlja vse potrebne povezave z izbranim PLC-jem,
- stikala za izbiro programa in izbiro posameznih funkcij,
- svetlobni indikatorji, ki ponazarjajo potek programa ali opozorilo.

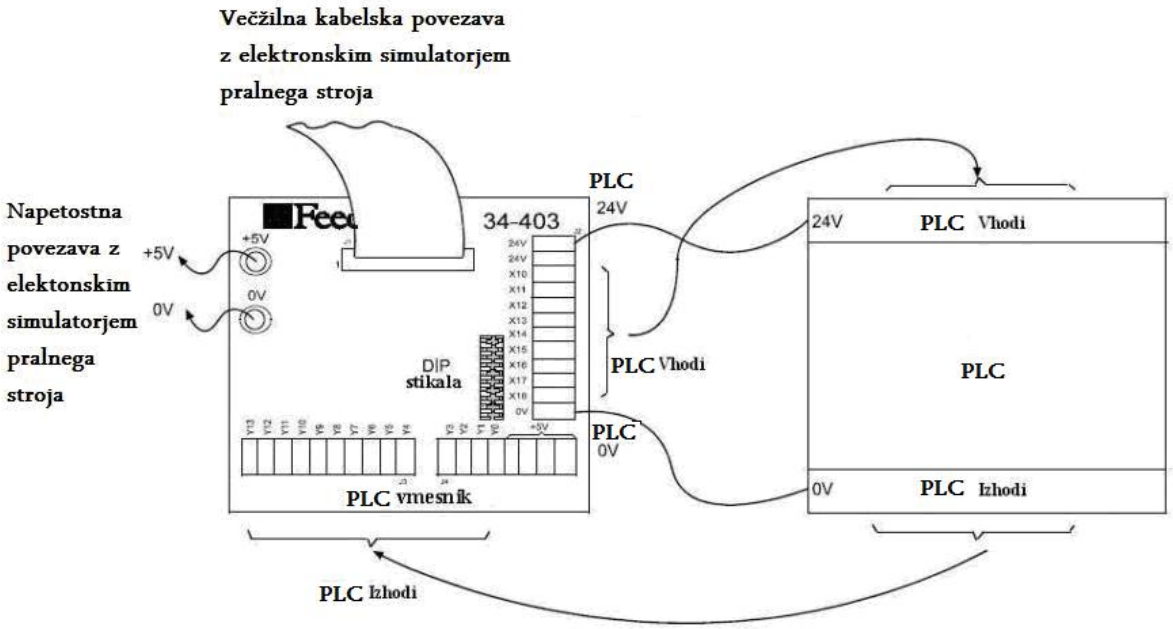


Slika 5.2: Kontrolna plošča elektronskega simulatorja avtomatskega pralnega stroja

b) PLC-vmesnik 34-403 je vezje z raznimi priključki, na katere privijačimo posamezne vodnike, da povežemo pralni stroj z izbranim krmilnikom (PLC). 24-voltno enosmerno električno izhodno napajanje krmilnika zagotavlja +5V enosmerne električne napetosti, ki jo elektronski simulator avtomatskega pralnega stroja potrebuje za delovanje. Priskrbljeni so 4-milimetrski priključki za zagotavljanje te napajalne povezave. Tipična povezava sistema je prikazana z diagramom na sliki 5.3. Na sliki 5.4 pa lahko vidimo diagram tipične povezave PLC-vmesnika s krmilnikom.



Slika 5.3: PLC-vmesnik



Slika 5.4: Diagram tipične povezave PLC-vmesnika s PLC-krmilnikom

c) Za vodenje modelne naprave in zagotavljanje vseh funkcij pralnega stroja, kot so opisane v primeru, potrebujemo krmilnik (PLC) z vsaj 9 vhodi in 14 izhodi ter s 24-voltno enosmerno električno izhodno napetostjo. Primer ustreznega krmilnika je prikazan na sliki 5.5.

5.2 Opis vhodov in izhodov za naš primer (PLC interface, XSoft globalnih spremenljivk, njihova imena ter prevod)

Tabela 1: Vhodni modul Moeller XIOC 16DI

Vhodi	PLC interface (PLC-vmesnik)	XSoft globalne spremenljivke	Imena vhodnih globalnih spremenljivk	Prevod, ime na panelu, oznaka na panelu
0	X10	IX6.0	xVhHalfLoad	Pranje s polovično polnim bobnom (Half Load) – SW8
1	X11	IX6.1	xVhSpin	Centrifuga (Spin) – SW7
2	X12	IX6.2	xVhCoolWash	Hladno pranje (Cool Wash) – SW6
3	X13	IX6.3	xVhHotWash	Toplo pranje (Hot Wash) – SW5
4	X14	IX6.4	xVhDoorClosed	Vrata (Door open /closed) – SW4
5	X15	IX6.5	xVhTubHalfFul	Polni boben do polovice (Tub Half Full) – SW1
6	X16	IX6.6	xVhTubFull	Polni boben do konca (Tub Full) – SW2
7	X17	IX6.7	xVhStart	Začni (Start) – SW3
8	X18	IX7.0	xVhVibration	Vnos vibracije (Vibration Sensor) – SW9

Tabela 2: Izhodni modul Moeller XIOC 16DO-S

Izhodi	PLC interface	XSoft globalne spremenljivke	Imena vhodnih globalnih spremenljivk	Prevod, ime na panelu, oznaka na panelu
0	Y0	QX2.0	xIzFastForwardSpin	Hitro vrtenje naprej (Fast Forward Spin) – LD1
1	Y1	QX2.1	xIzPumpUp	Črpalka (Pump On) – LD2
2	Y2	QX2.2	xIzDoorLocked	Blokirana vrata (locked) – LD3
3	Y3	QX2.3	xIzTubFilling	Polnjenje (Tub Filling) – LD4
4	Y4	QX2.4	xIzTubFull	Poln boben (Tub Full) – LD5
5	Y5	QX2.5	xIzTubHalfFull	Napol poln boben (Tub Half Full) – LD6
6	Y6	QX2.6	xIzSlowForward	Počasi naprej (slow Forward) – LD7
7	Y7	QX2.7	xIzCold	Hladno (cold) – LD8
8	Y8	QX3.0	xIzTepid	Mlačno (tepid) – LD9
9	Y9	QX3.1	xIzHot	Toplo (hot) – LD10
10	Y10	QX3.2	xIzHotOn	Hot on (odprto toplo) – LD11
11	Y11	QX3.3	xIzColdOn	Cold on (odprto hladno) – LD12
12	Y12	QX3.4	xIzStart	Začetek (Start) – LD13
13	Y13	QX3.5	xIzSlowReverse	Počasi nazaj (slow Reverse) – LD14

6 POSTOPEK NAČRTOVANJA SEKVENČNEGA VODENJA

6.1 Zahteve

Iz podrobnega opisa zahtev naredimo diagram poteka. Iz diagrama poteka lahko vidimo kompleksnost in druge lastnosti, kot so potrebna dolžina programske kode, število neznank itd. Nato napišemo program v izbranem programskem jeziku, npr. »Structured text« v programskem okolju Xsoft, ga simuliramo ter prenesemo program za vodenje procesa v krmilnik.

Pralni stroj mora izvajati naloge, kot so definirane v zahtevanem postopku. Zato je potrebno napisati program, ki bo omogočal postopek pranja. Pri tem moramo seveda upoštevati določene zahteve, kako naj se ukazi izvršujejo ter kdaj naj se nek ukaz izvrši in kdaj ne.

6.2 Začetni program

a) V kolikor je gumb za pričetek ***(Start)*** pritisnjen in so vrata odprta (vključeno stikalo ***Door Open***), naj se naprava ne odziva.

b) V kolikor je gumb za pričetek pritisnjen in so vrata zaprta, se prižgeta svetlobna indikatorja ***Start*** in ***Door locked***, ki predstavljata zagon pralnega stroja in zaprta vrata.

Program za pranje belega perila (Hot Wash):

a) S preklopom stikala na ***Hot Wash*** in hkrati z zaprtimi vrati, se prižgejo svetlobni indikatorji, ki sporočajo, da se je pričelo polnjene bobna z vročo vodo (***Tub Filling, Hot, Hot on***).

b) Ko je pritisnjeno stikalo za polovično napolnjen boben z vodo, se prižge indikator ***Tub Half Full***. Boben se napolni z vodo do vrha, ko je pritisnjeno stikalo ***Tub Full***, ki predstavlja ukaz za poln boben. Prižge naj se svetlobni indikator ob stikalu. Ko je boben poln, se indikatorja ugasneta (***Tub Filling*** in ***Hot on***).

c) Sledijo trije pralni cikli. Vsak cikel naj bo sestavljen iz dveh sekund dolgega počasnega vrtenja bobna naprej (indikator ***Slow Forward***), polsekundnega odmora ter iz dveh sekund dolgega počasnega vrtenja bobna nazaj (indikator ***Slow Reverse***).

d) Po koncu pranja sledi praznjenje bobna s črpalko, ki ga predstavlja prižgan indikator ***Pump on***, nakar se postopoma ugasne najprej indikator, ki predstavlja polno napolnjen boben ***(Tub Full)***, nato še indikator za ponazoritev napol polnega bobna ***(Tub Half Full)***. Čez nekaj trenutkov ugasneta še indikator, ki predstavlja pranje belega perila ***(Hot Wash)*** in indikator črpalke ***(Pump on)***.

e) Na vrsto pride polnjenje bobna kot prej, le da tokrat s hladno vodo, ki naj ga predstavljata prižgana indikatorja ***Cold on*** in ***Cold***.

f) Sledi postopek izpiranja, ki je enak kot postopek pranja, le da traja le dva cikla, in sicer cikel počasnega vrtenja naprej (indikator ***Slow forward***) in zatem cikel počasnega vrtenja nazaj (indikator ***Slow reverse***).

g) Črpalka naj prazni boben, dokler ta ni popolnoma prazen.

h) Nato se trikrat ponovi izpiranje s hladno vodo (glej zgoraj od točke e do točke g).

i) Indikatorja ***Pump on*** in ***Fast Forward Spin*** naj gorita 5 sekund, kar ponazarja končno praznjenje bobna.

j) V kolikor je bil na začetku ali pa med pranjem izbran ***Spin*** (centrifuga), se izvede dolga centrifuga, ki traja 30 sekund. Če med tem časom stisnemo tipko ***Vibration senzor***, se vsi indikatorji izključijo (pralni stroj se ustavi).

k) V kolikor preteče čas centrifuge, se pralni stroj ustavi (vsi kontrolni indikatorji ugasnejo).

l) Vsa stikala ugasnemo (postavimo na ***off***). V kolikor izklopimo stikalo ***Door closed***, se program resetira na tak način, da ga lahko ponovno zaženemo brez resetiranja samega PLC-ja.

Program za pranje belega perila z napol polnim bobnom:

Postopek naj bo enak kot pri navadnem programu za toplo pranje, le da naj se polnjenje bobna pri prvem polnjenju ustavi, ko je boben polovično napolnjen.

Program za pranje barvnega perila:

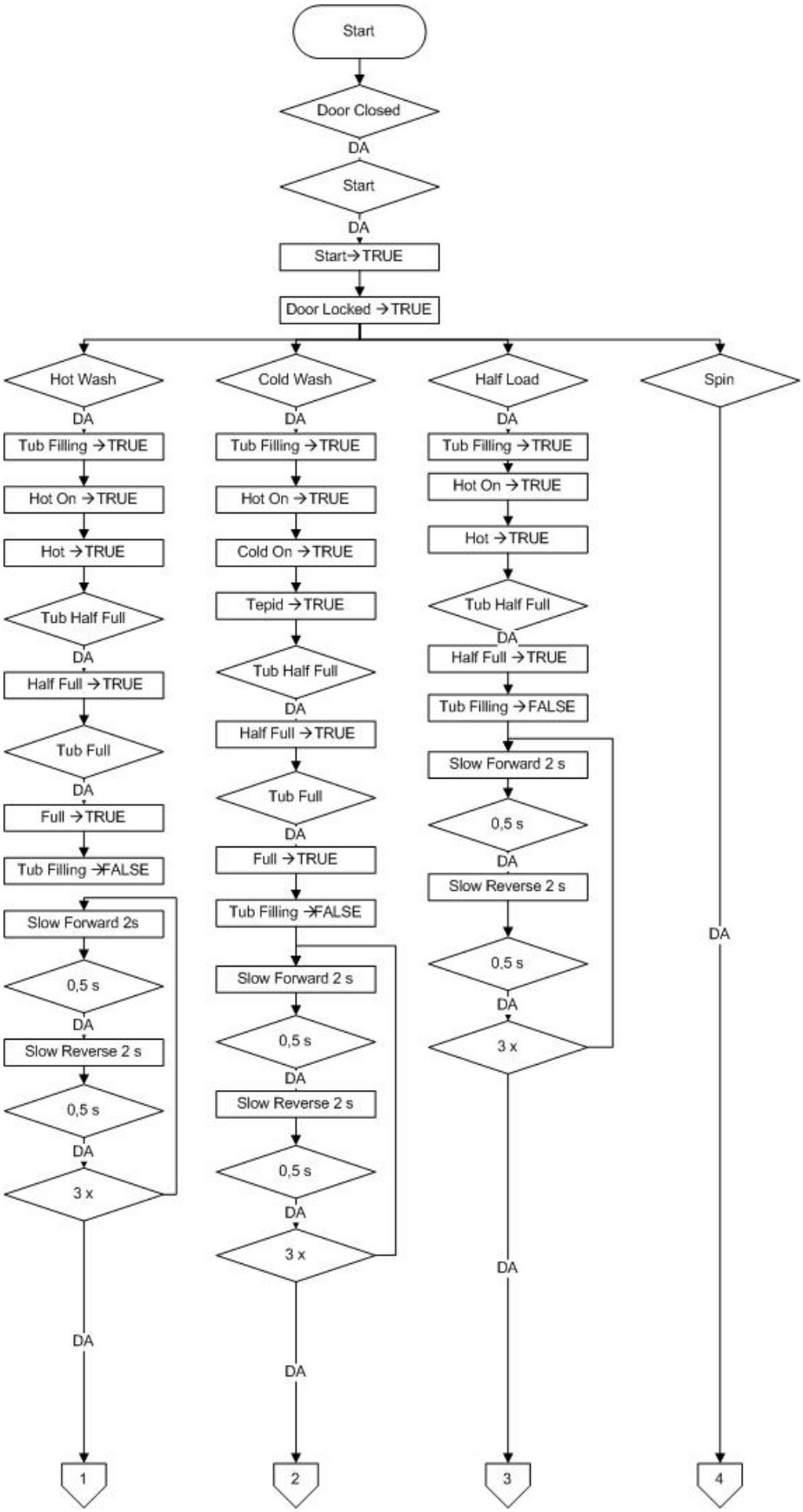
Z izbiro ***Cool Wash*** se z zaprtimi vrati prižgejo svetlobni indikatorji, ki sporočajo, da se je pričelo polnjene bobna z mlačno vodo (***Tub Filling, Hot On, Cold On, Tepid***).

Centrifuga brez izbranega programa za pranje:

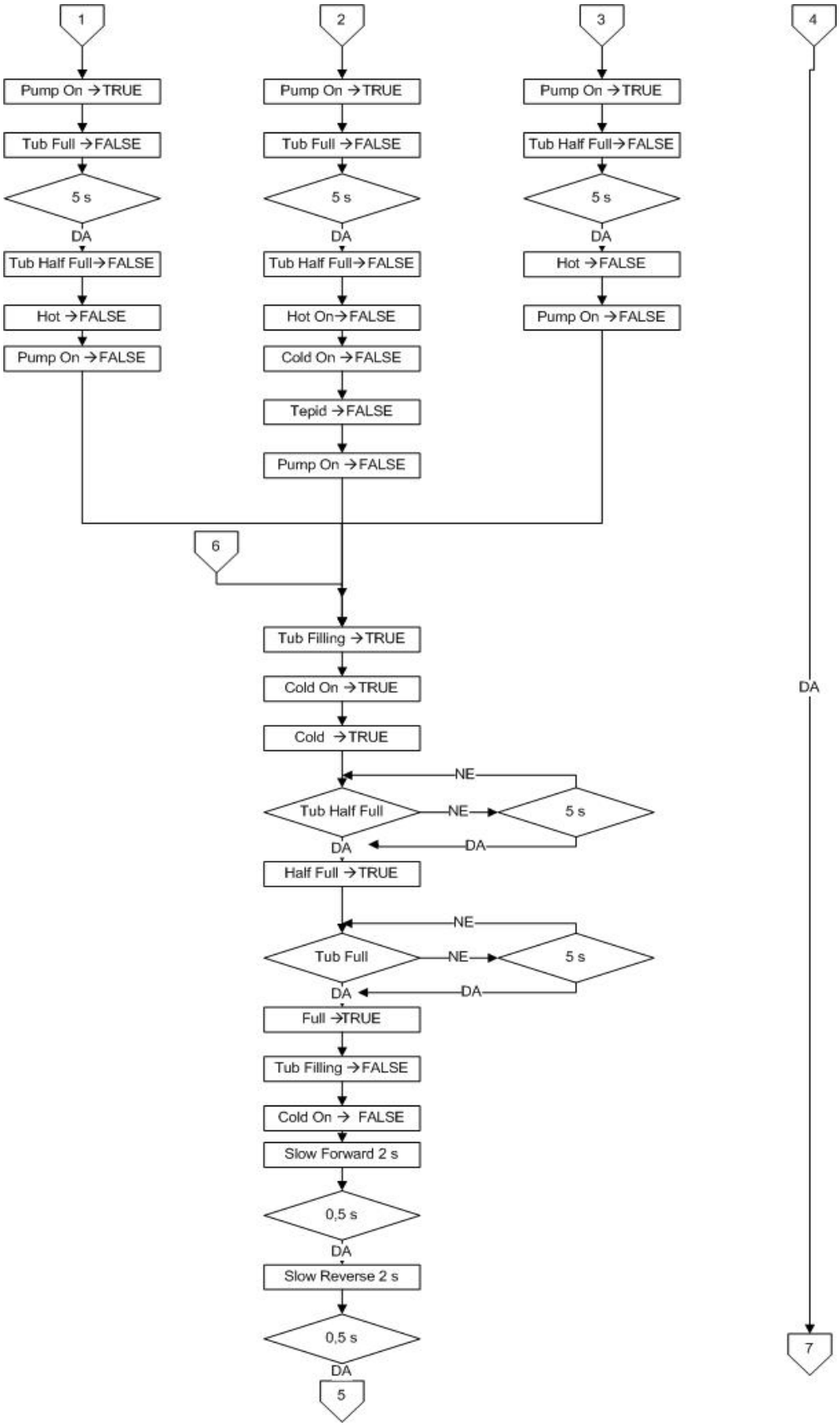
V kolikor je na začetku vključeno stikalo ***Spin***, stikali za pranje barvnega ali belega perila (Cold on ali Hot on) pa sta izključeni, se izvede le zaključni postopek dolge centifuge, ki naj traja 30 sekund.

6.3 Diagram poteka

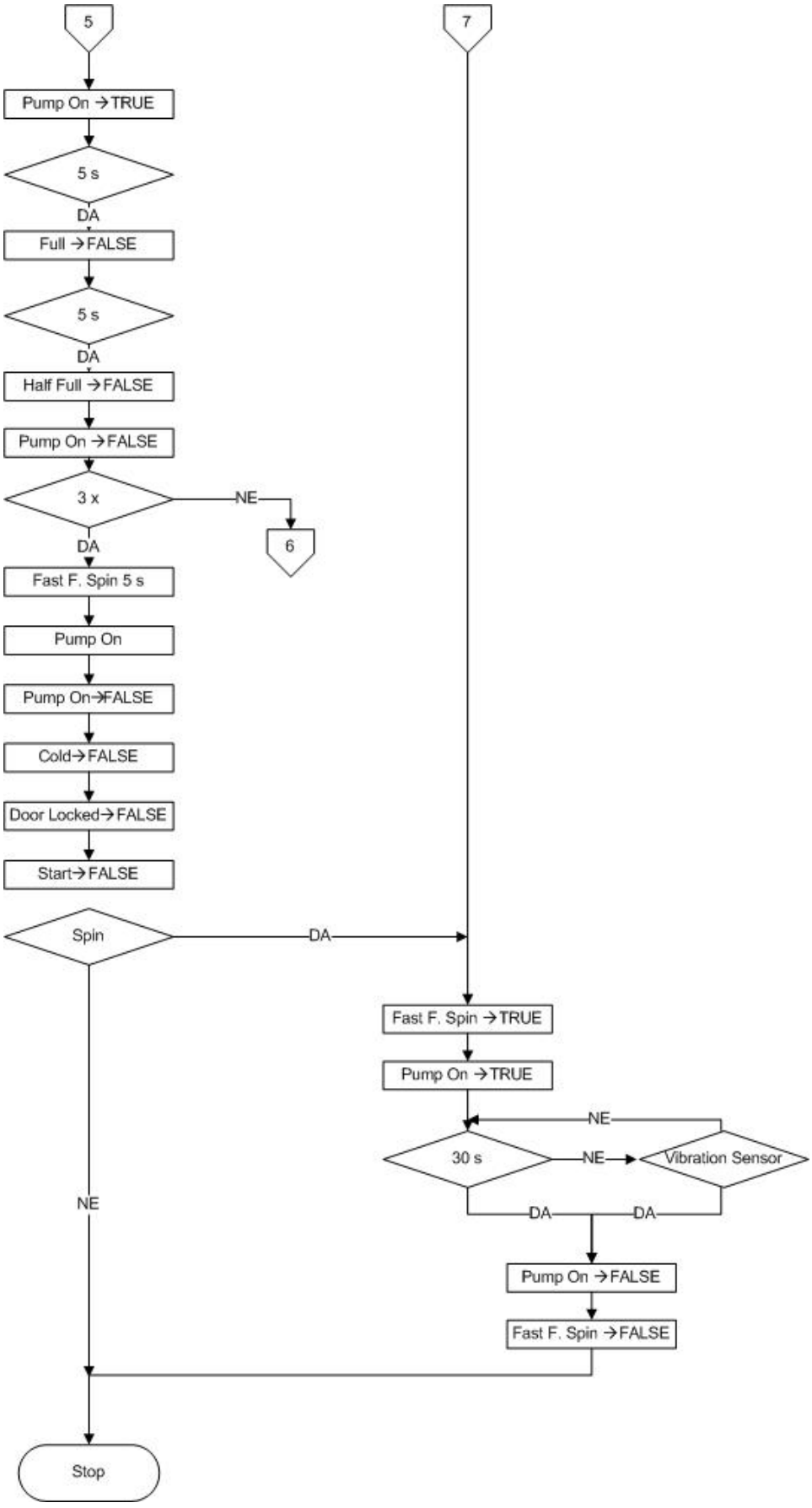
Diagram poteka, kot je prikazan na sliki 6.1, omogoča predstavitev poteka zaporedja posameznih ukazov, kot smo jih opisali v zahtevah za vodenje. Program za krmilnik smo nato napisali na osnovi tega diagrama poteka.



Slika 6.1 a: Diagram poteka – prvi del



Slika 6.1 b: Diagram poteka – drugi del



Slika 6.1 c: Diagram poteka – tretji del

6.4 Preizkus delovanja

Zadnja faza je preverjanje delovanja programa krmilnika na simulatorju pralnega stroja.

Simulacija delovanja

Najprej smo program preverili s simulacijo na računalniku. Da bi program simulirali oz. preverili delovanje, smo najprej zagnali program XSoft ter odprli program ***ProgramWashMaschine.pro.*** Računalnik smo predhodno povezali s krmilnikom (podpoglavje 4.2). Ko smo v programu XSoft, krmilniški program najprej razhroščimo. To smo naredili tako, da smo šli na jeziček ***Project → Rebuild all***. Program s tem ukazom pregleda programsko kodo in izpiše morebitne napake. Da bi program simulirali, smo kliknili na jeziček ***Online*** ter obkljukali ***Simulation Mode***. Zadnji korak za izvedbo simulacije pa je pod jezičkom ***Online → Run***. S tem ukazom v programu XSoft poženemo simulator. Prikaže se okno, ki je prikazano na sliki 4.11.

Prenos v krmilnik

Ko je bil program razhroščen, smo ga prenesli v krmilnik po postopku, ki smo ga opisali v podpoglavju 4.11.

Preizkus delovanja na krmilniku

Program krmilnika smo nato preizkusili na elektronskem simulatorju pralnega stroja. Najprej smo vzpostavili začetno stanje, in sicer da so vrata zaprta in je pritisnjena tipka ***start***. Preverili smo, ali pralni stroj deluje, kot je bilo postavljeno v zahtevah za vodenje oziroma v diagramu poteka. Posamezne napake smo sproti odpravili.

V postopku preverjanja smo odpravili tako napake v sami kodi kot tudi vsebinske napake. Samo delovanje elektronskega simulatorja pralnega stroja je sledilo programu ter izvedlo celoten postopek pranja tako, kot je bilo zahtevano.

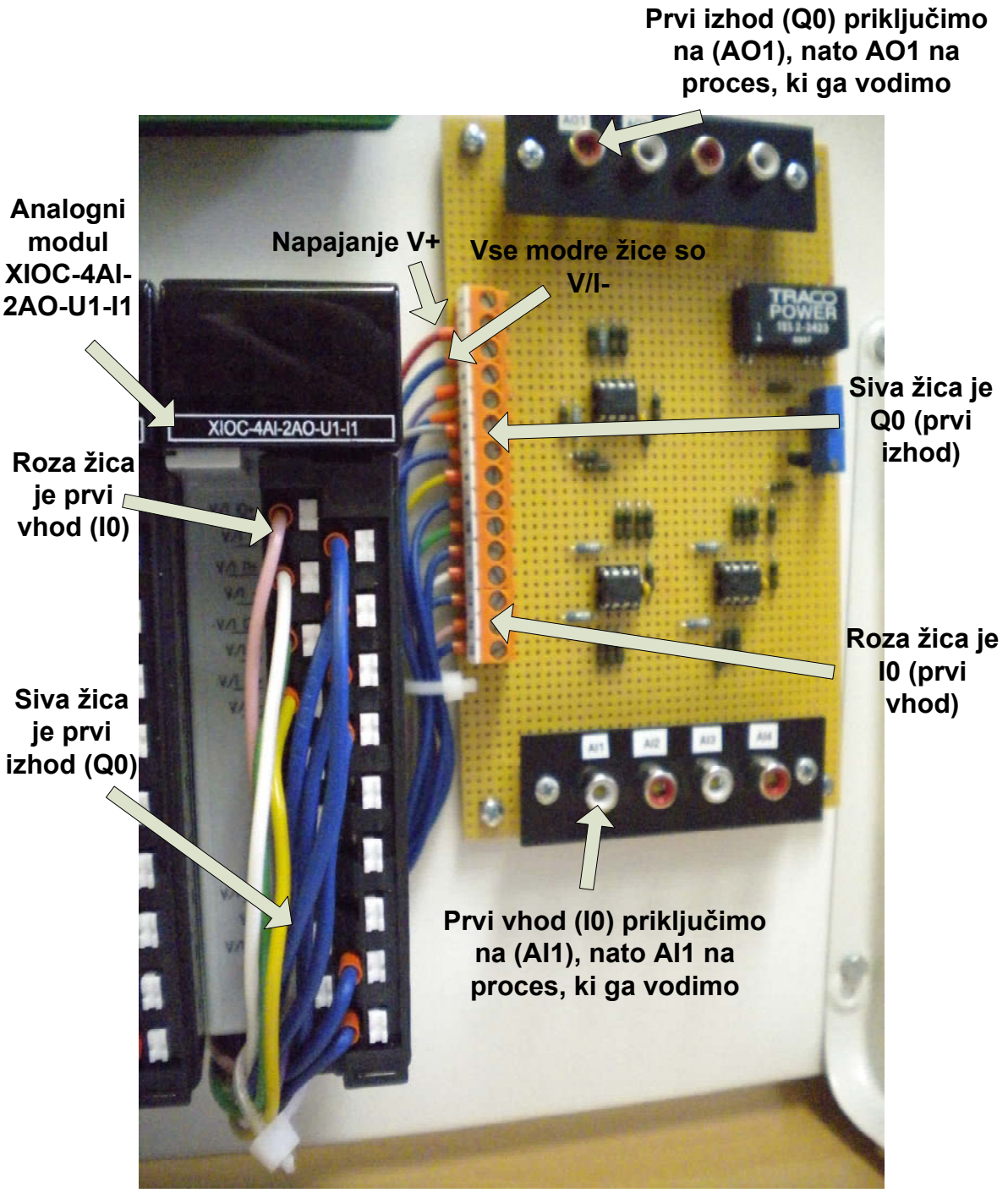
7 OPIS POSTAVITVE ZA ZVEZNO VODENJE

Za načrtovanje regulacijskega (zveznega) sistema bomo poleg procesorske enote XC200 potrebovali modul XIOC-4AI-2AO-U1-I1, ki je analogni vhodno-izhodni modul, ki lahko sprejema ali oddaja zvezno vrednost električnega toka (med 0 in 20 mA) ali napetosti (med 0 in 10 V) (za podrobnejši opis glej poglavje 3). Modula XIOC-16DI (digitalni vhodi) in XIOC-16DO-S (digitalni izhodi) pri regulaciji ne bosta potrebna.

Postavitev za zvezno vodenje se od postavitve za sekvenčno vodenje razlikuje predvsem po tem, da pri zveznem vodenju digitalnih modulov ne potrebujemo več, potrebujemo pa analogni modul, ki oddaja ali sprejema zvezno vrednost napetosti ali toka.

7.1 Modul XIOC-4AI-2AO-U1-I1

Na sliki 7.1 je analogni modul proizvajalca Moeller, ki ima 4 vhode in 2 izhoda. V vezavi na sliki 7.1 smo s procesom en vhod (I0) in en izhod (Q0).



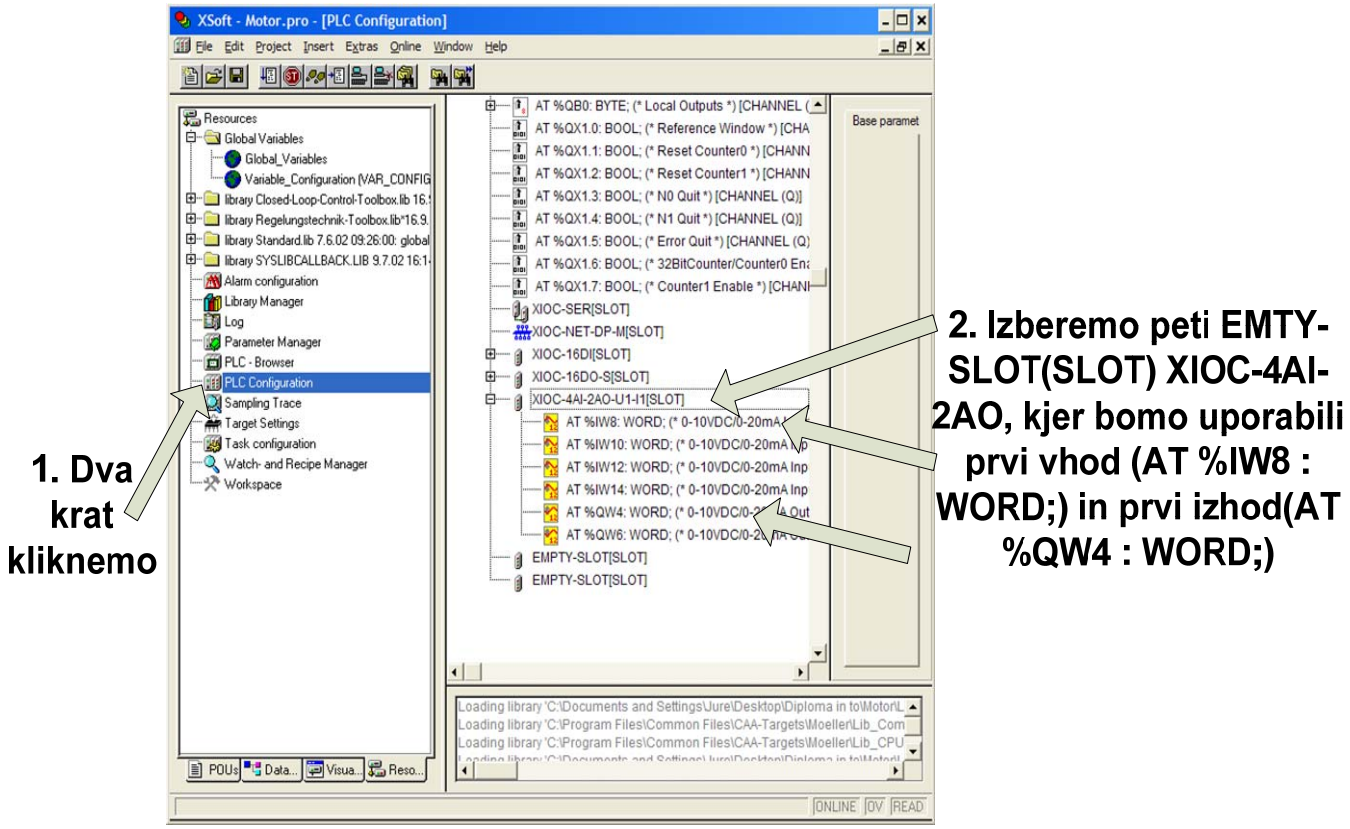
Slika 7.1: Analogni modul XIOC-4AI-2AO-U1-I1

8 OKOLJE Xsoft IN POVEZAVA RAČUNALNIKA S PROCESOM

Program za regulacijo v Xsoftu pišemo tako, kot smo pisali že v prejšnjih poglavjih pri programu za sekvenčno vodenje pralnega stroja (glej poglavje 4). V Xsoftu najprej nastavimo nov prazen projekt, kot smo naredili v podpoglavju 4.4. Nato določimo vhodne in izhodne module, dodamo knjižice za regulacijo ter določimo globalne spremenljivke.

8.1 Določitev vhodnih in izhodnih modulov

Kot smo v podpoglavju 4.6 opisali, kako se v Xsoftu določi modula XIOC-16DI in XIOC-16DO, moramo to storiti tudi za uporabo modula XIOC-4AI-2AO-U1-I1. Modul nastavimo na peto mesto, desno od procesorske enote XC200 (Slika 8.1).



Slika 8.1: PLC Configuration (analogni modul XIOC-4AI-2AO-U1-I1) ter vhodi in izhodi modula

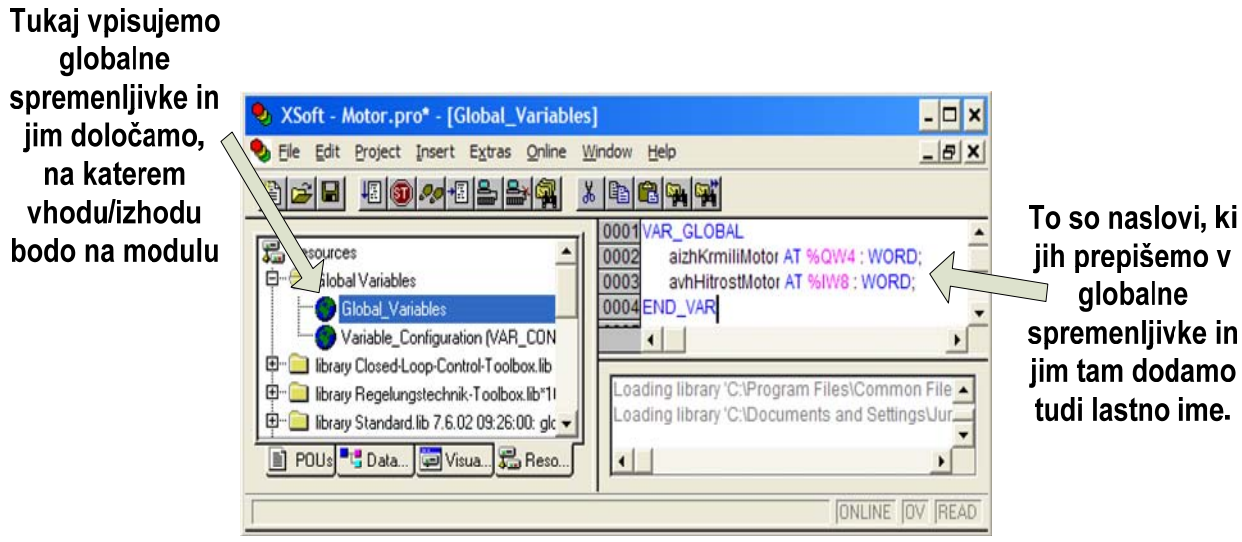
8.2 Dodajanje knjižic za regulacijo

Za regulacijo bomo potrebovali dodatne datoteke, ki jih dobimo na zgoščenki *XSOFT-APPLIB-REG.* V datotekah bomo našli knjižice, ki jih uporabljamo za programiranje zaprtozančnih regulacijskih sistemov. Kot smo opisali že v podpoglavju 4.7, z ukazom ***Aditional Library*** dodamo knjižice ***Regelungstechnik-Toolbox.lib*** in ***Closed-Loop-Control-Toolbox.lib.***

Te knjižice vsebujejo interpolatorje ter proporcionalno-integrirno-diferencirne regulatorje. Interpolatorje uporabimo npr. za interpolacijo med dvema vrednostima, npr. med 8200 in 2560, pri čemer te vrednosti pretvorimo v vrednosti 0 in 1000.

8.3 Določanje globalnih spremenljivk

Globalne spremenljivke določamo, kot smo že opisali v podpoglavju 4.8. Vhodno spremenljivko v analogni krmilnik smo imenovali ***avhHitrostMotor*** in jo deklarirali na prvem vhodu v krmilnik XIOC-4AI-2AO-U1-I1 z ukazom ***»avhHitrostMotor AT %IW8 : WORD;«***, izhodno pa smo poimenovali ***aizhKrmiliMotor*** in jo deklarirali na prvem izhodu istega analognega vhodno-izhodnega krmilnika (glej sliko 8.2). V glavnem programu ***PLC_PRG(PRG)*** pokličemo obe spremenljivki.



Slika 8.2: Določanje globalnih spremenljivk

8.4 Izvedba programa

Za izvedbo načrtovanega vodenja poleg osebnega računalnika potrebujemo programsko okolje Xsoft, knjižice iz zgoščenke *XSOFT APPLIB-REG*, PLC-vmesnik in krmilnik proizvajalca Moeller (procesorsko enoto in analogni modul).

V našem primeru bomo opisali dva najpogosteje uporabljena funkcijska bloka iz skupine standardni funkcijski bloki, in sicer interpolacijo in PID-regulator. Oba bloka se nahajata v knjižicah z naslovom *XSOFT APPLIB-REG*. Poleg teh je v knjižicah mnogo drugih blokov, enostavnejših in zahtevnejših. Ti so na primer:

- osnovni zaprtozančni bloki (diferenciator, integrator, oscilator itd.),
- regulatorji (proporcionalno-integrirno-diferencirni regulatorji),
- matematični bloki (prištevanje, odštevanje, deljenje, množenje, potenciranje itd.),
- filtri signalov (omejevalniki signala, filtri itd.),
- vizualizacijski bloki (osciloskop, izrisovanje podatkov).

8.4.1 Dvotočkovna interpolacija

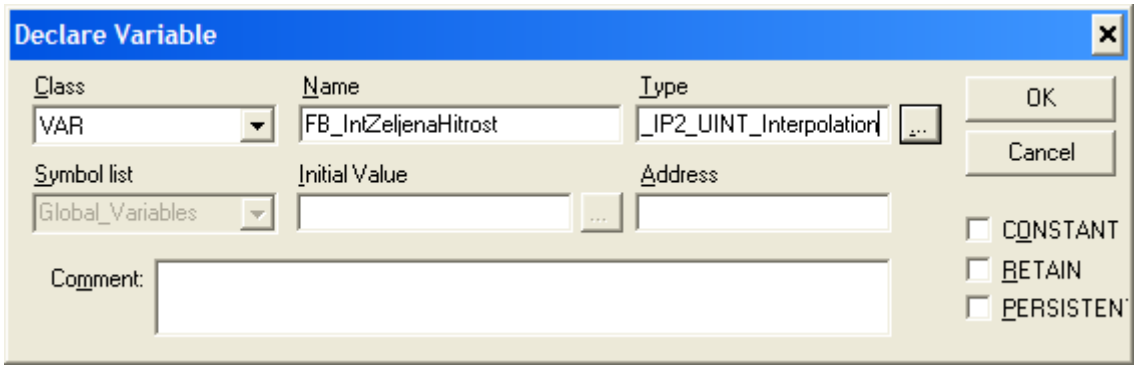
Dvotočkovna interpolacija pove, kako določiti neznano vrednost med dvema točkama. Dvotočkovno interpolacijo bomo potrebovali, da bomo napetostni signal, ki ga bomo dobili na vhod krmilnika, spremenili v števila od 0 do 1000.

Naš primer smo izdelali po preimeru *"Dvotočkovne interpolacije"* iz dokumentacije ***Docu-XSOFT-APPLIB-REG-gb.pdf***, ki se nahaja na zgoščenki *XSOFT-APPLIB-REG* (poglavje ***Basic Information and Technical Data***).

V Xsoftu najprej definiramo globalne spremenljivke želene in dejanske vrednosti regulirane spremenljivke. Za želeno spremenljivko smo izbrali ime ***uiZeljenaHitrost***, za dejansko regulirano spremenljivko pa ***uiTrenutnaHitrost***. Obema spremenljivkama smo dodelili tip ***UINT***, ki predstavlja 16-bitno spremenljivko.

Deklaracija interpolacije

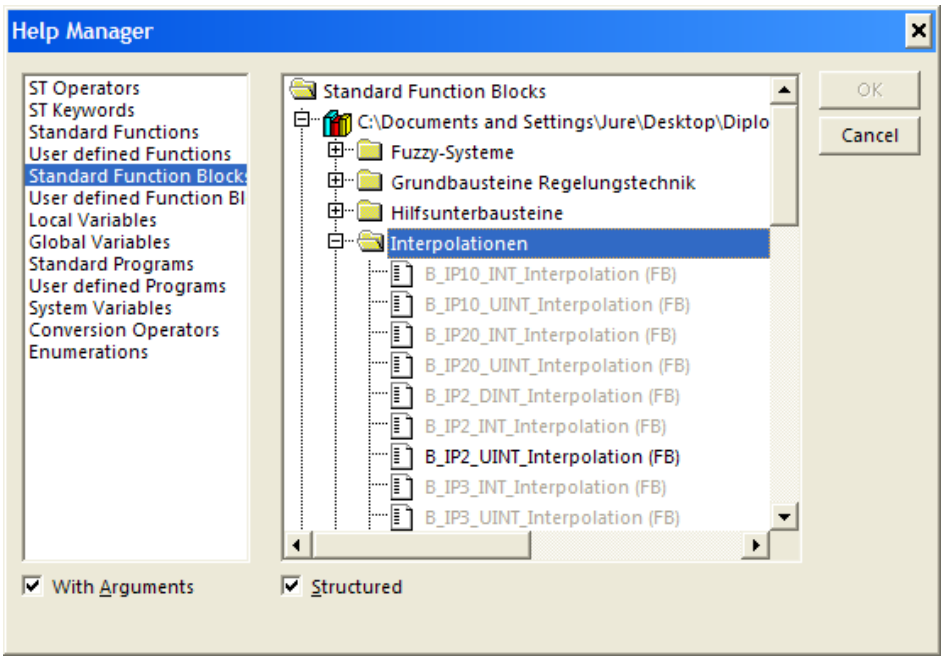
Nato v glavnem programu ***PLC_PRG(PRG)***, ki smo ga prej ustvarili, v deklarativnem delu Xsofta s klikom na desno tipko miške odpremo meni, ki omogoča avtodeklaracijo novih spremenljivk (angl. Auto Declare) ali pa pritisnemo tipko ***Shift+F2***. Pod ***Type*** poiščemo ***»B_IP2_UINT_Interpolation«***, izberemo ime spremenljivke ter deklaracijo potrdimo. S tem smo definirali interpolacijo. V našem primeru smo deklarirali 2 interpolatorja. Enega za želeno vrednost spremenljivke ***(FB_IntZelenaHitrost :B_IP2_UINT_Interpolation;)*** in enega za dejansko vrednost spremenljivke ***(FB_IntTrenutnaHitrost :B_IP2_UINT_Interpolation;)*** (slika 8.3).



Slika 8.3: Deklaracija spremenljivk za interpolacijo

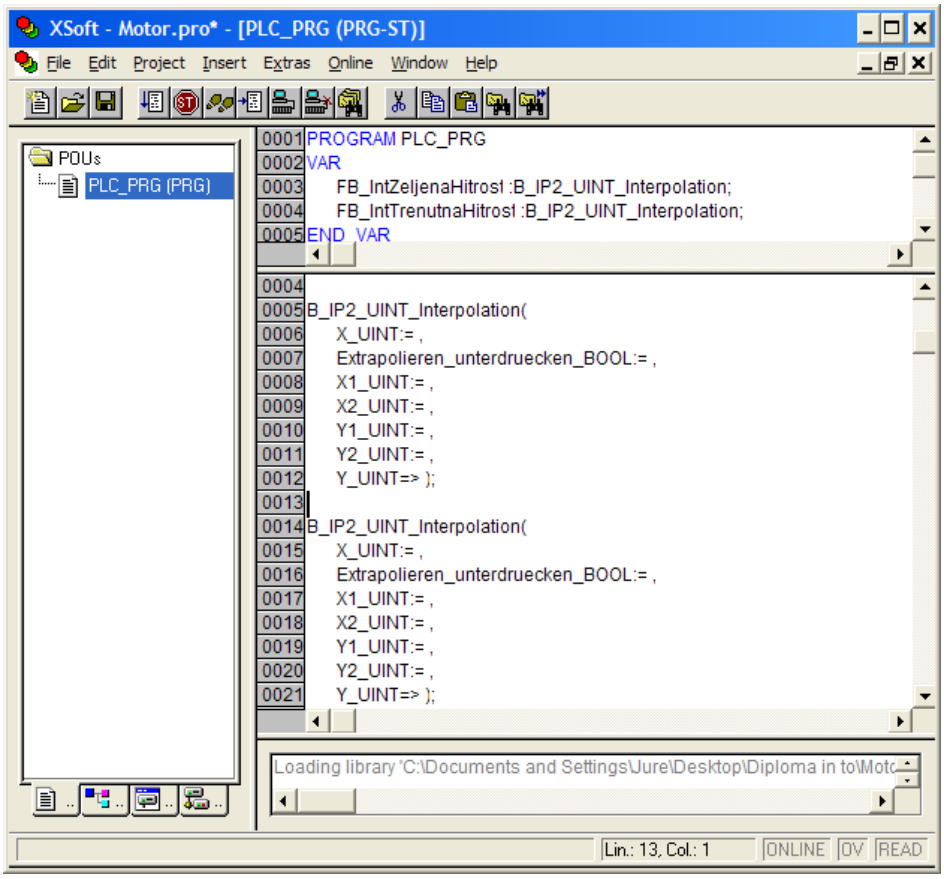
Dodanjanje interpolacije

Sedaj lahko v oknu, kjer pišemo glavni program, izberemo ***Input Assistant*** s pritiskom na tipko ***F2***. Pod ***Standard Function Blocks*** (slov. Standardni funkcijski bloki) razširimo pogled na ***»Interpolationen«***, kot prikazuje slika 8.4. Nato izberemo ***B_IP«_UINT_Interpolation*** (FB), ki predstavlja funkcijski blok, ki vsebuje interpolator.



Slika 8.4: Dodajanje interpolatorjev

Po deklaraciji in dodanih interpolatorjih dobimo, kar prikazuje slika 8.5.



Slika 8.5: Prazna funkcijska bloka interpolacije in deklaracija blokov

B_IP2_UINT_Interpolation spremenimo v ***FB_IntTrenutnaHitrost***, saj smo spremenljivko že deklarirali in ji dodelili ime. Za vhod ***X_UINT:=*** napišemo ***avhHitrostMotor***. Za ***Suppress_extrapolation_BOOL:=*** napišemo ***FALSE***. Ugotovili smo, da ima spremenljivka ***avhHitrostMotor*** vrednost 8191, ko miruje, zato smo to vpisali za vrednost ***X1_UINT***. Največja vrednost regulirane spremenljivke je 2560, zato smo to vpisali za vrednost spremenljivke ***X2_UINT***. Vrednost regulirane spremenljivke bomo merili do desetinke odstotka natančno, zato bomo za ***Y1_UINT*** dali vrednost 0, za ***Y2_UINT*** pa 1000. Za izhod iz interpolatorja ***(Y_UINT)*** pa smo določili spremenljivko ***uiTrenutnaHitrost***.

8.5 PID-regulator (U_PID_controller) iz knjižic Closed-Loop Control Toolbox

Blok zaprtozančnega sistema s PID (proporcionalno-integrirno-diferencirnim) regulatorjem imamo v knižici ***Closed-Loop-Control-Toolbox.lib.*** Moramo ga samo dodati v Xsoft, mu dodeliti ime, mu definirati vhoda ***(Setpoint_value_12Bit_UINT, Actual_value_12Bit_UINT)***, izhod ***(Manipulated_variable_12Bit_UINT)*** ter ostale parametre. Najprej s tipko ***F2*** prikličemo meni za dodajanje knjižic v program. V podmeniju ***Standard Function Block*** poiščemo ***Controllers***. Nato v podmeniju poiščemo ***U_PID_Controller (FB)*** in ga s potrditvijo dodamo v naš program.

$$Y = KpE + \frac{Kp}{Tn}\int Edt + KpTv\frac{dE}{dt})$$

Kjer so:

Y – izhod iz regulatorja (manipulirna spremenljivka)

E – vhod v regulator (pogrešek)

Kp – proporcionalno ojačanje

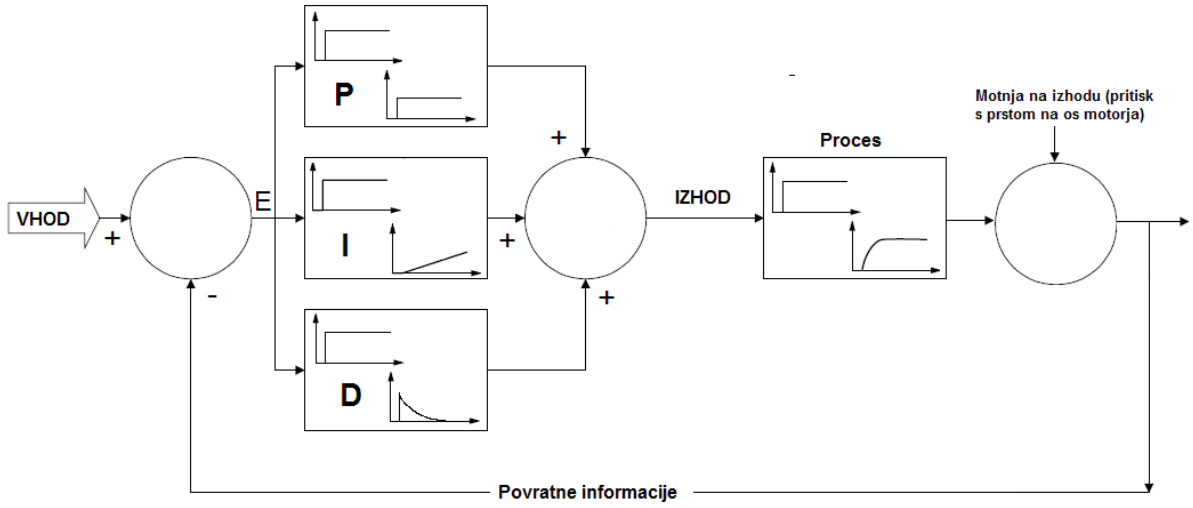
Tn – integrirna časovna konstanta

Tv – diferencirna časovna konstanta

PID-regulator ima tri sestavne dele. Prvi je proporcionalni del, ki ima to lastnost, da po nenadni spremembi na vhodu izhodna vrednost po nekem času znova prevzema vrednost iz konstante (referenčno vrednost). Drugi je integrirni del, izhodna vrednost tega sistema se bo prikazala kot nenaden padec oziroma rast vhodne veličine. Tretji je diferencialni del z mrtvim časom. Mrtvi čas v diferenciranem delu pripomore k temu, da se sistem ne odziva prehitro in

s tem ne povzroči prenihajev napetosti. Grafi vhodnih in izhodnih signalov PID-regulatorja so skicirani na sliki 8.6.

Proporcionalno-integrirno-diferencirni (PID) regulator združuje vse prej navedene funkcije in se pogosto uporablja. Prednosti so, da reagira hitro na odstopanja in deluje na tak način, ki zagotavlja, da izgine pogrešek v ustaljenem stanju. Poudarek je torej na posameznih regulacijskih komponentah, ki so odvisna od določenih parametrov.



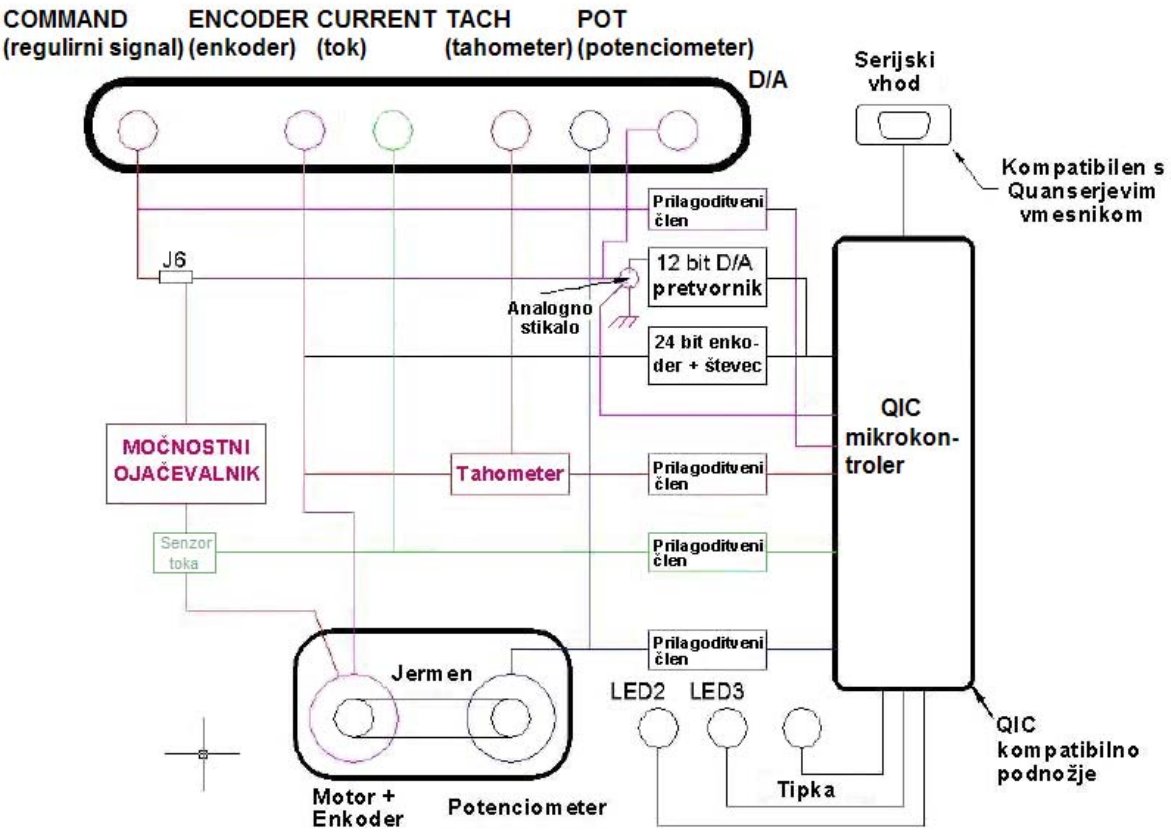
Slika 8.6: Shema zaprtoznačnega sistema s PID-regulatorjem z grafičnim prikazom odziva posameznih komponent

Ko dodamo blok ***U_PID_controller*** v program na sliki 8.5, dobimo naslednje:

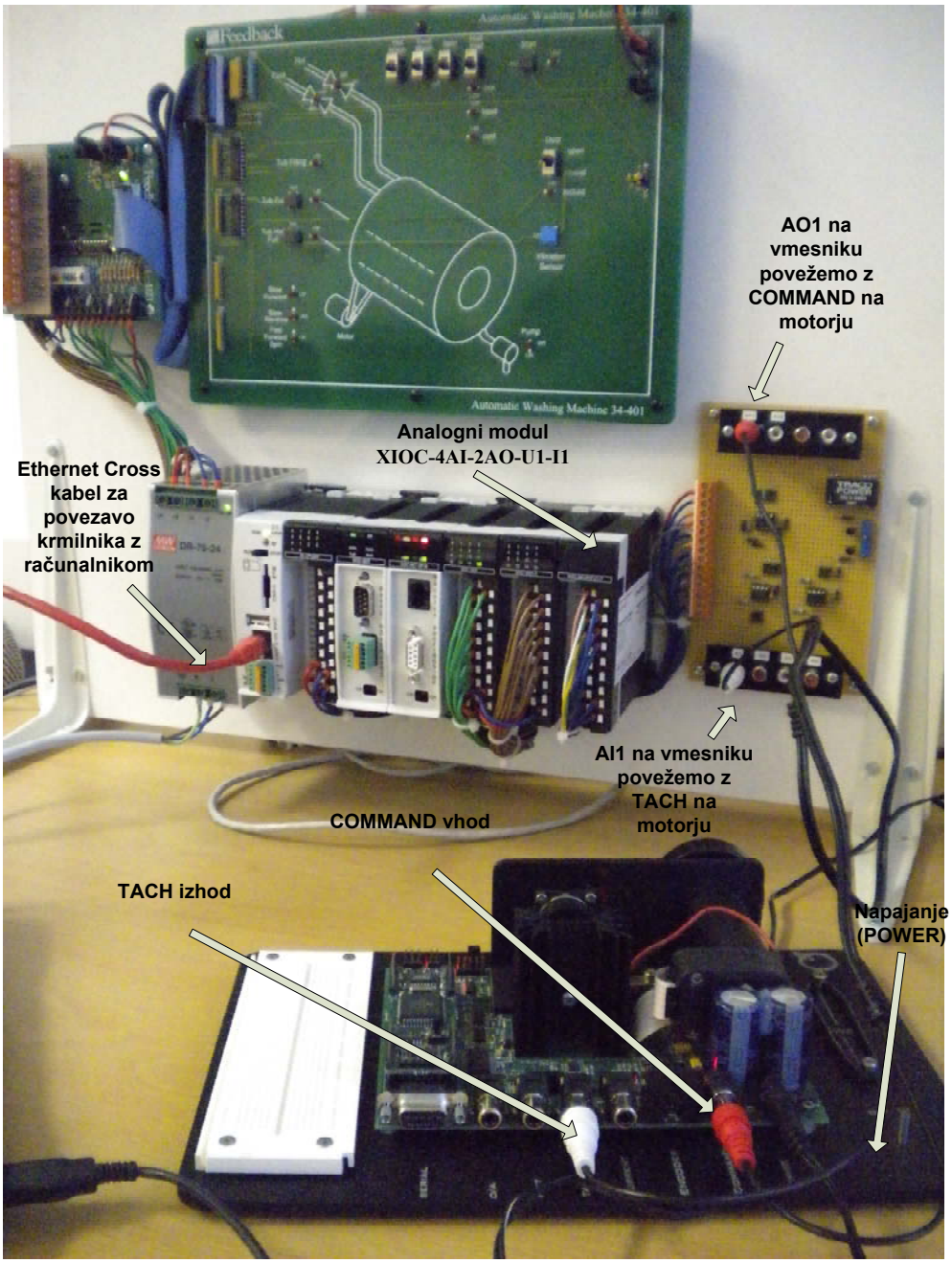
```
U_PID_controller(
      Setpoint_value_12Bit_UINT:= ,                (želena ali nominalna vrednost)
      Actual_value_12Bit_UINT:= ,                  (dejanska vrednost regulirane spr.)
      P_activate_BOOL:= ,                          (aktivira P komponento, če je TRUE)
      I_activate_BOOL:= ,                          (aktivira I komponento, če je TRUE)
      D_activate_BOOL:= ,                          (aktivira D komponento, če je TRUE)
      Proportional_rate_percent_UINT:= ,           (ime spremenljivke za Kp)
      Reset_time_10ths_UINT:= ,                    (ime spremenljivke za Tn)
      Derivate_action_time_10ths_UINT:= ,          (ime spremenljivke za Tv)
      Manipulated_variable_P_13Bit_INT=> ,         (manipulirana spremenljivka P dela)
      Manipulated_variable_I_13Bit_INT=> ,         (manipulirana spremenljivka I dela)
      Manipulated_variable_D_13Bit_INT=> );        (manipulirana spremenljivka D dela)
```

9 MODELNA NAPRAVA ENOSMERNI ELEKTRIČNI MOTOR DCMT (DC MOTOR TRAINER) ZA POUČEVANJE REGULACIJE

Modelna naprava *enosmerni električni motor* (DC motor) je prilagodljiva enota za poučevanje in prikazovanje osnov avtomatskega vodenja na različnih primerih, ki se vsebinsko razlikujejo. Sistem je pripravljen za regulacijo položaja in hitrosti vrtenja motorja. Posebno primeren je za prikaz delovanja proporcionalno-integrirno-diferencirnih regulatorjev (PID) in njihovega načrtovanja. Povezava enosmernega električnega motorja z Moellerjevim krmilnikom XC200 omogoča izvajanje preddefiniranih eksperimentov za modeliranje motorja, regulacijo hitrosti motorja ter preizkušanje robustnosti vodenja. Kot je bilo že omenjeno, pa smo se v nalogi omejili le na dve nalogi, to sta modeliranje in regulacija hitrosti motorja. Enosmerni električni motor je prikazan na sliki 9.2 in shematsko na sliki 9.1.



Slika 9.1: Shematski prikaz celotnega sistema DC-motorja



Slika 9.2: Modelna naprava Quanser enosmerni električni motor, povezan z napajanjem in modulom XIOC-4AI-2AO-U1-I1

9.1 Analogno merjenje hitrosti in linearni ojačevalnik

TACH- tahometer

Analogno merjenje hitrosti – analogni signal je sorazmeren s hitrostjo in je na voljo na vtičnici RCA z oznako ***Tach***. Njegovo območje je ± 5 voltov. Signal je skaliran in pretvorjen na območje 0–5 voltov in je na voljo glavni enoti QIC.

COMMAND – ukazovalnik

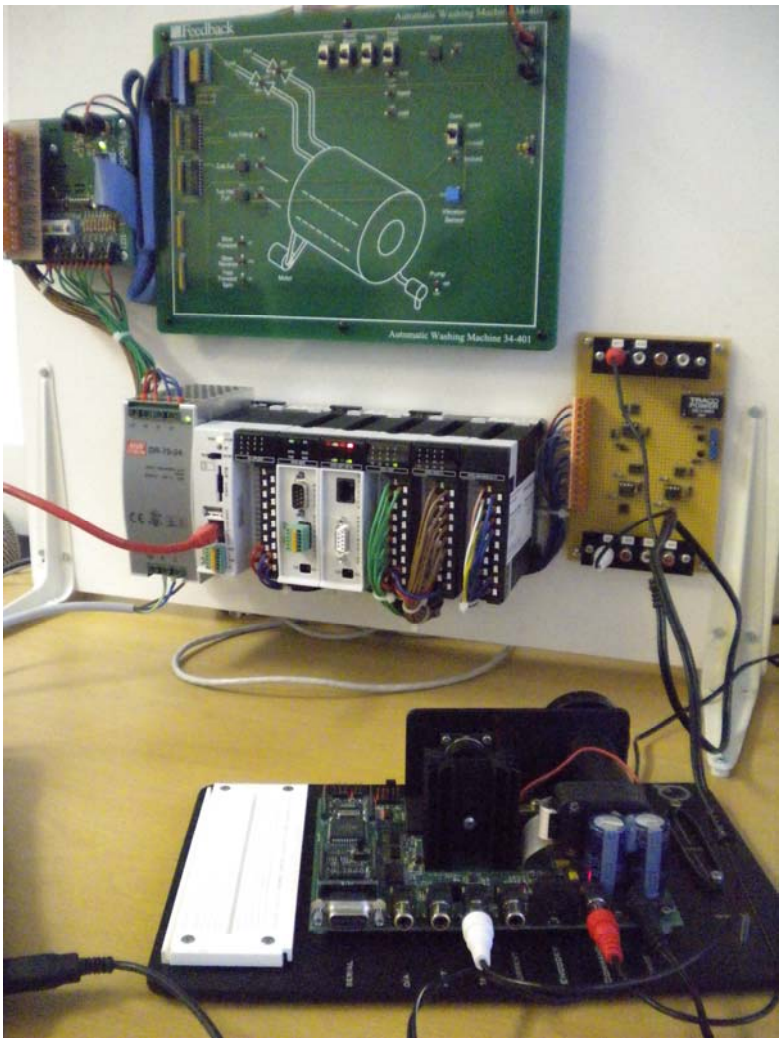
Linearni ojačevalec – linearni ojačevalnik moči se uporablja za pogon motorja. Vhod v ojačevalnik je lahko nastavljen tako, da je napetost bodisi na priključku RCA z oznako ***Command*** ali da je izhod vgrajenega D/A. Vgrajen D/A je mogoče uporabljati le, če je enota QIC povezana s sistemom in je nameščeno ustrezno stikalo (J6).

9.2 Povezava med osebnim računalnikom, krmilnikom z analognim modulom in vodenim procesom

Računalnik in krmilnik povežemo, kot smo opisali v poglavju 4.2. Za povezavo med krmilnikom in enosmernim električnim motorjem pa potrebujemo dodaten povezovalni kabel (slika 9.3). Na vmesniku en vodnik povežemo z izhodom ***AO1***, drugega pa z vhodom ***AI1***. Vodnik, ki je povezan na ***AO1***, na motorju povežemo na vhod ***COMMAND, AI1*** pa z izhodom motorja ***TACH***. Nato moramo na motor povezati še napajanje iz električne vtičnice v vhod z oznako ***POWER*** (slika 9.4).

Slika 9.3: Avdio kabel (4 x vmesnik RCA vtikač) za povezavo med motorjem in vmesnikom krmilnika

10 POSTOPEK NAČRTOVANJA ZVEZNEGA VODENJA



Slika 10.1: Povezava med motorjem in vmesnikom krmilnika z avdio kablom

10.1 Zahteve načrtovanja zveznega vodenja

Zamislili smo si primer, pri katerem se v nekem trenutku zažene motor na 50 % največje možne hitrosti, po približno petih sekundah pa naj se motor zopet ustavi. Pri tem si izberemo take parametre *Kp*, *Tn* in *Tv*, da bo motor dosegel želeno hitrost v manj kot treh sekundah brez prenihaja.

V drugem primeru naredimo enako, vendar ko motor deluje s 50% hitrostjo, z roko nežno pritisnemo na os motorja. S tem ustvarimo motnjo na izhodu sistema, ki jo mora regulator odpraviti. Tudi to mora regulator opraviti v manj kot treh sekundah.

Za izris grafa, ki prikazuje želeno hitrost in dejansko hitrost, uporabimo pripomoček za izrisovanje grafov *Sampling trace*. Program napišemo s knjižicami, ki smo jih opisali v poglavju 8.

10.2 Specifikacije in programiranje zveznega vodenja

Ime, ki ga nameravamo dodeliti regulatorju, moramo najprej definirati. Izbrali smo si ime in ga v deklarativnem oknu XSofta definirali: ***FB_RegulatorHitrosti :U_PID_controller***.

Za želeno vrednost izberemo ***ime uiZeljenaHitrost***:
Setpoint_value_12Bit_UINT:= uiZeljenaHitrost.

Za dejansko vrednost pa ***uiTrenutnaHitros***t:
Actual_value_12Bit_UINT:= uiTrenutnaHitrost.

P_activate_BOOL aktivira proporcionalno komponento, zato jo nastavimo na ***TRUE***
P_activate_BOOL:= TRUE,
I_activate_BOOL aktivira integracijsko komponento, zato jo nastavimo na ***TRUE***
I_activate_BOOL:= TRUE,
D_activate_BOOL aktivira diferencialno komponento, zato jo nastavimo na ***TRUE***
D_activate_BOOL:= TRUE.

Sedaj definiramo globalne spremenljivke *Kp*, *Tn*,*Tv*. Vse so tipa ***UINT***, kar pomeni, da imajo lahko vrednost od 0 do 65535. Ko jih dodajamo, označimo ***Retain*** in ***Persistent*** ali pa napišemo, kot je spodaj. Lastnosti ***Retain*** in ***Persistent*** pomenita, da se spremenljivke ne bodo izbrisale iz krmilnika niti po tem, ko izklopimo električno napajanje. To bo omogočilo izključitev vsakega od PID-komponent, brez da bi se PID-regulator v Xsoftu resetiral. *Kp* je proporcionalno ojačanje, *Tn* integrirna časovna konstanta in *Tv* diferencirna časovna konstanta.

```
VAR_GLOBAL RETAIN PERSISTENT
        Kp : UINT:= 100;
        Tn : UINT:= 30;
        Tv : UINT:= 10;
END_VAR
```

Na koncu program izgleda tako:

```
FB_RegulatorHitrosti(
        Setpoint_value_12Bit_UINT:= uiZeljenaHitrost,
        Actual_value_12Bit_UINT:= uiTrenutnaHitrost,
        P_activate_BOOL:= TRUE,
        I_activate_BOOL:= TRUE,
```

```
D_activate_BOOL:= TRUE,
Accept_manual_manipulated_variable_BOOL:= ,
Proportional_rate_percent_UINT:= Kp,
Reset_time_10ths_UINT:= Tn,
Derivate_action_time_10ths_UINT:= Tv,
Manipulated_variable_P_13Bit_INT=> ,
Manipulated_variable_I_13Bit_INT=> ,
Manipulated_variable_D_13Bit_INT=> );
```

To je proporcionalno-integrirno-diferencirni regulator, s katerim bomo regulirali hitrost motorja in kljub motnjam obdržali želeno hitrost. Od nastavitve parametrov *Kp, Tn* in *Tv* bo odvisen odzivni čas hitrosti motorja in njeno nihanje. Poiskati moramo take parametre, da bo hitrost motorja čim hitreje in brez prenihajev dosegla želeno vrednost. Vrednosti želene hitrosti lahko določimo s številom od 0 do 1000. Število 0 pomeni, da je želena hitrost 0, število 1000 pa pomeni 100% hitrost, ki jo povzroči napetost 5V.

10.3 Nastavljanje parametrov regulatorja

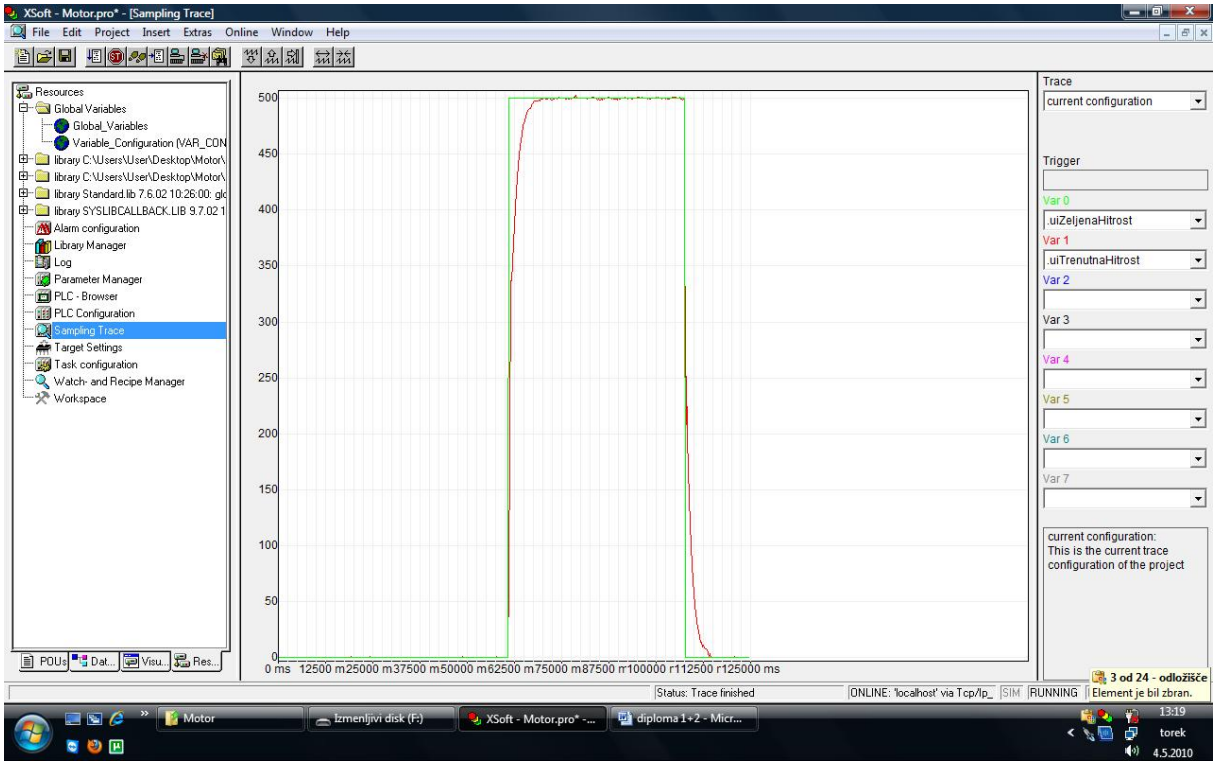
Zaključen program nato prenesemo iz osebnega računalnika na krmilnik (glej poglavje 4.12).

V našem primeru smo parametre nastavili na vrednosti *Kp*= 500, *Tn*= 10, *Tv*= 3, saj smo s preizkušanjem ugotovili, da tako nastavljeni parametri dosežejo želeno hitrost v času do dveh sekund. S tako nastavljenimi parametri smo s programom *Sampling trace*, ki se nahaja v paketu XSoft, izrisali graf, ki prikazuje želeno vrednost in dejansko hitrost (slika 10.2).

10.4 Preizkus delovanja

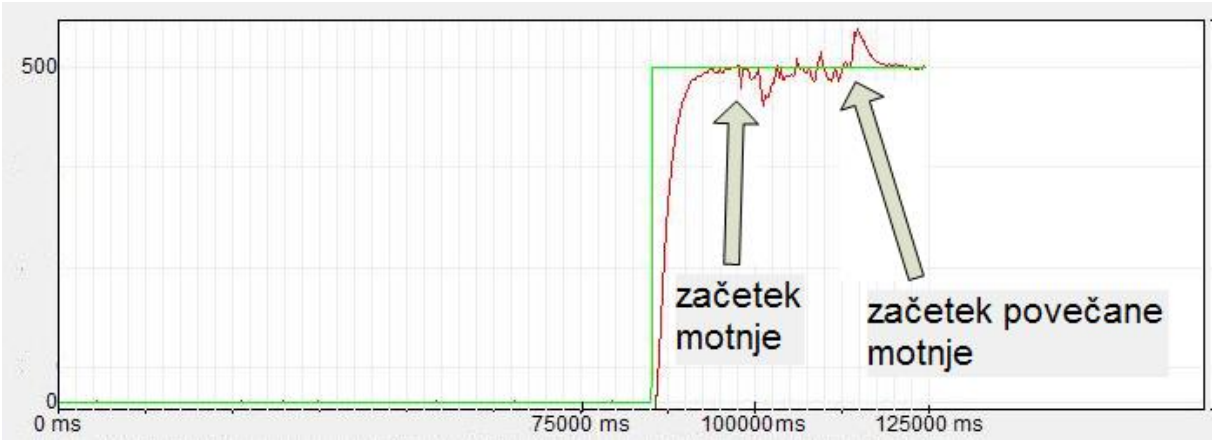
Pod jezičkom ***Resources*** v levem oknu programa XSoft kliknemo na ***Sampling trace***. Nato z izbiro spremenljivk določimo, katere spremenljivke bi radi izrisali na grafu. V našem primeru smo za prvo spremenljivko izbrali ***.uiZeljenaHitrost***, za drugo spremenljivko pa ***uiTrenutna hitrost***. Nato kliknemo z desno tipko miške na okno, kjer se bo izrisoval graf, ter iz menija izberemo ***Auto read trace***, kar sproži izrisovanje grafa v realnem času.

V samem programu dvakrat kliknemo na ***uiZeljenaHitrost*** in v okence, ki se odpre, napišemo neko vrednost od 0 do 1000. V našem primeru smo nastavili na vrednost 500, po nekem času (približno 2 sekundi) pa zopet na 0. Program ***Sampling trace*** izriše graf, kot ga prikazuje slika 10.2.



Slika 10.2: Graf želene in dejanske hitrosti motorja

V drugem preizkusu bomo prav tako želeli 50 % največje možne hitrosti, vendar bomo takoj, ko se hitrost ustali, dodali motnjo (pritisk z roko na os motorja) na izhodu procesa. S tem bomo preizkusili, ali deluje regulacija hitrosti tudi z dodano motnjo in kako hitro regulator motnjo odpravi. Na sliki 10.3 je graf, ki prikazuje odpravljanje motnje.



Slika 10.3: Graf želene in dejanske hitrosti motorja v prisotnosti motnje

Kot vidimo na sliki 10.3, je regulator odpravljal motnjo, ki smo jo dodali s pritiskom roke na os motorja. Ko smo roko odmaknili, se je hitrost motorja zopet izenačila z želeno.

Z delovanjem smo bili zadovoljni, saj smo uspeli sprogramirati tak regulator, ki doseže želeno vrednost (hitrost motorja) v manj kot treh sekundah, kljub morebitnim motnjam na izhodu procesa (os motorja). S tem smo izpolnili zahteve načrtovanja zveznega vodenja.

11 ZAKLJUČEK

Pri izvedbi načrtovanja vodenja se srečujemo z bolj ali manj težavnimi primeri. V diplomskem delu smo se odločili obravnavati dve različici avtomatskega vodenja. To sta sekvenčno vodenje in regulacija.

V prvem delu smo se osredotočili na načrtovanje, simulacijo in izvedbo sekvenčnega računalniškega vodenja s krmilniki. Načrtovanje smo izvedli s programom XSoft, ki omogoča programiranje, simulacijo ter vizualizacijo in je namenjen za programiranje krmilnikov. Za samo izvedbo vodenja pa smo uporabili sekvenčne krmilnike proizvajalca Moeller. Preizkus smo v prvem delu izvedli z elektronskim simulatorjem pralnega stroja znamke Feedback.

V drugem delu smo v ospredje postavili zvezno vodenje procesov, ki je za nas predstavljalo regulacijo hitrosti enosmernega električnega motorja. Tudi v tem delu smo potrebovali program XSoft, vendar smo namesto simulatorja pralnega stroja uporabili enosmerni električni motor znamke Quanser, namesto sekvenčnih vhodno-izhodnih modulov pa analogne vhodno-izhodne module istega proizvajalca. Za vizualizacijo delovanja smo uporabili pripomoček *Sampling trace*, ki se ravno tako nahaja v programu XSoft. Zgoščenko, na kateri so bile knjižice, razumljive programu XSoft, v katerih se nahajajo regulatorji in interpolatorji, ki smo jih potrebovali, nam je prijazno posredovalo podjetje Synatec d.o.o. iz Idrije.

S tem smo izpolnili naš namen, ki je bil podrobnejša obravnava dveh vrst avtomatskega vodenja, sekvenčnega in regulacije. Dosegli smo postavljene cilje, saj smo v prvem primeru uspeli sprogramirati sekvenčni program, ki izvaja različne programe pranja ter se tudi uspešno zaključi. V drugem delu smo tudi dosegli postavljene cilje, saj smo uspešno sprogramirali program za regulacijo enosmernega električnega motorja, ki v manj kot treh sekundah zregulira hitrost in pri tem ne povzroči prenihajev v napetosti.

V prvem delu smo imeli prednost v tem, da smo lahko programirali sekvence v relativno enostavnem jeziku (strukturiran tekst), slabost pa je bila dolžina in kompleksnost, ki smo si jo zadali v sami strukturi programa. Zaradi tega je programska koda relativno dolga. V drugem delu smo kot prednost imeli na razpolago knjižnice, razumljive programu XSoft, iz katerih

smo uporabili že vnaprej sprogramirane programske bloke (regulatorji, interpolatorji). Slabost tega dela pa je bila kompleksnost ter posledično netransparentnost ravno teh vnaprej sprogramiranih blokov.

V diplomskem delu seveda nismo prikazali vseh možnosti programske opreme, ki smo jo uporabljali. Tudi strojna oprema (simulator pralnega stroja, enosmerni električni motor in programirljivi krmilniki) omogoča številne dodatne možnosti, ki jih nismo izkoristili. To so na primer regualcija hitrosti motorja z mikrokrmilnikom Quanser, ki se nahaja na plošči poleg motorja.

Delo z Moellerjevimi krmilniki je kompleksno, vendar je s pomočjo zaposlenih v podjetju Synatec d.o.o. iz Idrije potekalo brez večjih zapletov. Simulator pralnega stroja smo povezali s krmilniki ter to opisali. Programske sekvence smo programirali strukturiranem jeziku. Pri regulaciji hitrosti motorja smo imeli opravka z analognimi signali in zveznim regulatorjem na krmilniku. Povezali smo krmilnik z motorjem ter napisali program za regulacijo hitrosti. S tem smo študentom omogočili boljše razumevanje principov avtomatizacije ter strojne opreme Moeller, jim omogočili boljši vpogled v krmilnik ter naredili osnovo za nove aplikacije v smeri realizacije računalniškega avtomatskega vodenja s krmilniki.

12 SEZNAM VIROV IN LITERATURE

Strmčnik, S. (ur.) (1998). Celostni pristop k računalniškemu vodenju procesov. Ljubljana: Fakulteta za elektrotehniko.

Feedback Instruments. (2006). Feedback PLC Applications 34-400. England: FI Ltd

Kocijan, J. (1996). Načrtovanje in vodenje dinamičnih sistemov. Ljubljana: Fakulteta za elektrotehniko.

Kocijan, J. (2006). Dodatno gradivo iz osnov avtomatskega vodenja. Interno gradivo. Nova Gorica

Matko, D. (1995). Računalniško vodenje procesov. Ljubljana: Fakulteta za elektrotehniko in računalništvo

Strmčnik, S. (ur.) (1998). Celostni pristop k računalniškemu vodenju procesov. Ljubljana: Fakulteta za elektrotehniko.

Zupančič, B. (1992). Zvezni regulacijski sistemi. 1. del. Ljubljana: Fakulteta za elektrotehniko in računalništvo.

Aström, K. Et al. (2006). DC Motor Control Trainer. Instructor Workbook. Canada: Quanser.

Wikipedia. Pridobljeno 26.05.2009 s svetovnega spleta: http://en.wikipedia.org/wiki/Mindstorms

SoftPerfect Network Scanner. Pridobljeno 17.6.2009 s svetovnega spleta: http://softperfect.com/products/networkscanner

Quanser. Pridobljeno 2.5.2010 s svetovnega spleta: http://www.quanser.com/english/downloads/products/QET_Motor_Trainer.pdf