Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство
Ниво студија: Мастер академске студије
Модул: Примењена механика и аутоматско управљање
Предмет: Индустријски рачунарски системи
Број индекса: 348/2015

Андрија С. Јестровић

Пројектовање SCADA апликације и система заснованог на раду четири једносмерна и асинхрона мотора

Мастер рад

Комисија за преглед и одбрану:

1. **др Милан Матијевић - ментор**
2. _____ Датум одбране:_____
3. _____ Оцена:_____

У оквиру овог мастер рада кандидат треба да:

Кандидату је на располагању опрема лабораторијског простора *C85* Центра за примењену аутоматику Факултета инжењерских наука Универзитета у Крагујевцу. Потребно је описати интеграцију система који се састоји од личног рачунара за управљање и надзор, главног (*master*) ПЛЦ рачунара, два једносмерна мотора, два фреквентна регулатора, два асинхрона мотора. У мери у којој је потребно описати систем у целости и његовим елементима, и посебно како се елементи система спајају. У тренутној ситуацији, потребно је направити интерфејс између ПЛЦ-а и једносмерног мотора са интегрисаним инкременталним енкодером. Изабрати адекватну апликацију засновану на раду четири мотора (два једносмерна и два асинхрона) и описати процес креирања *SCADA* апликације. Илустровани пример треба да садржи снимљене одзиве трајекторија (пут, брзина и евентуално убрзање) контролисаних мотора. Циљ је грађење једног спремног лабораторијског модела на коме могу да се програмирају различите *SCADA* апликације засноване на раду четири мотора са интегрисаним инкременталним енкодерима - и то остварење са илустрованим примером програмирања *SCADA* апликације која користи сва четири мотора је главни допринос овог Мастер рада.

Препоручена литература:

[1] Матијевић, М., Јакуповић, Г., Цар, Ј.: Рачунарски подржано мерење и управљање, Машински факултет у Крагујевцу, друго издање, Крагујевац, Септембар, 2008.

[2] Khamel, K., Khamel, E.: Programmable Logic Controllers - Industrial Control, McGraw-Hill Education, 2014, e-Book Cenveo Publisher Services, Version 1.0.

[3] Луковић, В.: Програмирање и примена ПЛЦ контролера *Siemens SIMATIC S7-1200*, Дипломски рад, Машински факултет у Крагујевцу, Универзитет у Крагујевцу, Крагујевац, 2010.

Крагујевац, 2017.

Ментор:
др Милан Матијевић, ред. проф.

_____

Садржај

Резиме

У раду су објашњени основни принципи рада ПИД контроле, ПЛЦ контролера и *SCADA* система, који су неопходни за разумевање рада.

Објашњен је принцип рада свих делова који сачињавају наш *SCADA* систем као и како су они повезани да би систем исправно радио. Развијена је прилагодна електроника која омогућава исправну комуникацију између једносмерног мотора и управљачког уређаја.

На крају је објашњен поступак развијања *SCADA* апликације која се диже на наше ПЛЦ-ове и *HMI* уређаје, како би они лепо комуницирали међусобно и омогућили кориснику да задаје дозволе рада као и жељене вредности брзине или позиције мотора, које систем касније сам налази.

Кључне речи: *SCADA*, ПИД, ПЛЦ, асинхрони мотори, једносмерни мотори, управљање.

Abstract

In this paper, the basic principles of PID control, PLC controllers and SCADA systems, are explained, which are necessary for the understanding of this paper.

The working principles of all the components which make up our SCADA system are explained, as well as the manner in which they are connected, so that our system would work properly. Adaptive electronics have been developed, so that our direct current motor can correctly communicate with the control device.

In the final part, the procedure for developing our SCADA application, which is uploaded on to our PLCs and HMI devices, is explained. The goal of the application is the establishment of communications between our devices and to enable the user to give work permissions and setpoint values for speed or position of motors, which the system then achieves by itself.

Key words: SCADA, PID, PLC, asynchronous motors, direct current motors, control.

Увод

Аутоматика је данас јако распрострањена у разним сферама, од корисничких потрошачких уређаја до индустријских система. Она представља коришћење система контроле како би се вршили управљање разним машинама, процесима у индустрији, контроле температуре, управљањем и стабилизацијом авиона и бродова, контролу телефонских мрежа, и многих других. Неки процеси су данас комплетно аутоматизовани, са тежњом да се што више њих аутоматизује комплетно.

Аутоматизација процеса је постигнута на разне начине коришћењем механичких, хидрауличких, пнеуматских, електричних и компјутерских решења. Код компликованих система се користи комбинација горе поменутих начина. Предности аутоматике су смањење неопходне радне снаге, мања потрошња енергије и сировина, као и повећање квалитета због веће прецизности.

Постоје два облика контролне петље – отворена и затворена. Код отворене петље је контролер независтан од излаза, док је код затворене петље управљање зависно од излаза. Практичан пример отворене спреге би био бојлер који греје воду на основу хронометра, док би затворена спрега била када би се додао термостат систему, како би бојлер грејао воду до жељене температуре, без обзира колико би му времена требало за то [1].

Слика 1. *Повратна спрега [1]*

ПИД контрола

ПИД дејство, односно пропорционално, интегрално и диференцијално дејство представља врсту контроле која је јако заступљена у индустрији, где је процена да је више од 90% контролера засновано на ПИД управљању. Јако су популарни због просте структуре коју је лако применити у пракси. Предности су што се може користити и ако нам није познат математички модел објекта управљања, и релативно лако је променити параметара након његове инсталације унутар система аутоматског управљања. Данас је његова имплементација искључиво дигитална. Сложенији начини управљања се користе само када ПИД управљање није довољно да испуни задате спецификације.

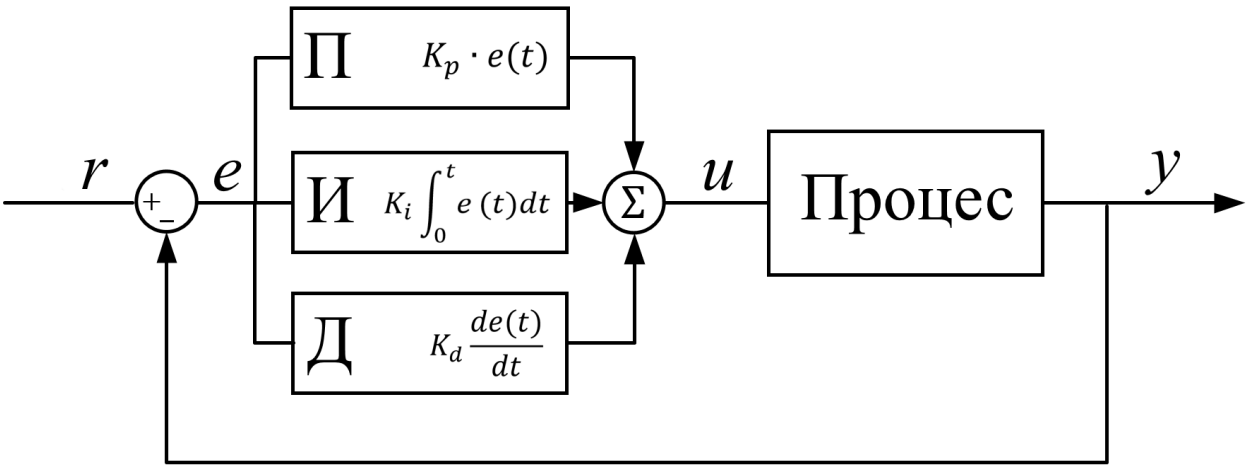
Закон ПИД управљања је добијен искуствено и у идеализованом облику се може представити као

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right),$$

где су параметри K - појачање, T_i - временска константа интегралног дејства и T_d - временска константа диференцијалног дејства. Закон управљања се састоји од три дејства – П, И и Д. П дејство представља пропорционално дејство прилагођава грешку у тренутном времену, и ради једино ако грешка постоји. Повећањем K појачања се повећава брзина одзива и тачност система, али се губи на стабилности.

Интегрално дејство И генерише управљање на основу сигнала грешке прошлог времена. Уколико је сигнал грешке нула, то не значи да је управљање нула. И дејство доприноси елиминацији грешке стационарног стања, и користи се да би повећао тачност рада система у стационарном стању. Његово појачање T_i, уколико је превелико, може довести до дестабилизације система, или смањења релативне стабилности. И дејство повећава инертност и деградира динамичке перформансе одзива система. Са П дејством чини ПИ систем, који, заједно са П регулатором, припада фамилији ПИД регулатора и често се користе у пракси.

Диференцијално дејство Д генерише управљање на основу будућег тренда сигнала грешке која се множи са T_d да би добили Д управљање. Д дејство повећава брзину и резерву стабилности система, али и осетљивост на високофреквенцијске сигнале, као што је мерни шум. Повећањем T_d је могуће у већој мери потиснути осцилације временског одзива система. На слици 2 је дат општи приказ ПИД контроле.



Слика 2. *ПИД контрола [2]*

Континуални закон ПИД алгоритма се чешће изражава функцијом преноса

$$G_c(s) = \frac{U(s)}{E(s)} = K\left(1 + \frac{1}{T_i s} + T_d s\right).$$

Претходно објашњење ПИД управљања представља његову књишку верзију коју није могуће у целости применити у реалном систему, већ само приближно. Разлог томе је што се чист диференцијатор не може тачно физички реализовати, и што повећање шум мерног сигнала повратне спреге. Ови ефекти су неприхватљиви за тачан рад система. Због тога се диференцијално дејство омекшава, тј. функција преноса се апроксимира изразом

$$T_d s \approx \frac{T_d s}{1 + T_d s/N},$$

који има особине диференцијалног компензатора, и који је могуће реализовати. Квалитет апроксимације је добар на ниском фреквенцијама, док је амплитуда одзива на високим фреквенцијама ограничена параметром N. Он се бира у опсегу од 3 до 20, што практично значи да се дозвољава да амплитуда шума буде појачана N пута. Ми нећемо користити Д дејство у нашем систему.

Постоје многе варијације ПИД управљања у зависности од жељених резултата, али ми их нећемо разматрати овде јер неће бити коришћени за реализацију нашег система.

До сада објашњена ПИД контрола се односи да континуалан домен, али пошто ћемо ми наш систем реализовати на процесорима, потребно је ПИД управљање изразити у дигиталном домену. Пошто су ПИД алгоритми у суштини хеуристички закони управљања, лако је могуће добити њихову дигиталну верзију чији се параметри могу подешавати процедурама које су познате у пракси континуалне регулације.

Када континуални сигнал пребацујемо у дигитално, тај поступак вршимо тако што са континуалног сигнала узоркујемо вредности на тачно одређено време, које зависи од природе сигнала. То време се назива периода одабирања. Са тиме на уму могуће је добити дискретан облик класичног ПИД алгоритма

$$u(kT) = k\left(e(kT) + \frac{T}{T_i}\sum_{i=0}^{k} e(iT) + \frac{T_d}{T}\big(e(kT) - r(kT - T)\big)\right),$$

где је T - периода одабирања. Интеграл се апроксимативно налази тако што се грешка множи са периодом одабирања, где добијамо површину правоугаоника, па онда сабирамо са свим осталим правоугаоницима грешака. Ако је периода одабирања довољно мала, добијамо резултат који је јако близу тачном решењу интеграла. Извод грешке се добија апроксимативном релацијом диференцирања уназад, односно где од садашње грешке одузимамо претходну грешку.

Одавде, аналогно континуалним системима, је овако добијен ПИД алгоритам могуће изразити функцијом дискретног преноса

$$G_c(z) = \frac{U(z)}{E(z)} = K\left(1 + \frac{\frac{T}{T_i}}{1 - z^{-1}} + \frac{T_d}{T}(1 - z^{-1})\right).$$

Ова добијена форма дигиталног ПИД алгоритма може даље бити тумачена и преправљана на исте начине као и код континуалне верзије. Пошто их ми нећемо користити, неће бити ни разматрани.

Постоје више начина за налажење појачања за ПИД. У нашем случају појачања су нађена емпиријски [1].

SCADA системи

SCADA (*Supervisory Control And Data Acquisition*) је систем за аквизицију података и надзорно управљање и представља комбинацију система телеметрије и даљинског управљања, односно система за прикупљање података, система за пренос података до централног рачунара и система за приказ и анализу тих података. Највише се користе код система где је потребно прикупљање великог броја података са удаљених локација. *SCADA* системи се углавном користе у отвореној петљи, пре свега као надзор, а врло ретко за управљање.

SCADA систем се састоји из више компонената, било хардверских, било софтверских, које су повезане заједно. Основне компоненте типичног *SCADA* система су:

Мерна и управљачка опрема - то су сензори и претварачи, уређају који мере неку нама потребну вредност, као што су температура, ниво течности, брзина, сила, итд. На основу ових података оператери могу да процене стање система. Претварачи мерне сигнале претварају у погодан облик, рецимо мењају напон или јачину струје сигнала. Помоћу актуатора и командних панела је могуће аутоматизовати процесе, контролисати моторе, отварати и затварати вентиле и друго.

Даљинске станице - да би могли да вршимо надзор, информације са сензора се морају претворити у облик који је компатибилан са *SCADA* системом. Ову функцију врше даљинске станице (*RTU - Remote Terminal Unit*) које представљају робусне индустријске рачунаре који узимају податке са сензора, врше примарну обраду података и локално управљање, као и комуникацију са центром. Ову функцију често врше програмабилни логички контролери - ПЛЦ-ови (*PLC - Programmable Logic Controllers*). Неке од функција даљинских станица су:

1. Узорковање вредности мерења у одређеним интервалима и њихово чување у меморији док их не преузме централна јединица, односно не буду преписане новим подацима.
2. Регистровање асинхроних догађаја као што су промене стања дигиталних величина или прекорачење вредности аналогних величина.
3. Извршавање команди на основу референци добијених од главне јединице.

4. Локална конверзија у инжењерске јединице, односно конверзија бинарних вредности у вредности које представљају измерене физичке величине.
5. Локално процесирање података које може бити усредњавање, дигитално филтрирање, налажење екстрема и друго.
6. Бележење аналогних података код наступа поремећаја ради касније анализе.
7. Логичке функције типичне за рад програмабилних логичких контролера, односно секвенцијално управљање.
8. Имплементацију локалних регулационих петљи, које су обично локални ПИД регулатори.

Комуникациона мрежа - за пренос свих ових података је потребна комуникациона опрема преко које се подаци шаљу са периферије до центра система. Код *SCADA* система је могуће користити разне методе, као што су оптичка влакна, радио сигнале, телефонске линије, а уколико је систем у оквиру једног објекта, може се користити и локална мрежа (*LAN - Local Area Network*). Пренос података се врши преко стандардних протокола, који одређују на који начин су подаци упаковани.

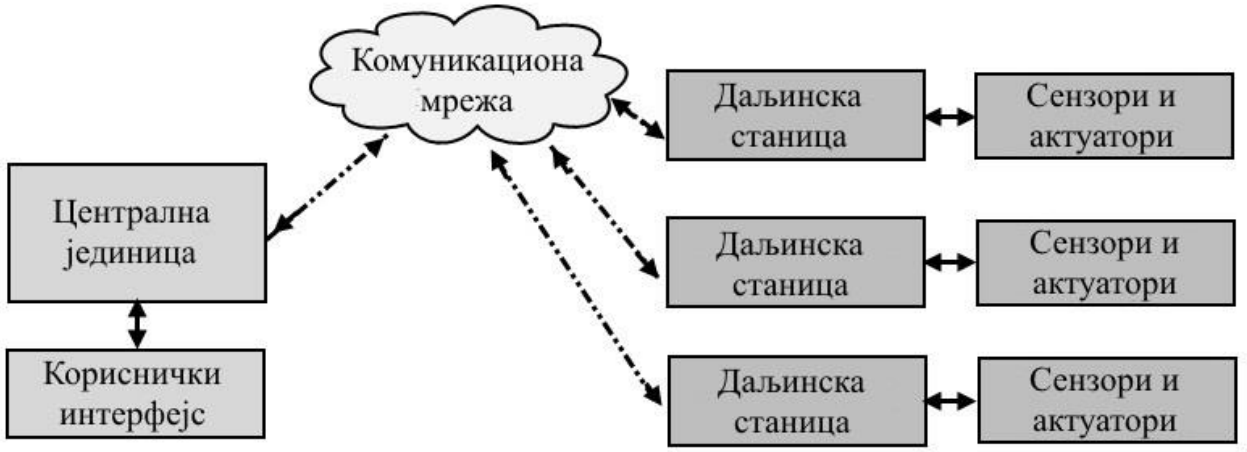
Централна станица (*MTU - Master Terminal Unit*) - је главни процесор *SCADA* система у који стижу сви прикупљени и телеметрисани подаци. Подаци са задњег циклуса се стављају у РАМ (*RAM - Random-acces memory*) меморију одакле могу бити преузети од стране различитих софтверских апликација *SCADA* система, као што су апликације за архивирање података. У неким системима се централна станица дели на више рачунара, тако да се аквизицијом бави примарни процесор који затим одмах вредност шаље преко локалне мреже у други рачунар који комуницира са осталим апликацијама. Централна станица типично има следеће функције:

1. Интеграција свих прикупљених података система.
2. Процесирање и конверзију прикупљених података.
3. Одржава и освежава садржај базе података.
4. Интерфејс са осталим *SCADA* апликацијама које омогућавају приказ информација.
5. Обезбеђује интерфејс другим софтверским апликацијама *SCADA* система према прикупљеним подацима.
6. Обезбеђивање интерфејса другим софтверским апликацијама *SCADA* система за издавање команди и управљањем акција.
7. Вржи функцију надзорног управљања.
8. Обезбеђује функцију креирања примарних краткотрајних историјских архива и обезбеђује информације историјском серверу податке потребне за трајно архивирање.
9. Обезбеђује интерфејс ка осталим деловима информационог система.

Могуће је затворити повратну спрегу управљачког система кроз цео *SCADA* систем, али је најчешће лоша идеја. затварање повратне спреге преко централног система, најчешће због телекомуникационих проблема, смањује поузданост регулатора, а самим тим и целог система. Кад год је могуће требало би све регулационе петље извршавати локално, док се *SCADA* система само шаљу референтне вредности, а ка централној јединици шаљу вредности које се добијају.

Историјски сервери - њихова сврха је архивирање података и у оквиру *SCADA* система их најчешће врше специјализовани сервери у склопу *SCADA* система.

Операторске радне станице - MMI/HMI(*Man Machine Interface/Human Machine Interface* - човек-машина интерфејс) се најчешће налазе налазе поред главног рачунара и омогућавају графички кориснички интерфејс који омогућава приступање архивираним подацима као и задавање управљачких команди систему од стране корисника. На слици 3 је дата је шема *SCADA* система.



Слика 3. *Шема SCADA система [3]*

Први патенти за даљинско командовање и надзор су се појавили крајем 19. века који су углавном коришћени само за надзор и командовање. Током 20-их и 30-их се појављују комерцијални системи са могућношћу преноса више мерења по једном каналу и елементарно командовање. Ово командовање је било јако просто, и углавном се односило на укључи/искључи, док су мерења била индукцијска, односно давала су повратну спрегу да ли је нешто укључено или није. Ови системи су били реализовани применом електромеханичких релеја, који су због своје сложености и непоузданости чинили слање већег броја података непрактичним.

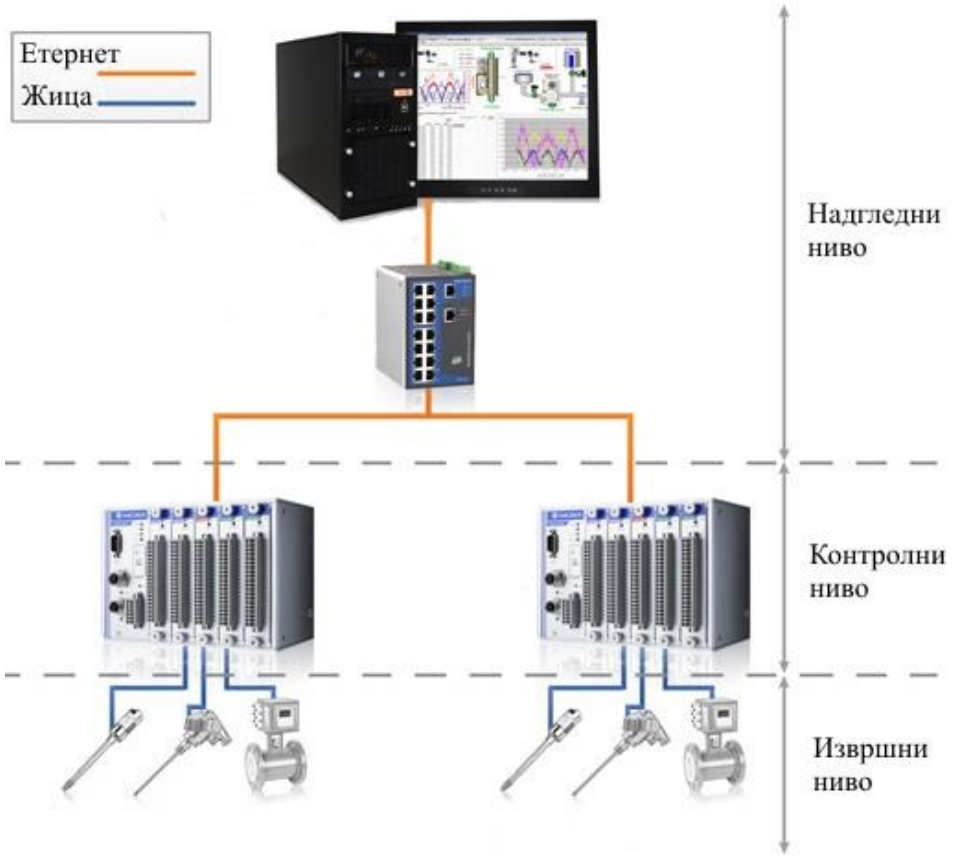
Тек 60-их година, са појавом рачунара и развојем телекомуникација и мерне опреме се појављују први рачунарски системи који су имали све функције система које ми данас називамо *SCADA* системи. Развој рачунара је допринео развоју *SCADA* система и они се могу поделити на три генерације:

- Централизовани *SCADA* системи
- Дистрибуирани *SCADA* системи
- Умрежени *SCADA* системи

Централизовани системи су први СЦАДА системи, где је један централни рачунар управљао целим системом. Локалне мреже нису постојале, а WAN (*Wide Area Network*) су настале због потребе за комуникацијом са спољним станицама, и нису коришћене у друге сврхе. Комуникациони протоколи су били развијани од произвођача RTU-ова и нису били стандардизовани. Њихове способности су биле јако ограничене и није били могући

преносити податке преко мреже, сем комуникације RTU-ова. Везу са централним системом су развијали произвођачи која се састојала од адаптера који се повезивао са процесором. Редундантност система се постизала тако што су коришћена два идентична система који су повезани преко исте магистрале, где помоћни надгледа главни, и преузима улогу ако дође до отказа главног система.

Дистрибуирани системи користе LAN мрежу и развој полупроводничке технологије која омогућава минијатуризацију и расподелу управљачких функција. Помоћу ових технологија је више станица повезано, тако да међусобно размењују податке. Станице су углавном мини рачунари, који имају подељене функције. Неки од њих играју улогу комуникационих процесора који преузимају податке, други имају улогу операторског уређаја, а неки су математички ко-процесори и сервери за базе података. Овим се постиже расподела процесорских снага и добијање веће укупне снаге него да се користи један централни процесор, док у исто време повећава редундантност. Углавном користе LAN протоколе, мада су неки развили и своје протоколе како би имали размену података у реалном времену, али тиме смањили могућност комуникације са уређајима других произвођача. Пошто је систем дистрибуиран, сви рачунари могу да раде у исто време уместо постојања помоћног. Као и претходна генерација *SCADA* система, и ова је ограничена хардвером, софтвером и периферном опремом коју је одредио произвођач.



Слика 4. *Општи приказ SCADA система [4]*

Умрежени *SCADA* системи су слични дистрибуираним системима, са том разликом што су системи отворени за разлику од затворених система претходне генерације. Функције централне јединице су и даље подељене између више рачунара. Побољшање је отворена архитектура која користи отворене протоколе за комуникацију тако да је могуће расподелити функције и кроз WAN мрежу, а не само LAN. Примена отворених стандарда елиминише разне проблеме претходних генерације, тако да је могуће користити комерцијалне софтвере и протоколе, лакше повезивање уређаја различитих произвођача. Такође омогућава произвођачима *SCADA* система да користе водећа решења системских произвођача у развоју основних рачунарских платформи и оперативних система, уместо да развијају своја, тако да могу да се посвете развоју ствари специфично везаних за *SCADA* системе. Најважније достигнуће треће генерације је могућност комуникације између уређаја преко WAN протокола, као што је IP протокол, као и комуникација преко Етернет везе [1].

ПЛЦ контролери

Настанак ПЛЦ контролера је 1968. године када је Џенерал моторс (*General Motors*) расписао конкурс за дизајнирање новог система управљања како би заменили претходни класични релејни систем. Решење које је победило је развила компанија Бедфорд асосиетс (*Bedford Associates*), и било је специјализовани дигитални рачунарски управљачки систем који је био оптимизован за брзу обраду релејних логичких функција, и пројектован тако да може да издржи рад у сложеним индустријским условима. Звао се *Modicon 084*(*MOdular DIgital CONtroler* - модуларни дигитални контролер) и представљао је први уређај које би смо ми данас називали ПЛЦ-овима. Није поседовао класичан оперативни систем, већ су електрични сигнали директно уписивани у меморију, а систем је поред процесора и меморије поседовао и систем за логичке прорачуне, који је омогућавао брзу анализу логичких услова у поређењу са класичним алгоритмима. Систем је могао у потпуности да замени релеје, уз лакше одржавање и већу флексибилност, али су у реалној индустрији они само заменили релеје у логичким функцијама, док су остале примене релеја и даље присутне. Многе фирме настављају развој ПЛЦ-ова, што доводи до пада у цени, и њихове веће примене.

Први систем за комуникацију између ПЛЦ-ова је био *Modicon Modbus*, развијен 1973. ПЛЦ-ови из тог времена су углавном били прости укључи/искључи уређају програмирани лествица логиком (*ladder logic*), тако да су системи били пројектовани на исти начин као релејни системи. Касније почиње рад на стандардизацији протокола за комуникацију и развоја графичког окружења за њихово програмирање. Сада могу да раде и са аналогним величинама, тако да улази и излази нису више само бинарне вредности, већ је било могуће измерити вредности величина на сензорима, а излази бити референтне вредности. Процењује се да се данас у око 80% случајева и даље користе само основне логичке функције са бинарним дигиталним улазима са јако малим коришћењем аналогних излаза и улаза и једноставним програмским решењима на јефтиним програмабилним логичким контролерима.

Задатак ПЛЦ-а је прикупљање десетина или стотина дигиталних и аналогних величина, обраду ових података, и формирање управљачких аналогних или дигиталних

излаза у реалном времену. Ради обезбеђивања оваквих перформанси, архитектура ПЛЦ-ова се разликује од архитектуре стандардних рачунара и мини рачунара. Пошто се користе у индустријском окружењу где постоје значајне електромагнетне сметње, велике разлике у температурама, корозивна испарења, а понекад и изложеност атмосферским условима, избор хардвера који улази у састав ПЛЦ-а је последица потребе да се задовоље ови радни услови. Основни елементи ПЛЦ-ова су:

Управљачка јединица или CPU(*Central Processing Unit* - централна процесорска јединица) - је задужена за извршавање управљачког програма. Сем што она подразумева микропроцесор, често се под њом сматрају и остали елементи неопходни за формирање интелигентног система као што су контролери прекида, контролери директног приступа меморији (*DMA - Direct Memory Access*), итд. Процесор ради на бази скупа системских програма, који заједно представљају оперативни систем ПЛЦ-а. Раније, када перформансе микро процесора нису биле задовољавајуће, уграђиван је и посебни хардверски део који је служио за решавање логичких проблема (*logic solver*), али се данас не користи због развоја микропроцесора.

Слика 5. *Мицубиши Алфа 2 ПЛЦ [5]*

Меморија - због индустријских захтева се у њих не уграђују хард дискови, флопи дискови и други хардвер чији је рад заснован на магнетном пољу, већ се користе флеш меморије, EEPROM(*Electrically Erasable Programmable Read-Only Memory*), батеријска CMOS RAM меморија за трајно смештање података. Ради бољег и бржег учитавања аналогних и дигиталних улаза, они су често мапирани на одређен део меморије који одговара улазу. То значи да се учитана вредност увек налази на једном истом месту. Уписивање вредности не врши процесор, већ специјални хардверски склоп. У програмској меморији се чувају програмске инструкције на основу којих процесор извршава програм. Само програмирање са најчешће врши на другом рачунару коришћењем специјалног софтвера. Програм је могуће дићи на ПЛЦ на неколико начина, често преко серијског порта, а данас све чешће преко Етернета јер онда није потребно да програмер буде у истој просторији.

Часовник реалног времена - је јако битна компонента која обезбеђује прецизне и тачне информације о времену, пошто су оне јако битне за тачан рад самог ПЛЦ-а. Часовник би требало да ради и када је ПЛЦ искључен. Новији уређаји тачност обезбеђују помоћу GPS система [1].

Комуникациони интерфејс - су сви уређаји и протоколи који омогућавају комуникацију између ПЛЦ-ова и остале опреме. Она је најчешће остварена серијским портом или рачунарском мрежом. При комуникациони начини су прављени серијским портовима, али су они били спори. Појава Етернета је значила знатно бржу комуникацију, али је довела до нових проблема, као што је смањен детерминизам. Наиме, серијски портови су спори али време слања података је увек познато. Код Етернета, оно није, и информациони пакети имају различито време стизања. Ово је лоше за рад мреже ПЛЦ-ова, па су развијени специјални индустријски Етернет и протоколи. Индустријски Етернет са хардверске стране је ништа више до робусније Етернет опреме која може да преживи на радном поду фабрике. Специјални протоколи омогућавају бржу и тачнију размену података од обичног Етернета, која може бити у реалном времену. Неки од протокола су *Modbus*, *EtherNet/IP*(*Industrial Protocol*), *PROFINET* и други [1, 6].

1. Развој *SCADA* система

Ми наш *SCADA* систем можемо поделити на два главна дела:

- Хардвер
- Софтвер

Хардверски део се односи на коришћене ПЛЦ-ове, једносмерне и асинхроне моторе, фреквенционе регулаторе, прилагодну електронику, као и каблове неопходне да се цео систем повеже.

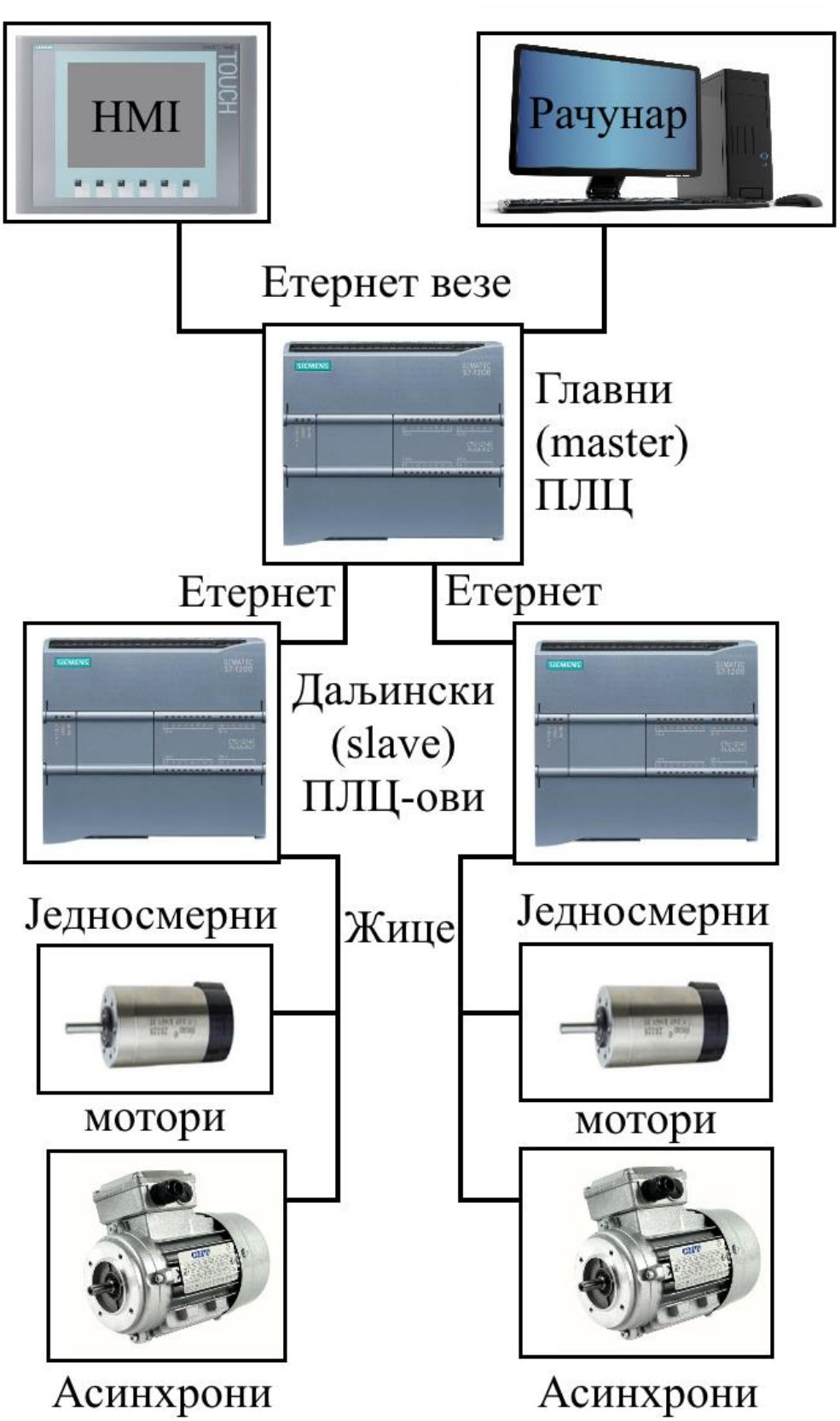
Софтвер се односи на програмирање самих ПЛЦ-ова тако да они комуницирају између себе и врше управљање одговарајућих мотора, као и прављење HMI–а и рачунарске станице.

1.1. Хардвер

Наш систем се састоји од:

- Три Сименс *S7 1200* ПЛЦ-а са *1214C DC/DC/DC* процесором
- Једног *HMI KTP600 Basic PN* уређаја
- Једне *PC* станице
- Два асинхрона мотора *Chiaravalli CHT63B4*
- Два енкодера за асинхроне моторе *Hontko HTR-W-360-2-PP*
- Два фреквентна регулатора марки:
  - *Telemchanique Altivar ATV61H075N4*
  - *Danfoss VLT Hvac Drive FC-102P1K5*
- Два једносмерна мотора *Escap 28D2R-219P* са уграђеним енкодерима резолуције 144
- Прилагодна електроника за једносмерне моторе
- Повезивање свих уређаја
  - *PROFINET* протокола
  - Ожичење ПЛЦ-ова и мотора

У даљем делу рада ће бити објашњени сви ови делови појединачно.



Слика 6. *Наша SCADA*

1.1.1. ПЛЦ Сименс *S7 1200 – 1214C DC/DC/DC*

Користимо три уређаја, где ће нам један бити главна станица (*master*), а два као даљинске станице (*slave*). Сва три уређаја користе процесор *1214C DC/DC/DC* који се користе за рачунање управљања једносмерних и асинхроних мотора, као и за одређивање дозволе рада.

ПЛЦ-ови се напајају са специјалног напајања *PM 1207* који се укључује на градску мрежу и даје 24V на два излаза и две масе. Димензије кућишта су 100x70x75mm.

Сами ПЛЦ-ови су димензија 110x100x75mm, има програмску меморију од 2MB и 50KB за чување података. Његови улази и излази су:

- 14 дигиталних улаза - који имају опсег од 0V до 24V, где је нула од 0V до 5V, а јединица од 15V до 24V. Улази се означавају са ”I” и приступа им се адресом Iбајт.бит, на пример I0.6. Он поседује и шест бројача великих брзина () који се налазе на првих 12 дигиталних улаза. Када су активирани, на пример први ће заузети места од I0.0 до I0.3 и позива се меморијом ID1000. Остали се позивају ID1004, ID1008, ID1012, ID1016 и ID1020. Постоје три начина рада бројача:
    - *Frequency* - броји промену улазног сигнала у одређеном периоду (1, 0,1 или 0,01 секунда) чиме добијамо фреквенцију улазног сигнала која се затим уписује као целобројна вредност у регистар.
    - *Counting* - броји промену улазног стања и уписују их као целе бројеве у регистар, без обзира на време између померања.
    - *Axis of motion* - користи се за бројање импулса код *PTO* (*Pulse Train Output*).
- 10 дигиталних излаза – могу да шаљу сигнал до 24V, у зависности колико напајање је доведено на излазе. Уколико имамо 5V, логичка нула ће бити 0V, а јединица 5V, а ако доведемо 24V, логичка нула ће бити 0V, а јединица 24V. Излази се означавају ”Q” и прате исту логику као и излази, на пример Q0.3. ПЛЦ поседује могућност генерисања два *PTO* или *PWM* (*Pulse Width Modulation*) сигнала на прва четири излаза, где сваки генератор заузима два. Позивају се на вредност QW1000 и QW1002.
- 2 аналогна улаза – имају опсег напајања од 0V до 10V. Ознаке адреса су AI0 и AI1, где се дигитална реч смешта у регистар величине 16 бита.
- 1 аналогни излаз – опсег излаза је 12 бита, исте меморије као улаз, а позива се AQ0 [7-8].

Слика 7. *ПЛЦ Simatic S7 1200 [9]*

1.1.2. *HMI KTP600 Basic PN*

Коришћени дисплеј је један од стандардних у понуди *Simatic S7*, и његова улога је обезбеђивање комуникације између главног уређаја и оператера који надгледа процес. Величина дисплеја је 6 инча и има шест тастера, док је цео екран је осетљив на додир, има 256 боја и резолуцију 320x240 пиксела, а произвођач наводи да је предвиђен за рад са контролерима *S7 1200*. За рад панела је потребно обезбедити напајање од 24V [10].

Слика 8. *KTP600 Basic PN HMI панел [11]*

1.1.3. *PC* станица

За програмирање је коришћен рачунар са процесором *Intel(R) Celeron(R) CPU 430 @1.80GHz, 2GB RAM* меморије, *Intel(R) GMA 3100* интегрисаном графичком картицом, 150GB хард диск меморије, оперативним системом *Microsoft Windows XP Professional*. На њу је инсталиран софтвер за програмирање ПЛЦ-ова *TIA Portal V11*.

1.1.4. Асинхрони мотори *Chiaravalli CHT63B4*

Мотори наизменичне струје се дале на асинхроне и синхроне. Разлика је у томе што се код синхроног мотора угаона брзина мотора једнака кружној фреквенцији напона, док код асинхроних није. Асинхрони мотори се најчешће користе у индустрији од свих електромотора јер имају високу поузданост, висок степен искоришћења, ниске цене и одржавања.

Главни делови мотора су статор и ротор. Статор се састоји од трофазних намотаја који су смештени у жлебове магнетског кола статора. Магнетско коло има цилиндричан облик и израђује се од динамо-лимова, како би се спречили губици. Лимови се пакују тако да формирају цилиндар. Са унутрашње стране лимова се налазе жлебови у које су стављени намотаји који су груписани у три фазе. Ротор има магнетно коло цилиндричног облика које у себи има жлебове на површини дуж главне осовине и можемо их поделити на клизноколитне машине где су трофазни намотаји у жлебовима спојени у звезду на крајевима, а почеци изведени на прстенове, и на кавезне машине које се састоје од шипки направљених од алуминијума или бакра чији су крајеви спојени једним, а почеци другим прстеном. Кућиште је шупљи цилиндар са жлебовима ради бољег хлађења. На њему се налази прикључна кутија у којој се налазе крајеви намотаја за напајање.

Принцип рада асинхроног мотора је у постојању обртног магнетног поља. Како напајамо намотаје статора трофазном наизменичном струјом, индукује се магнетно поље, које се кружно помера, због промена струје. Намотаје можемо везати у звезду или троугао. Када их вежемо у звезду, крајеве везујемо у једну тачко док су остали слободни. У заједничкој тачки, збир свих струја је једнак нули, за коју може бити везано уземљење. Везивањем у троугао спајају се крај једног и почетак другог намотаја, и овакав начин везивања није могуће уземљити.

У оба случаја дефинишемо фазне напоне и струје и линијске напоне и струје. Фазни напони и струје припадају појединим фазама, а линијске су напони и струје у проводницима напајања. Код звездастог везивања фазни напони су напони између фазе и нултог проводника, а линијски напони су разлика између фазних напона, а збир фазних напона је једнак нули.. Линијске и фазне струје су једнаке. Код троугаоне везе Фазни и линијски напони су једнаки, линијске струје су једнаке разлици фазних струја, а према правилу и збиру струја у чвору електричне мреже [12].

Асинхрони мотор који ћемо ми користити је *Chiaravalli CHT63B4* на чије вратило је прикачен редуктор са енкодером. Има кавезни ротор, а статорски намотаји су везани у

звезду и фазно померени за 120°. Може да ради са напајањем из електричне мреже фреквенције 50-60 херца и напона 230/400, тако да исти мотор може да ради на различитим мрежама. Карактеристике су му: Снага 0,18kW , струја 0,7A, момент 1,27Nm , ефикасност 57% и маса 4,5kg. Редуктор има ознаку CHM-30U и преносни однос му је 15 [13].

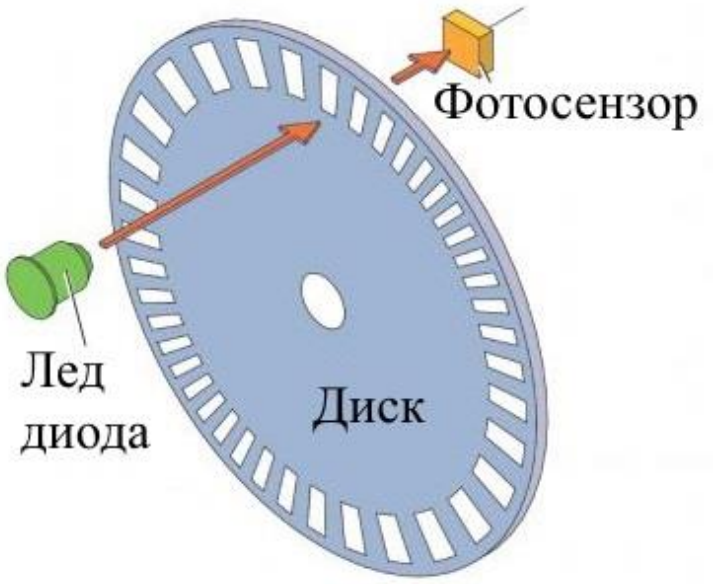
Слика 9. *Chiaravalli CHT63B4 асинхрони мотор [14]*

1.1.5. Енкодери *Honto HTR-W-360-2-PP*

Енкодери су сензори који служе за мерење угла окретања. Могу бити апсолутни и инкрементални. Апсолутни енкодери имају диск који се састоји од више концентричних прстенова при чему сваки следећи гледајући од центра ка ободу има двоструко више подеока Вредност угла се одређује тако што извор светла осветљава део диска, тако да захвати све прстенове. Са друге стране се налазе фотодетектори који детектују да ли светлост пролази, и када пролази пуштају напон. На тај начин добијамо бинарни код на излазу енкодера који представља вредност измереног угла.

Инкрементални енкодери имају прстен да прорезима кроз које пролази светлост, која пада на фоторецептор и даје бинарни код. Пошто он генерише један импулс по прорезу, довољан је један диск са прстеновима. Ми добијамо само сигнал, тако да је потребно додати бројач који броји импулсе, на основу којих налазимо угао. Прво налазимо тачан положај

тако што поделимо 360° са бројем прореза на диску, а затим помножимо са бројем добијених импулса. Да би смо одредили смер обртања потребан је још један прстен са фазно помереним прорезима у односу на први. Тако добијени сигнал касни или жури у односу на први, зависно од смера, па га је могуће одредити. Сви енкодери које ми користимо су инкрементални [15].



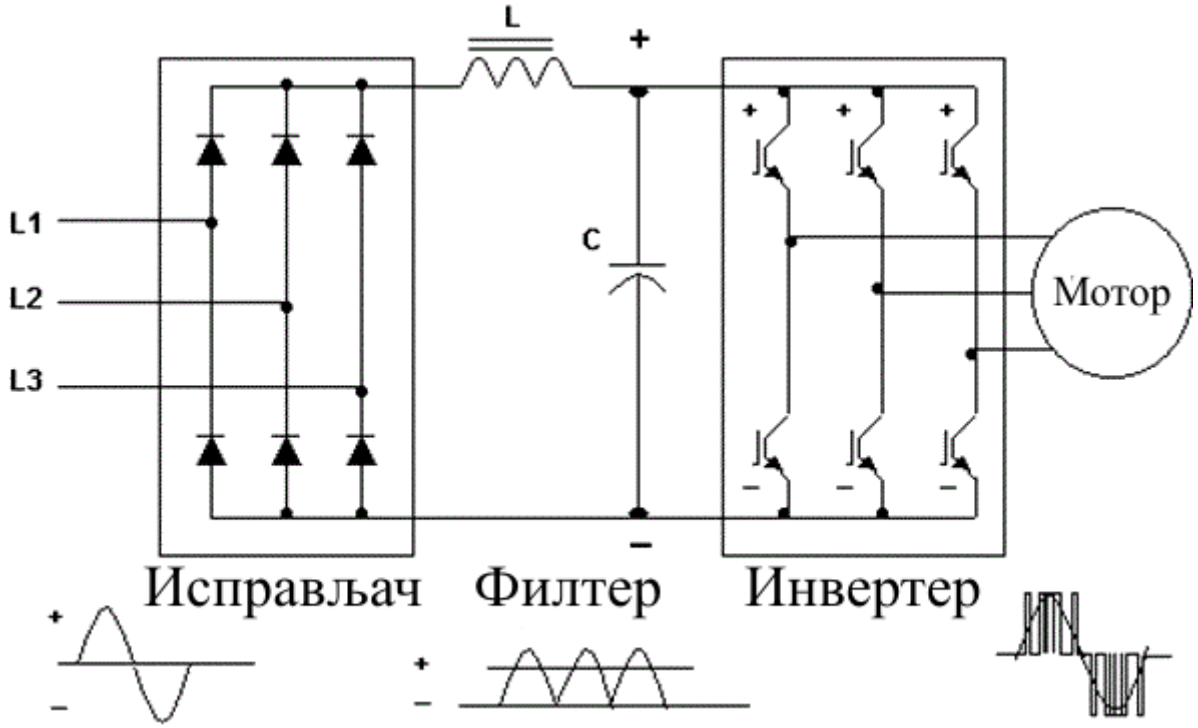
Слика 10. *Рад енкодера [15]*

Енкодер прикачен за асинхрони мотор *Hontko HTR-W-360-2-PP* са диском од 360°, који на излазу даје два правоугаона сигнала који су померени за 90°, и који се напаја са или 5V или 8-26V једносмерне струје. Ми користимо напајање са ПЛЦ-а од 24V. На излазу енкодера имам пет жица, где нам је црвена позитивно напајање, црна негативно напајање, бела жица је фаза А, а зелена фаза Б, док жута жица није у употреби. На ПЛЦ-у си А и Б фазе кратко спојене на два улаза како би користили два брза бројача [16].

Слика 11. *Hontko HTR-W-360-2-PP енкодер [17]*

1.1.6. Фреквентни регулатори

Да би контролисали брзину обртања мотора, морамо да контролишемо фреквенцију статора. Да би то постигли користимо фреквентни регулатор. Начин његовог рада је да се прво он прикачи на трофазну мрежу, и где се онда наизменична струја претвара у једносмерну помоћу исправљача. Он користи шест диода да би исправио смер струје тако што су за сваку фазну жицу везане две диоде. Тако добијену једносмерну струју шаљемо преко једносмерног међу кола, односно филтера, са кондензатором и калемом који исправља синусоиду струје тако да напон има већу или једнаку вредност од амплитуде вредности наизменичног напона. Та строја затим долази до инвертера које садржи шест транзистора, тако да су по два редно везана у грани, а имамо три гране, где свака одговара једној фази трофазне струје. Излаз инвертера је правоугаони ширински модулисан сигнал. Само коло ради тако што контролер отвара или затвара транзисторе, по један у свакој грани, великом брзином, тако да на излазу добијамо трофазни ширински модулисани сигнал, који ће мотор да види као аналогну синусоиду. Фреквенција зависи од брзине отварања и затварања транзистора а амплитуда од дужине пуштања струје. Битно је да оба транзистора у грани не буду укључени у исто време [18-19].



Слика 12. *Шема фреквентног регулатора [18]*

Наш регулатор *Telemchanique Altivar ATV61H075N4* се напаја из градске мреже напоном 380V, док су му границе напона од 323V до 528V, а фреквенција напона од 47,5Hz до 60Hz. На излазу даје струју од 2,3A при 12kHz на 380V трофазне струје или 2,1A при 12kHz на 460V. Излазни напон је једнак или мањи напону напајања [20].

Слика 13. *Telemchanique Altivar ATV61H075N4 [21]*

Други регулатор *Danfoss VLT Hvac Drive FC-102P1K5* има могућност рада ја напону 200V до 240V, од 380V до 480V или 525V до 600V. Фреквенција за сва три нивоа напајања је од 50Hz до 60Hz. Излазни напон је од 0 до 100% улазног, са фреквенцијом од 0 до 590Hz на 1.1 до 90kW. Можемо видети да су, бар за овај рад, регулатори јако слични [22].

Да би нам регулатор радио како треба, потребно је извршити нека подешавања. Прво постављамо да је референтна вредност фреквенције задата у зависности од једносмерног напона на улазу инвертера, тако да 10V задаје 50Hz на намотајима статора. За задавање референтне фреквенције користимо улаз AI2, односно 53, на регулатору, који добија сигнал са аналогног излаза на ПЛЦ-у. Смер окретања мотора се задаје тако што се на улазе LI1/18 за смер напред и LI2/19 за смер уназад доведе бинарни сигнал од 24V, са дигиталних излаза ПЛЦ-а. Напајање од 24V водимо са излаза +24, односни 13, да би напајали октокаплер, који ће бити објашњен касније. Могуће је користити и потенциометар за задавање фреквенције и прекидаче за задавање смера, али онда не постоји аутоматска контрола [20, 22].

Слика 14. *Danfoss VLT Hvac Drive FC-102P1K5 [23]*

1.1.7. Једносмерни мотори *Escap 28D2R-219P*

Једносмерни мотор има стационарне магнете у статору. На статору, односно арматуру, се доводи једносмерна струја, која индукује магнетно поље. Због деловања магнетног поља статора, ротор креће да се обрће. Брзина зависи од јачине магнетног поља, која зависи од напона струје. Ротор је везан на комутатор преко четкица које доводе струје на ротор. Како се индукује струја, један крај бива одбијен а други привучен и ротор прави полукруг. Комутатор сада окреће смер струје како би процес кренуо поново, јер у супротном би ротор стао [24].

Коришћени једносмерни мотор је *Escap 28D2R-219P*, који ради на максималном напону 12V и максималне брзине од 5800 обртаја у минуту. У додатку је могуће наћи ширу документацију мотора. У себе има уграђен енкодер са 144 подеока, односно резолуције од 2.5°, који даје синусни излазни сигнал. Тај сигнал је потребно претворити у коцкасти, тако да ПЛЦ може да га чита [25]. За управљање мотором нам је потребан и *H* мост. Рад обе компоненте ће бити објашњен у следећем поглављу.

Слика 15. *Escap 28D2R-219P мотор [25]*

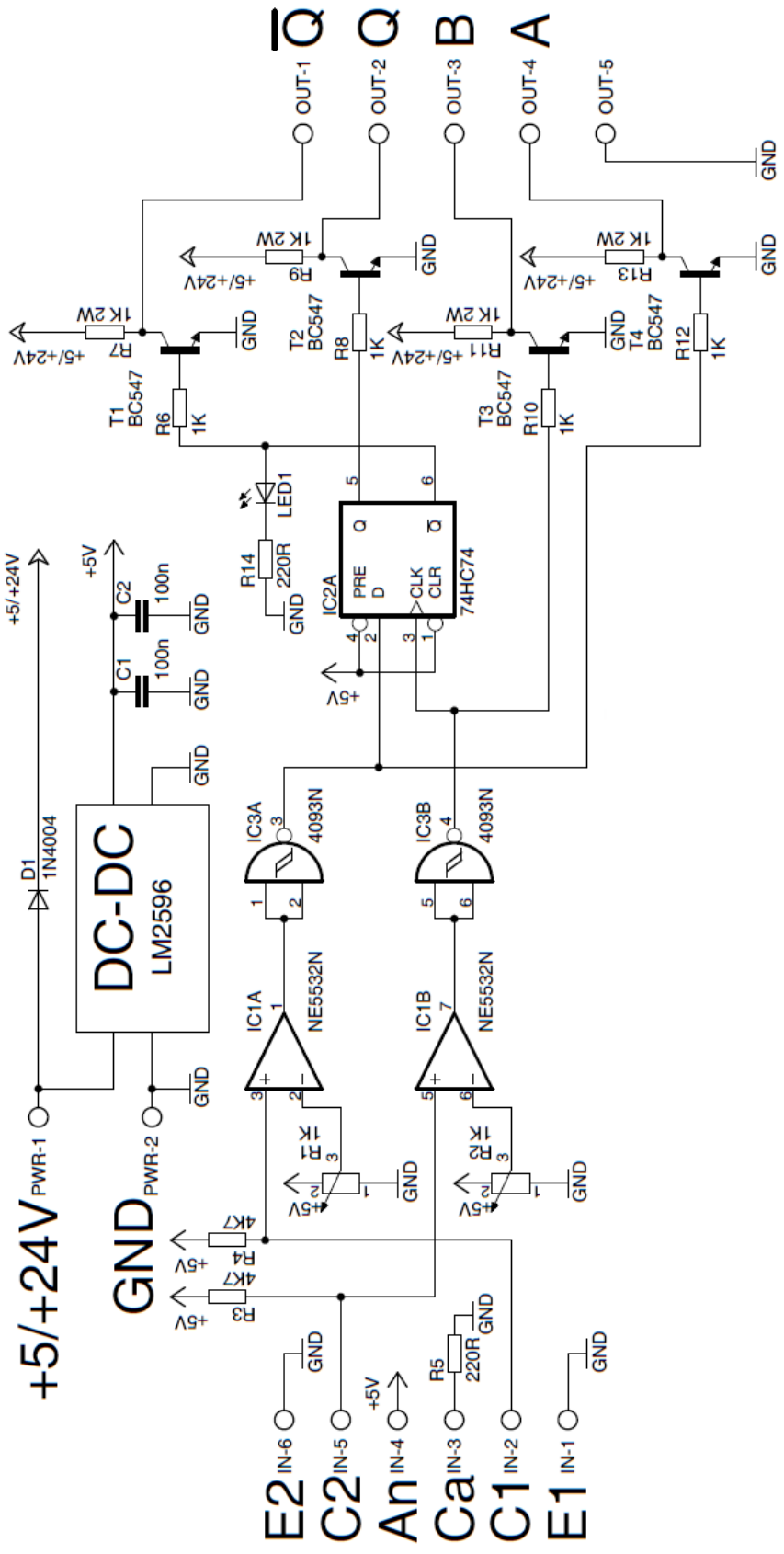
1.1.8. Прилагодна електроника за једносмерне моторе

Потребно је направити прилагодну електронику која ће синусоидни сигнал енкодера једносмерног мотора претворити у квадратиће. Излаз електронике би требало да може да избаци и 24V и 5V, у зависности од напајања, да би иста електроника могла да се користи и за ПЛЦ и за Ардуино.

1.1.8.1 Техничко решење

На слици 15 је дата шема кола прилагодне електронике. На енкодер доводимо напајање од 5V, и отпорник од 220 ома како би заштитили диоду. Излази са енкодера са доводе на потенциометре, а они су у суштини подешавајући отпорници. Сигнал са њих иде на операциони појачавач, који појачава разлику између два улазна сигнала. Ова да дела заједно чине компаратор, који нам синусни сигнал претвара у коцкице. Он у суштини проверава да ли је напон већи или мањи од средње вредности коју добија и избацује максимум или минимум. На шеми се на плус доводи сигнал са енкодера, а на минус се доводи референтна вредност. Подешавањем потенциометра ми доводимо средњу вредност на средину амплитуде синусоиде. Након изласка са компаратора, сигнал водимо на Шмитов компаратор. Он ради на сличан начин као и обичан, са том разликом што има две границе уместо једне. Да би сигнал постао јединица, напон мора да буде већи од горње границе, а да би пао на нулу, мора да буде мањи од доње. Циљ овог поступка је елиминација шумова. Код несавршене синусоиде, можемо имати брзо померање горе-доле малих амплитуда, па би у кратком року имали брзо померање горе-доле на излазу пре него што би се стабилизовало на једном од оложаја, сваки пут када би мењали положај. Шмит компаратор елиминише то понашање. Такође спушта вредност напона доње вредности на нулу.

Сигнали се даље води на *D* флип-флоп. Принцип његовог рада је прост. Вредност доведена на улаз *D* се уписује у њега када на улаз *CLK* (*Clock*). На излаз Q се доводи уписана вредност, а на $\bar{Q}$ обрнута вредност. Вредности које он чува су бинарне. Излаз А је везан за улаз, а Б је везан за *CLK*. Ако је А испред Б због смера, прво се на улаз доводи јединица, а након тога и јединица на *CLK* па се та вредност уписује. Када А падне на нулу, и Б пада. Када поново порасте на један, и Б ће, па ћемо увек имати уписану јединицу ако се

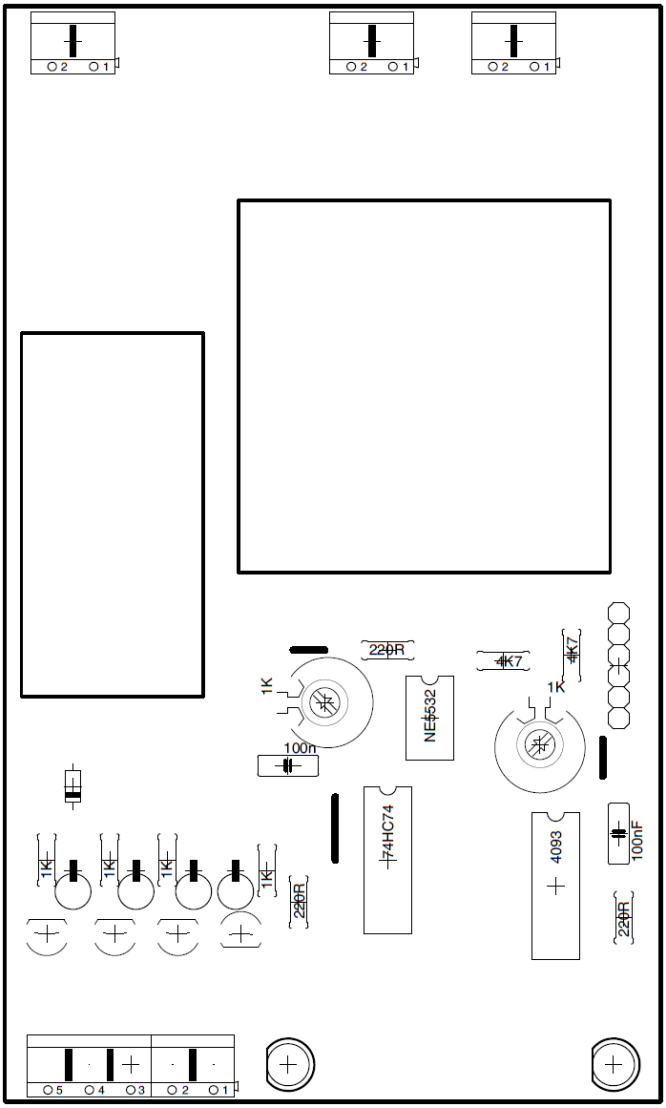


Слика 16. *Шема прилагодне електронике*

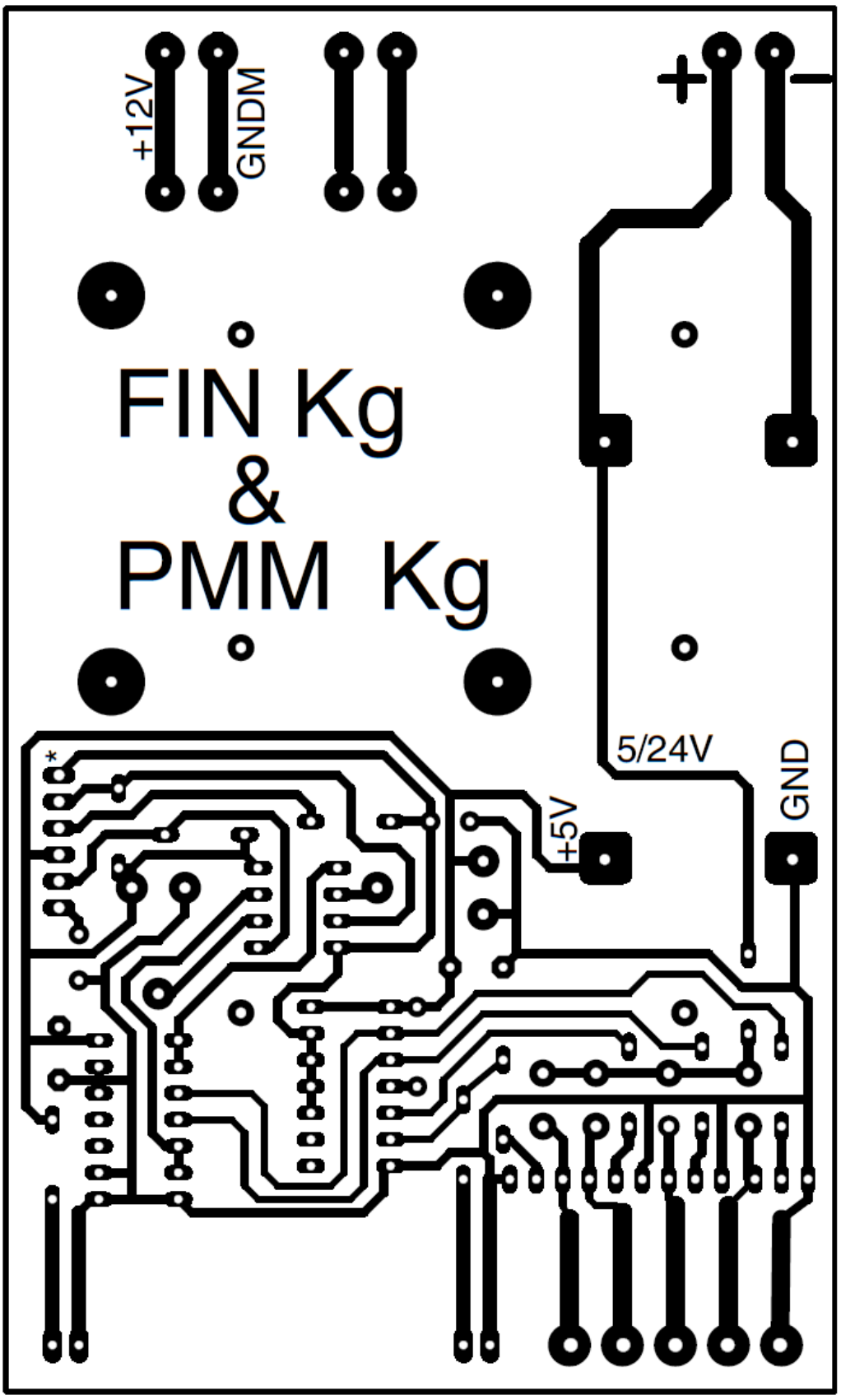
мотор врти у том смеру. У обрнутом смеру ће Б ићи прво, па ће уписивати нулу са А. Када А порасте један неће бити уписано, јер је Б већ на јединици. Сада добијемо да је излаз нула.

Излазе са енкодера и излазе са флип-флопа сада водимо на четири транзистора који повећавају њихов напон у зависности од улазног напона. Ако је напон 5V, крајњи излаз ће бити 5V, а ако је улаз 24V, крајњи ће бити 24V.

Наша цела електроника, сем крајњих излаза ради на 5V, тако да морамо обезбедити потребно напајање без обзира на улазно. Томе служи *DC-DC* конвертер, који увек даје напон који смо му ми одредили без обзира на улаз. Излазни напон одређујемо преко потенциометра који се налази на њему. Отпорници везани у колу служе да заштите остале делове, а два кондензатора на улазу исправљају струју ради бољег рада. На сликама 16 и 17 су дати распореди компонената на површини и матрица за штампање плочице. Матрицу пре штампања треба инвертовати, пошто се штампа са доње стране [25-26].

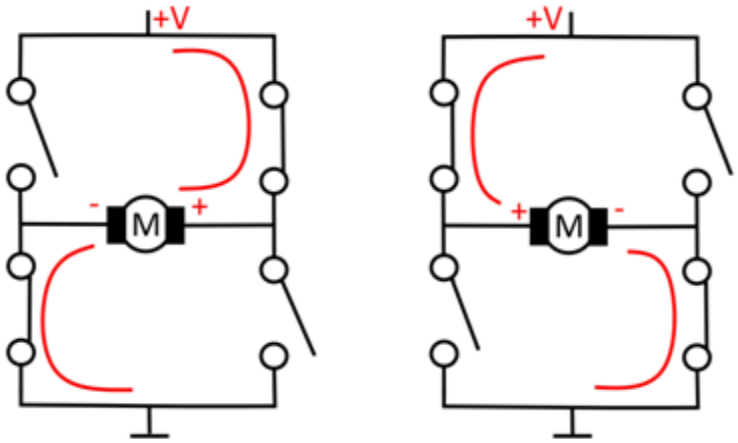


Слика 17. *Распоред компоненти на горњој површини плочице*



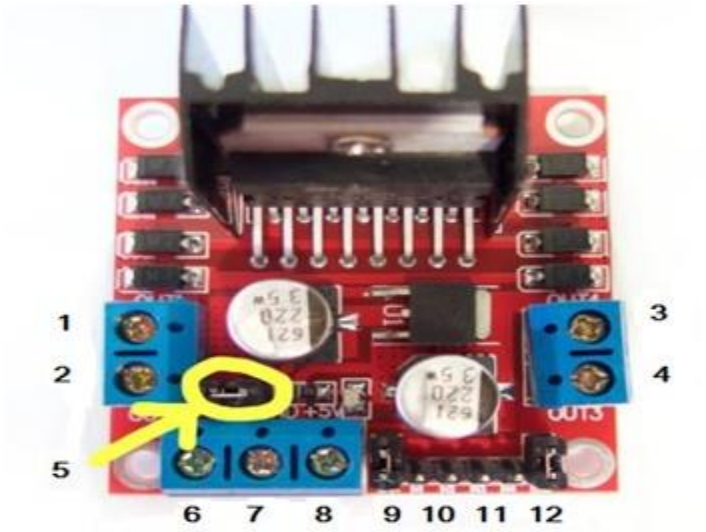
Слика 18. *Матрица за штампање*

За управљање мотора потребан нам је *H* мост уз помоћ којег можемо окретати смер мотора. На слици 18 је дат његов приказ. На једној дијагонали су прекидачи увек затворени, а на другој отворени. Избором дијагонале бирамо смер окретања мотора.



Слика 19. *H мост*

У нашем случају ми смер бирамо довођењем напона на један од пинова под бројем 10, а брзину *PWM*-ом. *H* мост напајамо са 12V преко улаза 6 и 7. Краткоспојником спајамо пинове 5 како би исти напон користила и електроника самог моста, која може да издржи до 12V. Кратим спајањем пинова 9 ми обезбеђујемо да ће на излазе 1 и 2 ка мотору увек бити довођено 12V тако да *PWM* сигналом можемо да управљамо мотором. Када имамо сигнал, на мотор иде максимални напон, када га немамо, иде нула. Користимо одвојено напајање од остатке електронике, јер мотор зна да повуче струју, па би могао да спали цео склоп у том случају [25].



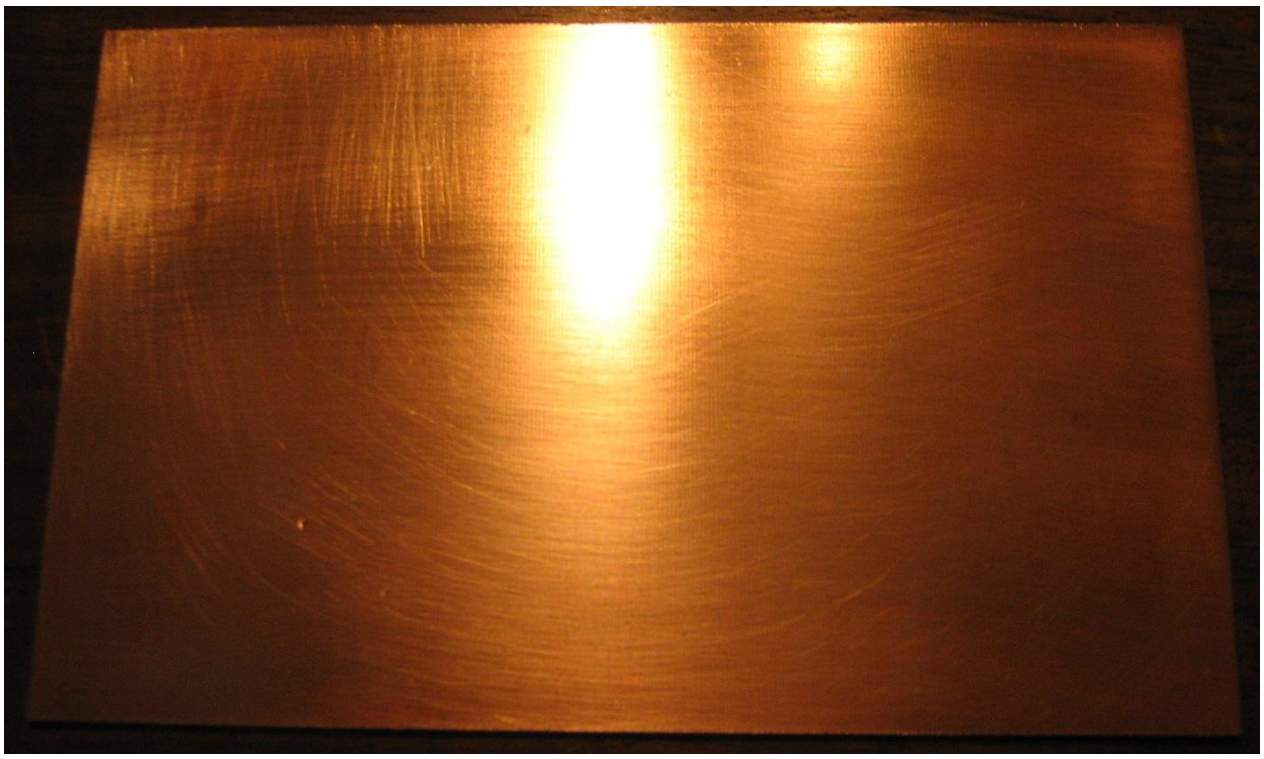
Слика 20. *Плочица са H мостом*

Табела 1. *Излази Н моста*

1,2	Изводи за први мотор
3,4	Изводи за други мотор
5	Селектор извора напајања
6	Извор напајања за мотор
7	Заједничко уземљење
8	Извор напајања за управљачку логику
9	Сигнал за први мотор
10 (2 извода)	Избор смера првог мотора
11 (2 извода)	Избор смера другог мотора
12	Сигнал за други мотор

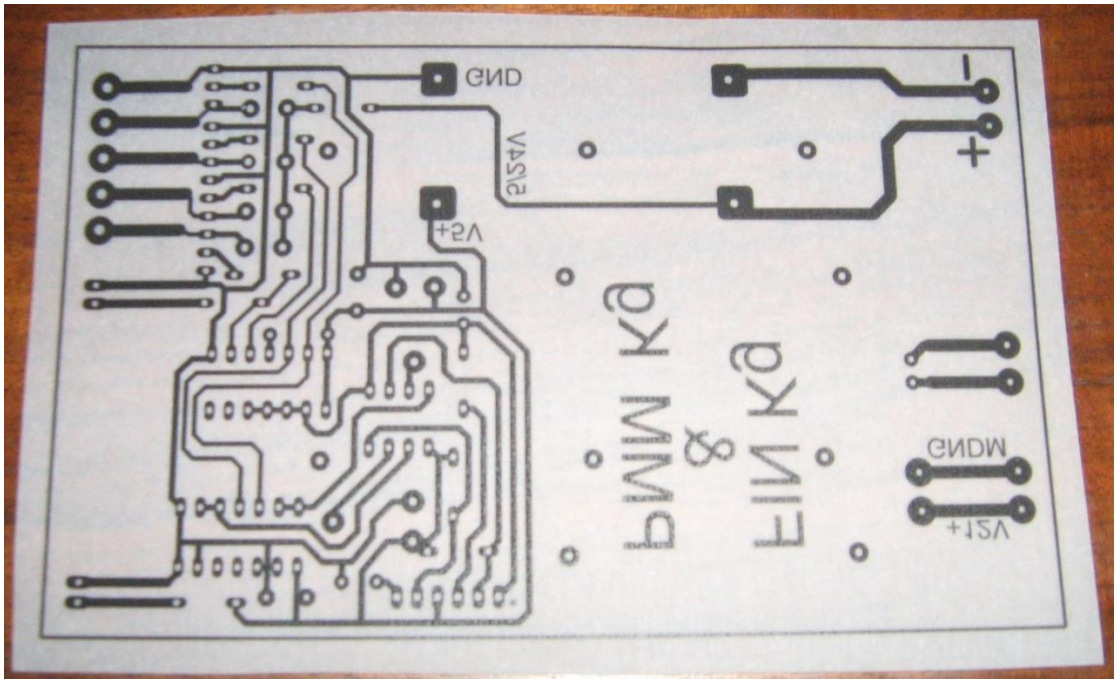
1.1.8.2. Израда електронике

Сада је потребно направити штампану плочицу. Њу правимо фото поступком. Прво сечемо једнострано каширани витропласт *FR4*, дебљине 1,55 милиметара и дебљине бакра 0,33 микрометра, на величину мало већу од наше плочице, која име димензије 13 пут 8 центиметара. Матрица коју ћемо да штампамо је приказана раније, а одрађена је у софтверу *Eagle V5*. Плочицу је потребно добро опрати како би се отклониле нечистоће и масноће на њеној површини које би могле да сметају при изради.



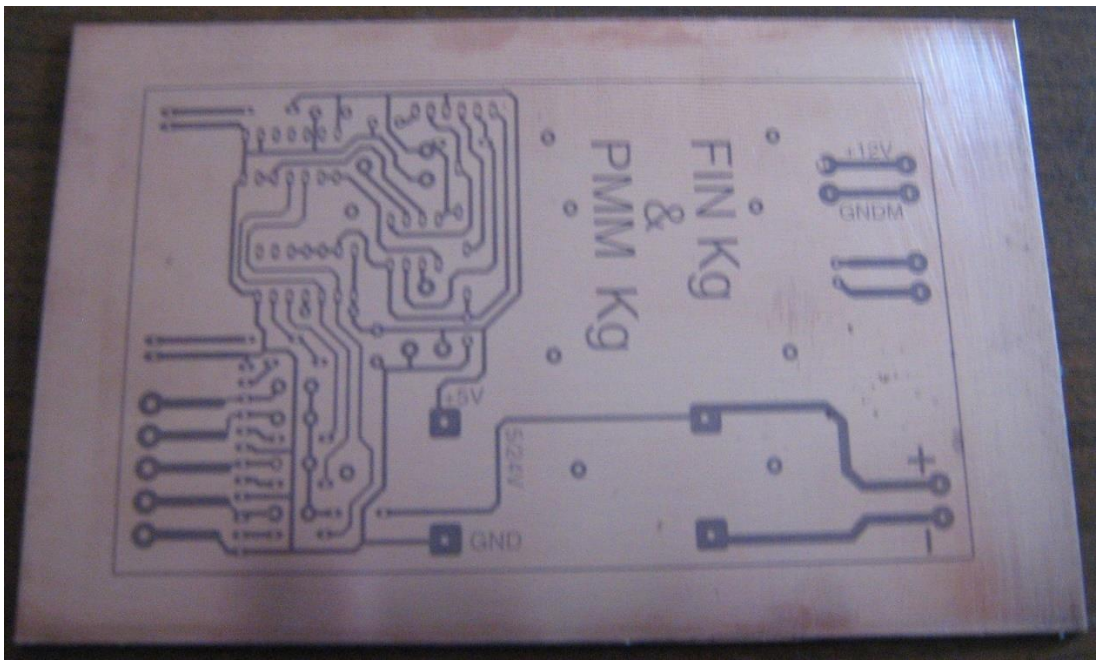
Слика 21. *Очишћена плочица*

Бакарну површину прекривамо слојем фотосензитивног спреја POSITIV 20, и затим печемо плочицу на 70° 20 минута, да би се спреј стегао. Затим на плочицу стављамо нашу матрицу одштампану на паус папиру, и стављамо под ултраљубичасто светло на 20 минута. Овако ослабљујемо спреј на свим местима сем под матрицом.



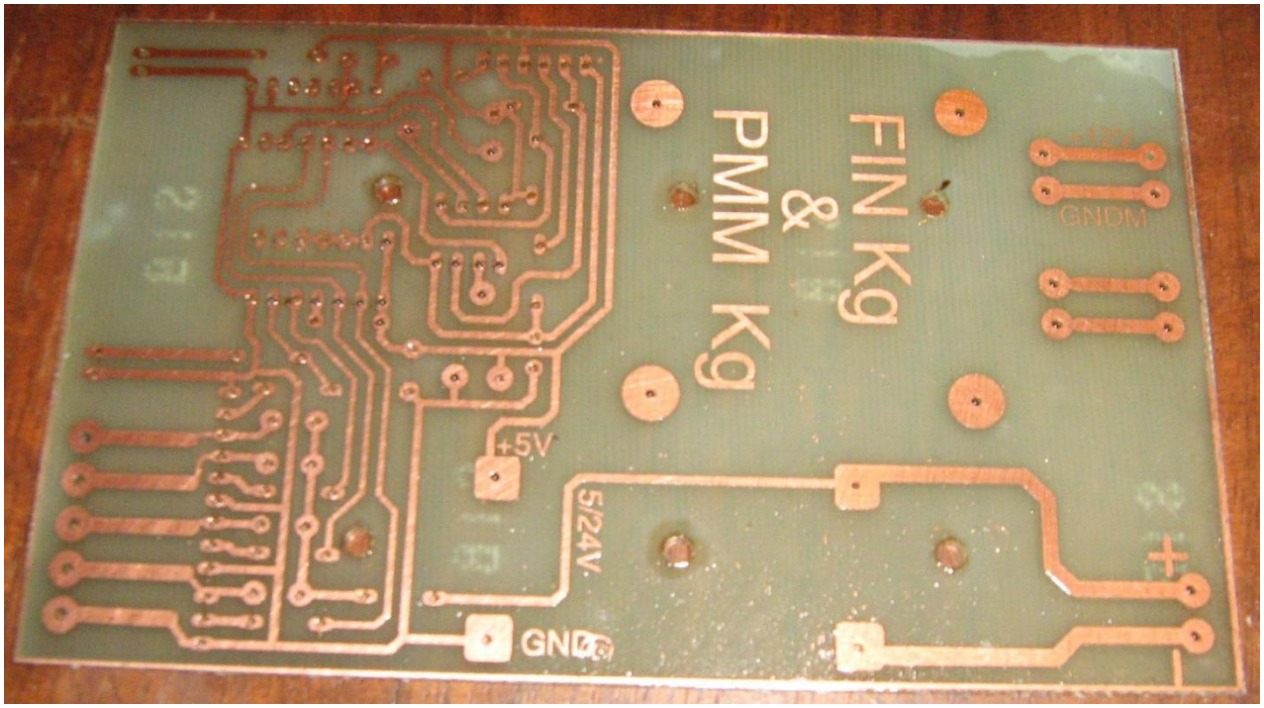
Слика 22. *Одштампана матрица*

Након експозиције, отклањамо ослабљени спреј користећи развијач - раствор од 7 промила натријум-хидроксида и воде.



Слика 23. *Плочица пред нагризање бакра*

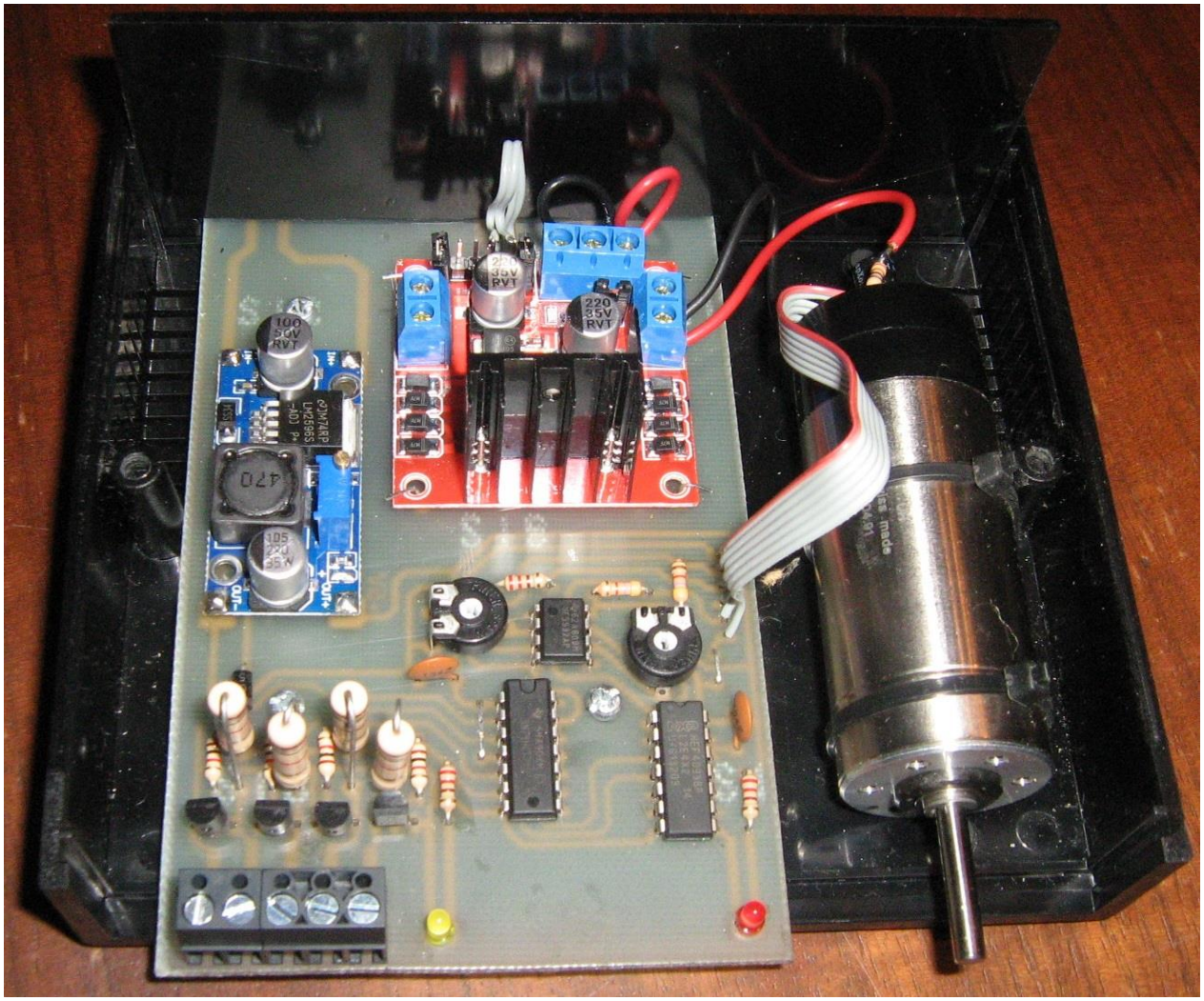
Након скидања боје, плочицу стављамо у киселину која нагриза непокривени бакар и раствара га. Киселина је раствор 15% хидрогенске киселине и 18% хлороводоничне киселине у односу 1 према 3. Након уклањања бакра остаје да се нитро разређивачем отклони остатак боје. Сада бушимо рупе на плочици бургијом од 1,1 милиметар, и прекривамо бакарну страну калофонијумом растворeним у алкохолу који штити од корозије [26].



Слика 24. *Готова штампана плочица*

Сада остаје да се на плочицу напакују компоненте и залеме према већ одрађеном техничком решењу. Онда подешавамо потенциометре тако да добијамо сигнал. Плочице и мотори морају бити упарени, јер постоје мале разлике између енкодера мотора, тако да лепо подешен компаратор код једног мотора не мора да ради и код другог. На крају их све заједно стављамо у једну кутију са излазима како би могли да повежемо мотор са остатком управљачког система.

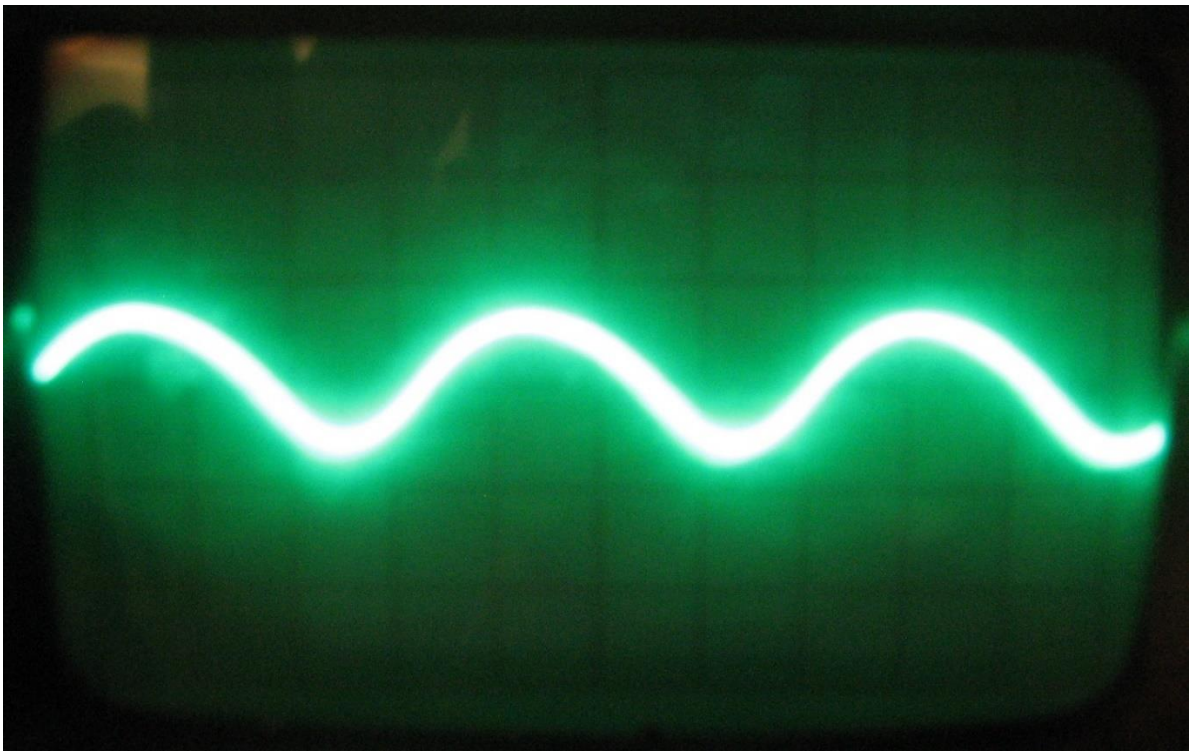
На предњој страни кутије имам пет излаза - маса, излаз А енкодера, излаз Б енкодера, смер окретања мотора, инверзна вредност смера окретања мотора. Позади имамо по три улаза. На први се доводи напајање од 12V за напајање *H* моста и мотора, на други се доводе излази за *PWM* са ПЛЦ-а, и на задња два напајање електронике. На последњи улаз се доводи напајање, које може бити 5V или 24V у зависности од управљачког уређаја. У нашем случају доводимо 24V директно са ПЛЦ-а.



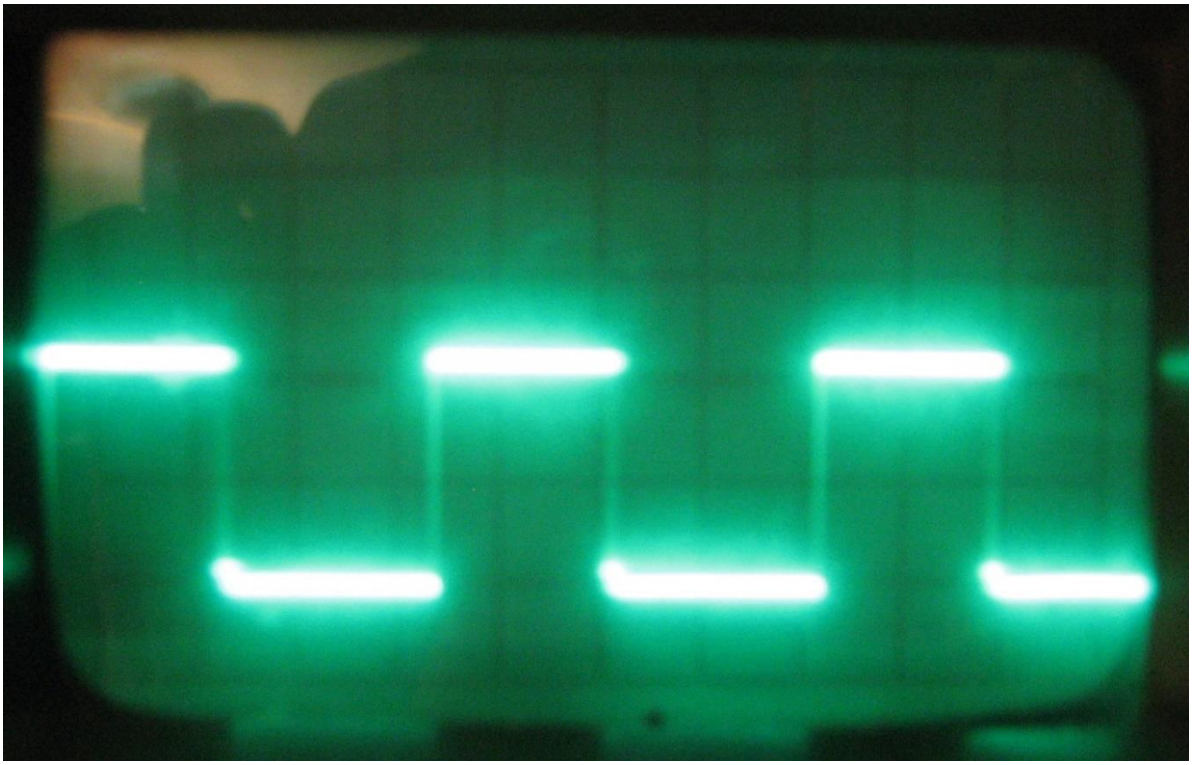
Слика 25. *Електроника спакована у кутију*

Слика 26. *Изглед предњег дела кутије*

На сликама 27 и 28 се може видети излазни сигнал одмах са енкодера једносмерног мотора, и након проласка кроз прилагодну електронику.



Слика 27. *Синусоида са енкодера*



Слика 28. *Сигнал након прилагодне електронике*

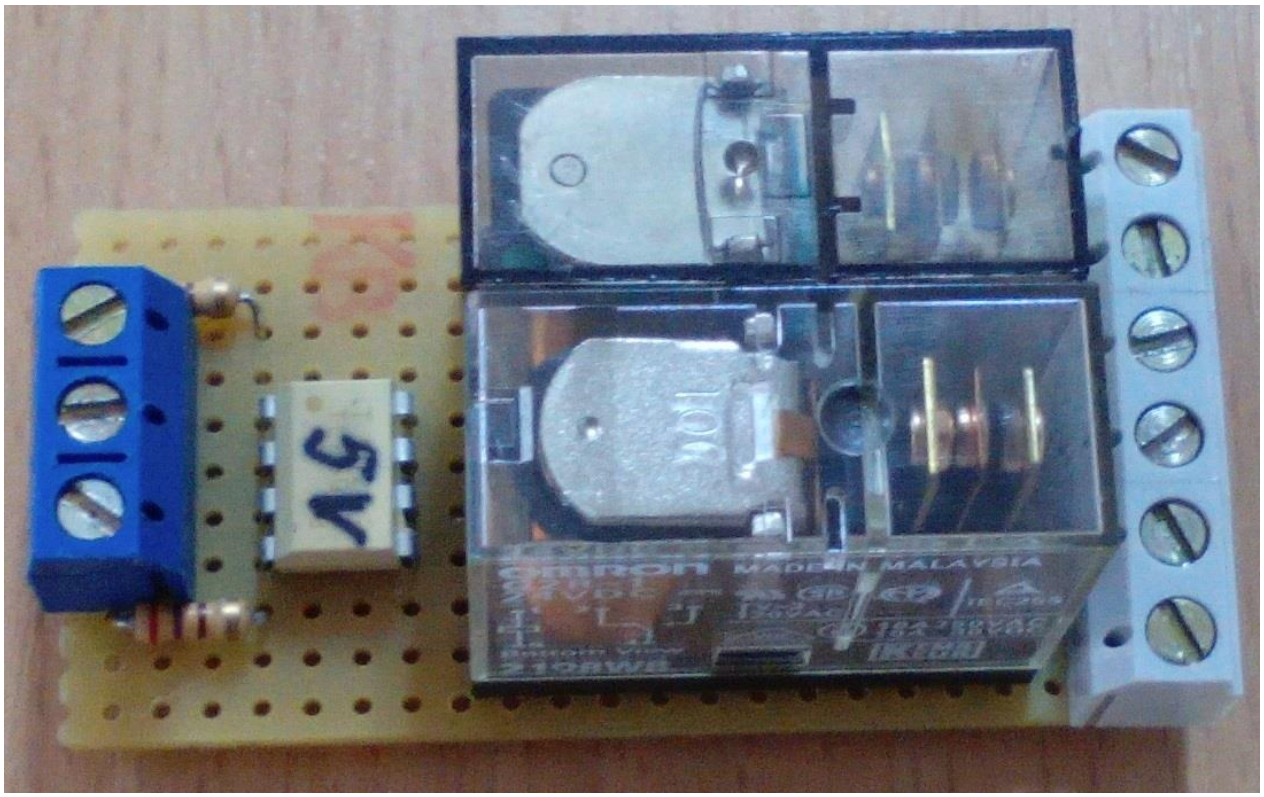
1.1.9. Повезивање свих уређаја

1.1.9.1. *PROFINET*

У лабораторији *C85* су сви рачунари и већина осталог хардвера (као што су ПЛЦ-ови) умрежени коришћењем *RJ45* Етернет каблова и *PROFINET* протокола. Ми ћемо за наш пројекат користити већ постојећу мрежу [6].

1.1.9.2. Ожичење ПЛЦ-ова и мотора

Све масе на даљинским ПЛЦ-овима су повезане. На улазе 0.0 и 0.1 се доводи енкодер са асинхроног мотора, и кратко спаја на улазе 0.2 и 0.3. Са излаза 1.0 и 1.1 се изводе каблови до октокаплера који излазну вредност од 5V подиже на 24V. На њему се налазе и два релеја која служе да одвоје кола ради безбедности. Са октокаплера се воде жице до фреквентног регулатора на улазе који одређују смер окретања мотора, LI1/18 за смер напред и LI2/19 за смер уназад. Повезујемо и масе са октокаплером преко улаза 0V и 20. Октокаплер се напаја са фреквентног регулатора напоном од 24V са излаза +24, односни 13. Са аналогног излаза се воде жице ка фреквентном регулатору и оне носи информацију о фреквенцији на улазе AI2 и 53 респективно, а маса са ПЛЦ-а на улазе COM односно 55. Оба фреквентна регулатора на својим заштитним поклопцима имају упутство шта који улаз/излаз ради, и на основу њих је рађено повезивање.



Слика 29. *Октокаплер са релејима*

На улазе 0.4 и 0.5, аналогно енкодеру за асинхрони мотор, се доводе улази са енкодера једносмерног мотора, и кратко се спајају на улазе 0.6 и 0.7. Уколико се прилагодна електроника напаја са ПЛЦ-а, није потребно спајати и масу са излаза, јер је маса за напајање заједничка. Уколико имамо други извор напајања, потребно је повезати масу излаза електронике и масу улаза ПЛЦ-а. Са излаза 0.0 и 0.2, који одговарају *PWM*-овима, водимо жице на *H* мост. Прва жица иде на леви улаз гледано у мост, а друга на десни. Ако обрнемо жице, управљање брзином ће радити, али у погрешном смеру, а управљање позицијом неће радити како треба. Пошто је наш ПЛЦ способан да одреди смер окретања мотора на основу два канала енкодера, не користимо излазе за смер окретања мотора. Излаз ПЛЦ-а напајамо посебним напајањем од 5V.

На главном ПЛЦ-у није потребно никакво ожичење сем напајања од 24V.

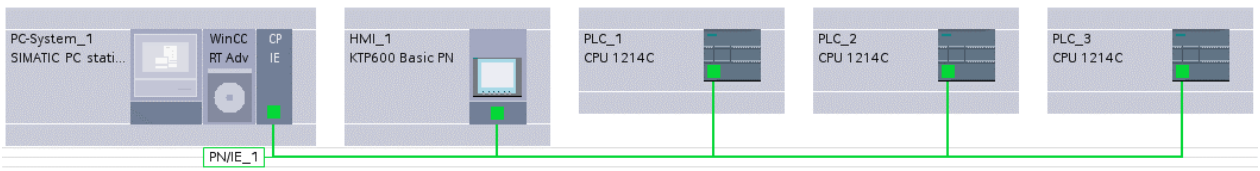
1.2. Софтвер

У овом делу рада ће бити објашњена логика рада програма, како међусобно ПЛЦ-ови размењују податке, како се одређују дозволе рада и како се врши управљање. Објашњавање сваке појединачне команде софтвера неће бити објашњено јер би заузело превише места. За оне који нису упознати са основама програмирања наших ПЛЦ-ова препоручује се да се прво прочита дипломски рад Луковић Видоја под називом "Програмирање и примена ПЛЦ контролера *Siemens SIMATIC S7-1200*". Овде ће бити објашњене сва логичка решења програма једанпут, без обзира колико пута се понављале у програму. Комплетан програм је могуће наћи на *CD*-у који је приложен уз рад. Програмирање се врши на принципу лествичасте логике (*ladde logic*), у сименсовом *TIA Portal*-у, њиховом софтверу за програмирање ПЛЦ-ова [28-30].

Наш *SCADA* програм можемо поделити на четири дела:

- Програм главног ПЛЦ-а
- Програми даљинских ПЛЦ-ова
- Програм *HMI*-а
- Програм *PC* станице

Најпре је потребно повезати наше ПЛЦ-ове у програму, како би знао да су сву у једном систему, и успоставити комуникационе везе између свих компоненти. На слици испод је дат приказ софтверског повезивања.



Слика 30. *Повезане компоненте*

Даље треба успоставити *S7* везе између ПЛЦ-ова како би могли да размењују информације. Ово радимо на исти начин као и успостављање опште везе, само што сад

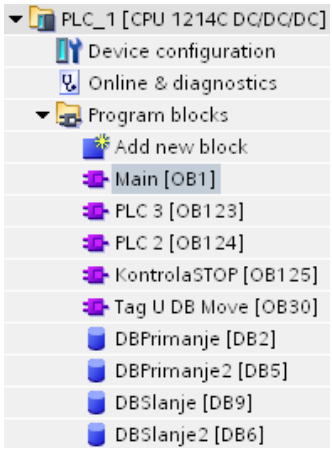
уместо *Network* бирамо *Connections* и из падајућег менија бирамо *S7 Connection*. Умрежени уређаји који могу да подрже ову везу ће бити обојени светло плавом бојом. Ми само треба да зелене тачке спојимо између жељених уређаја. На рају би требало да имамо листу веза која је слична листи на слици испод. Везе за *HMI* прави сам програм када повежемо одговарајуће меморијске јединице између њега и неког ПЛЦ-а докле год се оба уређаја налазе на заједничкој мрежи.

S7_Connection_2	PLC_1	101	101	PLC_2	S7 connection
S7_Connection_1	PLC_1	100	100	PLC_2	S7 connection
S7_Connection_4	PLC_1	103	101	PLC_3	S7 connection
S7_Connection_3	PLC_1	102	100	PLC_3	S7 connection
S7_Connection_2	PLC_2	101	101	PLC_1	S7 connection
S7_Connection_1	PLC_2	100	100	PLC_1	S7 connection
S7_Connection_4	PLC_3	101	103	PLC_1	S7 connection
S7_Connection_3	PLC_3	100	102	PLC_1	S7 connection
HMI_connection	HMI_1			PLC_1	HMI connection
HMI_connection_1	PC-System_1 \ HMI_...			PLC_1	HMI connection

Слика 31. *Везе између уређаја*

1.2.1. Програм главног ПЛЦ-а

Улога главног програма је да узима вредности и дозволе рада са *HMI*-а и да на основу покреће и гаси управљачке ПЛЦ-ове. Такође узима добијене вредности и враћа их на *HMI* како би корисник могао да провери тачност рада система. На слици испод су дати блокови.



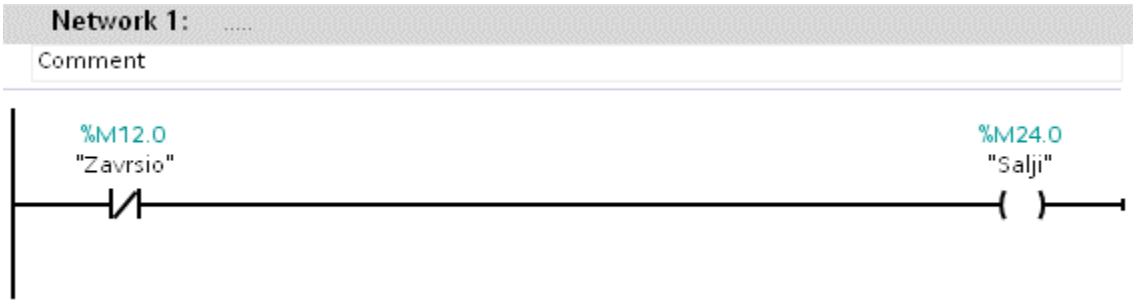
Слика 32. *Блокови програма главног ПЛЦ-а*

1.2.1.1 Блок *"Main"* главног ПЛЦ-а

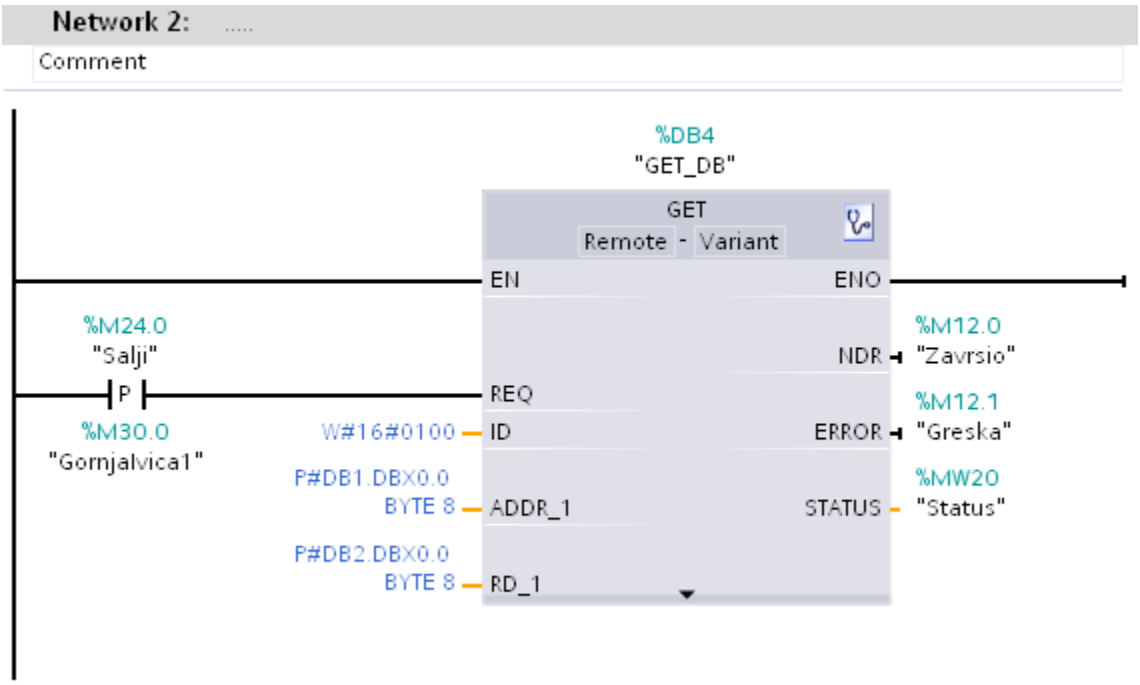
Овај блок служи само ради комуникације главног ПЛЦ-а са управљачким. Он користи функцију *GET* (мреже 2 и 4) како би са *Data Blocka* (*DB*) одговарајућег управљачког ПЛЦ-а узео вредности тренутне позиције или брзине, у зависности од тога које управљање је одабрано. Друго поље по ред покреће процес преноса података и када се заврши, даје

логичку јединицу на првом излазу. Зато ми користимо мрежу 1 да би смо ресетовали слање, и на тај начин вршили константну размену података између уређаја. У четврто поље уносимо адресу локације са које узимамо податке, а у пету адресу локације где их копирамо. Треба напоменути да функција *GET* не копира целу *DB* већ само копира битове који су записани у одговарајућем редоследу. Зато је на нама да се постарамо да оба *DB* буду једнаке дужине и истог редоследа података или ће у супротном доћи до грешке. Када уносимо адресу, треба напоменути да дужина података које преносимо не сме бити дужа од величине *DB*-а, или ће доћи до грешке. Можемо и бирати облик података које шаљемо, али ако их има више различитих, као код нас, најбоље је користити бајт облик. Треће поље носи индентификацију одговарајуће везе коју смо раније успоставили.

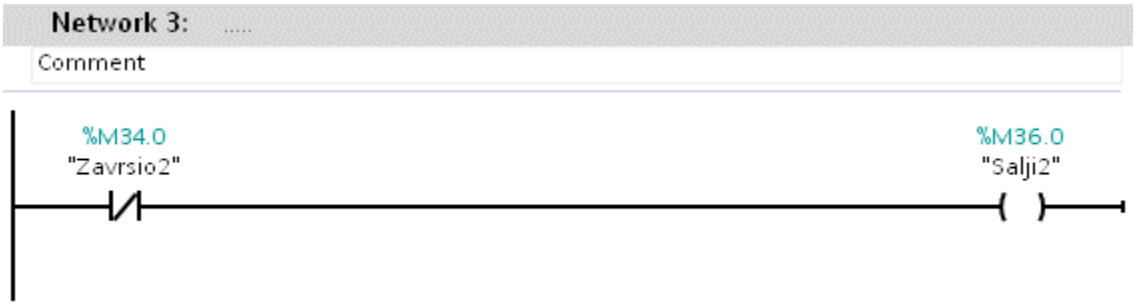
Начин адресирања функције *GET* је следећи: DB1.DBX0.0 BYTE 8 где су - DB1 индекс блока података из кога или у који уписујемо податке. DBX0.0 почетно место копирања података, у овом случају на самом почетку блока, први бајт, први бит. BYTE 8 нам говори да копирамо податке у облику осам бајтова. Можемо копирати и податке у другом облику, као на пример реалне вредности [31-32].



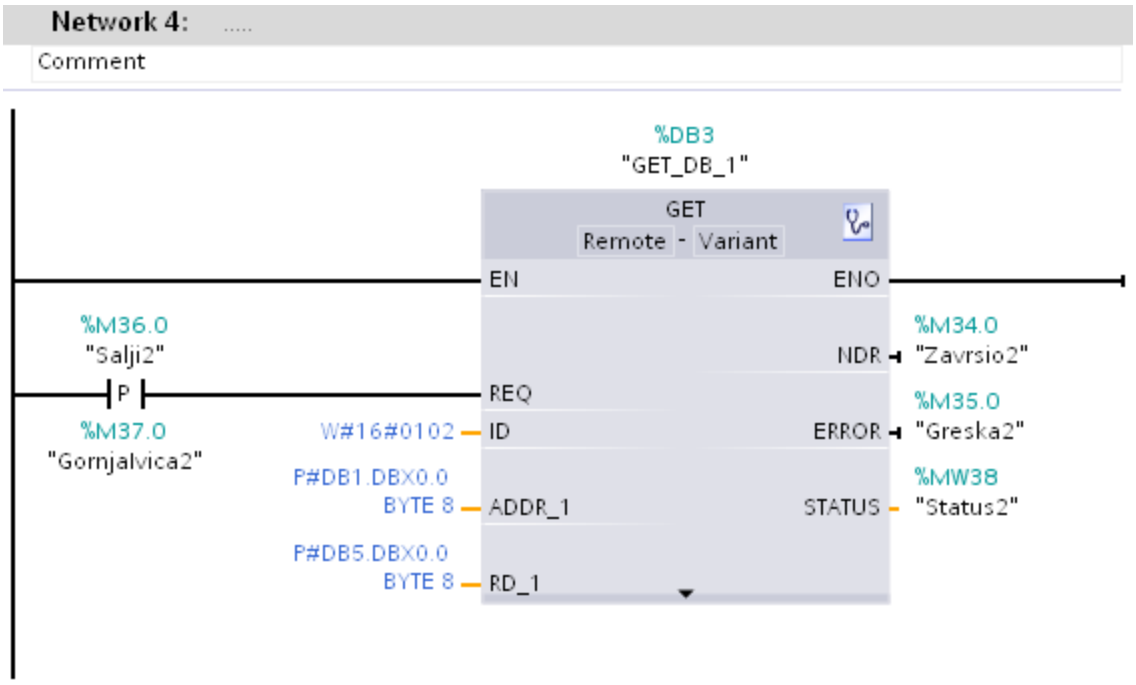
Слика 33. *Блок Main/Мрежа 1*



Слика 34. *Блок Main/Мрежа 2*



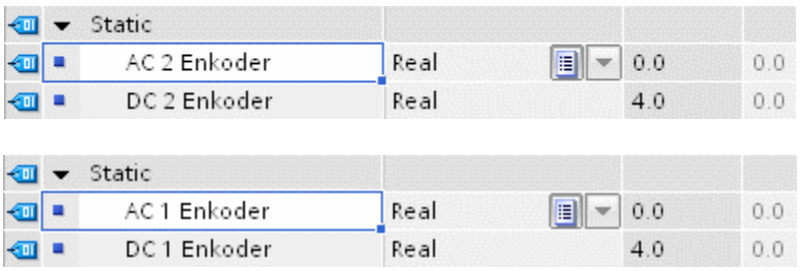
Слика 35. *Блок Main/Мрежа 3*



Слика 36. *Блок Main/Мрежа 4*

1.2.1.2. Блокови података (*Data Blocks* (*DB*))

Имамо четири блока, два у којима се налазе подаци за слање на управљачке ПЛЦ-ове, и два за примање података. Већ је објашњено како се примају подаци. За слање података су одговорни сами управљачки ПЛЦ-ови који користе функцију *GET* да би узели податке. На сликама испод се може видети да су блокови јако слични.



Слика 37-38. *Блокови за примање података*

Static			
AC 2 Zeljena Vrednost	Real	0.0	0.0
DC 2 Zeljena Vrednost	Real	4.0	0.0
AC 2 Brzina Dozvola	Bool	8.0	false
AC 2 Pozicija Dozvola	Bool	8.1	false
AC 2 Napred Dozvola	Bool	8.2	false
AC 2 Nazad Dozvola	Bool	8.3	false
DC 2 Brzina Dozvola	Bool	8.4	false
DC 2 Pozicija Dozvola	Bool	8.5	false
DC 2 Napred Dozvola	Bool	8.6	false
DC 2 Nazad Dozvola	Bool	8.7	false
AC 2 Reset Nula	Bool	9.0	false
DC 2 Reset Nula	Bool	9.1	false
Space 1	Bool	9.2	false
Space 2	Bool	9.3	false
Space 3	Bool	9.4	false
Space 4	Bool	9.5	false
Space 5	Bool	9.6	false
Space 6	Bool	9.7	false

Static			
AC 1 Zeljena Vrednost	Real	0.0	0.0
DC 1 Zeljena Vrednost	Real	4.0	0.0
AC 1 Brzina Dozvola	Bool	8.0	false
AC 1 Pozicija Dozvola	Bool	8.1	false
AC 1 Napred Dozvola	Bool	8.2	false
AC 1 Nazad Dozvola	Bool	8.3	false
DC 1 Brzina Dozvola	Bool	8.4	false
DC 1 Pozicija Dozvola	Bool	8.5	false
DC 1 Napred Dozvola	Bool	8.6	false
DC 1 Nazad Dozvola	Bool	8.7	false
AC 1 Reset Nula	Bool	9.0	false
DC 1 Reset Nula	Bool	9.1	false
Space 1	Bool	9.2	false
Space 2	Bool	9.3	false
Space 3	Bool	9.4	false
Space 4	Bool	9.5	false
Space 5	Bool	9.6	false
Space 6	Bool	9.7	false

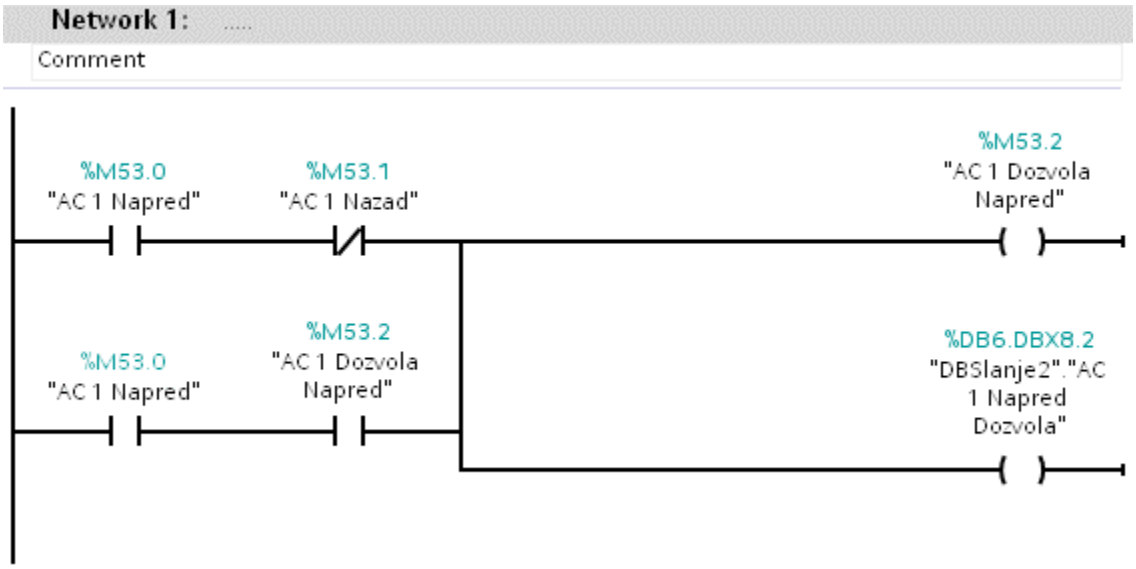
Слика 39-40. *Блокови за слање података*

Постоји још један блок података, а то је *Tagtable*. Његове податке или користи само главни ПЛЦ, или их користи и *HMI*, па се они касније пребацују у одговарајући блок података. Све *Tagtable*-ове је могуће наћи у додатку.

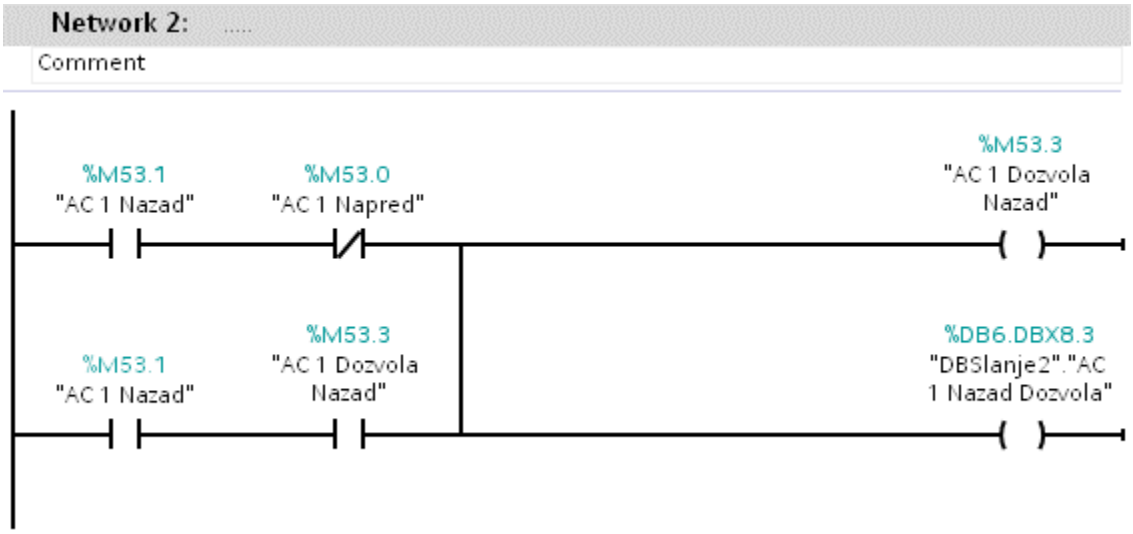
1.2.1.3. Блокови *"PLC 3"* и *"PLC 2"*

Улога ова два блока је да на основу улаза са *HMI*-а одреди дозволе и режиме рада, као и да спречи да се укључе неки други режими. Блокови су јако велики, али се логички понављају.

Ако погледамо мрежу један блока *PLC 3*, видимо да ако притиснемо дугме за избор смера "Напред" на *HMI*-у, и ако у исто време није притиснуто дугме за "Назад", добија се дозвола за тај смер све док се дугме не притисне поново. Исто то се догађа и у мрежи 2, само је случај обрнут. Прва дозвола је за *Tagtable* како би га ПЛЦ користио даље, а друга се уписује у одговарајући блок података да буде послат на управљачки ПЛЦ.

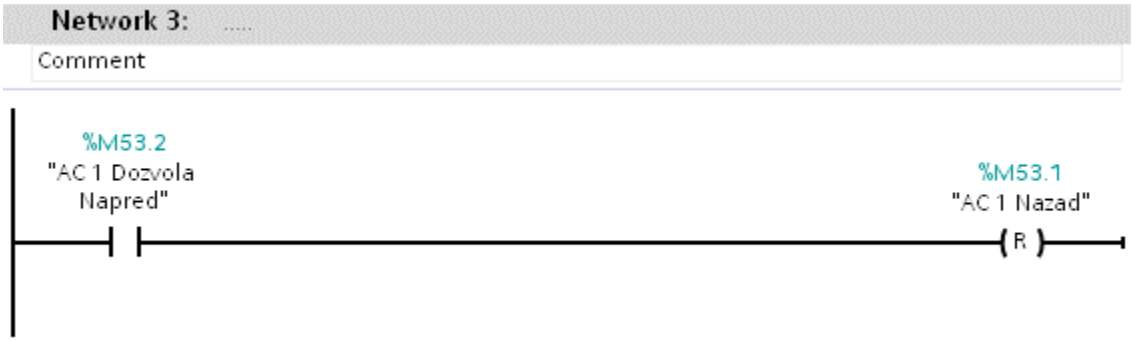


Слика 41. *Блок PLC 3/Мрежа 1*

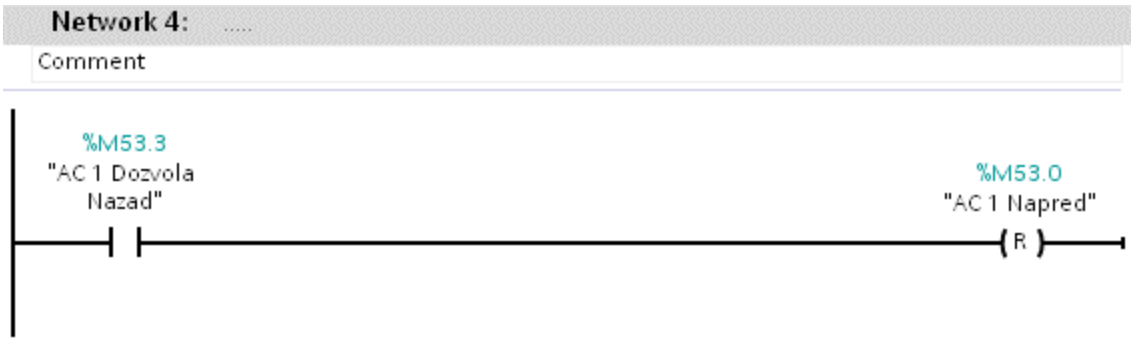


Слика 42. *Блок PLC 3/Мрежа 2*

Претходне две мреже спречавају да се активиране две дозволе у исто време, али бит који мења тастер за дозволе остаје укључен. Тако на пример, ако одаберемо дозволу за "Напред" која ће се онда укључити, па одмах затим "Назад", можемо видети да он није укључен, али ће у програму он бити сетован на јединицу. Ако затим угасимо "Напред", дозвола за "Назад" ће се одмах укључити. Следеће две мреже ресетују бит на нулу докле год је укључена друга дозвола, тако да се ово не би десило.

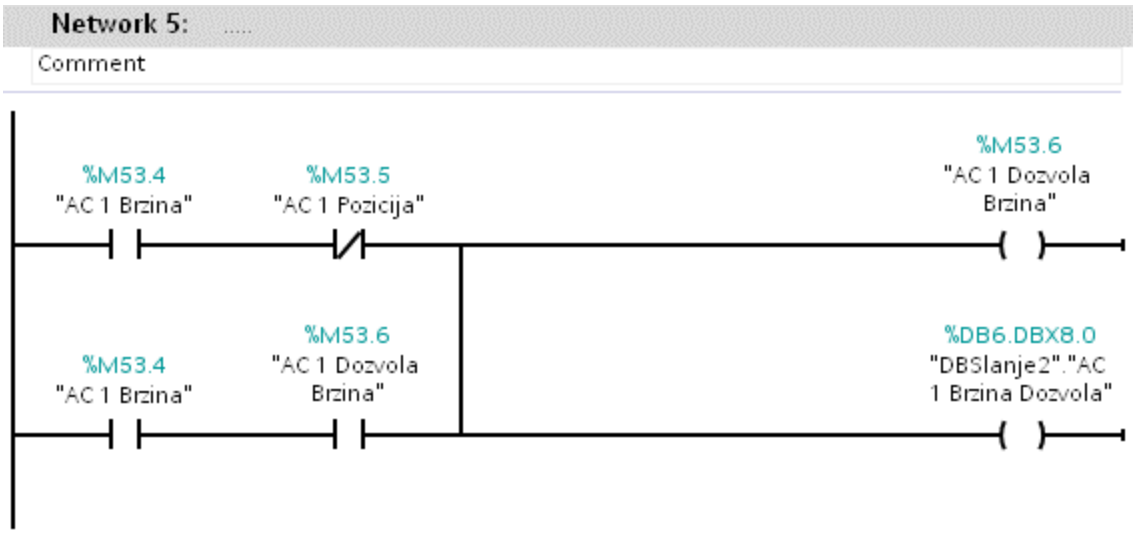


Слика 43. *Блок PLC 3/Мрежа 3*

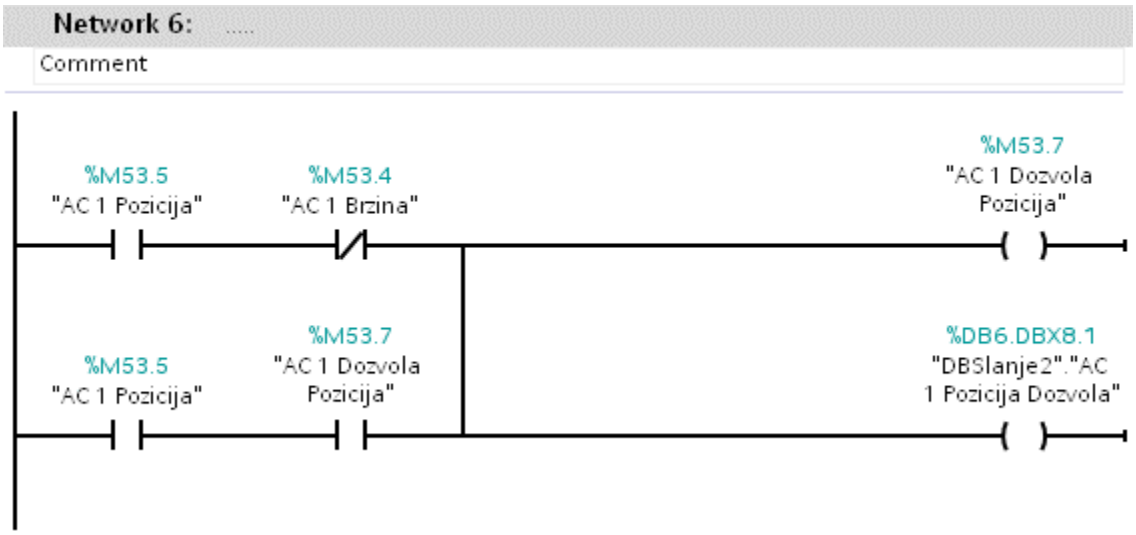


Слика 44. *Блок PLC 3/Мрежа 4*

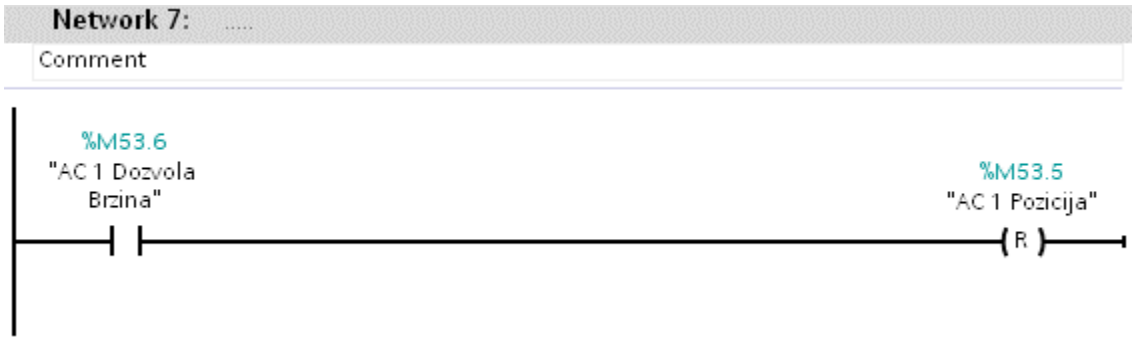
Следеће четири мреже раде на истом принципу као и претходне, само што сада оне одређују дозволу за контролу брзине и позиције мотора. Логика је потпуно иста.



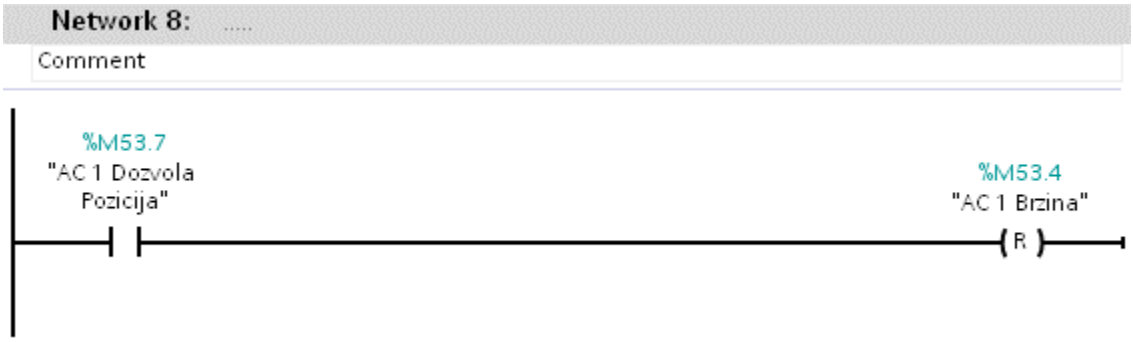
Слика 45. *Блок PLC 3/Мрежа 5*



Слика 46. *Блок PLC 3/Мрежа 6*

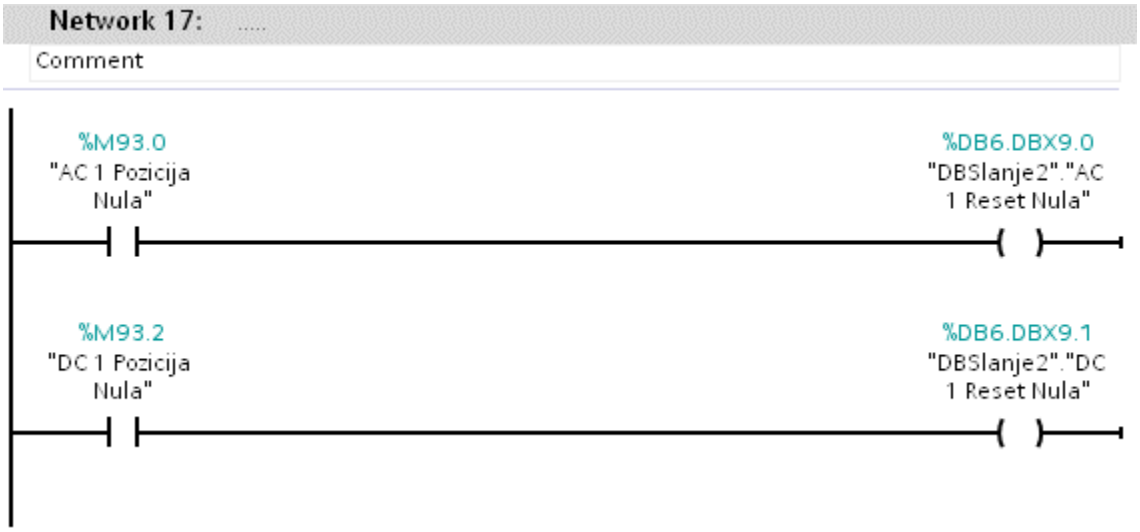


Слика 47. *Блок PLC 3/Мрежа 7*



Слика 48. *Блок PLC 3/Мрежа 8*

Идентично овим мрежама постоји и мрежа за дозволу једносмерног мотора у блоку *PLC 3*. На крају блока имамо мрежу 17, која служи да бит за ресет нуле позиције пренесе у одговарајући блок података. Улога тих дугмића ће бити објашњена код контроле позиције.

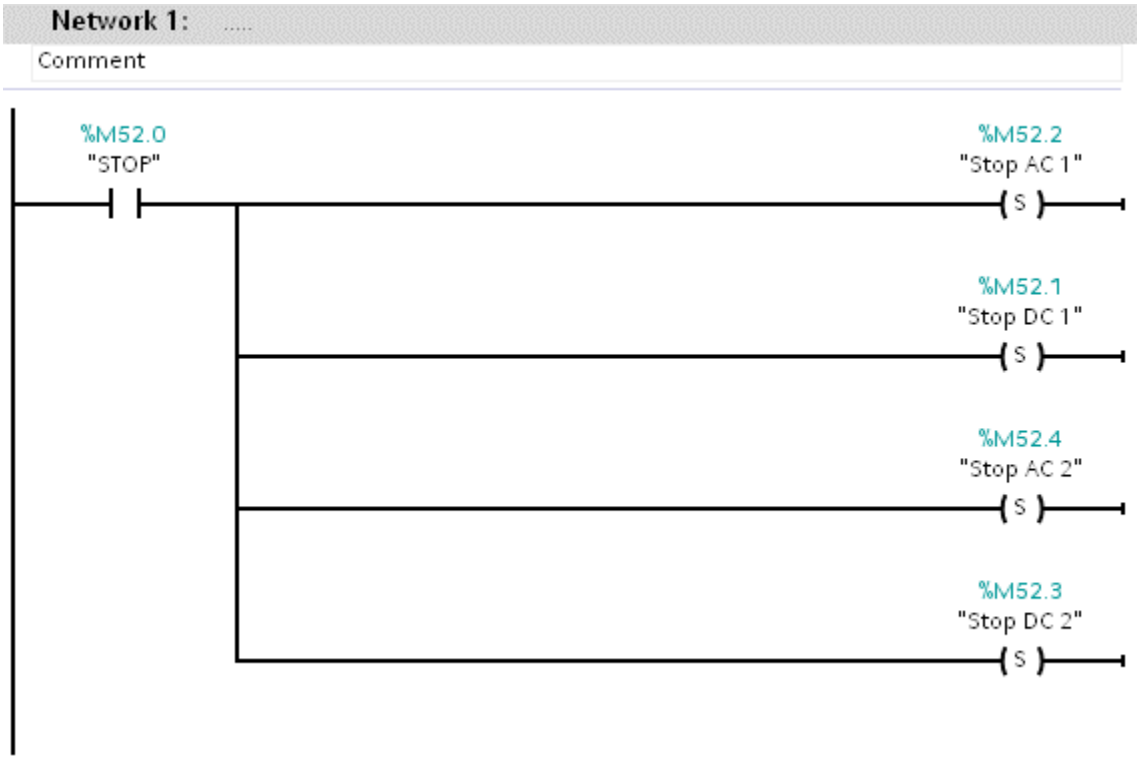


Слика 49. *Блок PLC 3/Мрежа 17*

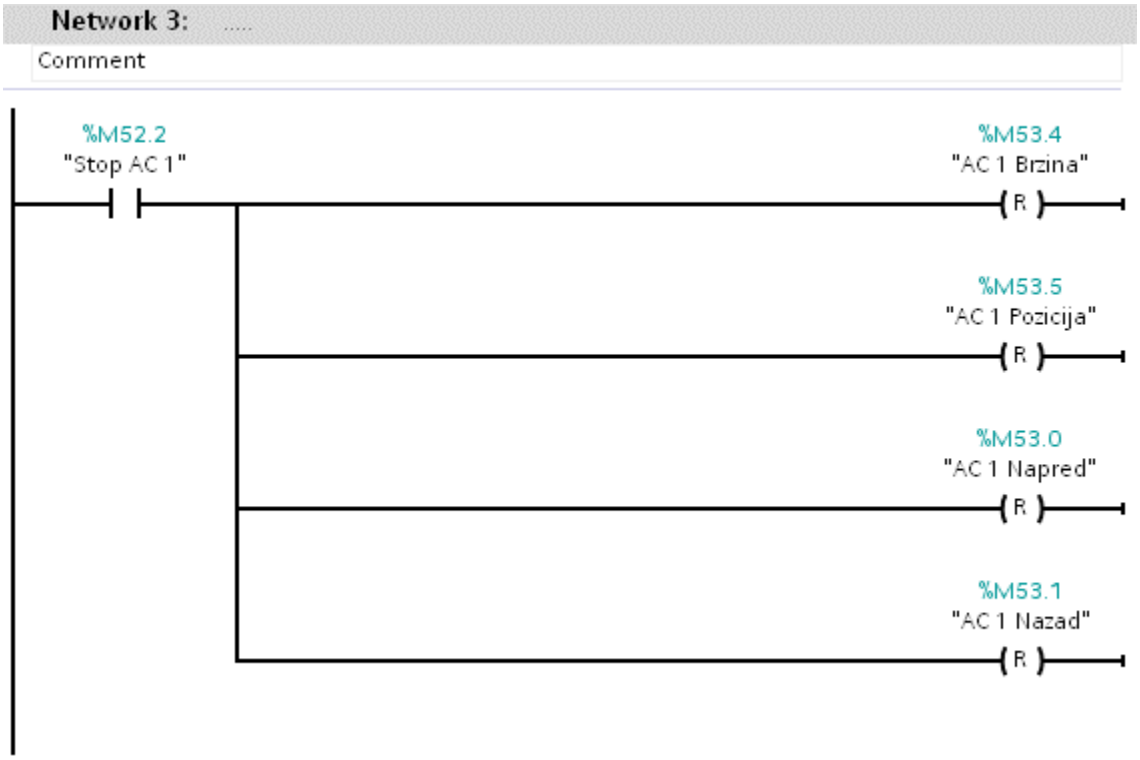
Идентично блоку *PLC 3*, и у блоку *PLC 2* се налазе логички исте мреже, само за друга два мотора.

1.2.1.4. Блок "*КонтролаSTOP*"

Овај блок регулише дозволе рада приликом притиска на неко од дугмића за заустављање. Једино што они раде јесте да ресетују битове дугмића на нулу без обзира на њихову претходну вредност. На слици испод можемо видети мрежу 1, где притиском на главно дугме за заустављање ресетујемо појединачне дугмиће заустављања, а на мрежи 3, ресетовање дозвола приликом притиска на појединачно дугме. Цео блок се састоји од логички истих мрежа где одговарајуће дугме за заустављање ресетује одговарајуће дозволе кретања рада мотора. Дугмиће за заустављање сетујемо да би их укључили, за разлику од осталих дугмића које желимо да искључимо.

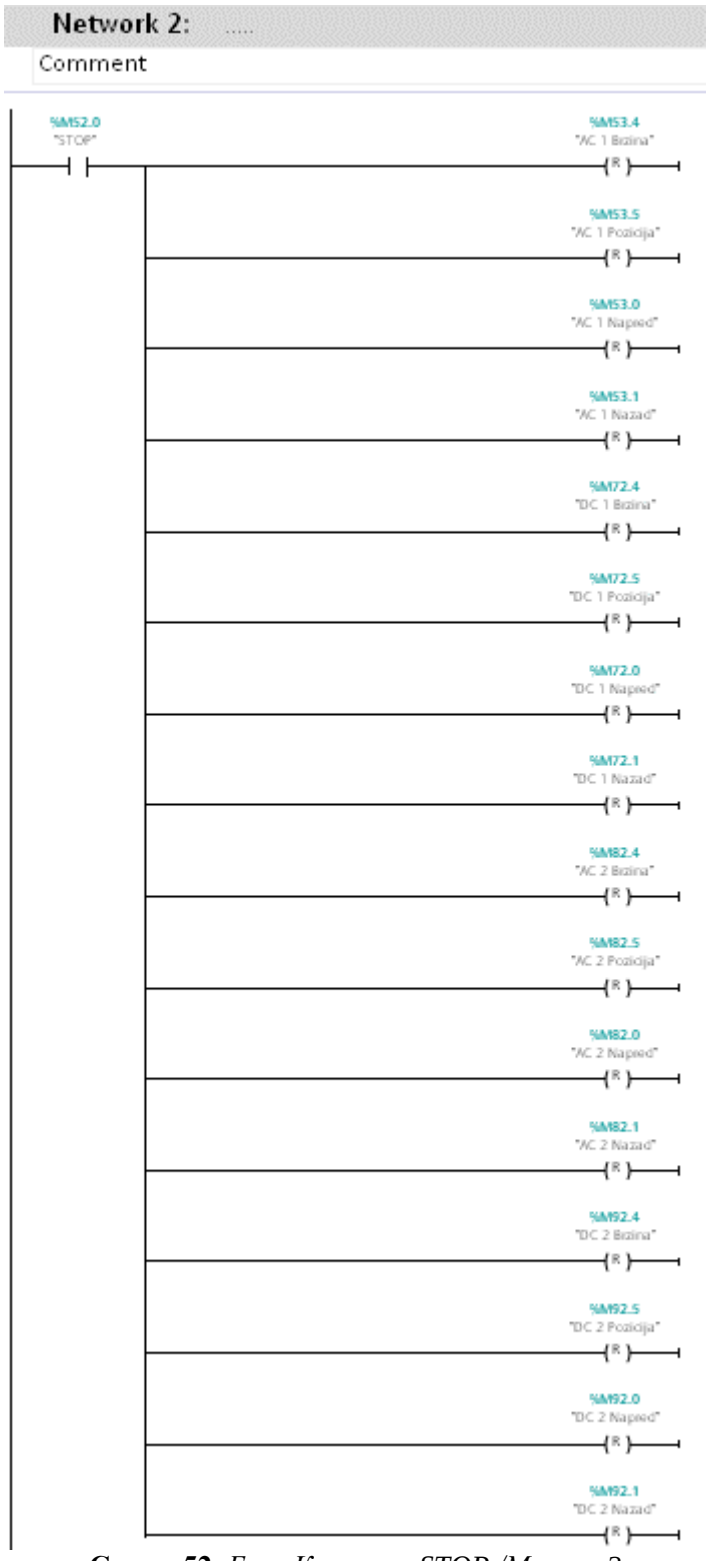


Слика 50. *Блок КонтролаSTOP /Мрежа 1*



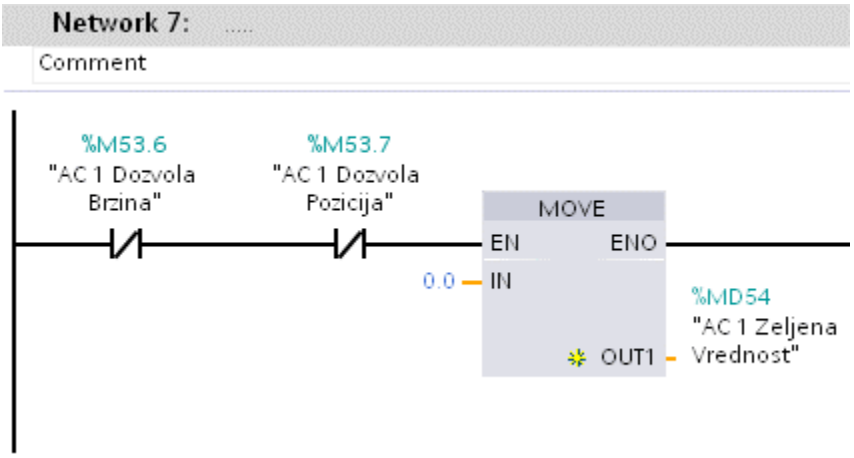
Слика 51. *Блок КонтролаSTOP /Мрежа 3*

И глави и појединачни дугмићи за заустављање ресетују остале дугмиће, у мрежи 2. Ово није неопходно, али се ради због бржег одзива система, него када имамо да главно дугме пали појединачне, па да они онда гасе остале.



Слика 52. *Блок КонтролаSTOP /Мрежа 2*

Мрежа 7 служи, да у случају да није одабран ни један режим управљања, гура вредност нула у жељену вредност. До овога долази и код притиска дугмића за заустављање, и код простог гашења оба дугмета за дозволу без гашења целог управљачког система. Постоје логички још 3 идентичне мреже за остале моторе.



Слика 53. *Блок КонтролаSTOP /Мрежа 7*

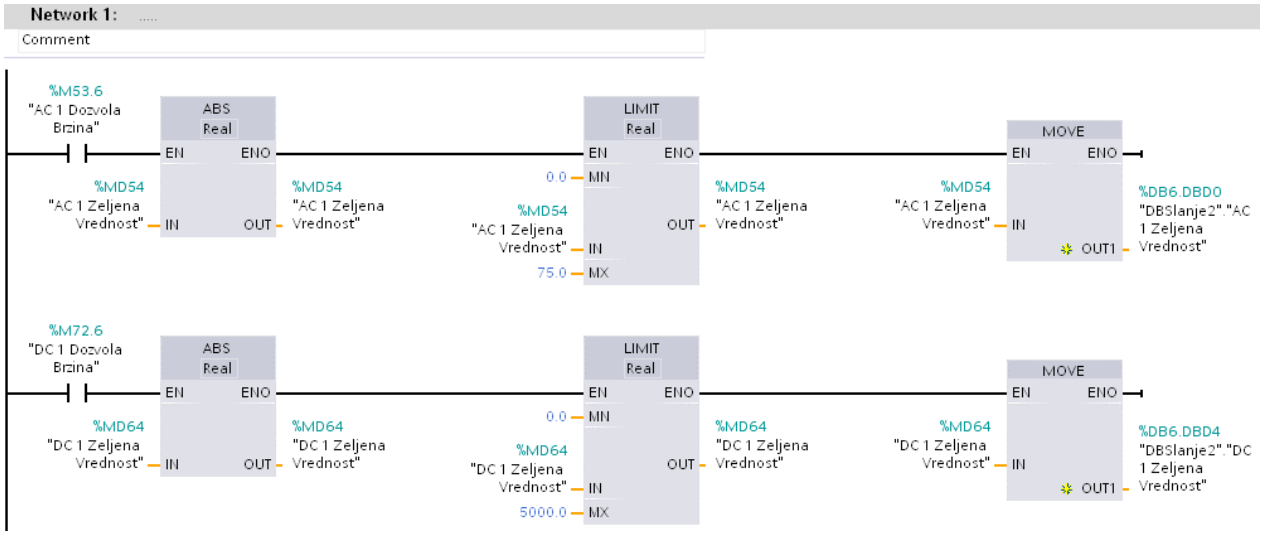
1.2.1.5. Блок *"Tag U DB Move"*

Овај блок служи да премешта вредности из тагова у блокове за податке и обрнуто да би жељене вредности са *HMI*-а слали на управљачке ПЛЦ-ове и назад.

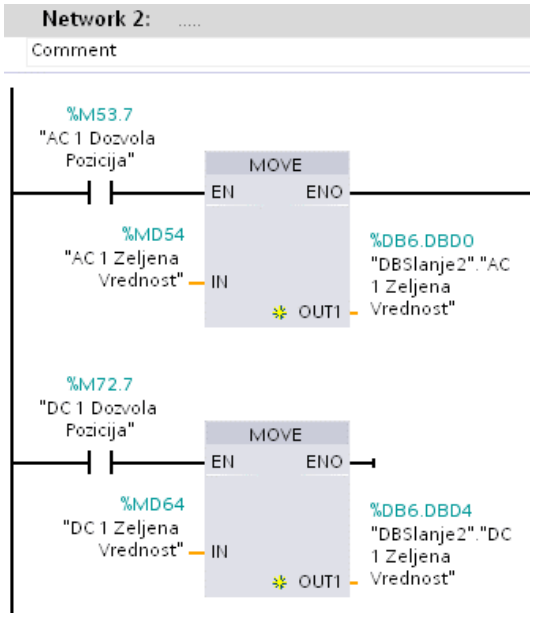
Мрежа 1 служи да нађе апсолутну вредност жељене брзине, јер смер одређујемо помоћу дигмића, а не предзнаком жељене вредности, а затим ограничава на 75, односно 5000, у зависности о ком мотору је реч. Ова ограничења су максималне брзине наших мотора. Једносмерни мотор има већу теоретску брзину, али она никада није достигнута у реалним условима.

Мрежа 2 има исту улогу као и мрежа 1, само овде не тражимо апсолутну вредност јер смер мотора одређујемо помоћу предзнака плус или минус, и не ограничавамо жељену вредност, јер не постоји максимално заокретање које мотор може да изврши.

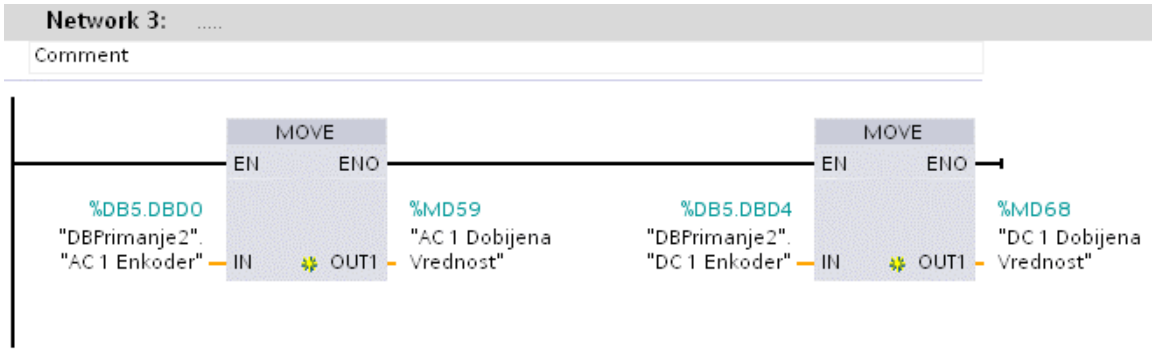
Мрежа 3 нам вредности које добијамо са управљачких ПЛЦ-ова шаље на *HMI* како би их приказали кориснику система да би он био обавештен о тренутном стању брзина и позиција мотора.



Слика 54. *Блок Tag U DB Move /Мрежа 1*



Слика 55. *Блок Tag U DB Move /Мрежа 2*

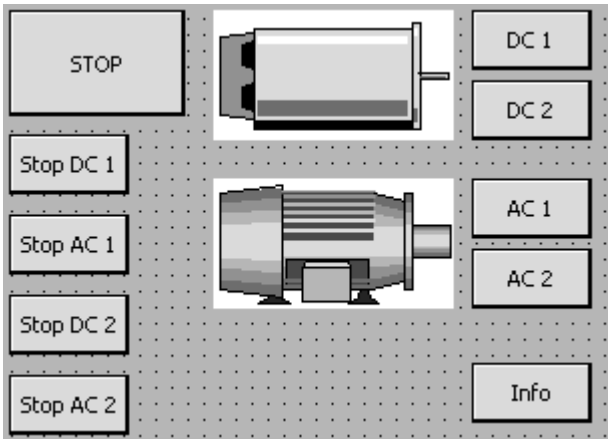


Слика 56. *Блок Tag U DB Move /Мрежа 3*

1.2.2. Програм *HMI*-а

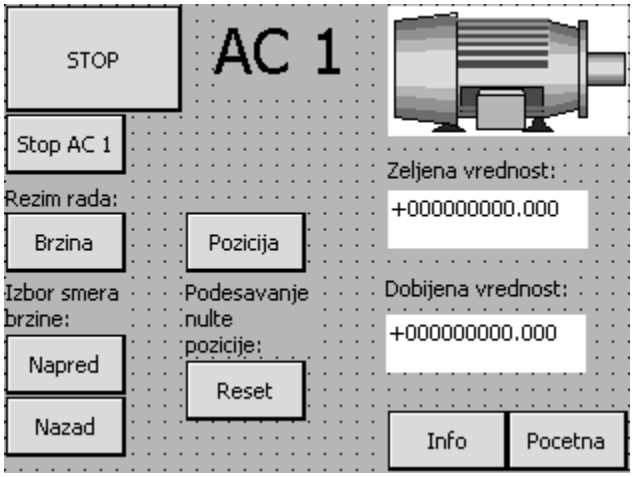
Овај програм је релативно лако одрадити и састоји се од превлачења дугмића или поља са понуђене листе, и њиховим повезивањем са одговарајућим промењивима у *Tagtable*-у главног ПЛЦ-а.

Испод је дата прва страна на којој је могуће зауставити све моторе по вољи, као и приступити контролама појединачних мотора.



Слика 57. *HMI почетна страна*

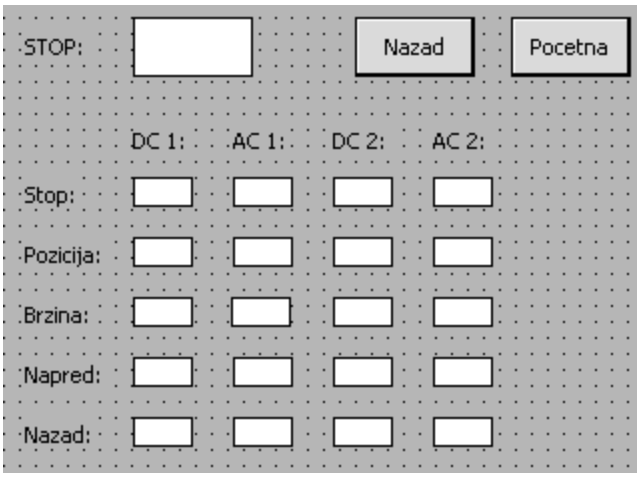
Даље можемо да видимо једну од страница за контролу мотора. Опет имамо главно дугме за заустављање као и дугме за заустављање одговарајућег мотора. Дугмићима можемо одабрати режим рада, који одлазе на главни ПЛЦ да буду сортирани као што је већ објашњено. У поље за жењену вредност уносимо брзину или позицију коју желимо да мотор достигне, а на добијену вредност нам систем враћа реалну вредност. Имамо и дугме за враћање на почетну страницу. Остале три странице за остале моторе су идентичне овој.



Слика 58. *Страна за управљање првим асинхроним мотором*

На крају имамо страницу за информације. На њој имамо преглед стања и режима свих мотора, али немам приступ контролама. Можемо се вратити на почетну страницу, или назад

на претходну страницу са које смо дошли. Приступ овој страници имамо са било које друге странице.

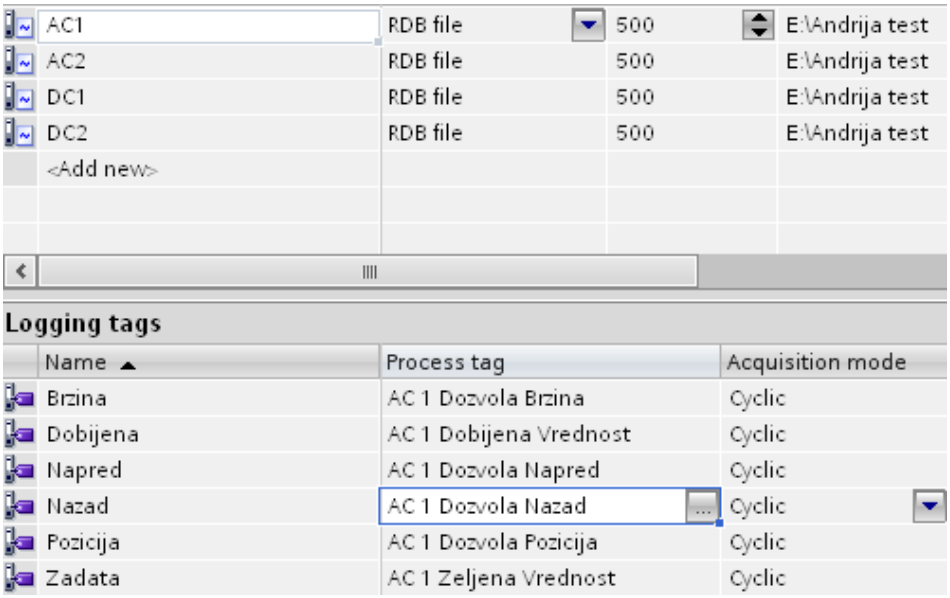


Слика 59. *Страна за информације о стању свих мотора*

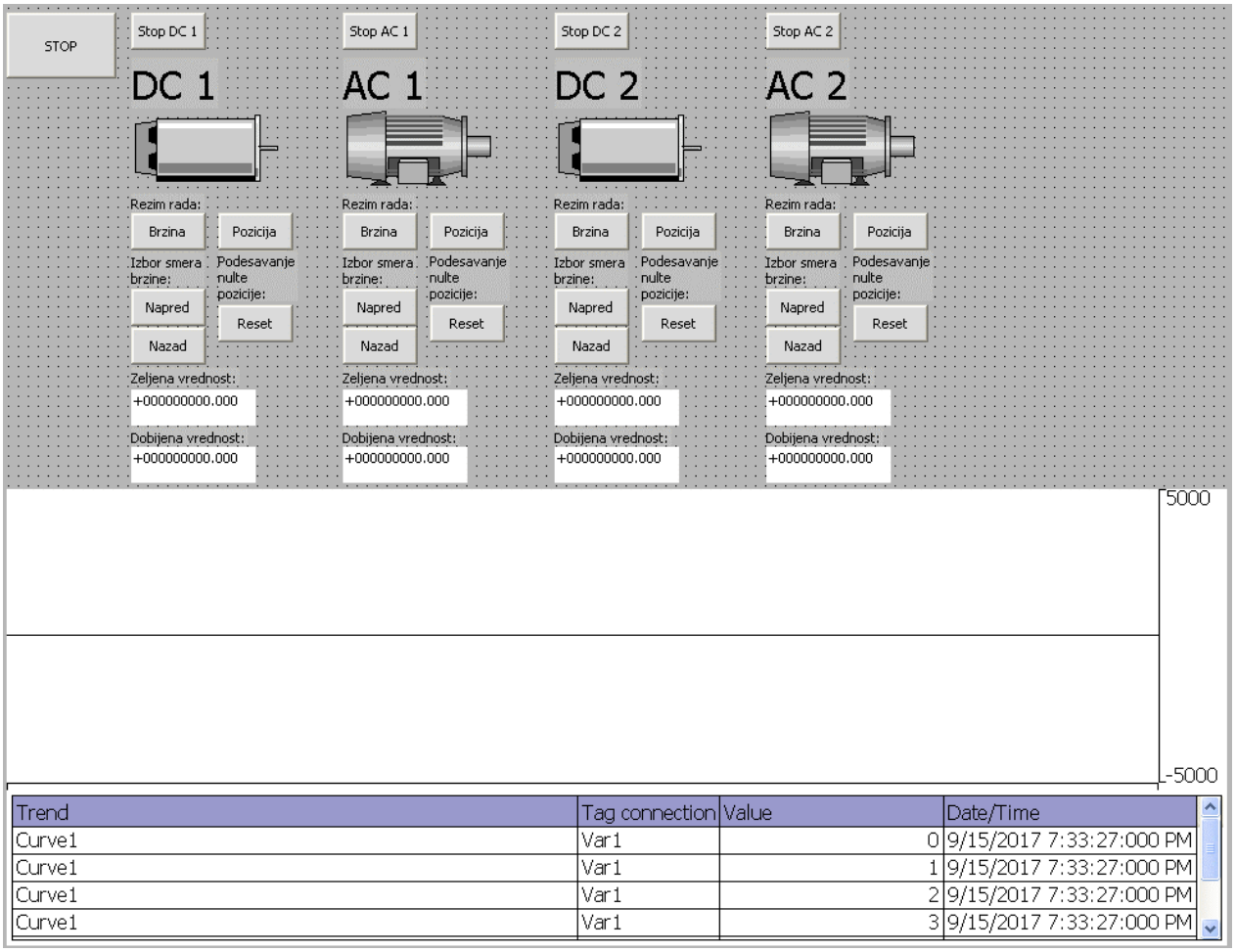
1.2.3. Програм *PC* станице

Програм је сличан као и код *HMI*-а, са том разликом што је монитор довољно велик да могу све контроле да стану на њега па нам није ни потребна страница за информације. Такође је додато и поље за исцртавање графика, на коме је могуће видети исцртавање промене реалне вредности пуном линијом, а жељну вредност тачкастом.

Један од циљева *SCADA* система је бележење историје рада, тако да је направљен систем за бележење тих података у *.xls* фајлу како би касније могли да их обрадимо у неком другом софтверу. Постоје белешке о историји за сва четири мотора одвојено.



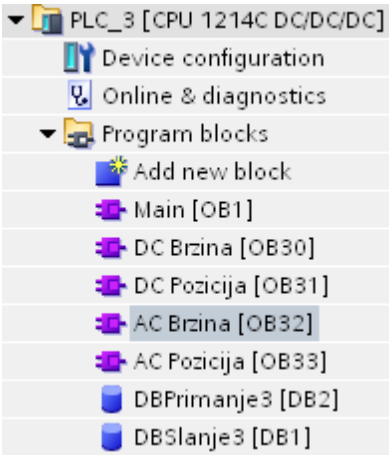
Слика 60. *Записивање историје првог асинхроног мотора*



Слика 61. *PC станица*

1.2.4. Програми управљачких ПЛЦ-ова

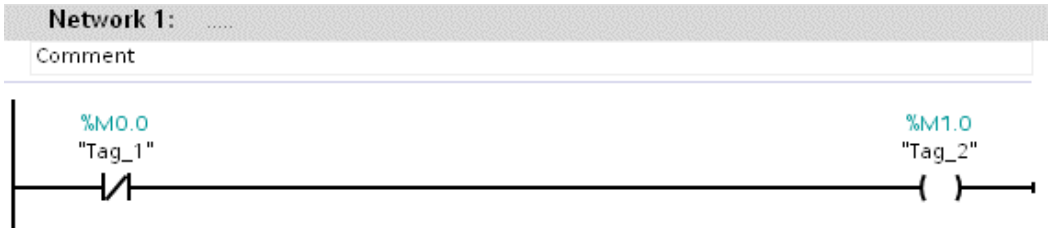
Пошто оба ПЛЦ-а управљају са једним асинхроним и једним једносмерним мотором, који имају исте карактеристике, онда су нам и програми идентични. једина разлика постоји у функцији *GET* јер се позивају различити блокови података, који садрже податке за одговарајући ПЛЦ. Инверзијом блокова би систем радио, само би се други мотори окретали. Ако би узимали исту базу онда би у исто време управљали са истим једносмерним или асинхроним моторима. Остали делови програма могу бити идентични, чак и до назива меморијских јединица, јер управљачки ПЛЦ-ови не деле меморије. Код нас постоји разлика у називима података у блоковима где се примају подаци ради лакшег распознавања, али она није потребна за исправан рад система. Даље ће бити објашњено како ради један од ПЛЦ-ова, пошто је програм за други копиран. За управљачке блокове је потребно користити посебан блок *Cyclic interrupt*. Обичан блок има периоду одабирања која се креће око одабране вредности, али није тачно дефинисана, док у блоку *Cyclic interrupt* она јесте. Без тачног времена одабирања ПИД управљање не може да ради како треба. Слично као и за главни ПЛЦ, програм је подељен у неколико блокова ради лакше навигације и исправке кода.



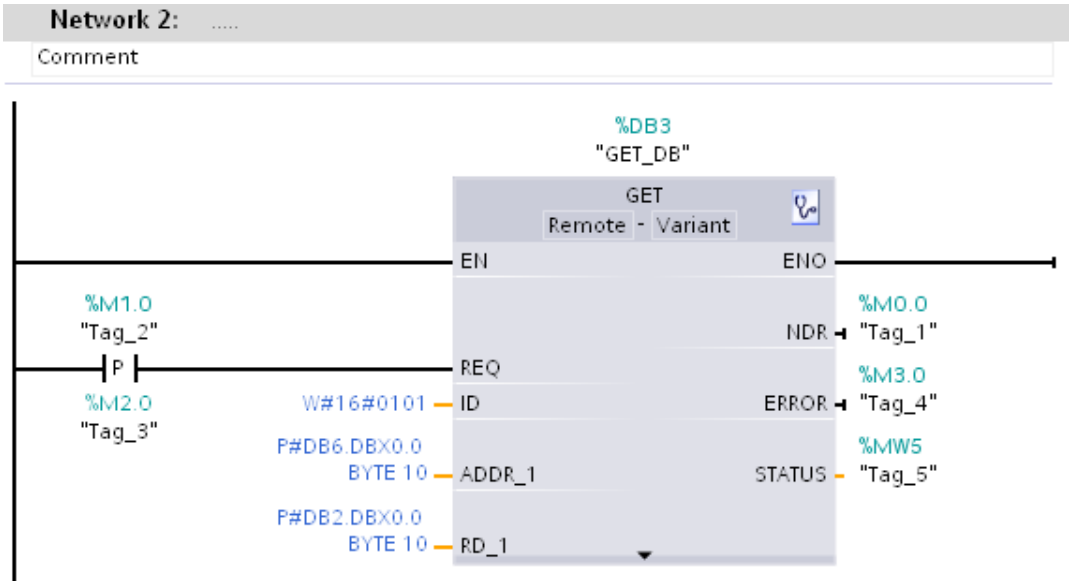
Слика 62. *Блокови управљачког ПЛЦ-а*

1.2.4.1. Блок "Main" управљачког ПЛЦ-а

У овом блоку се налазе све функције заједничке за више других блокова, као што је размена података и слање управљачких вредности на излазе ПЛЦ-а. Прве две мреже имају исту функцију као и код главног ПЛЦ-а, где узимају вредности из блока који садржи дозволе управљања и жељене вредности и копира их у свој идентичан блок.

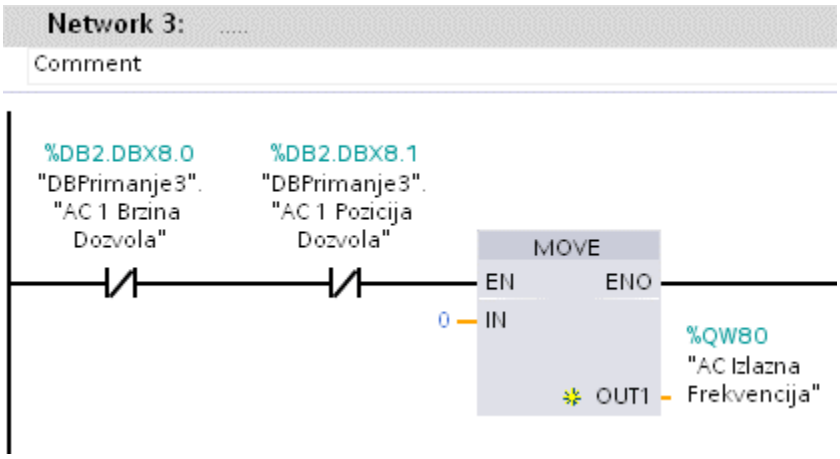


Слика 63. *Блок Main /Мрежа 1*

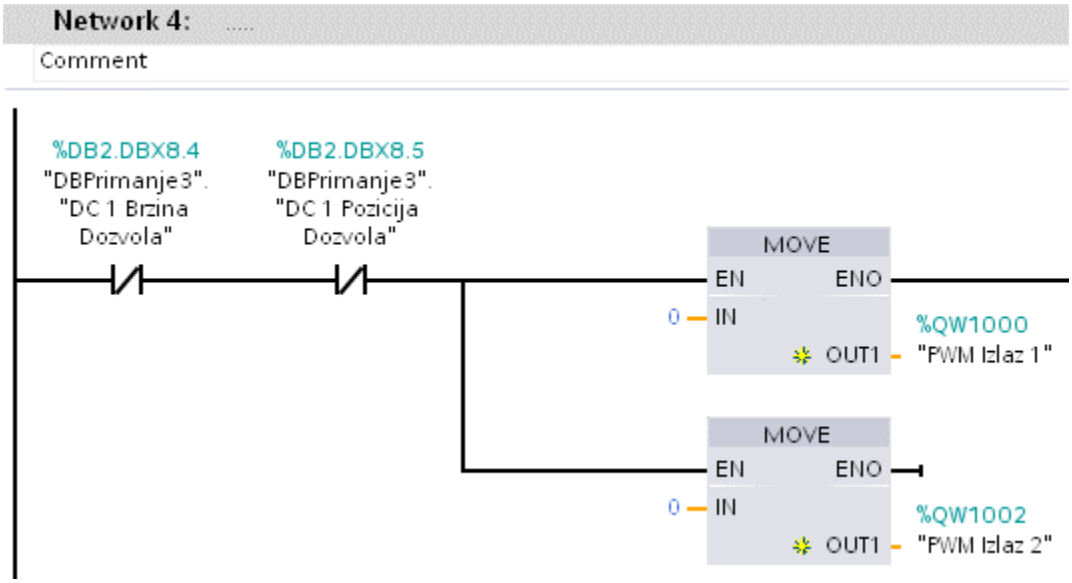


Слика 64. *Блок Main /Мрежа 2*

Мреже 3 и 4 служе, да у случају да ни контрола брзине ни контрола позиције немају дозволо рада, шаљу вредности нула на излаз за фреквенцију асинхроног мотора, у случају мреже 3, односно излазе за *PWM*-ове у случају мреже 4.

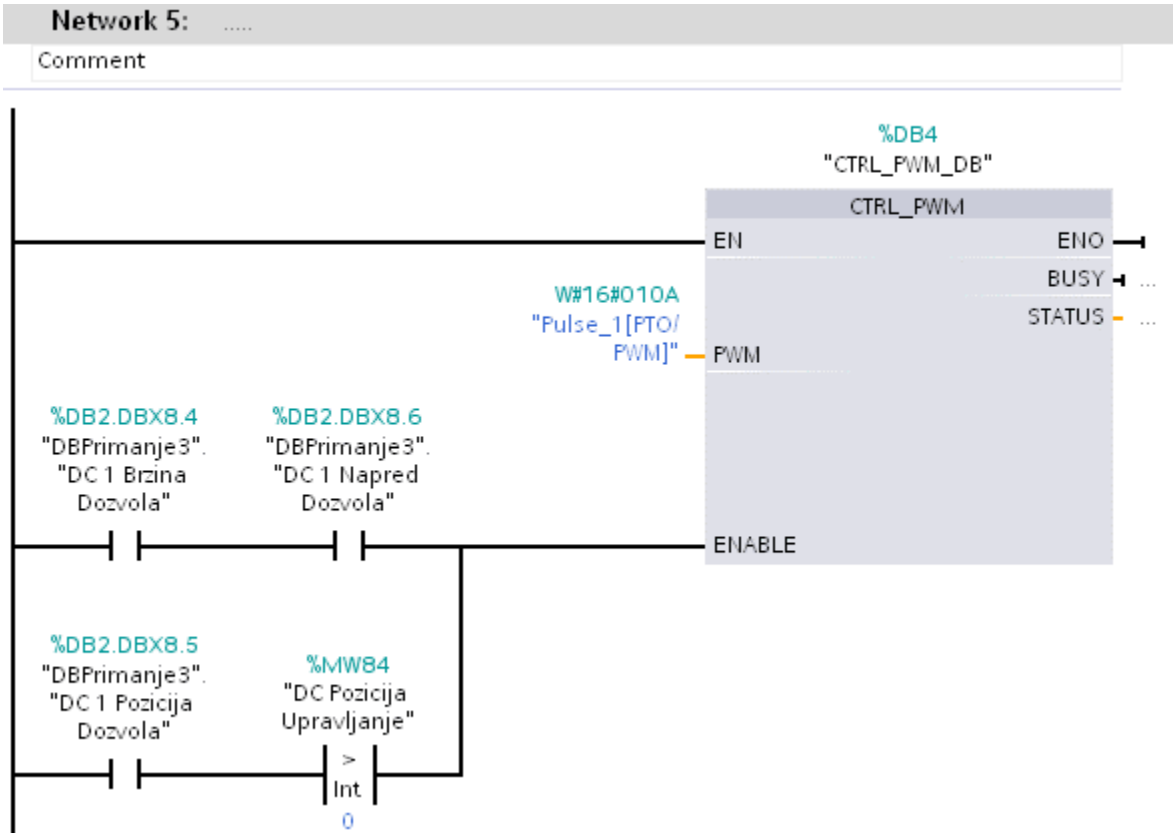


Слика 65. *Блок Main /Мрежа 3*

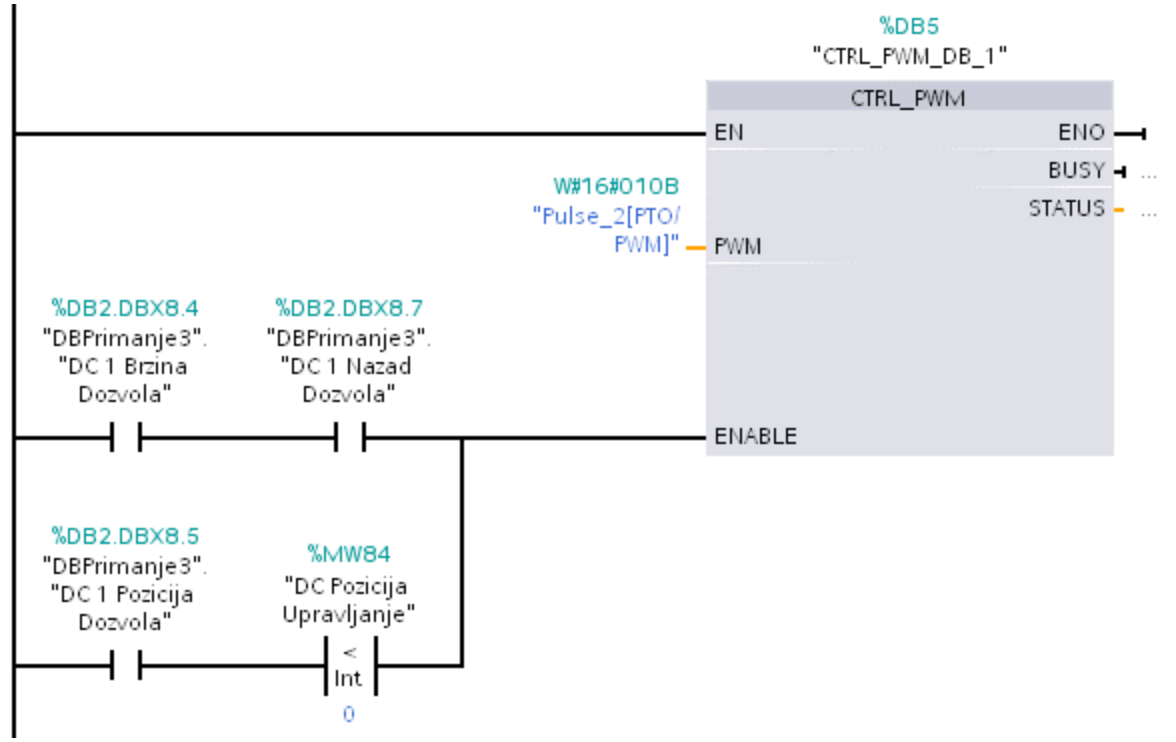


Слика 66. *Блок Main /Мрежа 4*

Мрежа пет у себи садржи два *PWM* блока који одговарају нашим физичким излазима. Када се на поље *Enable* доведе сигнал, *PWM* креће са радом пропорционално вредношћу која је доведена на меморијску локацију QW1000 односно QW1002. Ова мрежа проверава који је смер одабрао корисник код контроле брзине мотора, односно у ком смеру је програм израчунао да би мотор требало да се окреће како би дошао у жељени положај и укључује одговарајући *PWM*.

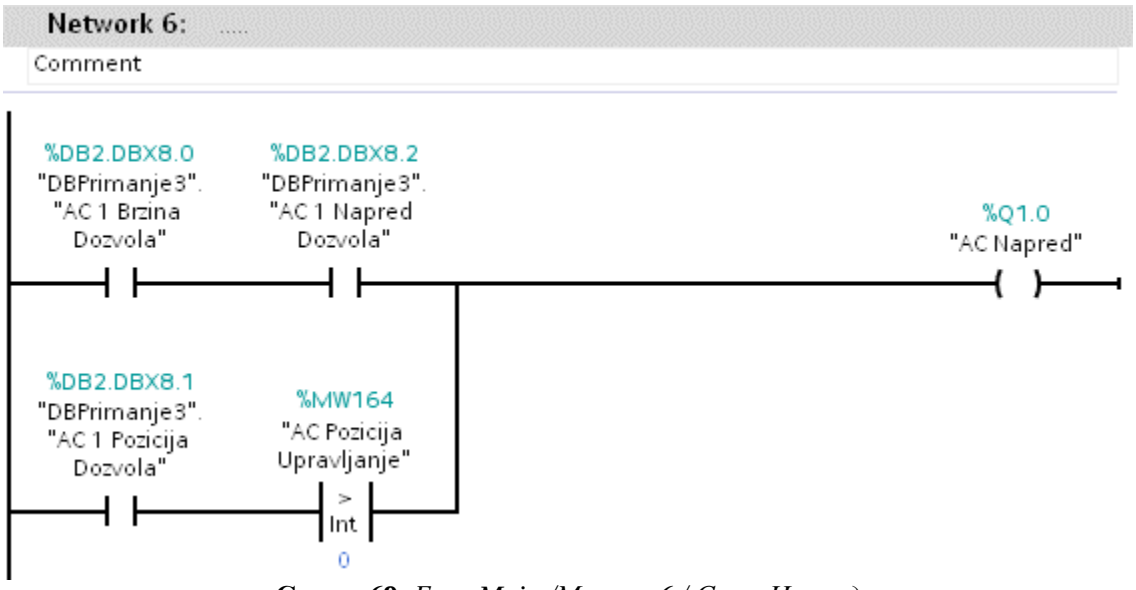


Слика 67. *Блок Main /Мрежа 5 / PWM 1*

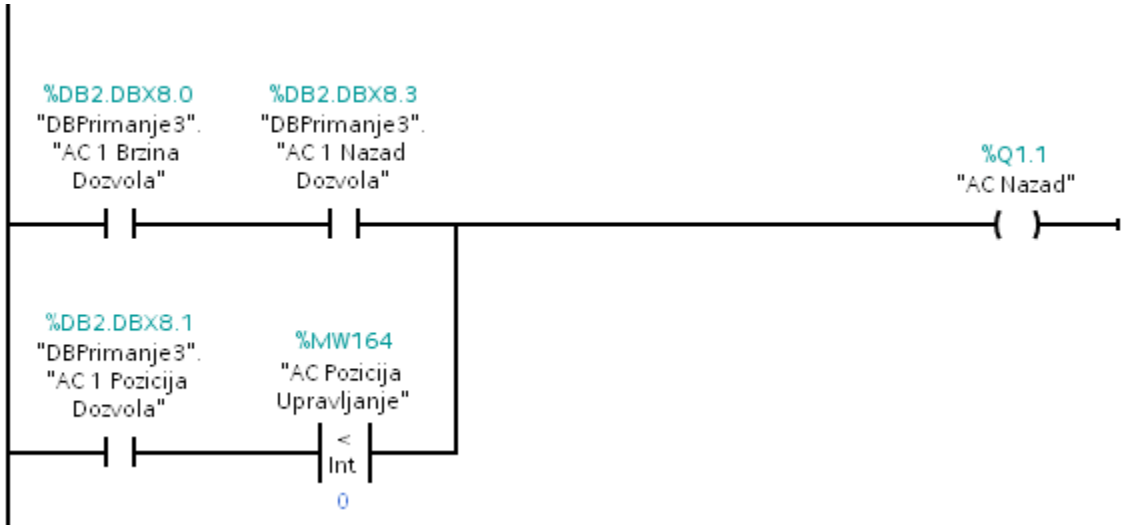


Слика 68. *Блок Main /Мрежа 5 / PWM 2*

Мрежа шест има потпуно исту улогу као и претходна, само што у овом случају се врши управљање асинхроним мотором, па се сигнал за смер шаље преко одговарајућег излаза на фреквентни регулатор.



Слика 69. *Блок Main /Мрежа 6 / Смер Напред*



Слика 70. *Блок Main /Мрежа 6 / Смер Назад*

1.2.4.2. Блокови података управљачког ПЛЦ-а

Имамо два блока, један већи у који уписујемо вредности које добијамо, и мањи из кога ће тренутне вредности брзине или положаја бити послате са главног ПЛЦ-а.

	Static			
	AC 1 Zeljena Vrednost	Real	0.0	0.0
	DC 1 Zeljena Vrednost	Real	4.0	0.0
	AC 1 Brzina Dozvola	Bool	8.0	false
	AC 1 Pozicija Dozvola	Bool	8.1	false
	AC 1 Napred Dozvola	Bool	8.2	false
	AC 1 Nazad Dozvola	Bool	8.3	false
	DC 1 Brzina Dozvola	Bool	8.4	false
	DC 1 Pozicija Dozvola	Bool	8.5	false
	DC 1 Napred Dozvola	Bool	8.6	false
	DC 1 Nazad Dozvola	Bool	8.7	false
	AC 1 Reset Nula	Bool	9.0	false
	DC 1 Reset Nula	Bool	9.1	false
	Space 1	Bool	9.2	false
	Space 2	Bool	9.3	false
	Space 3	Bool	9.4	false
	Space 4	Bool	9.5	false
	Space 5	Bool	9.6	false
	Space 6	Bool	9.7	false

Слика 71. *Блок за примање података*

	Static			
	AC 1 Enkoder	Real	0.0	0.0
	DC 1 Enkoder	Real	4.0	0.0

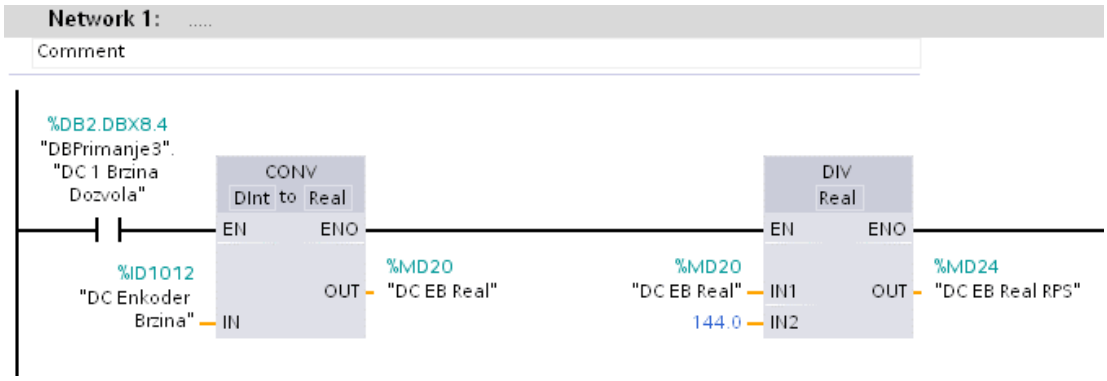
Слика 72. *Блок за слање података*

1.2.4.3. Блок *"DC Brzina"*

За све мреже овог блока прво проверавамо да ли је дозвола за контролу брзине једносмерног мотора присутна. Ако јесте онда вршимо претварање сигнала са енкодера у одговарајућу величину, налазимо грешку у односу на жељену вредност и на основу ње управљање које шаљемо на одговарајући излаз.

Мрежа један служи за трансформацију вредности са енкодера у одговарајућу вредност. Пошто нам енкодер даје вредност у облику промењиве *Int* (*integer* - цео број), ради бољег рачуна ми је прво пребацујемо у вредност *Real* (реалан, децимала број). Нама енкодер даје фреквенцију подеока енкодера, односно колико пута је диода осветлила фотосензор. Пошто наш енкодер има 144 подеока, реалну вредност са енкодера делимо са тим бројем како би добили број обртаја у секунди. Затим тражимо апсолутну вредност. У зависности од

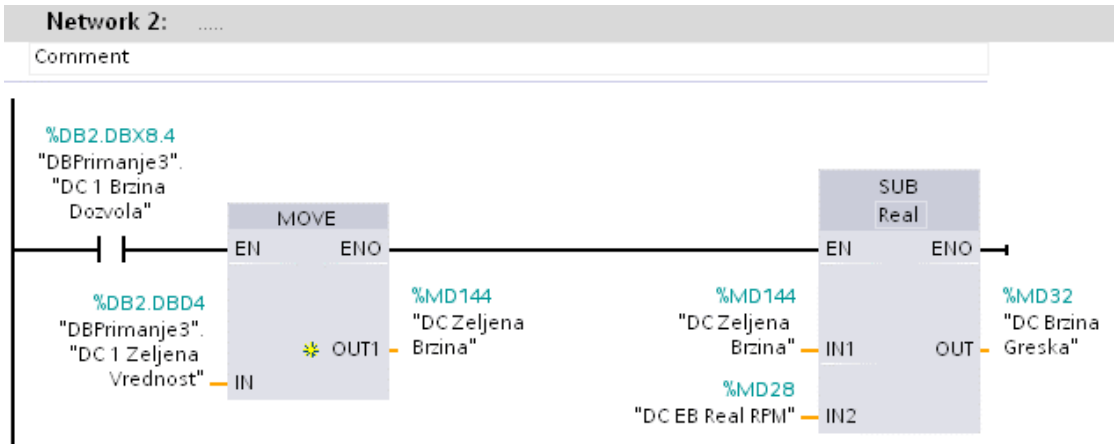
смера обртања, вредност са енкодера ће бити позитивна или негативна. Пошто користимо избор смера одвојен од управљања, ми ћемо га налазити увек као позитивну вредност. На крају нашу вредност множимо са 60 како би смо добили обртаје у минути.



Слика 73. *Блок DC Brzina /Мрежа 1/1*

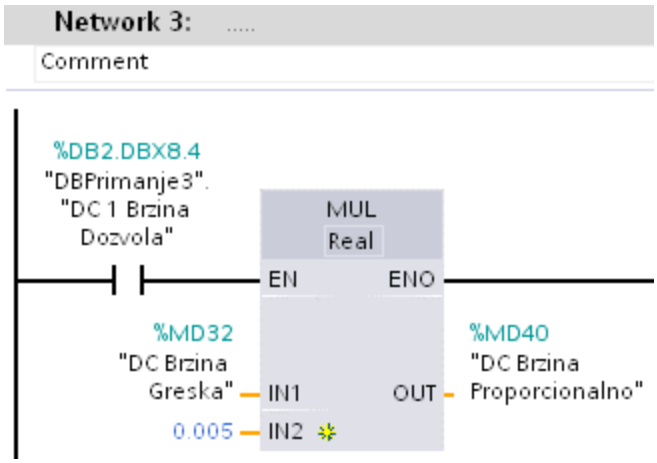
Слика 74. *Блок DC Brzina /Мрежа 1/2*

Мрежа два нам жељену вредност из промењиве из блока података, која служи и за брзину и позицију, пребацује у специјалну локалну меморију ради лакшег пражњења меморије приликом кашњења контроле брзине. Затим од наше жељене вредности одузима претходно добијену вредност обртаја у минути и даје нам грешку.



Слика 75. *Блок DC Brzina /Мрежа 2*

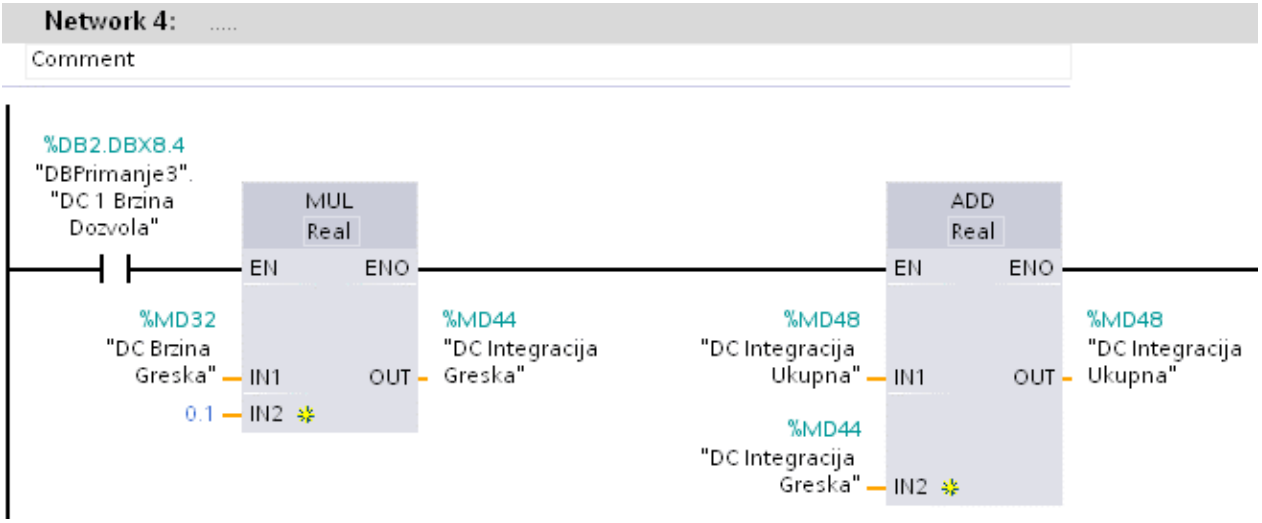
Мрежа три нам налази пропорционални део управљања тако што нашу грешку множи са П појачањем које је нађено емпиријски.



Слика 76. *Блок DC Brzina /Мрежа 3*

Мрежа четири нам тражи интеграционо управљање. Прво је неопходно наћи интеграл грешке. Интеграл представља површину испод неке криве. Ми је можемо апроксимирати као збир квадрата које се налазе испод те функције. Висина квадрата одговара вредности од нуле до вредности функције у том тренутку, а ширина периоди одабирања. Што је периода одабирања мања то нам је наш интеграл тачнији.

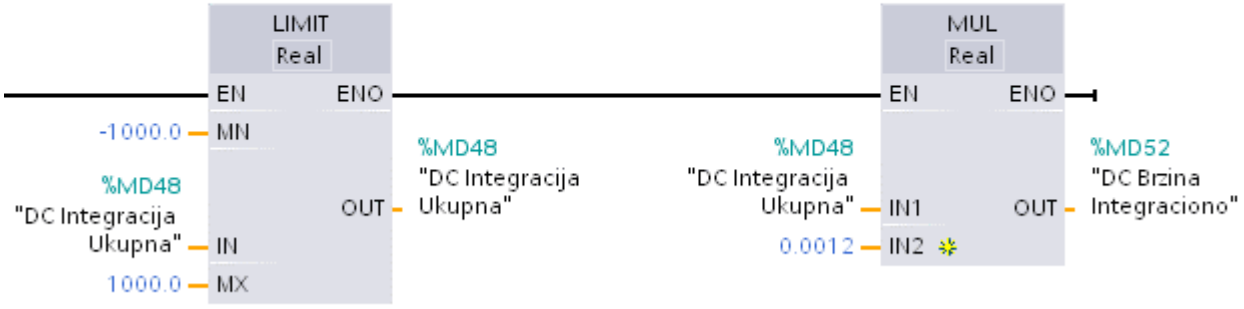
Ми прво множимо тренутну грешку са периодом одабирања, што нам даје површину најновијег квадрата грешке, а затим ту површину додајемо укупној површини свих осталих квадрата.



Слика 77. *Блок DC Brzina /Мрежа 4/1*

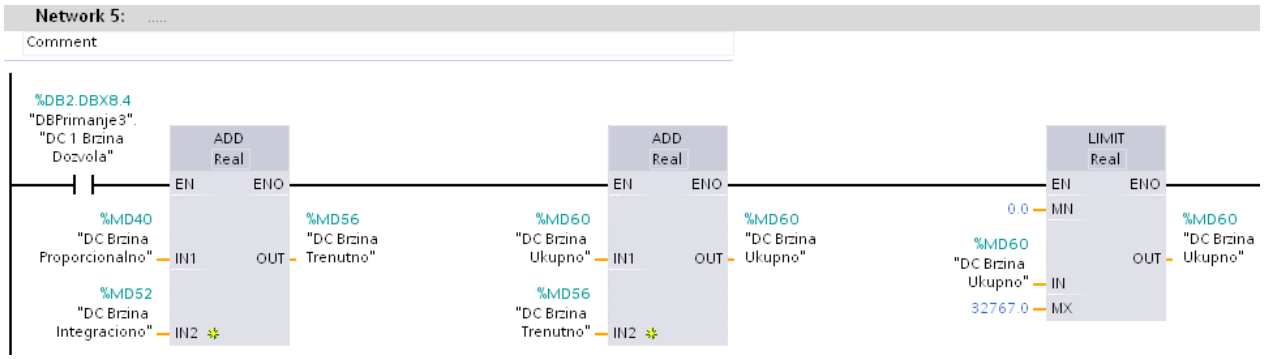
Дуги део мреже нам ограничава укупну вредност интегратора. Ово радимо да би спречили превелико намотавање грешке интегратора, чиме смањујемо време потребно за доласка у жељени положај. Уколико превише ограничимо интегратор, можемо уништити

његово дејство да систем, па би било као да он не постоји. На крају нашу добијену вредност множимо са И појачањем и добијамо интегрални део управљања.



Слика 78. *Блок DC Brzina /Мрежа 4/2*

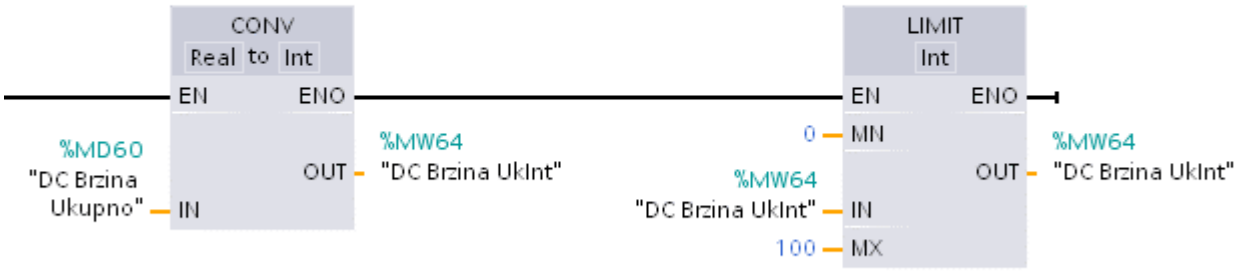
Први део мреже пет нам само сабира пропорциналну и интегралну вредност управљања и тиме добијмо ПИ управљање. Овим се завршава наше ПИ управљање.



Слика 79. *Блок DC Brzina /Мрежа 5/1*

Ми смо сада нашли нашу вредност управљања, али начин физичког управљања, као и ограничења коришћених система доводе до неких проблема са којима би смо морали да се изборимо. Први проблем је што ми наше управљање шаљемо на *PWM*. Када ми достигнемо жељену вредност, наша грешка пада на нулу, а тиме и сигнал управљања, што доводи до тога да и проценат *PWM*-а пада на нулу, па престаје ток струје у мотор, па он успорава. Ово повећава грешку, па ми поново добијамо управљање, *PWM* добија сигнал и мотор поново креће. Овакво понашање би се вртело у бесконачно и никад не бисмо имали стабилно стање. ово исправљамо тако што управљања свих циклуса сабирамо у једну вредност. Ово доводи до тога да ће време смирења и добијања тачне вредности бити дуже, али зато када тачно управљање за одржавање брзине буде нађено, и грешка буде нула, *PWM* ће моћи да користи већ записано управљање, уместо да га узима дирекно са ПИ управљања.

Трећи и четврти блок мреже 5 служе да претворе реалну вредност у цео број. Наше управљање је реална вредност, али *PWM* може да користи само целе бројеве. Овде долази до проблема што реална промењива има резервисано 32 бита информација, док цео број има 16 битова. То значи да реална вредност може да буде много већа, што доводи до проблема при пребацивању. Меморија целог броја се креће од -32767 до 32767, где нам последњи бит одређује знак. Уколико покушамо у њега да ставимо већи број, он ће мењати наш знак, тако да ће довести до наглог скакања са јако великих вредности, на негативне и обрнуто. Пошто наш *PWM* може да користи сигнал од 0 до 100 процената, то доводи до скакања његове вредност са минимума на максимум, што доводи у нестабилно стање, где ће грешка да распе, па тим и управљање, све док се на крају и меморија реалне вредности препуни и када програм креће да прави још веће проблеме. То исто скакање вредности *PWM* ће нам правити и проблеме на хардверу јер константна промена сам максималне вредности напона на нулу у кратком временском року може да направи оштећење на хардверу. Зато ми нашу реално вредност ограничавамо од 0 до 32767 јер не користимо вредности мање од нуле [33].



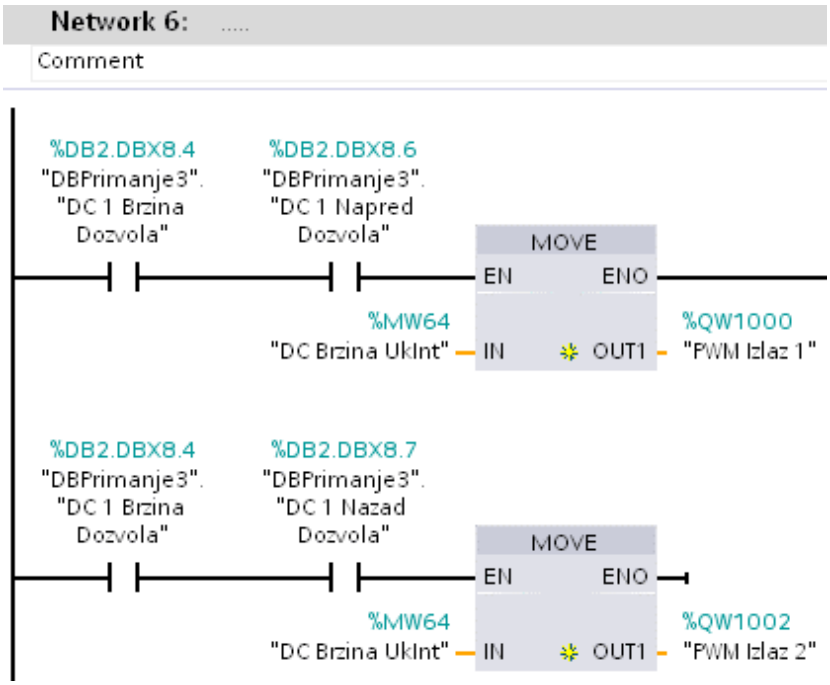
Слика 80. *Блок DC Brzina /Мрежа 5/2*

На крају имамо блок који управљачку вредност ограничава на вредности од 0 до 100, које одговарају проценту искоришћења *PWM*-а. Ово на први поглед изгледа нелогично јер губимо већи део управљања. Логичније би било да нађемо максимално могуће управљање, које имамо кад имамо максималну грешку, у овом случају би то било када би мотор мировао, а добио би задатак да убрза до максималне вредности. Онда би ми могли да управљачку вредност од нуле до максимума скалирамо на вредности од 0 до 100. ово није добро, јер ћемо максимално искоришћење *PWM*-а имати само у тренутку максималне грешке. Ако је она мања управљање ће одмах користити мањи проценат *PWM*-а. Ово доводи до тога да што је грешка мања, мање је и искоришћење, па ће нам време раста бити јако дуго. На крају када би пришли жељеној вредности, имали би само неколико процената *PWM*-а за подешавање финог управљања да би довели мотор до што тачније вредности. Можда и никад не бисмо достигли жељену вредност, већ би она била асимптота наше реалне вредности.

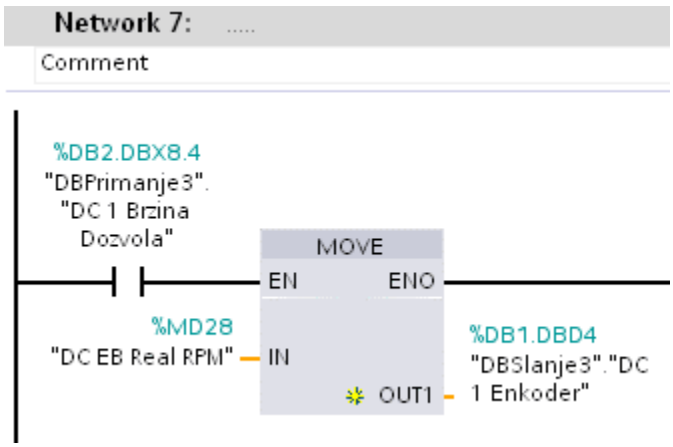
Начин који ми користимо је тачнији јер ћемо користити максимално убрзање све док не прићемо близу жељене вредност, када ћу управљање пасти на испод 100, па ће и

коришћење процента *PWM* бити мање. Вредности управљања веће од 100 нам просто говоре да је грешка још увек велика, и да би требали да наставимо да убрзавамо ка жељеној вредности. На овај начин имамо много бржи раст брзине, тако да је време достизања жељене вредности много мање.

Мрежа шест нам претходно добијену вредност процента *PWM* сигнала шаље на одговарајући *PWM* у зависности од одабраног смера.



Слика 81. *Блок DC Brzina /Мрежа 6*

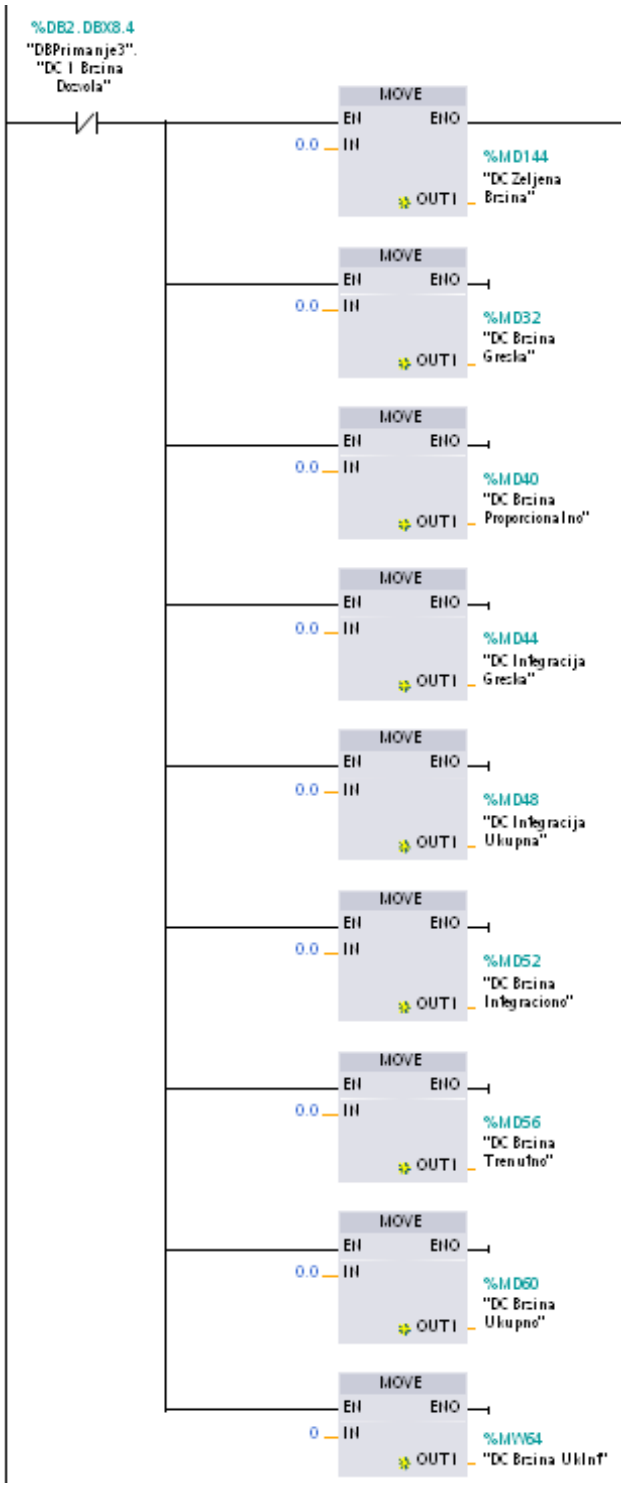


Слика 82. *Блок DC Brzina /Мрежа 7*

Мрежа седам нам вредност са енкодера у обртајима у минути шаље у блок података који ћемо да шаљемо на главни ПЛЦ како би кориснику приказали тренутну вредност брзине.

Мрежа осам нам, у случају да не постоји дозвола за контролу брзине, гура нула вредности у све наше промењиве које користимо за налажење управљања. Ово радимо, да

ако је контрола брзине радила неко време и нашла неку вредност, па онда угашена, а затим упаљена након неког времена, мотор ће одмах кренути да се окреће према старој вредности. Ово је непожељно, па ми просто празнимо све вредности.

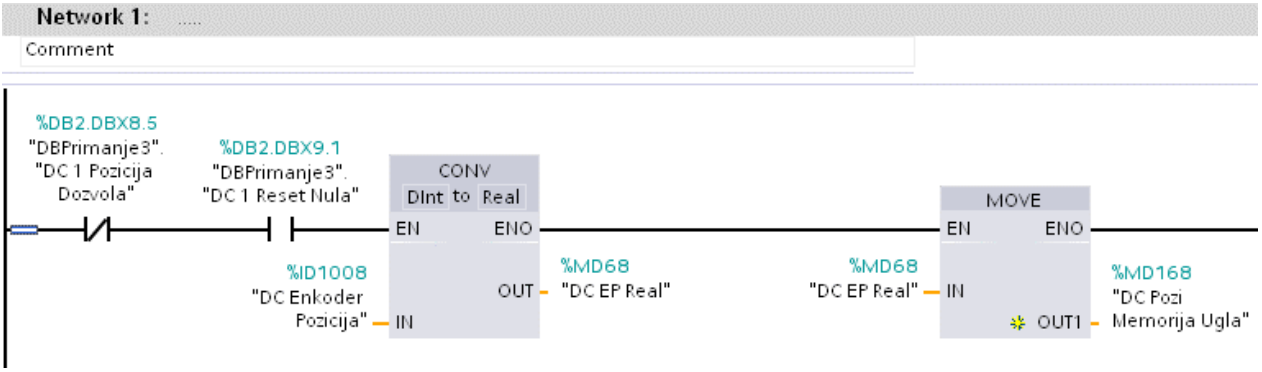


Слика 83. *Блок DC Brzina /Мрежа 8*

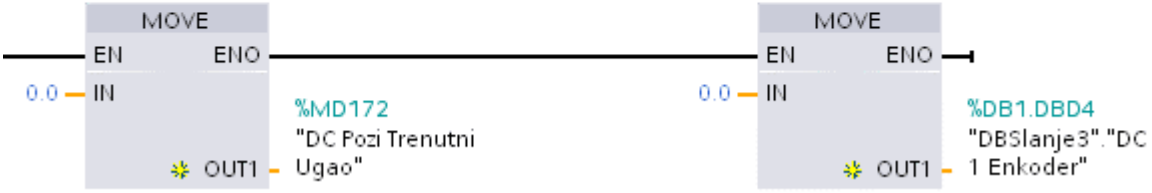
1.2.4.4. Блок *"DC Pozicija"*

Уколико смо неко време возили наш мотор одређеном брзином, енкодер за позицију је у исто време радио, и он је за то време могао лако да изброји неколико милиона импулса. Ако би му ми сада задали вредност 60, он не би ту вредност додао на већ избројене вредности, него би кренуо да се врти назад. Ако корисник жели да на тренутну позицију дода вредност од 60 степени, он мора да сабере жељену и тренутну вредност. Ово је непотребно јер притиском на дугме је могуће поставити да тренутну вредност померања програм третира као нулу, односно као почетак.

Мрежа један има ту улогу. Ако је дугме за ресетовање притиснуто, и програм тренутно нема дозволу тражења позиције, како не би било могуће ресетовати нулу сред рада програма, ми прво пребацујемо вредност из целог броја у реални као и код програма за управљање брзином мотора. Затим вредност енкодера пребацујемо у меморију која ће да је запамти како би је користили као референтну тачку. У вредност тренутног угла гурамо вредност нула, и у блок за податке шаљемо нулу, како би обавестили корисника програма да је позиција ресетована.

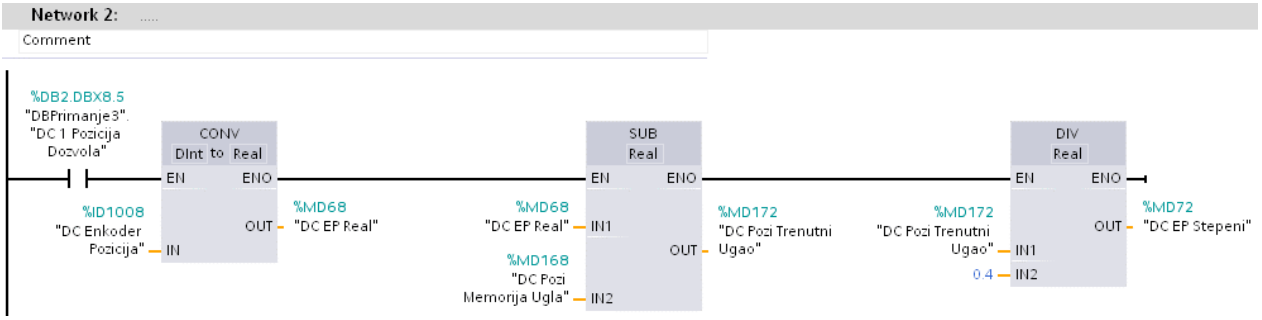


Слика 84. *Блок DC Pozicija /Мрежа 1/1*



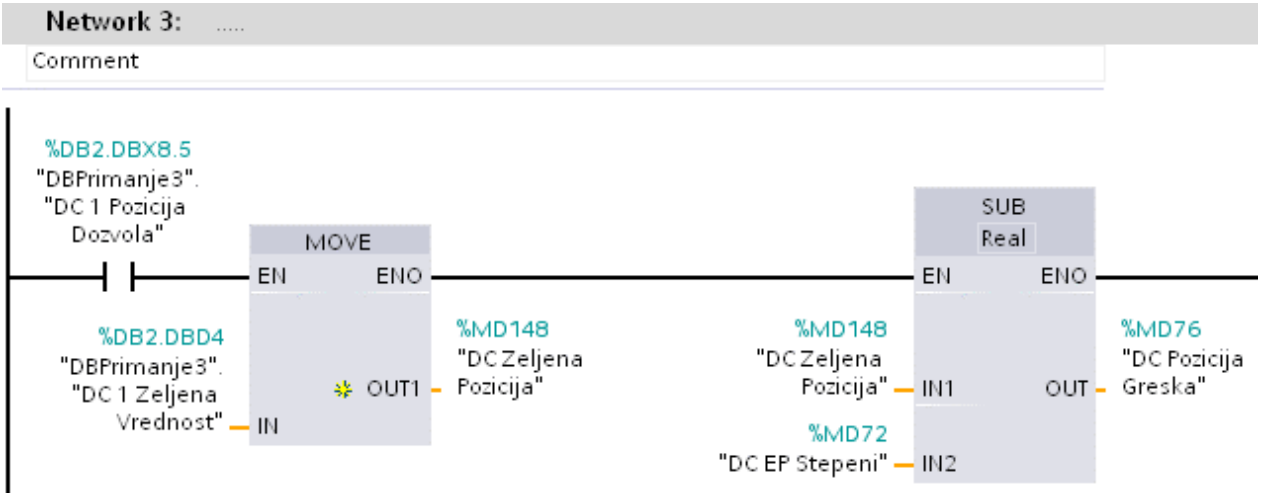
Слика 85. *Блок DC Pozicija /Мрежа 1/2*

Мрежа два нам поново мења вредност из целог броја у реалну, због различитих услова рада мрежа, затим од тренутно вредности енкодера одузима вредност меморије угла коју смо запамтили у мрежи један, како би нашли померање у односу на њу, пошто је она сада наша нула. Ту вредност затим делимо са 0,4. Математички гледано, дељење са 0,4 је једнако множењу са 2,5. Ово желимо да одрадимо јер наш енкодер има 144 подеока, тако да између сваког подеока имамо простор од 2,5°. овим множењем добијамо заокретање мотора у степенима.



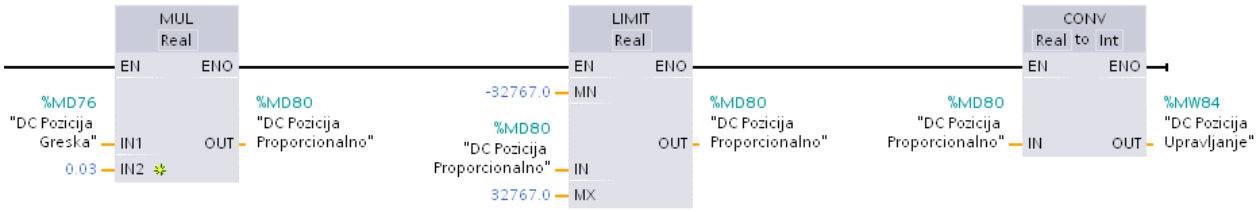
Слика 86. *Блок DC Pozicija /Мрежа 2*

Мрежа три нам, као и у претходном блоку, вредност из заједничког места у блогу података пребацује у посебну меморијску јединицу, а затим налази грешку жељене и реалне вредности.



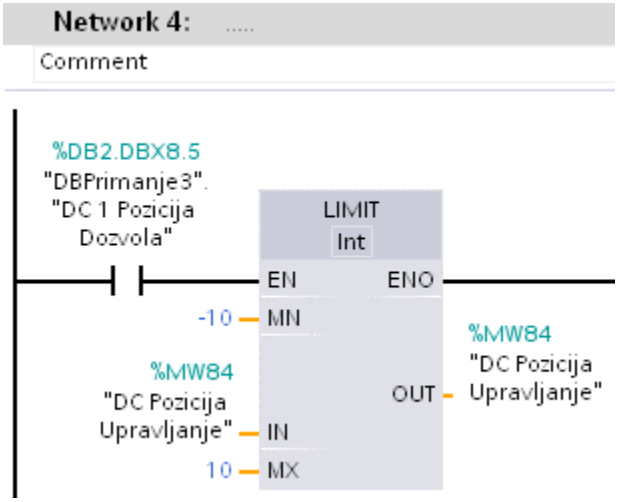
Слика 87. *Блок DC Pozicija /Мрежа 3/1*

Ту грешку затим множимо са П појачањем како би добили управљање. за контролу позиције користимо само П управљање. Добијено управљање сада ограничавамо на вредности од -32767 до 32767, због истих разлога као и раније, са том разликом што сада користимо негативне вредност. На крају ми наше управљање мењамо из реалне вредности у целобројну вредност.



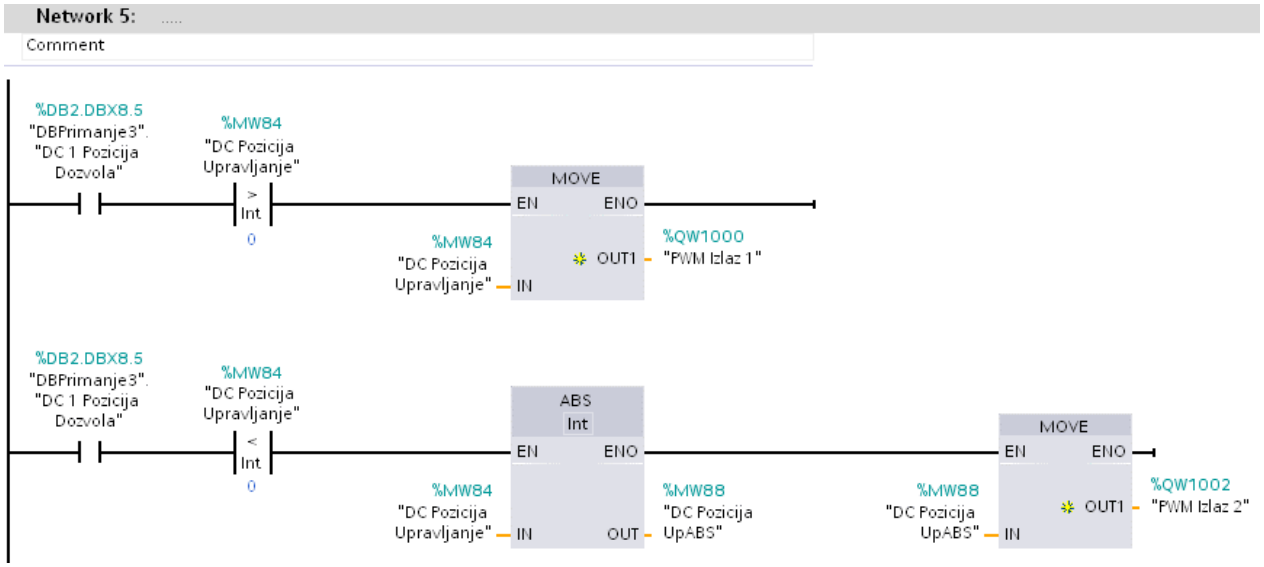
Слика 88. *Блок DC Pozicija /Мрежа 3/2*

Мрежа четири нам даље наше управљање ограничава на вредности од 0 до 10 које одговарају *PWM*-у. Ради веће тачности користимо само 10 процената снаге *PWM*-а. Негативни део управљања користимо само за одређивање смера, јер *PWM* не може да ради са негативном вредностима.



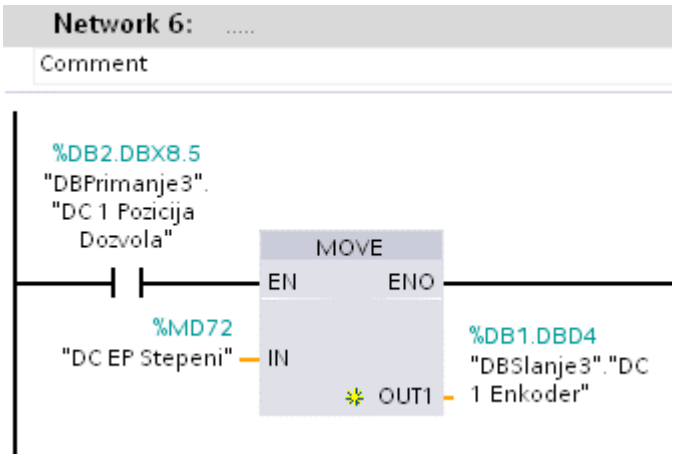
Слика 89. *Блок DC Pozicija /Мрежа 4*

Мрежа пет нам, на основу тога да ли је вредност управљања позитивна или негативна, шаље управљачку вредност на одговарајући *PWM*. Ако је вредност негативна, прво налази њену апсолутну вредност након одређивања смера, како би могли да је пошаљемо на *PWM*.



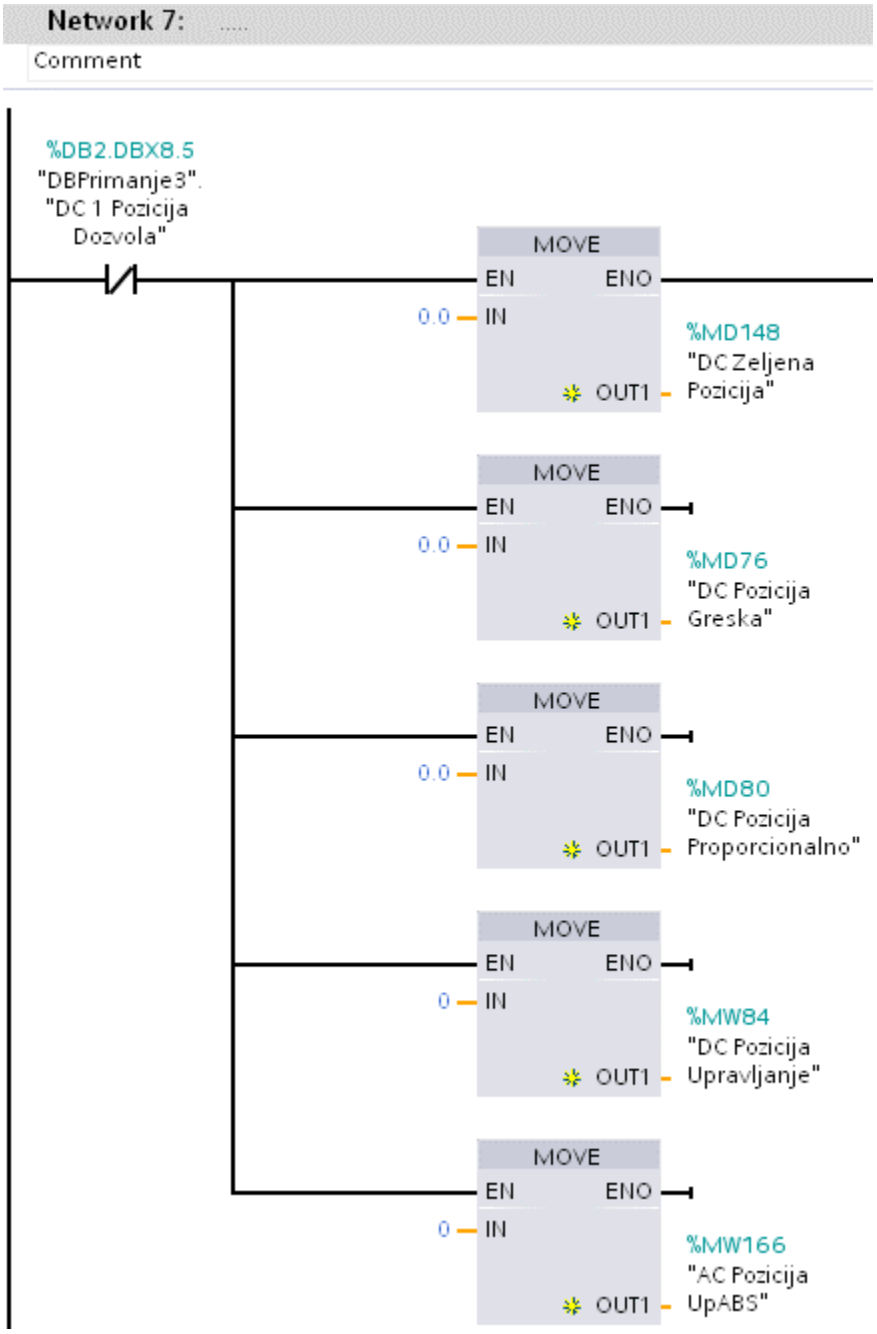
Слика 90. *Блок DC Pozicija /Мрежа 5*

Мрежа шест нам има исту улогу као и мрежа седам у претходном блоку, где нам вредност са енкодера у степенима шаље у меморију блока за размену података како бисмо обавестили оператера о тренутном положају мотора.



Слика 91. *Блок DC Pozicija /Мрежа 6*

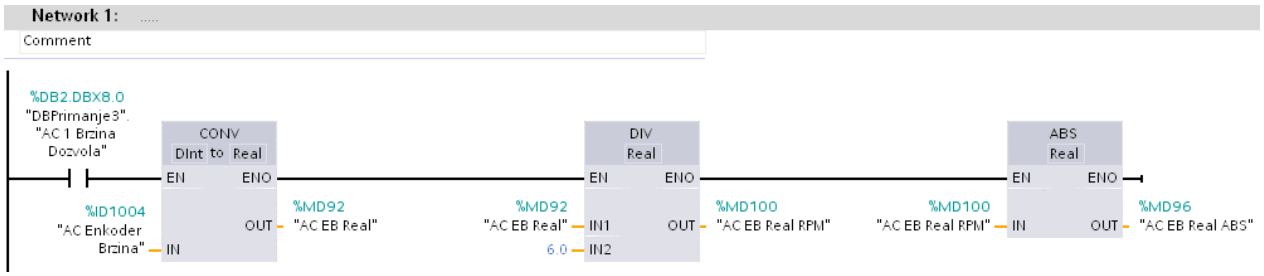
Мрежа седам нам гура нуле у све наше промењиве које користимо у овом блоку у случају да не постоји дозвола управљања позицијом, из истих разлога као и код управљања брзином.



Слика 92. *Блок DC Pozicija /Мрежа 7*

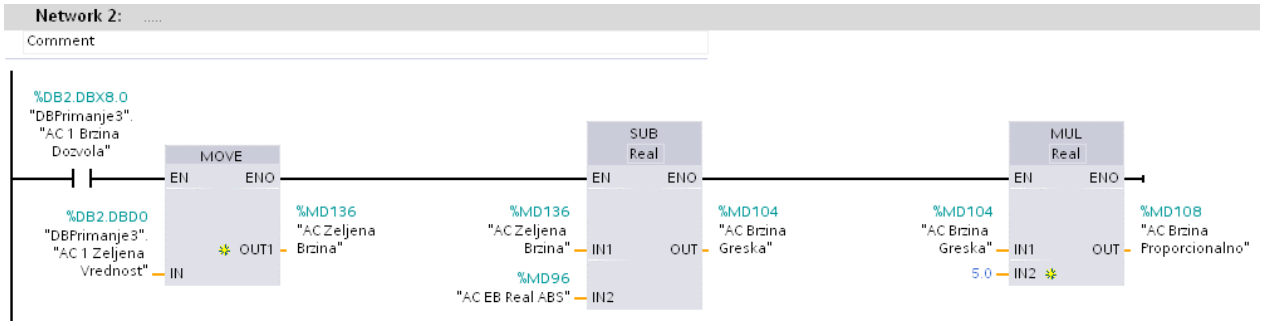
1.2.4.5. Блок *"AC Brzina"*

Слично као и код брзине једносмерног мотора, вредност енкодера пребацујемо из целог броја у децималну вредност, и онда делимо са бројем 6. Код једносмерног броја смо делили са 144 па множили са 60, док би смо овде морали да делимо са 360 јер толико подеока има енкодер, па множили са 60, односно скраћено, делили са 6. На крају налазимо апсолутну вредност, као и код једносмерног мотора.



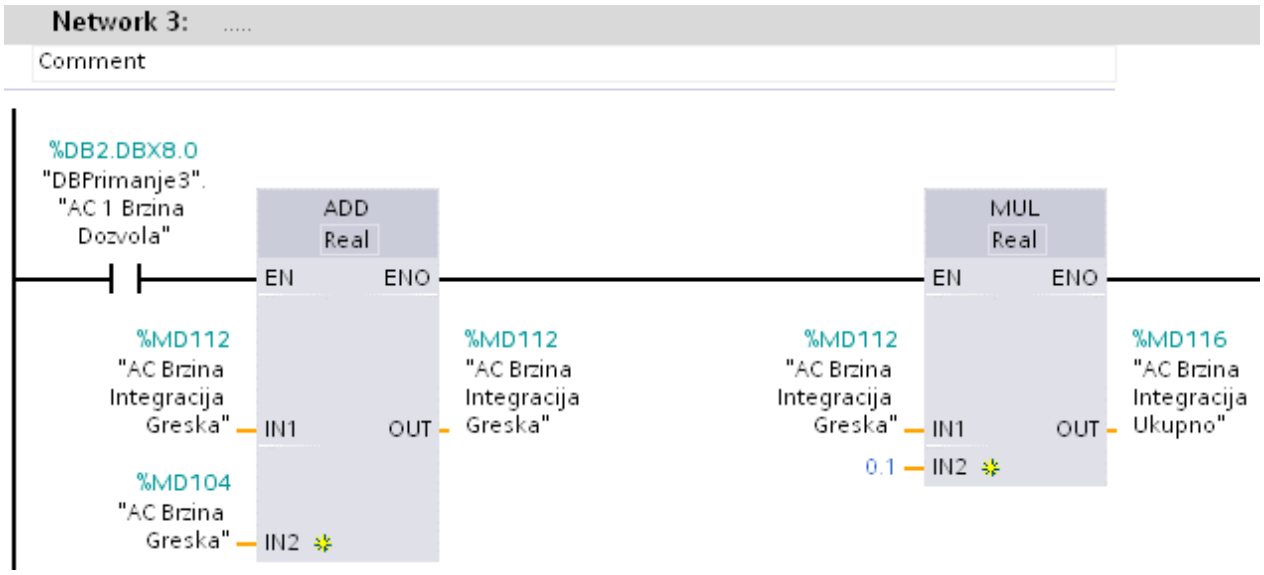
Слика 93. *Блок AC Brzina /Мрежа 1*

Мрежа два нам прво пребацује жељену вредност из заједничке у посебну меморију, налази грешку, и затим одређује П управљачку компоненту.



Слика 94. *Блок AC Brzina /Мрежа 2*

Мрежа три нам налази интеграцију истом логиком као и код једносмерног мотора, и затим множи са појачањем да би смо добили И управљање.

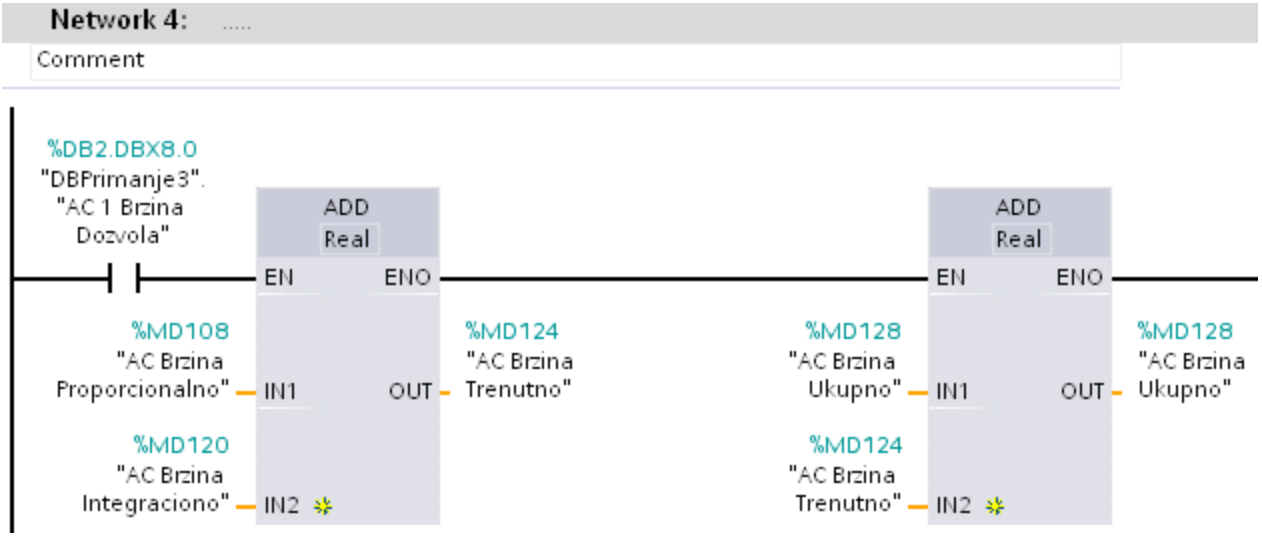


Слика 95. *Блок AC Brzina /Мрежа 3/1*

Слика 96. *Блок AC Brzina /Мрежа 3/2*

Мрежа четири нам налази ПИ управљање, и опет имамо сличан проблем као и пре. Пошто ми управљање водимо на излаз директно, који одређује фреквенцију асинхроног мотора, уколико би достигли жељену вредност, управљање би пало на нулу и имали би осцилације слично као и код једносмерног мотора. Овај проблем решавамо на исти начин.

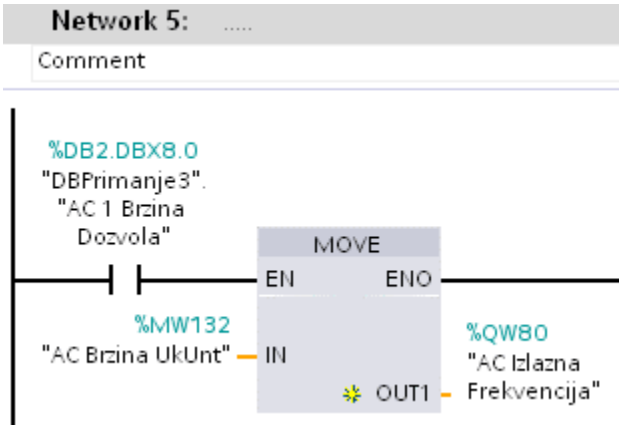
Због истих разлога ограничавамо вредност пре него што променимо врсту меморије у којој је чувамо. У овом случају је та вредност 27645, јер је Сименс сам одредио да та вредност на излазу одговара вредности од 10 . Пошто је то максимум који наш излаз може да пошаље, ограничавамо га на ту вредност. Уколико је вредност већа, ПЛЦ ће пријавити грешку и стати са радом [34-35].



Слика 97. *Блок AC Brzina /Мрежа 4/1*

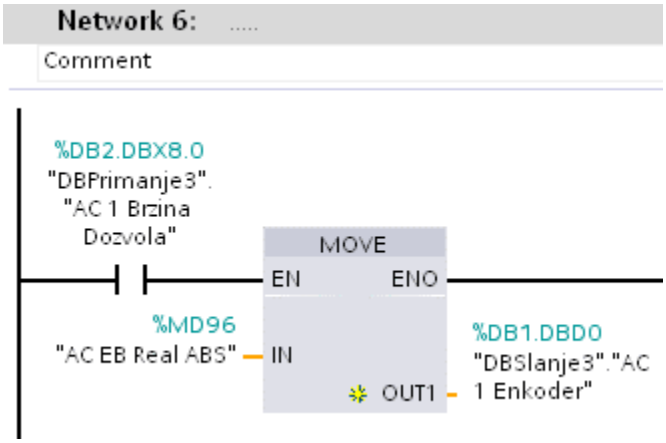
Слика 98. *Блок AC Brzina /Мрежа 4/2*

Мрежа пет нам управљачку вредност шаље на аналогни излаз.



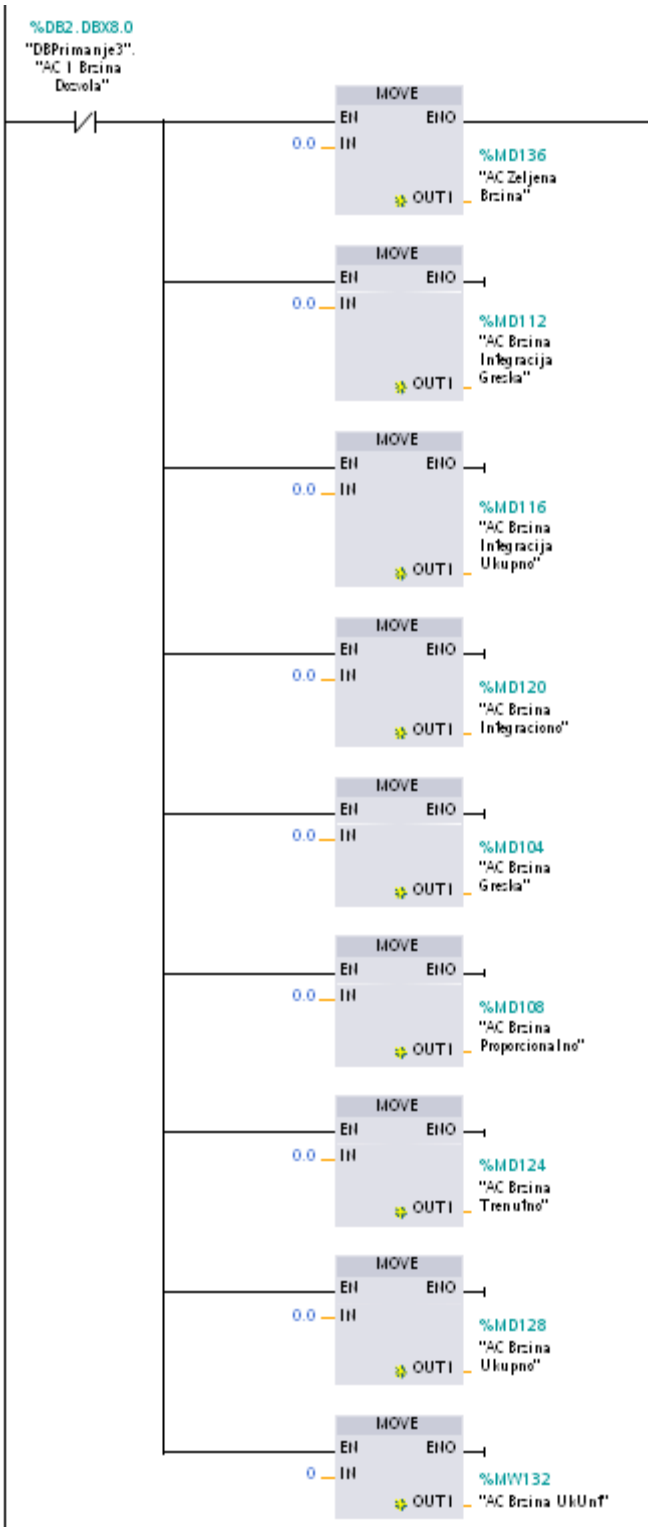
Слика 99. *Блок AC Brzina /Мрежа 5*

Мрежа шест враћа стварну вредност на главни ПЛЦ.



Слика 100. *Блок AC Brzina /Мрежа 6*

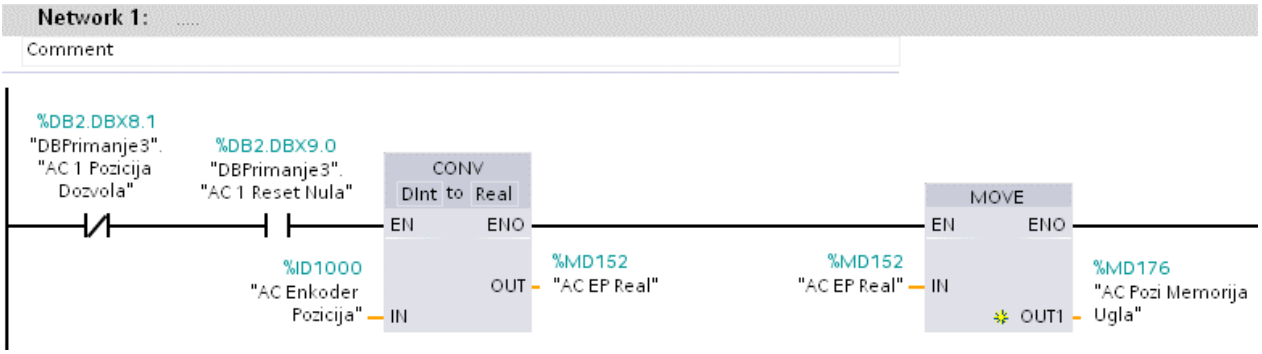
На крају имамо мрежу седам која нам шаље нуле у све промењиве када не користимо управљање брзином.



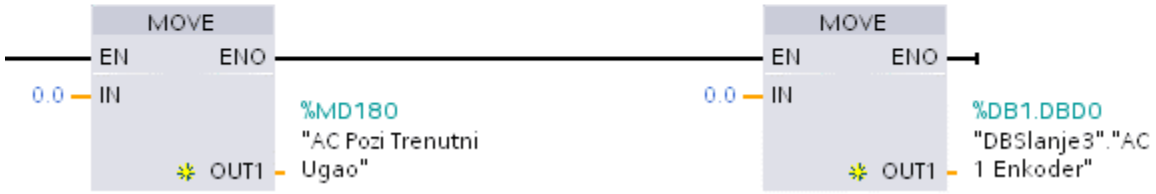
Слика 101. *Блок AC Brzina /Мрежа 7*

1.2.4.6. Блок *"AC Pozicija"*

Мрежа један ради на потпуно исти начин као код једносмерног мотора и ресетује нам почетну позицију на нулу.

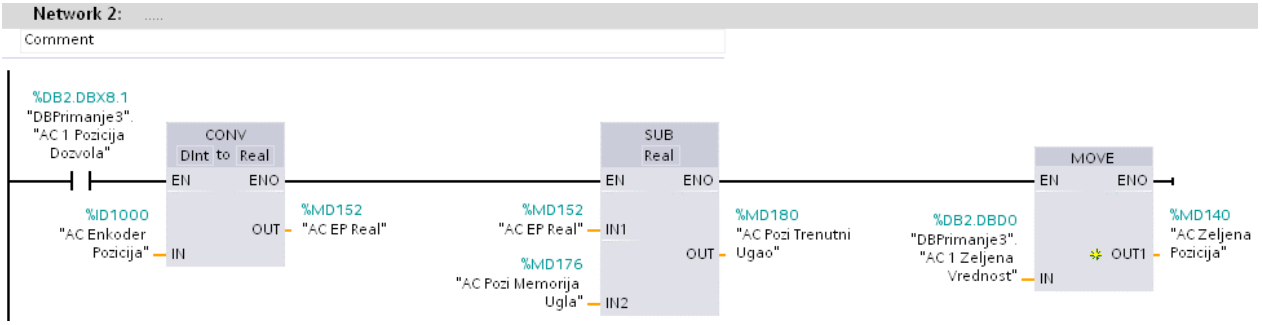


Слика 102. *Блок AC Pozicija /Мрежа 1/1*



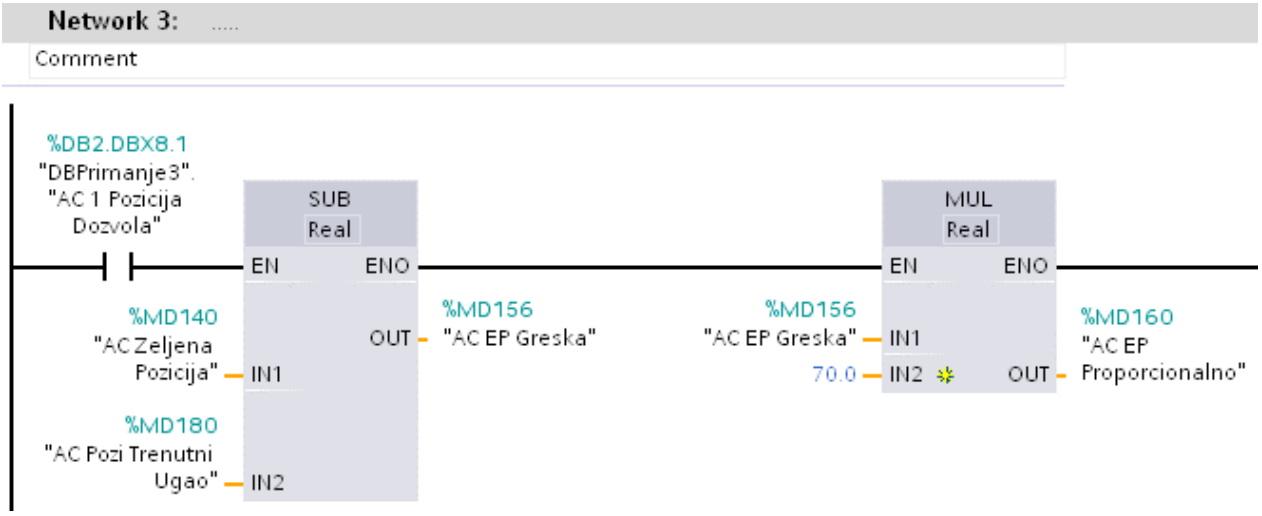
Слика 103. *Блок AC Pozicija /Мрежа 1/2*

Потпуно идентично као и раније, мрежа два нам налази тренутни угао у односу на постављену нулу, и пребацује жељену вредност у локалну промењиву.

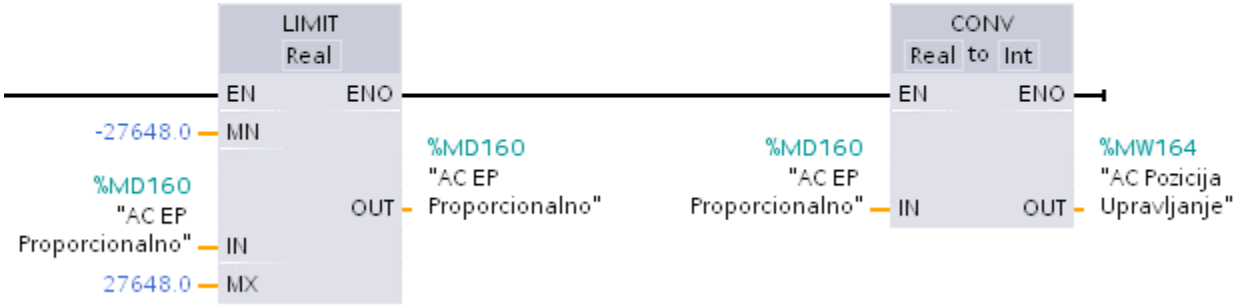


Слика 104. *Блок AC Pozicija /Мрежа 2*

Мрежа три налази грешку, налази П управљање, ограничава максималну вредност због истих разлога као и код контроле брзине асинхроног мотора, и добијену вредност пребацује у сео број.

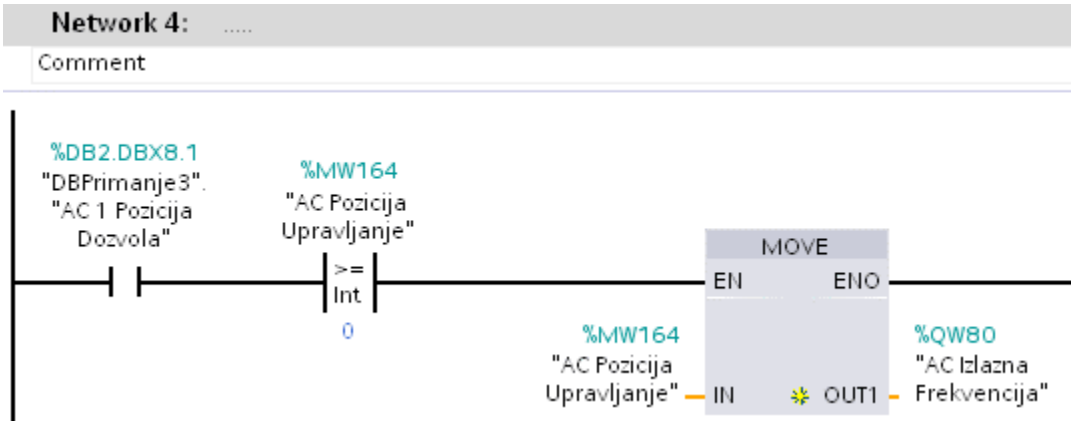


Слика 105. *Блок AC Pozicija /Мрежа 3/1*

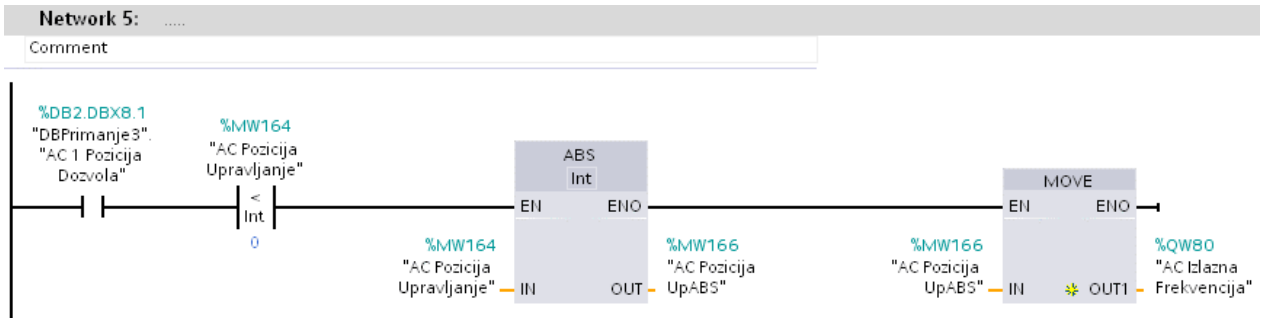


Слика 106. *Блок AC Pozicija /Мрежа 3/2*

Мреже четири и пет проверавају да ли је управљање позитивно или негативно, да би нашао апсолутну вредност управљања, јер смер окретања одређујемо у *Main* блоку. Логика је потпуно иста као и код једносмерног, само што сад вредност не шаљемо на *PWM* већ на аналогни излаз.

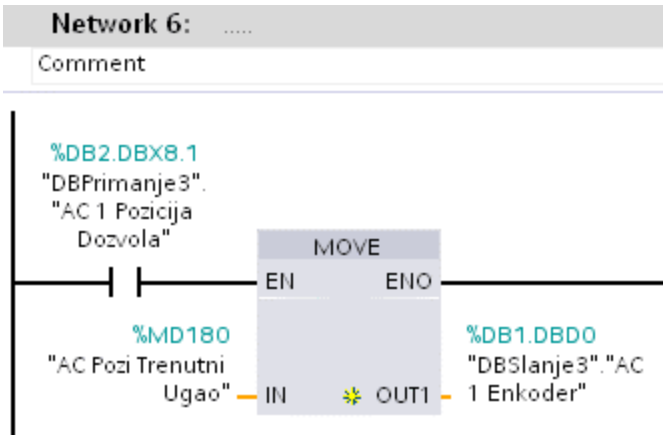


Слика 107. *Блок AC Pozicija /Мрежа 4*



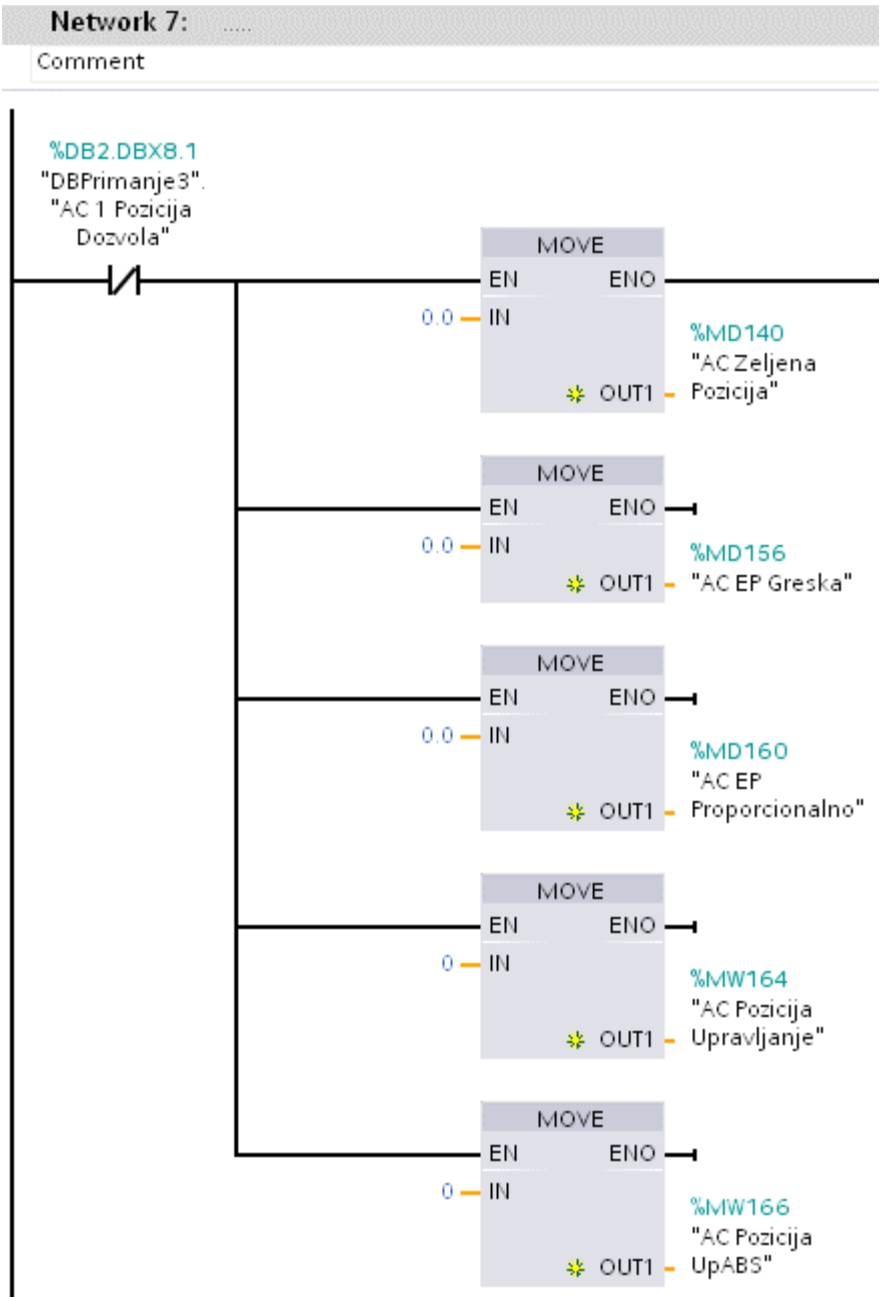
Слика 108. *Блок AC Pozicija /Мрежа 5*

Мрежа шест нам враћа реалну вредност ради приказивања на *HMI*-у.



Слика 109. *Блок AC Pozicija /Мрежа 6*

И на крају нам мрежа седам поништава вредности промењивих уколико не користимо управљање позицијом.



Слика 110. *Блок AC Pozicija /Мрежа 7*

Закључак

У овом раду је су објашњени основни принципи рада ПИД управљања, ПЛЦ контролера и *SCADA* система. Изложене су све компоненте које чине наш систем и објашњени принципи рада асинхроних мотора, енкодера, фреквентних регулатора и једносмерних мотора. За потребе контроле једносмерног мотора је развијена прилагодна електроника која нам синусни сигнал енкодера претвара у квадратасти сигнал који управљачки уређај може да користи, направљене су штампане плоче, и цео систем је упакован у једну кутију, заједно са једносмерним мотором. На крају је дато објашњење како су сви они заједно повезани у једну целину коју би смо ми назвали *SCADA* систем.

Даље је у раду објашњен развој *SCADA* апликације у *TIA Portal*-у, како главни ПЛЦ комуницира са *HMI*-ем, како ПЛЦ-ову међусобно размењују податке како би остварили цео систем, како ради програм који одређује дозволе рада, смера окретања мотора и вршења брзинског или позиционог управљања, на основу одабира корисника система, што је у ствари и главна сврха *SCADA* система. Даље је објашњено како даљински ПЛЦ-ови, задужени за управљање моторима, то управљање и врше на основу дозвола рада, жељених вредности корисника и вредности добијених са енкодера мотора. Сваки од два управљачка ПЛЦ-а може у исто време да управља са по једним асинхроним и једним једносмерним мотором, чији режими рада су независни.

Цео овај систем представља основу на коју је касније могуће надоградити и развити системе са специјалном наменом. На пример, могуће је узети наша два синхрона мотора и направити да они описују један квадрат. За то би било потребно направити механички систем који ће ротационо кретање мотора да претвори у транслаторно, и да се одреди преносни однос, односно колико померање окретање на мотору од један степен прави на нашем транслаторном делу механизма. После је на главном ПЛЦ-у потребно одрадити блок где он, на основу унетих координата корисника и преносног односа, рачуна за колико степени би морали да се окрену мотори, и те вредности шаље на наше блокове за управљање позицијом. Даље је потребно извршити мање измене што се тиче дозвола рада, како би сам програм могао да активира управљање положајем кад је оно неопходно, независно од корисника, као и додати још једну страницу у коју би корисник уносио жељене координате. Овакав систем не би само могао да опише квадрат, него би могао да доведе жељени механизам у било коју тачку тог квадрата. Касније би систем могао да се прошири да се направи аутоматска бушилица, или да систем може да потпише име. Ово су све идеје које су ван теме овог рада, али представљају путоказ за даљи развој овог система.

Литература

[1] Матијевић, М., Јакуповић, Г., Цар, Ј.: Рачунарски подржано мерење и управљање, Машински факултет у Крагујевцу, друго издање, Крагујевац, Септембар, 2008.

[2] DEWESoft - PID Control, Интернет страница: https://www.dewesoft.com/pro/course/pid-control-53, приступљено 16.09.2017.

[3] Electronics Hub - SCADA System, Интернет страница: http://www.electronicshub.org/scada-system/, приступљено 16.09.2017.

[4] Amplicon - What is SCADA?, Интернет страница: https://www.amplicon.com/Process-Control/scada.cfm, приступљено 16.09.2017.

[5] Direct Industry, Интернет страница: http://www.directindustry.com/prod/mitsubishi-automation/product-7025-589566.html, приступљено 16.09.2017.

[6] AutomationWorld - Understanding Ethernet Speed and Determinism, Интернет страница: https://www.automationworld.com/article/technologies/networking-connectivity/ethernet-tcp-ip/understanding-ethernet-speed-and, приступљено 16.09.2017.

[7] Siemens - The S7-1200 Basic Controller - All in one!, Интернет страница: w3.siemens.com/mcms/programmable-logic-controller/en/basic-controller/s7-1200/cpu/Pages/Default.aspx, приступљено 16.09.2017.

[8] Siemens 6ES7214-1AE30-0XB0, CPU 1214C, DC/DC/DC, 14DI/10DO/2AI, Интернет страница: https://support.industry.siemens.com/cs/pd/481697?pdti=td&pnid=13683&lc=en-WW, приступљено 16.09.2017.

[9] RS - Siemens S7-1200 PLC CPU, Ethernet Networking Profinet Interface, 100 kB Program Capacity, Интернет страница: http://sg.rs-online.com/web/p/plc-cpus/8624483/, приступљено 16.09.2017.

[10] Automatyka - HMI_KTP400_KTP600_KTP1000_TP1500.pdf, Интернет страница: https://www.automatyka.siemens.pl/docs/docs_ia/HMI_KTP400_KTP600_KTP1000_TP1500.pdf ХМИ, приступљено 16.09.2017.

[11] Dostlar Otomasyon - 6AV6647-0AD11-3AX0 Siemens Simatic HMI KTP600 Basic Color PN, Интернет страница: http://www.dostlarelektrik.com/6av6647-0ad11-3ax0-siemens-simatic-hmi-ktp600-basic-color-pn.html, приступљено 16.09.2017.

[12] Виша техничка школа струковних студија у Новом Саду - Асинхроне Машине, Интернет страница: https://www.google.rs/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjRwfCbgpXSAhXDA5oKHV-VA0QQFggaMAA&url=http%3A%2F%2Fvtsns.edu.rs%2Fwp-content%2Fuploads%2F2013%2F10%2FEMP1-zbirkaasinhrone-

masine.doc&usg=AFQjCNFwgJCqdcqy13iSaO4q1gtzHl80bg&sig2=OY6Y7xrnILNT90udjYvBnw, приступљено 16.09.2017.

[13] Chiaravalli group - Motori_Elettrici.pdf, Интернет страница: http://old.chiaravalli.com/pol/pdf_motori/MOTORI_ELETTRICI.pdf, приступљено 16.09.2017.

[14] NxtMarket - Chiaravalli CHT71C4, Интернет страница: http://nxtmarket.info/item/545060216836, приступљено 16.09.2017.

[15] Machine Design- basic of Rotary Encodes: Overview and New Technologies, Интернет страница: http://www.machinedesign.com/sensors/basics-rotary-encoders-overview-and-new-technologies-0, приступљено 16.09.2017.

[16] HQHeating - S4.3.pdf, Интернет страница: http://www.hqheating.com.my/files/S4.3.pdf, приступљено 16.09.2017.

[17] Sion Electronics - HTR-W-360-2-PP, Интернет страница: http://sion.rs/index.php?route=product/product&path=33&product_id=112, приступљено 16.09.2017.

[18] Gozuk - What is Frequency Converter? How it works? Интернет страница: http://www.frequencyinverter.org/what-is-frequency-converter-how-it-works-631601.html, приступљено 16.09.2017.

[19] SolarPro - How Inverers Work, Интернет страница: http://solarprofessional.com/articles/products-equipment/inverters/how-inverters-work, приступљено 16.09.2017.

[20] Octopart - Datasheets, Electronic Parts, Components - Altivar_61_ATV61H075N4.pdf, Интернет страница: http://datasheet.octopart.com/ATV61H075N4-Schneider-Electric-datasheet-14410185.pdf, приступљено 16.09.2017.

[21] Specialized Electronics - Telemecanique Altivar 61 ATV61H075N4 ATV 61 H075N4, Интернет страница: http://www.specializedelectronics.com/telemecanique-altivar-61-atv61h075n4-atv61-h075n4/, приступљено 16.09.2017.

[22] Systemair - Catalogue 2 - FileHandler.axd, Интернет страница: http://catalogue2.systemair.com/FileHandler.axd?hash=nKL85a1ctzaPE73i8ogRbA!! , приступљено 16.09.2017.

[23] eBay - Danfoss - HVAC Frequenzumrichter - FC-102 - 1,1kW, Интернет страница: http://www.ebay.ie/itm/DANFOSS-HVAC-Frequenzumrichter-FC-102-1-1kW-/141239033074, приступљено 16.09.2017.

[24] Automatika - DC Motori, Интернет страница: https://www.automatika.rs/baza-znanja/mehatronika/dc-elektromotori.html, приступљено 16.09.2017.

[25] Рачунарско управљање мотора једносмерне струје *Escap 28D2R-219P* - ОПИС ЛАБОРАТОРИЈСКОГ МОДЕЛА И ЊЕГОВЕ УПОТРЕБЕ, Индустријски рачунарски системи, Moodle portal, Интернет страница: moodle.mfkg.rs/course/view.php?id=228, приступљено 16.09.2017.

[26] Левит, Н. Б., Подгайный, В. К.: Основи електротехнике - аутоматика, Државни секретаријат за народну одбрану, 1969, штампано у војној штампарији Сплит.

[27] Automatika - Izrada štampanih pločica, Интернет страница: https://www.automatika.rs/baza-znanja/tutorijali/izrada-stampanih-plocica.html, приступљено 16.09.2017.

[28] Луковић, В.: Програмирање и примена ПЛЦ контролера Siemens SIMATIC S7-1200, Дипломски рад, Машински факултет у Крагујевцу, Универзитет у Крагујевцу, Крагујевац, 2010.

[29] Милановић, М.: Домаћи задатак - *SCADA* систем управљања *AC* и *DC* мотором, , Индустријски рачунарски системи, Moodle portal, Интернет страница: moodle.mfkg.rs/course/view.php?id=228, приступљено 16.09.2017.

[30] Khamel, K., Khamel, E.: Programmable Logic Controllers - Industrial Control, McGraw-Hill Education, 2014, e-Book Cenveo Publisher Services, Version 1.0.

[31] Siemens - Formum - Sample program: S7 communication GET and PUT instructions, Интернет страница: https://support.industry.siemens.com/tf/WW/en/posts/sample-program-s7-communication-get-and-put-instructions/68281?page=0&pageSize=10, приступљено 16.09.2017.

[32] Siemens - How do you program "GET" and "PUT" instructions in the user program of the SIMATIC S7-1200 CPU in order to transfer more than 160 bytes of data?, Интернет страница: https://support.industry.siemens.com/cs/document/65975617/how-do-you-program-the-get-and-put-instructions-in-the-user-program-of-the-simatic-s7-1200-cpu-in-order-to-transfer-more-than-160-bytes-of-data-?dti=0&lc=en-WW, приступљено 16.09.2017.

[33] Siemens - Forum - need help in size of word and integer, Интернет страница: https://support.industry.siemens.com/tf/WW/en/posts/need-help-in-size-of-word-and-integer/72899?page=0&pageSize=10, приступљено 16.09.2017.

[34] Siemens - Forum - How 27648 Analaog maximum Value?, Интернет страница: https://support.industry.siemens.com/tf/WW/en/posts/how-27648-analaog-maximum-value/26057?page=0&pageSize=10, приступљено 16.09.2017.

[35] Siemens - For an S7-1200/S7-1500 controller in STEP 7 (TIA Portal), how do you scale integer values in real numbers and vice versa for analog inputs and outputs?, Интернет страница: https://support.industry.siemens.com/cs/document/39334504/for-an-s7-1200-s7-1500-controller-in-step-7-(tia-portal)-how-do-you-scale-integer-values-in-real-numbers-and-vice-versa-for-analog-inputs-and-outputs-?dti=0&lc=en-WW, приступљено 16.09.2017.

Прилог А

Табела 2. *Спецификације једносмерног мотора [25]*

Величине	Вредност	Јединице
Максимални напон побуде мотора (-12 V ≤ $u(t)$ ≤ 12 V)	12	V
Максимална брзина неоптерећеног мотора	5800	o/min
Момент закочења (момент оптерећења под којим мотор не може да оствари било какву брзину)	94	mNm
Јачина струје неоптерећеног мотора (јачина струје у ротору при максималној брзини обртања неоптерећеног мотора)	44	mA
Стартни напон (минимални напон који је потребно довести да би се мотор покренуо –да би савладао Кулоново трење – величина мртве зоне)	0.15	V
Максимална континуална струја (максимално радно оптерећење струје ротора је 1.5 A)	1.5	A
Максимални континуални момент (максимално радно оптерећење вратила мотора према статичкој карактеристици произвођача, Сл.2.1)	28.4	mNm
Максимално угаоно убрзање	48	$10^3 rad/s^2$
Електрична константа (K_{me} = 0.0196 V/(rad/s)=0.0196 Vs)	2.05	V/(1000o/min)
Механичка константа (K_{em}=0.0195 Nm/A)	19.5	mNm/A
Отпорност арматуре (R= 2.5 Ω)	2.5	Ω
Индуктивност ротора (L=0.0003 H)	0.30	mH
Момент инерције ротора (J= 17.60· 10^{-7}kgm^2)	17.60	10^{-7}kgm^2
Механичка временска константа (T_s =0.012 s)	12	ms
Коефицијент вискозног трења (β= 1.41264797e-6Nm/(rad/s))	1.41264797e-6	Nm/(rad/s)
Број импулса по обртају енкодера Резолуција инкременталног енкодера је дакле 2.5 степена. Ради се о два сигнала која генеришу 144 импулса по обртају, а фазно су померена за 90 степени ради утврђивања смера обртања.	144	
Напон напајања енкодера	5	V
Пин +/- (прикључци за напајање мотора, и истовремено за управљање мотора напонским сигналом)	12VDC Motor	
Пин 1 (прикључак за емитер транзистора 1 унутар енкодера)	Емитер 1	
Пин 2 (прикључак за колектор транзистора 1 унутар енкодера)	Колектор 1	
Пин 3 (прикључак за катоду ЛЕ диоде унутар енкодера)	Катода	
Пин 4 (прикључак за аноду ЛЕ диоде унутар енкодера)	Анода	
Пин 5 (прикључак за емитер транзистора 2 унутар енкодера)	Колектор 2	
Пин 6 (прикључак за колектор транзистора 2 унутар енкодера)	Емитер 2	

Прилог Б

Zavrsio	Bool	%M12.0	DC 2 Zeljena Vrednost	Real	%MD84
Greska	Bool	%M12.1	DC 2 Dobijena Vrednost	Real	%MD88
Status	Word	%MW20	DC 1 Napred	Bool	%M72.0
Salji	Bool	%M24.0	DC 1 Nazad	Bool	%M72.1
Gornjalvica1	Bool	%M30.0	DC 1 Dozvola Napred	Bool	%M72.2
MasterProvera	Bool	%Q0.0	DC 1 Dozvola Nazad	Bool	%M72.3
Zavrsio2	Bool	%M34.0	DC 1 Brzina	Bool	%M72.4
Greska2	Bool	%M35.0	DC 1 Pozicija	Bool	%M72.5
Salji2	Bool	%M36.0	DC 1 Dozvola Brzina	Bool	%M72.6
Gornjalvica2	Bool	%M37.0	DC 1 Dozvola Pozicija	Bool	%M72.7
Status2	Word	%MW38	AC 2 Napred	Bool	%M82.0
STOP	Bool	%M52.0	AC 2 Nazad	Bool	%M82.1
Stop DC 1	Bool	%M52.1	AC 2 Dozvola Napred	Bool	%M82.2
Stop AC 1	Bool	%M52.2	AC 2 Dozvola Nazad	Bool	%M82.3
Stop DC 2	Bool	%M52.3	AC 2 Brzina	Bool	%M82.4
Stop AC 2	Bool	%M52.4	AC 2 Pozicija	Bool	%M82.5
AC 1 Napred	Bool	%M53.0	AC 2 Dozvola Brzina	Bool	%M82.6
AC 1 Nazad	Bool	%M53.1	AC 2 Dozvola Pozicija	Bool	%M82.7
AC 1 Dozvola Napred	Bool	%M53.2	DC 2 Napred	Bool	%M92.0
AC 1 Dozvola Nazad	Bool	%M53.3	DC 2 Nazad	Bool	%M92.1
AC 1 Brzina	Bool	%M53.4	DC 2 Dozvola Napred	Bool	%M92.2
AC 1 Pozicija	Bool	%M53.5	DC 2 Dozvola Nazad	Bool	%M92.3
AC 1 Dozvola Brzina	Bool	%M53.6	DC 2 Brzina	Bool	%M92.4
AC 1 Dozvola Pozicija	Bool	%M53.7	DC 2 Pozicija	Bool	%M92.5
AC 1 Zeljena Vrednost	Real	%MD54	DC 2 Dozvola Brzina	Bool	%M92.6
AC 1 Dobijena Vrednost	Real	%MD59	DC 2 Dozvola Pozicija	Bool	%M92.7
DC 1 Zeljena Vrednost	Real	%MD64	AC 1 Pozicija Nula	Bool	%M93.0
DC 1 Dobijena Vrednost	Real	%MD68	AC 2 Pozicija Nula	Bool	%M93.1
AC 2 Zeljena Vrednost	Real	%MD74	DC 1 Pozicija Nula	Bool	%M93.2
AC 2 Dobijena Vrednost	Real	%MD78	DC 2 Pozicija Nula	Bool	%M93.3

Слике 111-112. *Tagtable главног ПЛЦ-а*

Zavrsio1	Bool	%M0.0
Greska1	Bool	%M0.1
Status1	Word	%MW4
Tag_1	Bool	%M0.3
Tag_2	Bool	%M0.4
AC Enkoder Pozicija	DInt	%ID1000
AC Enkoder Brzina	DInt	%ID1004
DC Enkoder Pozicija	DInt	%ID1008
DC Enkoder Brzina	DInt	%ID1012
DC EB Real	Real	%MD20
DC EB Real RPS	Real	%MD24
DC EB Real RPM	Real	%MD28
DC Brzina Greska	Real	%MD32
DC EB Real RPS ABS	Real	%MD36
DC Brzina Proporcionalno	Real	%MD40
DC Integracija Greska	Real	%MD44
DC Integracija Ukupna	Real	%MD48
DC Brzina Integraciono	Real	%MD52
DC Brzina Trenutno	Real	%MD56
DC Brzina Ukupno	Real	%MD60
DC Brzina UkInt	Int	%MW64
PWM Izlaz 1	Int	%QW1000
PWM Izlaz 2	Int	%QW1002
DC EP Real	Real	%MD68
DC EP Stepeni	Real	%MD72
DC Pozicija Greska	Real	%MD76
DC Pozicija Proporcionalno	Real	%MD80
DC Pozicija Upravljanje	Int	%MW84
DC Pozicija UpABS	Int	%MW88

AC EB Real	Real	%MD92
AC EB Real ABS	Real	%MD96
AC EB Real RPM	Real	%MD100
AC Brzina Greska	Real	%MD104
AC Brzina Proporcionalno	Real	%MD108
AC Brzina Integracija Greska	Real	%MD112
AC Brzina Integracija Ukupno	Real	%MD116
AC Brzina Integraciono	Real	%MD120
AC Brzina Trenutno	Real	%MD124
AC Brzina Ukupno	Real	%MD128
AC Brzina UkUnt	Int	%MW132
AC Izlazna Frekvencija	Int	%QW80
AC Zeljena Brzina	Real	%MD136
AC Zeljena Pozicija	Real	%MD140
DC Zeljena Brzina	Real	%MD144
DC Zeljena Pozicija	Real	%MD148
AC EP Real	Real	%MD152
AC EP Greska	Real	%MD156
AC EP Proporcionalno	Real	%MD160
AC Pozicija Upravljanje	Int	%MW164
AC Pozicija UpABS	Int	%MW166
DC Pozi Memorija Ugla	Real	%MD168
DC Pozi Trenutni Ugao	Real	%MD172
AC Pozi Memorija Ugla	Real	%MD176
AC Pozi Trenutni Ugao	Real	%MD180
AC Nazad	Bool	%Q1.1
AC Napred	Bool	%Q1.0

Tag_1	Bool	%M0.0
Tag_2	Bool	%M1.0
Tag_3	Bool	%M2.0
Tag_4	Bool	%M3.0
Tag_5	Word	%MW5
AC Enkoder Pozicija	DInt	%ID1000
AC Enkoder Brzina	DInt	%ID1004
DC Enkoder Pozicija	DInt	%ID1008
DC Enkoder Brzina	DInt	%ID1012
DC EB Real	Real	%MD20
DC EB Real RPS	Real	%MD24
DC EB Real RPM	Real	%MD28
DC Brzina Greska	Real	%MD32
DC EB Real RPS ABS	Real	%MD36
DC Brzina Proporcionalno	Real	%MD40
DC Integracija Greska	Real	%MD44
DC Integracija Ukupna	Real	%MD48
DC Brzina Integraciono	Real	%MD52
DC Brzina Trenutno	Real	%MD56
DC Brzina Ukupno	Real	%MD60
DC Brzina UkInt	Int	%MW64
PWM Izlaz 1	Int	%QW1000
PWM Izlaz 2	Int	%QW1002
DC EP Real	Real	%MD68
DC EP Stepeni	Real	%MD72
DC Pozicija Greska	Real	%MD76
DC Pozicija Proporcionalno	Real	%MD80
DC Pozicija Upravljanje	Int	%MW84
DC Pozicija UpABS	Int	%MW88

AC EB Real	Real	%MD92
AC EB Real ABS	Real	%MD96
AC EB Real RPM	Real	%MD100
AC Brzina Greska	Real	%MD104
AC Brzina Proporcionalno	Real	%MD108
AC Brzina Integracija Greska	Real	%MD112
AC Brzina Integracija Ukupno	Real	%MD116
AC Brzina Integraciono	Real	%MD120
AC Brzina Trenutno	Real	%MD124
AC Brzina Ukupno	Real	%MD128
AC Brzina UkUnt	Int	%MW132
AC Izlazna Frekvencija	Int	%QW80
AC Zeljena Brzina	Real	%MD136
AC Zeljena Pozicija	Real	%MD140
DC Zeljena Brzina	Real	%MD144
DC Zeljena Pozicija	Real	%MD148
AC EP Real	Real	%MD152
AC EP Greska	Real	%MD156
AC EP Proporcionalno	Real	%MD160
AC Pozicija Upravljanje	Int	%MW164
AC Pozicija UpABS	Int	%MW166
DC Pozi Memorija Ugla	Real	%MD168
DC Pozi Trenutni Ugao	Real	%MD172
AC Pozi Memorija Ugla	Real	%MD176
AC Pozi Trenutni Ugao	Real	%MD180
AC Nazad	Bool	%Q1.1
AC Napred	Bool	%Q1.0

Слике 113-116. *Tagtable управљачких ПЛЦ-ова*