Универзитет у Београду
Електротехнички факултет

МАСТЕР РАД

Развој планарног рехабилитационог робота – програм за управљање планарним рехабилитационим роботом у повратној спрези

Кандидат:
Милош Радуловић

Ментор:
Проф. др. Дејан Б. Поповић

Београд, 2012.

ЗАХВАЛНИЦА

Желео бих да се захвалим професору Дејану Б. Поповићу за помоћ и подршку током развоја овог рада.

Такође бих желео да се захвалим Милошу Костићу за несебичну помоћ и корисне савете пружене током рада на развоју овог мастер рада.

Овај рад је производ истраживачких активности групе за Биомедицинско инжењерство и технологије на Електротехничком факултету Универзитета у Београду

Овај рад је подржан од стране FP7 EC *strep* пројекта HUMOUR, No. 231724 и пројекта број 175016, финансираног од стране Министарства за Науку и Технолошки Развој, Републике Србије, Београд и Швајцарског Националног Фонда, Берн (*Swiss National Foundation, Berne*) пројекат *InRES*

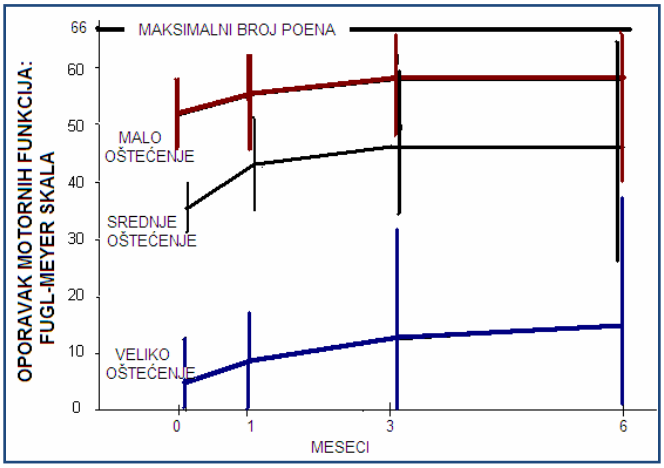
САДРЖАЈ

УВОД

Снажан технолошки развој у претходних двадесет година унео је драстичне промене у свакодневни живот људи у читавом свету. Све бржи развој нових технологија доноси и бржи начин живота који све чешће обилује стресним ситуацијама. Повећање стреса уз све неправилније животне навике има за последицу велико повећање разних обољења, а поготово обољења централног нервног система узрокована можданим ударом.

Статистике организације *American Heart Association*, у извештају за 2011. годину, *Heart Disease and Stroke Statistics – 2011 Update*, говоре да сваке године у Америци 795.000 особа доживи мождани удар, од чега је чак 610.000 оних којима је то први мождани удар, а њих 144.000 умре, што ову болест чини трећим узроком смрти становништва. Око 5.7 милиона Американаца који су преживели мождани удар живе са веома тешким последицама. Према статистикама Канадске Heart&Stroke фондације од 100 људи који доживе мождани удар, 15 има фаталан исход, 10 се потпуно опорави, 25 остане са веома благим последицама, 40 са средњим нивоом, док преосталих 10 са веома високим степеном оштећења моторних функција.

Последице можданог удара се могу поделити у три степена оштећења: велико, мало и средње оштећење, по критеријуму процентуалног оштећења моторних функција [1]. Као што се са слике 1.1 види, захваљујући пластицитету нервног система, у случају великог оштећења долази само до опоравка око 15%, у случају средњег оштећења око 45%, док у случају малог оштећења нешто мање од 60% моторних функција. Оно што је заједничко за сва три јесте да се углавном максималан степен опоравка постиже после око три месеца, након чега су доста смањене индиције за даљи опоравак.



Слика 1: Степен опоравка након можданог удара (преузето из скрипти проф. Д.Б.Поповића са предмета Неурално инжењерство)

Када је реч о блажем степену оштећења, тада у највећем броју долази до смањене функционалности одређених делова тела, од којих се најчешће издвајају отежано кретање и манипулација рукама. Особе са таквим последицама неопходно је подвргнути интензивним терапијама (извођење специјалних вежби и функционалној електричној стимулацији - ФЕТ), ради успостављања функционалности оштећење функција, а све то у циљу независног функционисања у условима свакодневног живота.

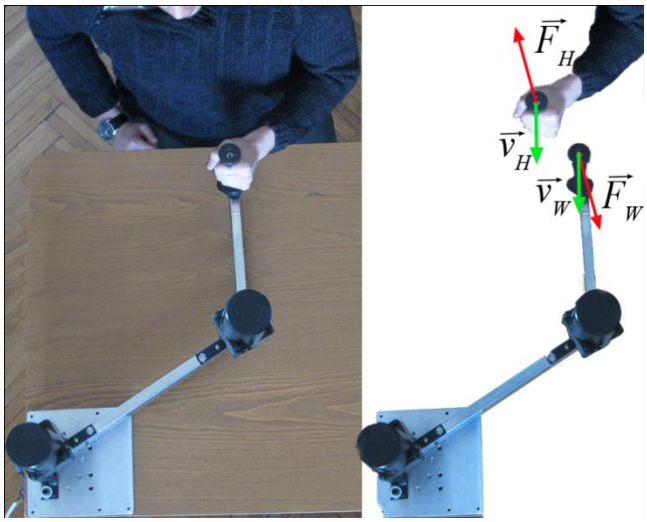
Висок проценат људи који преживе мождани удар имају као последицу оштећену моторну контролу и принуђени су да поново уче како да користе паретичне удове. Учење моторних вештина постиже се понављањем покрета са различитим облицима повратне информације, укључујући визуелне и проприоцептивне информације [2]. Последњи резултати у рехабилитацији горњих екстремитета код пацијената који су преживели мождани удар указују да интензивно вежбање уз асистенцију значајно унапређује овај процес [3-6]. Такође, показано је да вежбе које укључују континуално понављање покрета везаних за конкретан задатак доприносе дуготрајној реорганизацији кортекса која је карактеристична за области мозга које се користе за тај специфични задатак [7]. С обзиром да и након рехабилитације пацијенти настављају да користе функционалне покрете и тиме утврђују новостечене моторне шеме, овакав приступ рехабилитацији дугорочно даје потенцијално боље резултате.

Како би успешно обављали задатак током оваквих вежби пацијентима је неопходна асистенција. Уобичајено, пацијентима помажу терапеути, што изискује много њиховог времена и физички је напорно. Рехабилитациони роботи развијени су са циљем да помогну терапеутма у пружању подршке и асистенције пацијентима [8]. Рехабилитациони роботи су, уз прецизне механичке и кинетичке информације, у стању да пацијентима пруже високу дозу адаптивних вежби током терапије [9] растерећујући терапеуте од напорних сесија са пацијентима. Такође омогућавају да се у рехабилитационе вежбе имплементирају различите кинематичке шеме и закони покрета, прилагођени индивидуалним потребама пацијента.

Овакви роботски системи уведени су пре десетак година [10,11], а у последње време појављују се и као комерцијални производи за рехабилитацију, као на пример *InMotion Arm Robot* [12] и *Armeo* Power [13]. Овај вид терапије је у последње време врло актуелан, јер је увиђено да рехабилитациони роботи побољшавају покретљивост пацијената и дају додатан импулс њиховом опоравку. Уз правилну представу покрета, рехабилитациони роботи могли би да помажу процес терапије уз минималан надзор.

Један тип робота који се примењују у рехабилитацији су планарни манипуландуми. који асистирају при покрету тако што наводе шаку пацијента у једној равни. Овакви роботи имају изглед који подсећа на руку човека. Њихова рука је крута у Z оси, а покретљива је у

XY равни. Они асистирају пацијенту тако што наводе шаку пацијента током покрета у једној равни. Пример планарних манипуландума су системи Braccio di Ferro [11] и InMotion Arm Robot [12]. Највећи потенцијал оваквих роботских система као рехабилитационих уређаја је управо у рехабилитацији покрета дохватања заснованих на извршавању конкретног задатка.



Слика 2: Пацијент са планарним манипуландумом и расчлањени систем

Широко прихваћена метода за управљање оваквим системима је управљање импедансом. При управљању импедансом,осим једноставне контроле покрета манипулатора врши се и додатно управљање при девијацијама од референтног покрета. Овде контролер настоји да имплементира динамичку зависност између варијабли манипулатора као што су позиција и сила. У методи управљања импедансом робот се понаша као импеданса, а човек као адмитанса, што значи да робот производи силу према измереном покрету човека.

У [14] предложен је оригиналан метод за управљање асистенцијом робота, користећи представу покрета карактеристичну за покрете здраве руке. Према овој методи, планарни манипуландум не намеће „идеалну“ стратегију покрета, већ дозвољава пацијенту да користи њему најприроднију стратегију, коју робот временом поправља. Овај вид асистенције назива се „нежном“ асистенцијом. Управљање подразумева да, током вежбе, рука треба да прати референтну трајекторију, каква је код покрета здраве особе, а робот треба да пружи минималну подршку како би олакшао покрет. Метода за добијање репрезентације кинематике покрета, коришћена за управљање планарним манипуландумом, развијена је у Лабораторији за Биомедицинску инструментацију и технологије, Електротехничког факултета у Београду. Ова метода се заснива на стохастичком моделу покрета у коме је садржана природна варијабилност здравог покрета. Детаљан приказ ове методе дат је у [15]. Једно од могућих решења за избор трајекторије потребне за управљање роботом који асистира током рехабилитације манипулисања руком

представљено је у [16], где се предлаже да покрети здравих особа треба да формирају основу за представе покрета које погодују у процесу моторног учења.
Циљ овог рада је реализација софтверског програма за управљање планарним рехабилитационим роботом на основу методе „нежног“ управљања предложене у [14], при чему ће се покрети здраве особе добијени користе као референтни покрети, а представа варијабилности динамике покрета добијена је методом изложеном у [15].

Рад је организован у седам целина. У првом делу наведени су елементи аквизиционог система и дате су њихове карактеристике, након чега је описан комуникациони конвертор који служи за повезивање инвертора са рачунаром. У трећем поглављу рада дат је опис најбитнијих карактеристика ModBus комуникацијског протокола којим се одвија комуникација рачунара и инвертора. Детаљан опис начина рада инвертора и потребних подешавања приказан је у четвртом поглављу. Пето поглавље рада бави се описом реализованог софтверског програма. Приказан је и објашњен кориснички интерфејс, и дат је детаљан преглед функционалности главног програма и свих битнијих подпрограма. Закључци овог рада дати су упоследњем поглављу.

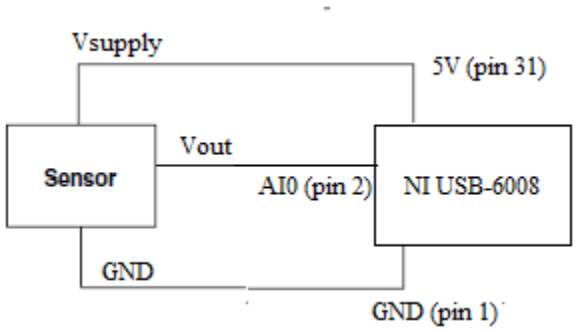
СИСТЕМ ЗА АКВИЗИЦИЈУ

Систем за аквизицију сензорских сигнала први је део управљачког система, за који је развијена апликација која је предмет овог рада. Он треба да обезбеди апликацији информацију о тачном положају ручке планарног манипуландума, како би се на основу те информације вршило управљање моторима система. Овај систем састоји се од сонзора положаја, аквизиционе картице и аквизиционе петље апликације.

Сензор

Како је потребно обезбедити управљање у затвореној спрези, неопходно је омогућити контролеру, у овом случају реализованом софтверском програму, информацију о тренутној позицији осовине мотора у сваком тренутку. У ту сврху је на осовине оба мотора постављен по један сензор угаоног положаја произвођача *Vishay* модел *351HE*. Овај сензор ради на принципу безконтактног потенциометра са Холовим претварачем. Ови сензори дозвољавају један пун покрет од 360°, што је за овај систем и више него довољно. Сензор захтева напајање од 5V, а даје излазни сигнал у опсегу 10-90% улазног сигнала, тј. 0.5-4.5V.

Излазни сигнал са сензора дововоди се на аналогни улазни канал аквизиционе картице. Шема повезивања сензора и картице дата је на слици испод.



Слика 3: Шема повезивања сензора положаја и аквизиционе картице

Аквизициона картица

За аквизицију излазних сигнала сензора положаја користи се аквизициона картица произвођача National Instruments, NI USB-6008.

Слика 4: Сензор положаја и аквизициона картица

Картица има осам аналогних улазних канала чији опсег улазног сигнала од +/-10V одговара опсегу излазног сигнала сензора од 0.5-4.5V. А/Д конвертор који се налази у картици ради са 12-битном резолуцијом и са максималном учестаношћу одабирања од 10kHz. С обзиром да се у овом случају врши аквизиција на два канала, максимална учестаност одабирања по каналу је 5 kHz. NI USB-6008 картица поседује и стабилан извор 5V напајања који се користи за напајање сензора, а поседује и два додатна аналогна излазна сигнала опсега 0-5V као и дванаест дигиталних улазно-излазних линија који се у овом тренутку не користе, али се у будућности по потреби могу искористити за напајање неких додатних сензора или других елемената система. NI USB-6008 картица се са рачунаром повезује преко USB порта и захтева да на рачунару буду инсталирани NI-DAQmx драјвери.

Софтверски програм

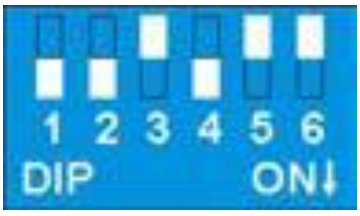
Реализовани софтверски програм врши аквизицију сигнала, њихово процесирање и задавање команди инверторима који управљају моторима. Процесирање сигнала подразумева конверзију напонских вредности у радијане и потом примену одговарајућих тригонометријскх трансформација како би се одредила позиција ручке механичког система у X,Y равни. Помоћу добијених информација о тачној позицији ручке и *look-up* табеле која одговара покрету који се тренутно изводи а која је претходно увезена у апликацију, примењује се алгоритмика која ће омогућити одређивање одговарајућих референци за моменте мотора. На основу добијених вредности формирају се одговарајуће ModBus поруке које се потом шаљу инвертору. Детаљан опис реализованог програма дат је у наредном поглављу.

USB/RS-485 КОМУНИКАЦИОНИ КОНВЕРТОР

Комуникација рачунара са инверторима који управљају моторима обавља се преко ModBus комуникацијског протокола. Сам инвертор подржава ModBus комуникацију путем RS-485 стандарда, а како рачунар који је коришћен при изради софтверског програма, као и већина данашњих рачунара, нема порт који би директно омогућио комуникацију путем RS-485 стандарда, за повезивање рачунара и инвертора коришћен је комуникациони конвертор DC USB/422-485 произвођача DECODE.

Слика 5: USB/RS-485 комуникациони конвертор

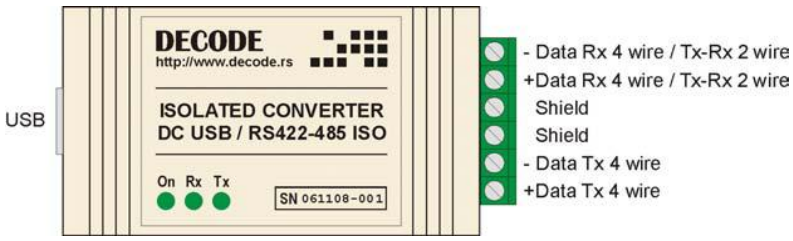
Сам конвертор је једноставан за употребу. Подешавање брзине и формата података врши се аутоматски према захтеву RS-485 уређаја. Модул се напаја из USB порта рачунара и галвански је изолован од RS-485 уређаја. Након прикључивања конвертора на рачунар и инсталације драјвера, конвертору се може приступити као виртуелном серијском COM порту, што омогућава LabVIEW програмском пакету, који је коришћен за развој софтверског програма, да користи NI VISA драјвере за успостављање комуникације. Да би се успешно успоставила RS-485 комуникација са инвертором, потребно је на адекватан начин конфигурисати конвертор. Са бочне стране конвертора налази се шест прекидача чијим се подешавањем дефинише мод рада конвертора. За остваривање RS-485 комуникације потребно је поставити прекидаче у положај као на слици:



Слика 6: Подешавање прекидача за RS-485 комуникацију

На слици испод приказан је распоред прикључака на USB/422-485 конвертору. Како се за RS-485 комуникацију користе две жице, које служе и за предају и за пријем података, потребно је повезати терминале +Data Tx-Rx и -Data Tx-Rx на конвертору са терминалима

SP и SN на инвертору. На конвертору се налазе и три ЛЕД –а за сигнализацију да је конвертор укључен (ON) и да постоји пренос података (TX и RX).



Слика 7: Конектори конвертора

MODBUS ПРОТОКОЛ

Како се комуникација између рачунара и инвертора обавља путем ModBus протокола, у овом делу рада биће дат кратак преглед основних карактеристика овог протокола. Овај преглед даје се како због упознавања са основама самог протокола, тако и због лакшег и бољег разумевања начина на који се инвертор конфигурише за ModBus комуникацију иначина на које се у реализованој апликацији формирају ModBus поруке.

ModBus серијски комуникациони протокол представила је 1979. године компанија Modicom за употребу са њиховим PLC контролерима, а данас је то један од најзаступљенијих стандардних протокола у индустрији. Овај протокол користи мастер/слејв архитектуру, где у истој мрежи може постојати само један мастер и до 247 слејв уређаја. Најчешће је базиран на серијској комуникацији, мада се користи и у варијанти где се као физички слој користи *Ethernet ModBus,* а комуникација се одвија у *half-duplex* моду. У својој мастер/слејв архитектури ModBus користи клијент/сервер комуникацију, где мастер игра у логу клијента, а слејв се понаша као сервер. У ModBus мрежи комуникацију може да иницира само мастер и то слањем два различита типа порука:

1. *unicast* порука – шаље се само једном слејву који потом одговара мастеру на примљену поруку. Како сваки слејв уређај у мрежи мора имате јединствену адресу вредност адресног поља поруке која се шаље тачно одређује слејв уређај којем се шаље порука.
2. *broadcast* порука – порука се шаље свим слејв уређајима у мрежи и на ову поруку слејв уређај не одговара. При слању овог типа поруке вредност адресног поља поруке је 0. Након примања *broadcast* наредбе сви слејв уређаји симултано извршавају примљену наредбу.

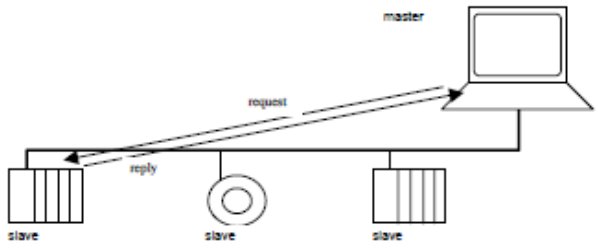


Figure 3: Unicast mode

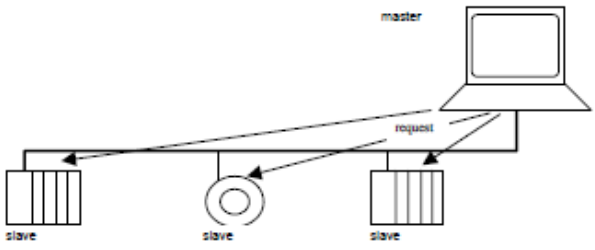
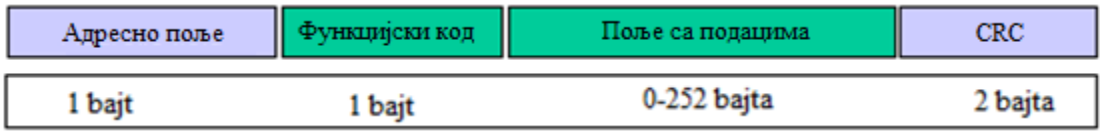


Figure 4: Broadcast mode

Слика 8: *Broadcast* и *unicast* поруке

ModBus протокол дефинише структуру поруке која се шаље и коју уређаји у мрежи могу препознати. Подаци се у ModBus мрежи могу преносити у две варијанте које одређују начин паковања података у поруку и њиховог декодирања – ASCII и RTU. Сви уређаји на

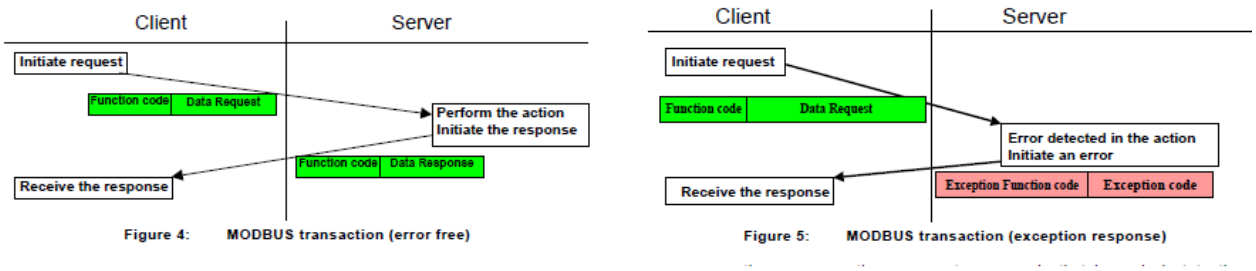
мрежи морају имати подешен исти вид трансмитовања порука. Како инвертор који се користи за управљање мотором подржава RTU мод, биће представљен само формат порука за овај мод преноса. При RTU (*Remote Terminal Unit*) моду сваки бајт састоји се од два хексадецимална четворобитна карактера. Максимална величина једног фрејма износи 256 бајтова, а формат је приказан на слици:



Слика 9: Формат ModBus поруке

. Свака порука састоји се од четири дела:

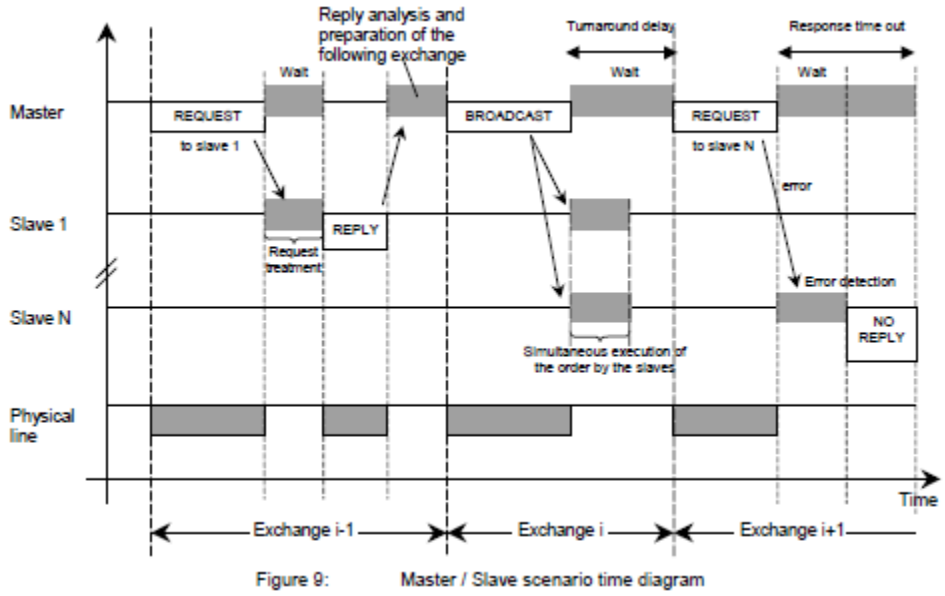
1. адресно поље – указује на који слејв се односи порука. Величина овог поља је 1 бајт, а може да има вредности 0 (*broadcast* порука) или 1-247 када означава адресу слејва којем се шаље порука, или са којег стиже одговор на претходно послату команду
2. функцијски код – дефинише операцију која се извршава. Има величину од 1 бајта и садржи функцијски код поруке, у распону од 0 до 255, који слејв јединици говори коју наредбу треба извршити. Примери наредби које мастер може послати су читање статуса *coil* променљивих или промена њиховог стања, читање или промена вредности регистара, провера исправности комуникације са слејв јединицом... Уколико је примљена порука исправна, слејв у свом одговору мастеру уписује исти функцијски код у поруку, док у случају немогућности извршења наредбе због грешке у садржају поруке, слејв уређај враћа функцијски код у коме је највишем биту додељена вредност 1.
3. подаци – сет података који се шаљу или примају. Овај део поруке састављен је од парова хексадецималних карактера где мастер уписује адресе регистара или *coil* променљивих којима треба приступити, број тражених података, или број података које шаље и саме те податке. Слејв јединица у овај део поруке уписује тражене податке, или код грешке уколико из неког разлога није у могућности да изврши примљену наредбу.
4. провера исправности поруке. При RTU моду преноса исправност поруке се проверава помоћу CRC (Cyclic Redundacy Check) прорачуна. Мастер јединица врши прорачун при слању поруке и уписује резултат на крај поруке. При пријему поруке слејв јединица врши исти прорачун и уколико се резултати разликују то значи да је дошло до грешке током преноса података



Слика 10: Клијент сервер комуникација

Између два узастопна слања обавезна је пауза у трајању од 3.5 знаковна интервала. Ово значи да на пример, за пренос поруке у RTU моду са подешеним 1 битом парности (1+1+8+1) при брзини преноса од 9600bps, време тишине мора бити 4.01 ms.

Мастер/слејв комуникација у ModBus протоколу одвија се по следећем дијаграму тока.



Слика 11: Дијаграм тока мастер/слејв комуникације

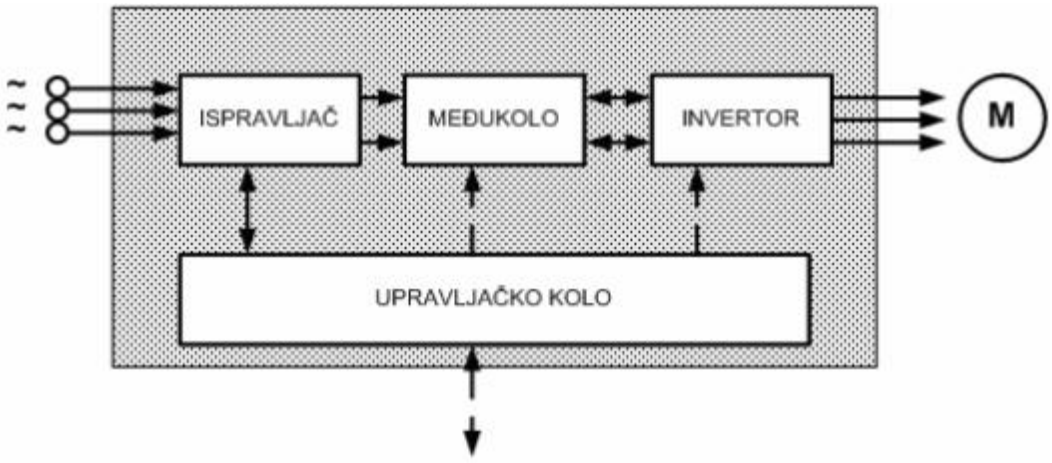
На самом почетку, након подизања система мастер се налази у почетном стању мировања. Ово је једино стање у којем мастер може да шаље поруке. Уколико мастер пошаље *unicast* поруку, прелази у стање у коме током одређеног временског периода чека на одговор слејва. Трајање овог интервала је одређено вредношћу параметра *Wait time for response* који се подешава и може да узима вредности 0-1000ms. Након што прими одговор од слејва, мастер прелази у стање у којем процесира одговор након чега се враћа у стање мировања из ког може поново да шаље поруке. Уколико мастер пошаље *broadcast* поруку, на њу не очекује никакав одговор, али ипак улази у стање у коме се садржава одређени временски период током којег даје времена свим слејв

јединицама да испроцесирају примљене поруке (*Turnaround delay*). Након истека тог периода, чије је трајање отприлике 100-200ms за брзину преноса од 9600bps, мастер се враћа у стање мировања из којег може поново да шаље поруке.
Што се тиче слејва, он се на почетку такође налази у стању мировања. Када стигне порука од мастера, прво се проверава да ли у поруци има грешака. Уколико је све у реду, порука се процесира и у зависности да ли је у питању *unicast* или *broadcast* порука слејв шаље или не шаље одговор.

ИНВЕРТОР

Након аквизиције сензорских сигнала, њиховог процесирања и одређивања референци за моменте мотора, те формирања ModBus порука, ове поруке се шаљу инверторима (фреквентним регулаторима) који управљају моторима роботског система. За ову сврху коришћени су OMRON MX2 инвертори.

Фреквентни регулатори су електронски уређаји који омогућавају управљање брзином трофазних асинхроних мотора претварајући улазни мрежни напон и фреквенцију, које су константне вредности, у промењљиве величине. Фреквентни регулатор је један од најчешће коришћених термина за ове уређаје, али постоје и други називи нарочито и у енглеској терминологији као што су инвертор, *Variable Frequency Drive* (VFD), *Adjustable Frequency Drive* (AFD)... Интерна структура фреквентног регулатора приказана је на слици испод. Исправљач претвара мрежни AC напон у пулсирајући DC напон. Међуколо стабилише овај DC напон и ставља га на располагање инвертору, док инвертор генерише фреквенцију напона на мотору претварајући DC напон у контролисани AC напон. Управљачко коло је микропроцесорски систем који прима и шаље сигнале из исправљача, међукола и инвертора, и на основу својих алгоритама управљања дефинише побуду за мотор како би се добио жељени одзив.



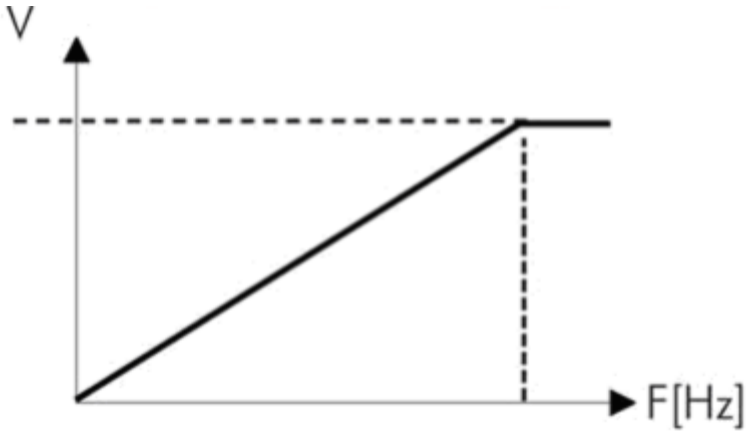
Слика 12: Структура фреквентног регулатора

Као што је речено, фреквентни регулатори контролишу брзину рада мотора мењањем фреквенције напона мотора. Синхрона брзина тј. брзина обртања магнетног поља статора у ротацијама по минути (RPM) добија се по формули:

$$SinhronaBrzina = \frac{120 * Frekvencija}{BrojPolovaMotora}$$

Номинална брзина обртања мотора представља брзину обртања осовине мотора са номиналним оптерећењем и при номиналној фреквенцији напона напајања (50Hz). Да би мотор био у стању да развије моменат, брзина обртања осовине мора бити нешто мања од синхроне брзине. Та разлика у брзинама назива се клизање и услов је за стварање обртног момента.

Два основна начина којима фреквентни регулатор може да управљања асинхроним мотором су скаларно и векторско управљање.
При скаларном управљању, или управљању у отвореној петљи, инвертор контролише рад асинхроног мотора одржавајући константним однос напона и фреквенције, U/f=const, где је са U дата ефективна вредност наизменичног напона, а f представља његову фреквенцију, при чему је брзина мотора пропорционална фреквенцији. При оваквом виду управљања, моменат мотора директно је сразмеран овом односу и на свим брзинама до номиналне брзине, моменат је константан и једнак је номиналном моменту. Регулатор може да напаја мотор и са већом фреквенцијом изнад номиналне, али како није могуће повећавање напона изнад номиналне вредности доћи ће до смањења момента, што може да доведе до тога а испоручени моменат не буде довољан за покретање датог оптерећења.



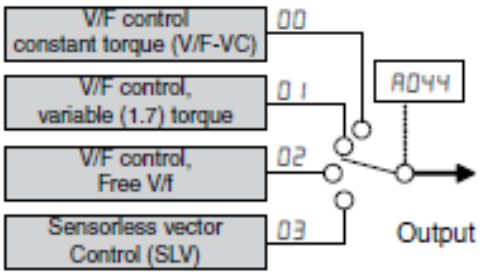
Слика 13: Скаларно управљање константним V/f односом

Код скаларног управљања, фреквентни регулатор нема никакву повратну информацију о брзини којом се мотор заиста окреће, па ће под различитим оптерећењима мотора и његова брзина може бити различита од оне која је задата регулатором.
При векторском управљању контролер у сваком тренутку има информацију од тачној позицији мотора и на основу те информације одлучује да ли треба да модификује излазне параметре контролног сигнала како би одржао жељену брзину и моменат мотора. Информацију о положају мотора регулатор може да добије или помоћу неког сензора (најчешће енкодера који се налази на осовини мотора) или поредећи излазне струјне сигнале са одређеним математичким моделом (*Sensorless Vector Control – SLV*).

Фреквентни регулатор OMRON MX2 може да ради у 4 различита контролна мода:

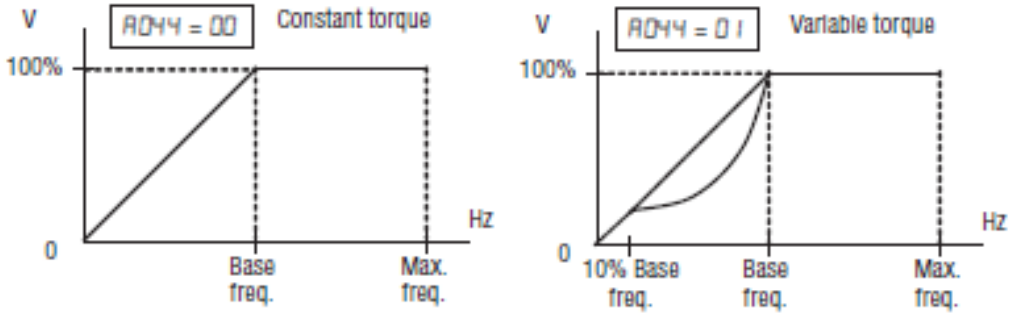
- V/f контрола са константним моментом
- V/f контрола са променљивим моментом
- V/f контрола са слободним постављањем V/f тачака
- Векторска контрола без сензора (*SLV*)

Избор жељеног алгоритма врши се дефинисањем одговарајуће вредности за параметар инвертора A044, а вредности су 00 за V/f са константним моментом, 01 за V/f контрола са променљивим моментом, 02 за слободно постављање V/f тачака и 03 за *SLV*.



Слика 14: Контролни алгоритми фреквентног регулатора

V/f управљање константним и променљивим напоном представљају две варијанте скаларног управљања чија је разлика приказана на слици испод. График са леве стране одговара управљању са константним напоном, и показује да је моменат константан на свим фреквенцијама од 0Hz до вредности основне фреквенције, која се дефинише у параметру регулатора A003. График са десне стране приказује промену V/f односа, па самим тим и момента мотора, при управљању са променљивим напоном. У овом случају, моменат је константан на опсегу од 0Hz до 10% основне фреквенције. Овим начином управљања постижу се веће вредности момента при мањим брзинама мотора и смањеном односу при већим брзинама.



Слика 15: V/f управљање са константним и променљивим моемнтом

Слободна V/f контрола омогућава кориснику да дефинише произвољну V/f карактеристику дефинишући седам тачака криве кроз постављање седам вредности напона и фреквенције у параметрима регулатора b100 до b113. Овај вид управљања, иако омогућава дефинисање произвољних V/f односа, а самим тим и лако дефинисање момента, има туману да је дате тачке могуће дефинисати само у *offline* моду рада, па није погодан за овакав вид управљања.
Векторска контрола без сензора (*SLV*) омогућава регулатору да контролише брзину и моменат мотора на основу познатог излазног напона и струје, и реализованог математичког модела мотора. Математички модел заснива се на константама мотора које се дефинишу у инвертору.

Пре него што се изабере алгоритам управљања дефинисањем вредности параметра A044, потребно је подесити параметар b049 (*Dual Rating Selection*) који одређује у којем од два мода рада ће радити регулатор. Инвертор подржава два мода рада, у зависности од оптерећења – режим са константим момемнтом и са променљивим. Уколико параметар b049 има вредност 00, регулатор ће радити у режиму са константним моментом и тада је могуће изабрати било који од четири описана контролна алгоритма, а ова опција препоручена је за употребу у ситуацијама када се захтева велики моменат услед већег оптерећења. Уколико оптерећење мотора није велико и не захтевају се високе вредности момента тада се може радити у режиму са променљивим моментом и параметру b049 додељује се вредност 01. У овом режиму рада није могуће изабрати *SLV* као контролни алгоритам регулатора.

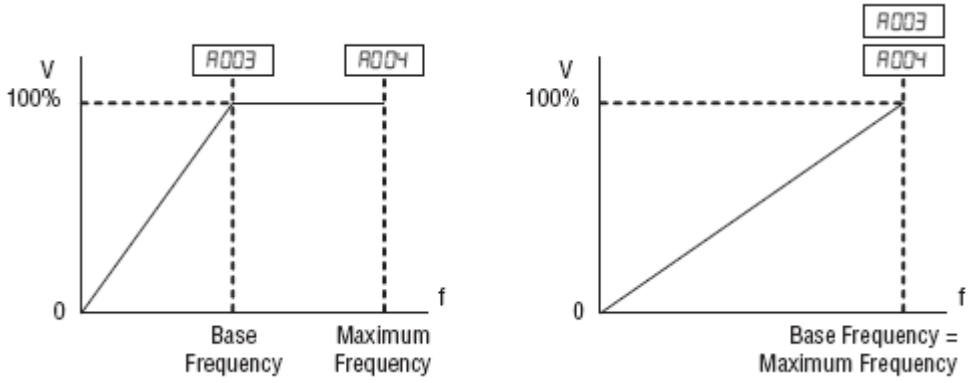
Након што се изабере режим рада регулатора (b049) и жељени контролни алгоритам (A044) потребно је подесити додатне параметре фреквентног регулатора пре него што се почне са управљањем мотора. Сви параметри регулатора подељени су по функцијама у групе:

- A група: стандардне функције
- b група: функције за фина подешавања контроле
- C група: функције за конфигурисање интелигентних терминала регулатора
- d група: функције за мониторинг рада регулатора
- F група: функције за подешавања основног профила рада мотора
- H група: функције за подешавање константи мотора
- P група: функције за разна додатна подешавања рада регулатора

У даљем делу текста биће дат опис релевантних параметара који су коришћени током развоја софтверског програма за управљање планарним роботом. Детаљан опис свих параметара налази се у одељку 3 упутства за употребу фреквентног регулатора.

Најпре ће бити дат преглед подешавања параметара који дефинишу понашање мотора при задатој фреквенцији. Пре свега потребно је дефинисати номиналне величине мотора чији рад регулатор контролише. Вредност параметра A003 одређује основну фреквенцију и може да узима вредности од 30Hz до максималне фреквенције. Максимална фреквенција дефинише се параметром A004 и може да има вредности у опсегу од основне фреквенције

(A003) до 400Hz. Однос између основне и максималне фреквенције приказан је на графику испод.



Слика 16: Однос основне и максималне фреквенције

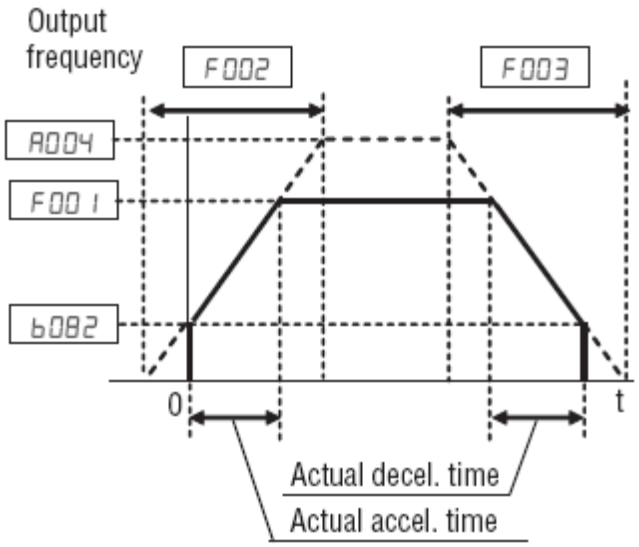
Излаз инвертора прати константан V/f однос све док не постигне максималан напон на основној фреквенцији (A003). На овом делу излазне карактеристике мотор има константан моменат, а повећање фреквенције до максималне вредности (A004) омогучава мотору брже окретање али уз пад вредности момента. Уколико је потребно обезбедити константан моменат мотору на читавом опсегу фреквенција, тада је потребно дефинисати једнаке вредности за основну и максималну фреквенцију (A003=A004).

Фреквентни регулатор поседује функцију аутоматске регулације напона којом прилагођава излазни напон номиналном напону мотора. Номинална вредност напона мотора дефинисана је параметром A082 у коме треба изабрати једну од понуђених вредности која одговара мотору који се користи. За номинални напон мотора од 220V вредност параметра A082 треба поставити на 02.

Параметри F002 и F003 одређују времена убрзања и успоравања мотора, респективно. Ови параметри имају вредности изражене у секундама и могу да се крећу у опсегу 0.01 до 3600 секунди. Вредности ових параметара представљају време које протекне од нулте до максималне фреквенције, или од максималне фреквенције до нуле.

Када се инвертор почне са радом, излазна фреквенција не креће да расте од 0Hz већ почиње од фреквенције дефинисане у параметру b082 (*Start frequency*) и наставља да расте до задате вредности. Параметар b082 може да узима вредности од 0.10 до 9.99Hz.

Излазна фреквенција, којом се одређује брзина окретања мотора, дефинише се параметром F001. Вредности које могу да се дефинишу налазе се у опсегу од 0Hz (односно стартне фреквенције дефинисане са b082) па до максималне фреквенције дефинисане параметром A004. На доњем графику приказано је како параметри убрзања и успорења (F002 и F003) као и почетне и максималне фреквенције (b082 и A004) одређују криву излазне фреквенције.



Слика 17: Карактеристика излазне фреквенције регулатора

Када се зада жељена излазна фреквенција F001, фреквентни регулатор даје излазни сигнал крећући од стартне фреквенције b082. Фреквенција излазног сигнала потом расте до F001 по нагибу дефинисаном односом максималне фреквенције A004 и времена убрзања F002. Иста зависност важи и у случају успоравања, када фреквенција опада са F001 до b082 са успорењем дефинисаним односом максималне фреквенције и времена успорења F003.

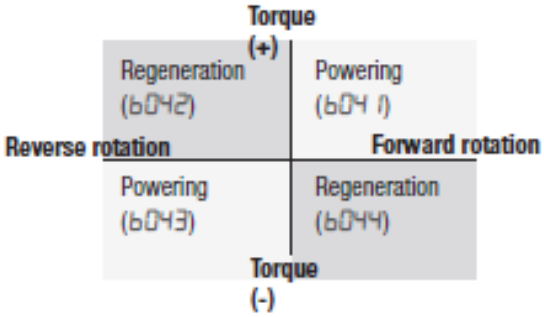
Начин заустављања мотора дефинише се параметрима b091 и b088. Параметар b091 одређује да ли ће након команде заустављања фреквентни регулатор бити тај који ће контролисати заустављање мотора (b091=00), или ће мотор природно да успорава услед трења све док се не заустави (b091=01). Уколико је вредност параметра b091 подешена на 01 тада треба дефинисати и фреквенцију од које ће се наставити контрола брзине мотора при поновном покретању. Уколико се за вредност овог параметра изабере 00 инвертор ће при поновном покретању почети од нулте фреквенције, а уколико је вредност 01 почеће од тренутне фреквенције (фреквенција којом се мотор тренутно окреће при успоравању уколико се још увек није зауставио).

Параметрима A001 и A002 бира се начин на који се задаје жељена фреквенција излазног сигнала као и начин на који се задаје команда за покретање и заустављање мотора. Уколико параметар A001 има вредност 02 излазна фреквенција одговараће вредности дефинисаној у параметру F001, а уколико је A001=03 тада се фреквенција задаје путем ModBus команде. За параметар A002, вредност 02 означава да се старт/стоп команда задаје путем тастера на самом инвертору, а вредност 03 омогућава да се ове команде издају путем ModBus поруке.

Параметри “H” групе користе се за дефинисање карактеристика и константи мотора. Параметрима H003 и H004 одређују се снага мотора и број полова. За вредност параметра H003, снагу мотора, може се изабрати једна од понуђених вредности у опсегу од 0.1kW 15.8kW. Како је снага мотора коришћеног при изради овог софтверског програма за

управљање 25W, за овај параметар одабрана је вредност 00 (0.1kW) као најближа вредност из скупа оних које су по спецификацијама инвертора подржане. Поред ова два поља, карактеристике мотора дефинишу се додатно параметрима H020 до H024, који служе за дефинисање константи мотора. Ове константе фреквентни регулатор користи при прављењу математичког модела мотора у случају када је за контролни алгоритам изабрана векторска контрола без сензора (SLV). Ових пет параметара редом представљају: две отпорности мотора, индуктивност, струју без оптерећења и инерцију. Након што се изабере снага мотора (H003) вредности параметара H020-H024 се аутоматски прилагоде изабраној снази. Вредности ових параметара могуће је унети и ручно, или сетовати путем ModBus команде. Математички модел мотора регулатор може да направи и помоћу константи дефинисаних у параметрима H030-H034. Ове константе представљају исте константе као и параметри H020-H024 али су добијене процесом аутоподешавања (Auto tunning). Вредношћу параметра H002 одређује се да ли ће се за формирање математичког модела мотора користити функције H020-H024 (H002=00) или H030-H034 (H002=02).

Уколико се за контролни алгоритам регулатора изабере SLV, тада је потребно дефинисати граничне вредности момената. Ове граничне вредности дефинишу се у параметрима b041 до b044 и представљају граничне вредности момента по квадрантима, као што је приказано на слици испод.



Слика 18: Граничне вредности момента по квадрантима

Ови параметри могу узимати вредности од 0 до 200%, при чему се 100% односи на напон при номиналној струји инвертора. Чињеница да је снага мотора са којим је рађено 25W, уноси ограничење при дефинисању граничних вредности момената. Наиме, уколико се користи мотор чија је снага реда величине мања од минималне снаге подржане од стране инвертора, граничне вредности момента требају прилагодити на начин да вредност α не прелази 200%, при чему је α дато са:

$$\alpha = \text{гранична вредност момента} \cdot \text{снага инвертора} / \text{снага мотора}$$

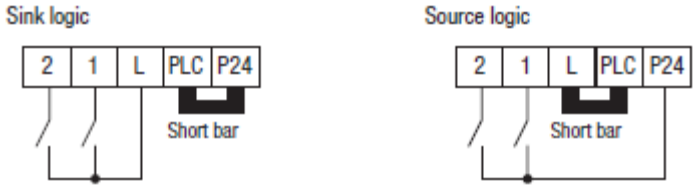
Како је снага коришћеног мотора 25W, а инвертора 0.2kW, добија се:

$$\alpha = \text{гранична вредност момента} \cdot 200W / 25W <= 200\%$$

тј.

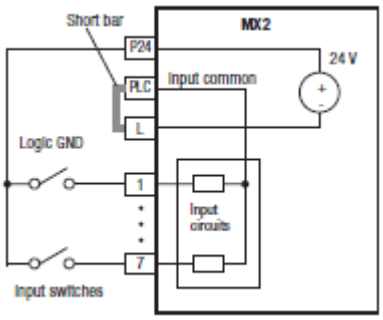
$$\text{гранична вредност момента} <= 25\%$$

Управљање моментом, при управљању у отвореној петљи, могуће је успоставити користећи параметре из „P“ групе функција. Параметар P033 дефинише улаз којим ће се уносити наредба о вредности момента. Вредност P033=03 одређује да вредност уноси оператер. што се потом може извршити и исправном ModBus командом. Параметар P034 може да има вредности од 0 до 200 (%) и њиме се дефинише вредност момента, где моменат од 100% одговара номиналној струји инвертора.
Како би се омогућила контрола момента, потребно је доделити функцију “ATR” (*Enable Torque Command Input*) неком од мултифункционалних улаза инвертора (C001 до C007) и поставити вредност функције на TRUE. Фреквентни регулатор поседује седам улазних терминала којима је могуће доделити неку од 72 различите функције. Једна од тих 72 функције је и “ATR” функција, чији је редни број 52. ATR функција се додељује неком од терминала тако што се једном од параметара C001 до C007 додели вредност 52. Параметри C001 до C007 редом одговарају улазним терминалима инвертора од 1 до 7. Начин на који се, након додељивања функције једном од терминала, поставља њена вредност на TRUE или FALSE, зависи од тога да ли је инвертор подешен да ради са *source* или са *sink* логиком, што опет зависи од начина на који је краткоспојен PLC терминал на инвертору. Два начина спајања приказани су на доњој слици.



Слика 19: *Sink* или *Source* логика

Фабричка подешавања инвертора су таква да су краткоспојени терминали PLC и L, односно користи се *sourcing* логика. У овом случају, да би се вредност одређеног терминала поставила на TRUE потребно је повезати тај терминал са P24 терминалом.



Слика 20: Повезивање при *sourcing* логици

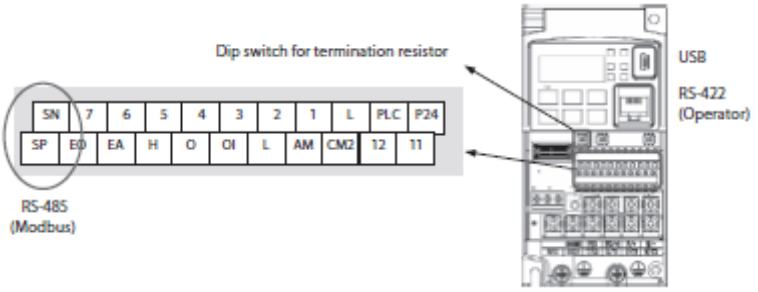
Према томе, да би смо омогућила контрола момента при управљању у отвореној петљи потребно је прво доделити једном од параметара C001-C007 вредност 52 (ATR) и потом повезати одговарајући терминал инвертора са P24 терминалом како би се доделила вредност TRUE функцији ATR.

Уколико се у било ком тренутку јави потреба за ресетовањем подешавања инвертора на фабричке вредности то се може врло лако урадити следећом процедуром:

- прво се параметру b084 додели вредност 02 чиме се одређује да ће се радити иницијализација параметара
- потом се параметром b094 одреди које све параметре хоћемо да ресетујемо при чему вредност 00 дефинише ресет свих параметара
- у пољу b085 подеси се вредност 01, што дефинише географски регион (Европа)
- на крају се у параметру b081 постави вредност 01, чиме се извршава претходно конфигурисано ресетовање вредности.

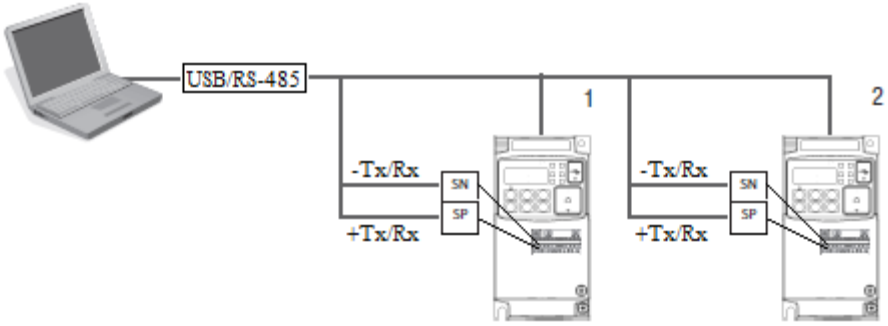
Ово је био кратак преглед неких основних подешавања фреквентног регулатора која је потребно конфигурисати како би смо омогућили правилан рад регулатора. Списак свих параметара по горе поменутим групама, као и њихов опис, налази се у упутству за употребу креираном од стране произвођача инвертора.

Као што је раније поменуто, OMRON MX2 фреквентни регулатор комуницира са реализованом апликацијом путем ModBus комуникацијског протокола, а као физички слој ModBus протокола користи RS-485 стандард. Повезивање инвертора са рачунаром обавља се преко USB/RS-485 комуникацијског конвертора, тако што се терминали +Tx/Rx и -Tx/Rx конвертора повежу на SP и SN терминале инвертора, респективно.



Слика 21: Терминали инвертора за повезивање на RS-485

Уколико је више инвертора повезано на исти рачунар, као што је случај у нашем систему у којем се налазе два инвертора, повезивање се врши према доњој шеми:



Слика 22: Шема повезивања рачунара и два инвертора

Ивертор поседује две врсте променљивих – *coil* променљиве, које су величине једног бита и могу имати вредности TRUE или FALSE, и 16-битни регистри чије се вредности такође могу мењати или читати. Приказ ModBus функција које подржава MX2 инвертор дат је у следећој табели:

Функцијски код	**Функција**
01h	читање статуса *coil* променљиве
03h	читање регистра
05h	упис у *coil* променљиву
06h	упис у регистар
08h	*loopback* тест
0Fh	упис у више *coil* променљивих
10h	упис у више регистара
17h	упис и читање узастопних регистара

Слика 23: ModBus функције које подржава MX2 инвертор

За било коју акцију коју желимо да урадимо потребно је да знамо адресу инвертора којем се обраћамо, функцијски код акције коју желимо да извршимо и редни број променљиве (*coil* или регистра) над којом желимо да обавимо акцију. Редни бројеви свих *coil* променљивих и регистара, њихова имена и параметре инвертора којим одговарају, могу се наћи у корисничком упутству инвертора у додатку Б „*ModBus Network Communications*“. Сада ће на два примера бити приказано како се правилно формирају ModBus поруке како би се регулатору задала одређена команда.

Први пример биће коришћење функције уписа вредности у *coil* променљиву. Рецимо на пример да желимо да покренемо мотор који је повезан на инвертор чија је ModBus адреса 1 тако што ћемо да задамо том инвертору „*Run*“ команду. У упутству можемо наћи да је редни број *coil* променљиве која одговара овој наредби 0001h.

Coil No.	Item	R/W	Setting
0000h	unused	–	(Inaccessible)
0001h	Operation command	R/W	1: Run, 0: Stop (valid when A002 = 03)
0002h	Rotation direction command	R/W	1: Reverse rotation, 0: Forward rotation (valid when A002 = 03)
0003h	External trip (EXT)	R/W	1: Trip

Слика 24: Тражење редног броја *coil* променљиве у корисничком упутству

Знајући да је вредност коју желимо да упишемо 1 (TRUE) потребно је формирати ModBus поруку облика:

0105 0000 FF00 8C3A

при чему 01 представља ModBus адресу инвертора, 05 је функцијски код наредбе уписа у *coil* променљиву, 0000 представља адресу *coil* променљиве која се при креирању поруке увек узима као број за 1 мањи од редног броја датог у упутству (јер су адресе променљивих индексиране почевши од нуле), FF00 су подаци које уписујемо у променљиву (FF00=ON, 0000=OFF), а 8C3A је CRC сегмент поруке који служи за проверу грешке при пријему.

У другом примеру приказан је начин на који се уписује вредност у више узастопних регистара. Уколико је акција коју желимо да извршимо сетовање излазне фреквенције на

15Hz, инвертору чија је адреса 2, тада знајући да се излазна фреквенција подешава у параметру F001 у корисничком упутству лако можемо наћи да је редни број регистара којима се може доделити вредност овом параметру 0001h и 0002h.

Register No.	Function name	Function code	R/W	Monitoring and setting items	Data resolution
0000h	unused	–	–	Inaccessible	
0001h	Frequency source	F001 (high)	R/W	0 to 40000 (valid when A001 = 03)	0.01 [Hz]
0002h		F001 (low)	R/W		
0003h	Inverter status A	–	R	0: Initial status 6: DC braking	–

Слика 25: Тражење редног броја регистра у корисничком упутству

ModBus порука би у овом случају имала облик:

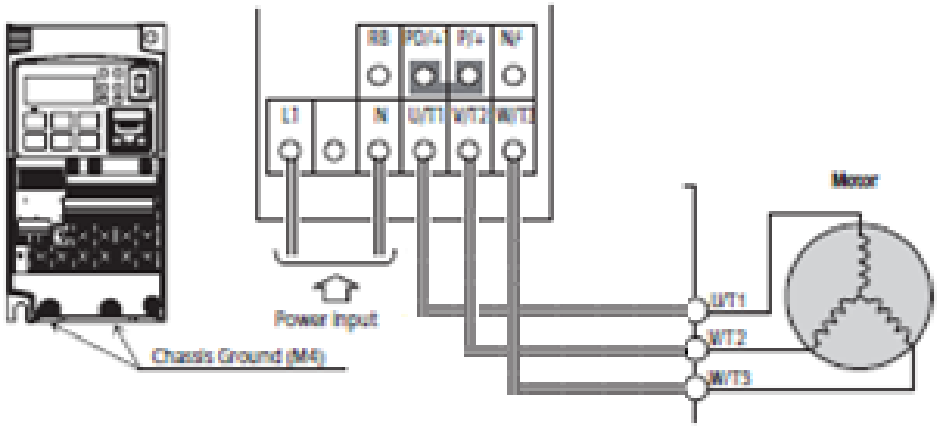
0210 0000 0002 0400 0005 DCFE 22

где 02 представља адресу инвертора, 10 означава функцијски код акције уписа у више регистара, 0000 је адреса првог регистра у који уписујемо (поново умањена за један јер су и адресе регистара индексиране почевши од нуле), 0002 дефинише број узастопних регистара у које се врши упис, 04 је број бајтова у које се врши упис (једнак је броју регистара помноженим бројем 2), 0000 и 05DC представљају хексадецималне вредности које се додељују регистрима. При уносу ових вредности треба имати у виду резолуцију са којом инвертор узима ове податке. Ова резолуција налази се поред сваке функције у приказаном списку, и видимо да за конкретан пример она износи 0.01Hz. То значи да, ако желимо да задамо фреквенцију од 15Hz, у регистре којима се дефинише излазна фреквенција треба да упишемо вредност 1500 (што је у хексадецималном запису 5DC). Последњи део поруке FE22 представља CRC сегмент.

Формирање ModBus порука на аналоган начин ради се и за остале функције, а примери за сваку од подржаних функција, као и комплетан списак параметара инвертора и адреса одговарајућих променљивих, могу се наћи у корисничком упутству. Детаљнији опис самог процеса формирања ModBus порука биће дат при опису реализованог софтверског програма.

МОТОР

Последњи елемент овог управљачког система чине два трофазна асинхрона мотора произвођача DONGZHENG. Номиналне вредности мотора су P=25W, U=220V, f=60Hz и I=0.2A. Као што се може видети на слици 2, мотори су постављени тако да се понашају као „зглобови“ роботских система. Тако постављени мотори омогућавају кретање планарног манипуландума у XY равни. По један фреквентни регулатор повезан је на сваки од мотора како би омогућио управљање манипуландумом током извођења покрета.један мотор повезан је на сваки инвертор а шема повезивања дата је на слици:

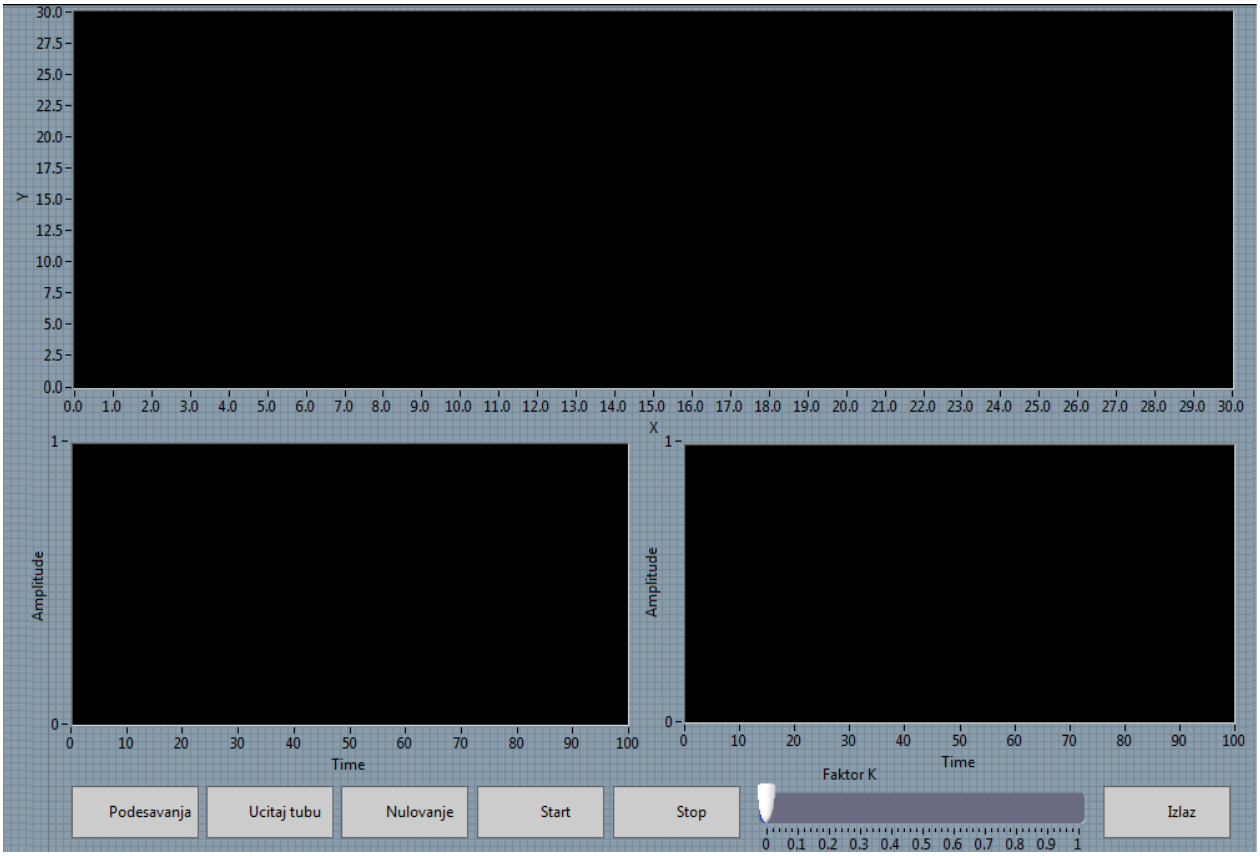


Слика 26. Шема повезивања мотора и инвертора

АПЛИКАЦИЈА

Софтверски програм за управљање планарним роботом за рехабилитацију реализован је у LabVIEW програмском окружењу. LabVIEW је софтверски пакет за графичко програмирање развијен од стране компаније National Instruments.
При опису реализоване апликације, прво ће се дати приказ корисничког интерфејса и опис елемената који се на њему налазе, као и кратко објашњење о томе како апликација ради. Након тога приступиће се опису програмског кода, где ће прво бити речи о изабраној архитектури апликације и њеним карактеристикама, а потом ће се детаљно анализирати и описати све функционалности главног програма и свих његових подпрограма.

Кориснички интерфејс апликације приказан је на слици испод. На слици се види да највећи део интерфејса заузимају три графика – централни график, на коме се исцртава покрет у XY равни, и два мања на којима се прати корелација X и Y координате покрета са учитаним тубама.



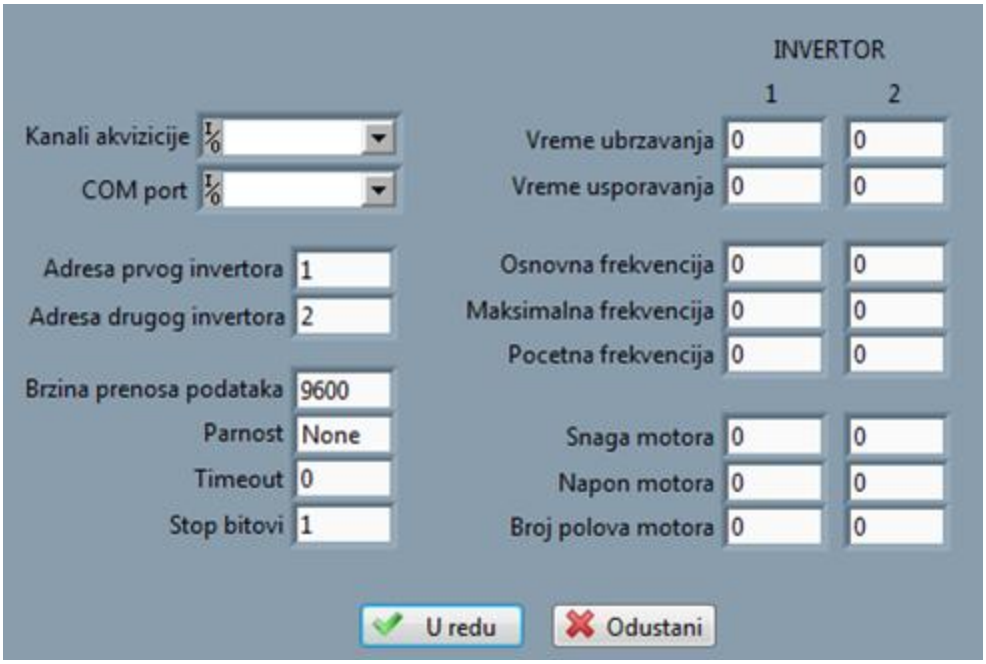
Слика 27: Изглед корисничког интерфејса

Поред графика, на корисничком интерфејсу налази се клизач којим се одређује вредност параметра “K”, као и контролни тастери:

- *Podesavanja* – отвара конфигурацијски прозор у коме је могуће подесити различите параметре везане за рад инвертора, канале аквизиције и виртуелни COM порт на који је прикључен USB/RS-485 ковертор
- *Ucitaj tubu* – отвара прозор за избор .TXT фајлова за X и Y тубу покрета.
- *Nulovanje* – користи се за одређивање офсета при почетном положају дршке роботског система
- *Start* – покреће апликацију и контролу роботског система
- *Stop* – зауставља апликацију и управљање системом

При покретању апликације, програм је у стању мировања све док се не притисне контролни тастер *Start*. До тог момента, могуће је притиском на дугме *Podesavanja* отворити конфигурацијски прозор. У овом прозору корисник може да дефинише следећа подешавања:

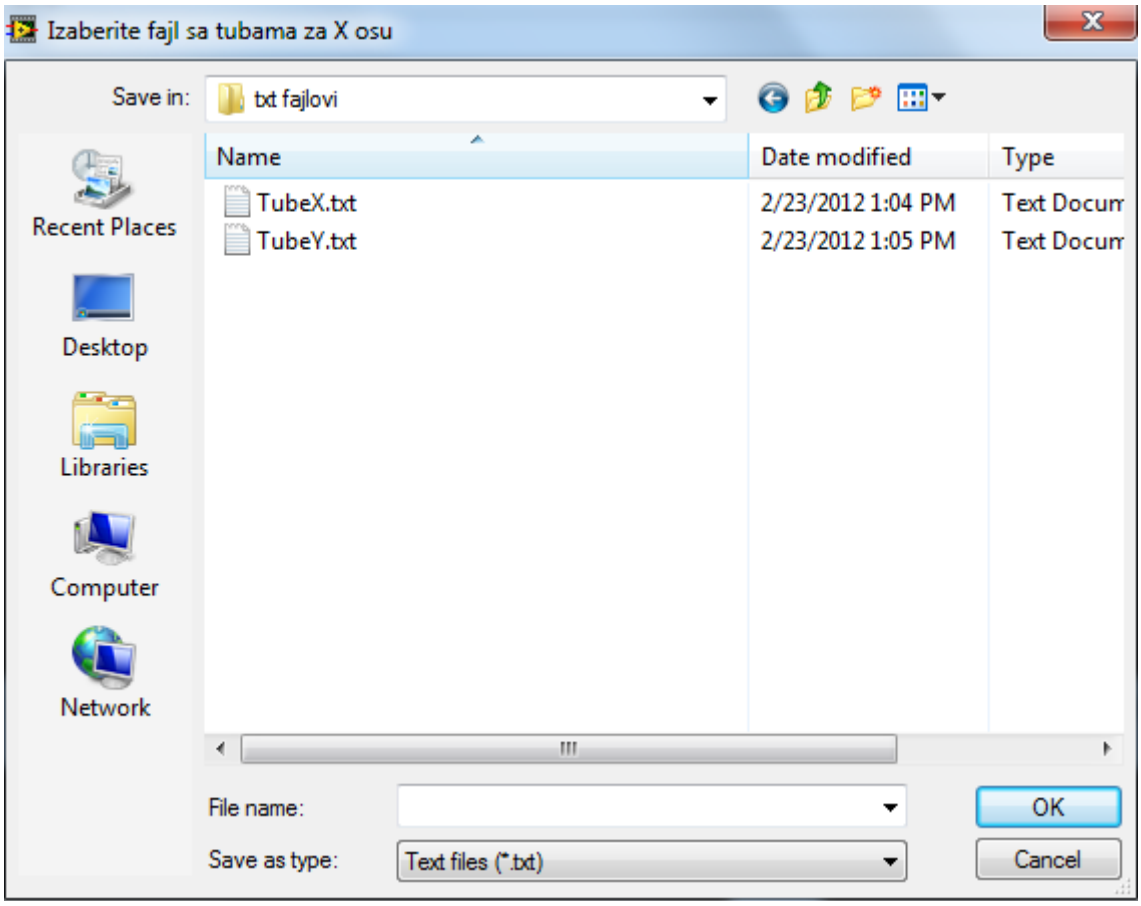
- избор улазних канала аквизиционе картице на које су повезани излази са сензора положаја
- избор COM порта рачунара на који је повезан USB/RS-485 конвертор
- основна подешавања везана за комуникацију рачунара и инвертора као што су брзина преноса података (*Baud rate*), парност, број стоп битова и време чекања на одговор слејва
- избор ModBus адреса инвертора са којима се одвија комуникација
- нека основна подешавања инвертора као што су избор контролног алгоритма, максимална, основна и почетна фреквенција, времена убрзавања и успоравања, номиналне вредности параметара мотора (снага, број полова, напон)...



Слика 28: Конфигурацијски прозор за подешавање параметара рада

Уколико се пре притиска тастера Start не дефинишу конфигурацијски параметри, програм ће радити са стандардним (*default*) вредностима. Када се притисне дугме *Start* конфигурацијске параметре није могуће променити све док се програм не заустави.

Пре покретања контролног мода рада, притиском на дугме *Start* потребно је учитати .txt фајлове са тубама за покрет од интереса. Притиском на тастер *Ucitaj tube* отвара се прозор у коме корисник треба да изабере прво фајл са тубама за X осу покрета, а одмах затим и фајл са тубама за Y осу покрета.



Слика 29: Прозор за избор .txt фајлова са тубама

Нове фајлове могуће је учитати сваки пут пре него што се притисне тастер *Start*, а једном учитане вредности остају активне све док се не заврши извршавање програма.

Пре покретања програма, притиском на тастер *Nulovanje* врши се прорачун офсета сензорског сигнала, односно рачуна се вредност напона када је ручка роботског система постављена у тачку (0,0). Притиском на тастер *Nulovanje* отвара се прозор у коме се исписује обавештење да је потребно поставити ручку у „нулти“ положај. Када корисник притисне тастер *U redu* програм израчунава вредност офсета. Ова операција не мора да се ради при сваком покретању апликације, али би је, ради добијања тачнијих резултата требало повремено извршити.

У тренутку када корисник притисне тастер Start апликација почиње континуално да врши аквизицију сензорских сигнала, потом да врши прорачун тренутног положаја ручке роботског система и момената за моторе, и да шаље инверторима ModBus команде на основу којих ће инвертори управљати радом мотора. Као што је већ поменуто у тексту, на графицима на корисничком интерфејсу тада се исцртавају трајекторија покрета који се прави ручком роботског система, као и корелација покрета са референтним вредностима добијеним учитавањем туба, за X и Y раван. Током рада оператер може да мења вредност параметра „K“ померајући хоризонтални клизач, и то у дозвољеном опсегу вредности од 0 до 1. Притиском на тастер *Stop* апликација престаје да врши аквизицију сигнала и слање команди инвертору, мотор се зауставља, и чека се на ново покретање притиском тастера *Start.* У сваком тренутку, корисник може да прекине извршавање програма притиском на дугме *Izlaz* или притиском на знак „X“ у горњем десном углу прозора.

При развоју кода водило се рачуна о томе да он задовољи основне захтеве које једна добро развијена апликација треба да испуњава, а то су:

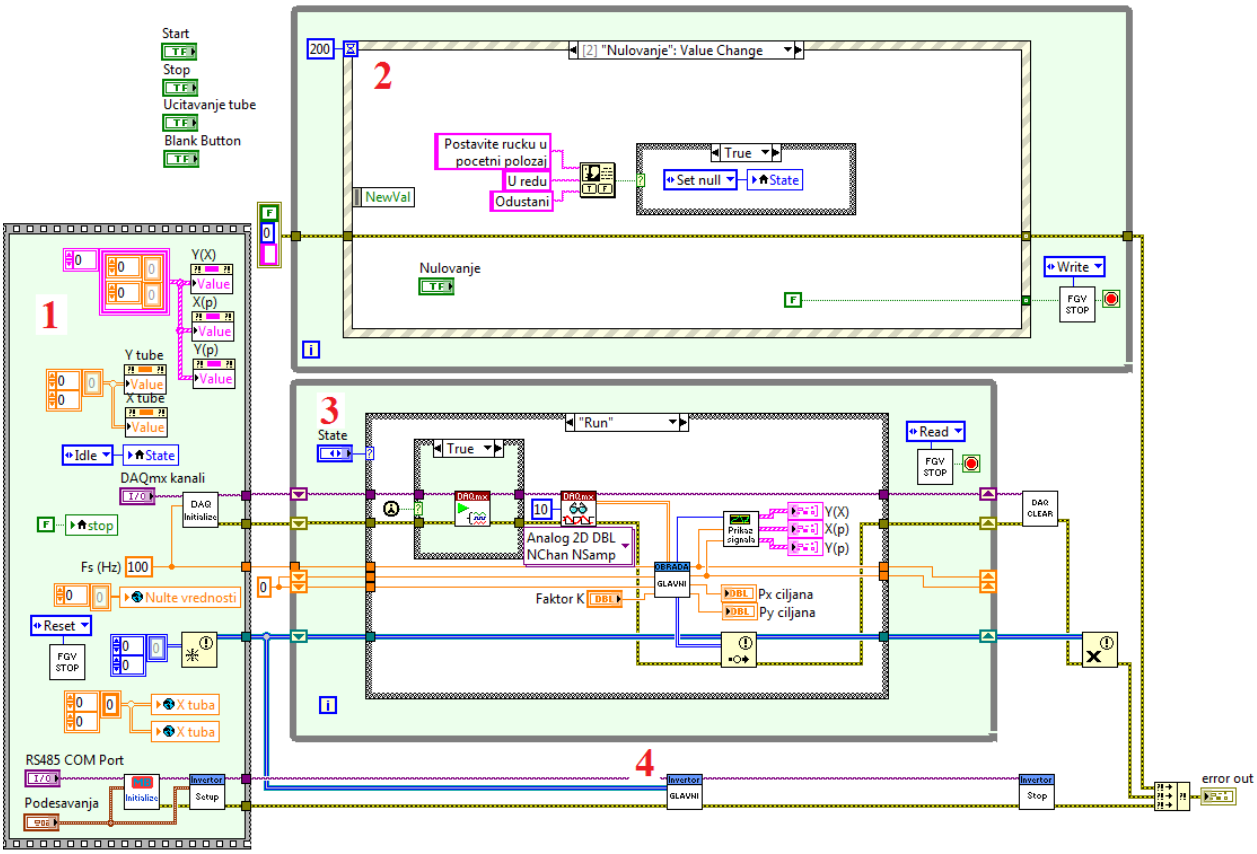
1. модуларност – код је подељен у засебне функционалне целине (модуле) на начин да измена кода у једном од реализованих модула има минималан или никакав утицај на рад других модула.
2. скалабилност – да обезбеди могућност даљег развоја функционалности кода без промена реализоване структуре апликације.
3. читљивост – да блок дијаграм главног програма и свих подпрограма буде уредан и документован на начин да посматрачу обезбеди јасну идеју о функционалности и улози било ког дела апликације.
4. лако одржавање – треба да обезбеди било ком наредном кориснику који буде радио на даљем развоју апликације да што лакше и што брже буде у стању да направи потребне измене оригиналног кода.

Архитектура апликације заснована је на произвођач/потрошач моделу контролисаном *Event* структуром (*Event driven producer/consumer design pattern*), где се комуникација и синхронизација „произвођачке“ и „потрошачке“ петље одвијају употребом *Notifier* функција. *Notifier*-и представљају алат за комуникацију између независних делова истог програма, или између различитих подпрограма на истом рачунару. Као и код осталих видова комуникације у LabVIEW-у, као што су нпр. *Queue*-ови или варијабле, рад *Notifier*-а се састоји од процеса слања и примања података. Оно што је предност *Notifier*-а у односу на варијабле је то што нема потребе за константним проверавањем да ли је стигла нова порука. Процес примања података се одвија тако што функција за пријем заустави даље извршавање петље у којој се налази, и наставља са извршавањем тек када прими нове податке. Ово елиминише потребу за константним очитавањем, а самим тим штеди CPU ресурсе. Друга врло битна карактеристика *Notifier*-а је то што немају бафер, по чему се и разликују од *Queue*-а. Наиме, функција за пријем *Notifier* поруке даје само прву вредност која стигне, од почетка њеног извршавања. Све вредности које су послате између два узастопна извршавања функције за читање *Notifier*-а су изгубљене. Ово је врло битна особина за проблематику управљања планарним манипуландумом јер се обезбеђује да инвертори управљају моторима искључиво на основу најсвежијих информација о

положају ручке манипуландума. Ова карактеристика *Notifier*-а синхронизује рад петље за пријем података, са радом петље за слање.

Програмски код, тј. блок дијаграм, реализоване апликације приказан је на доњој слици. Бројевима 1-4 на слици су означене функционалне целине кода које ће у наставку бити детаљно објашњене:

1. иницијализација
2. *Event* структура
3. петља за аквизицију и процесирање
4. модул за ModBus комункацију



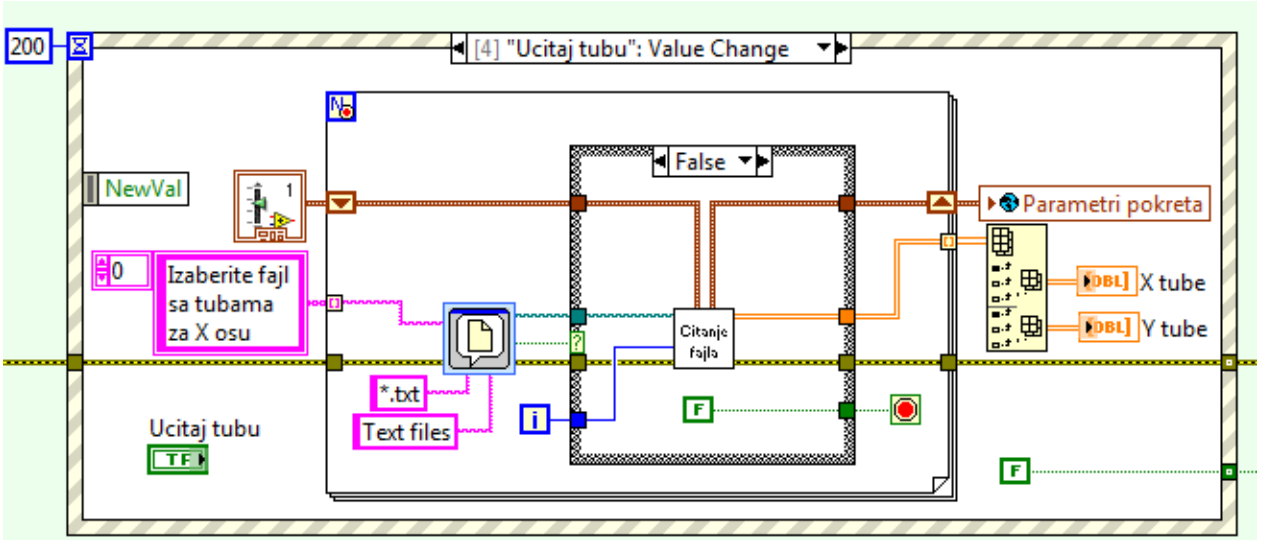
Слика 30: Блок дијаграм реализоване апликације

Иницијализација први део кода, који се извршава одмах при покретању апликације. У једној секвенци врши се упис почетних вредности у све локалне и глобалне варијабле које се користе касније при извршавању апликације, као и пражњење графика који се налазе на корисничком интерфејсу. Овде се врши и ресетовање функцијске глобалне варијабле *STOP* која се користи у свакој независној петљи, за пренос информације о заустављању програма. У овој секвенци се ствара *Notifier* који се користи за комуникацију и синхронизацију произвођачке и потрошачке петље и дефинише се тип података које *Notifier* може да преноси, тако што се на улаз *Obtain Notifier* функције повеже константа жељеног типа. У овом случају, тип података које *Notifier* може преносити је кластер који

се састоји од дводимензионалног низа целобројних вредности и једног *boolean* логичког елемента. Иницијализација свих ових елемената постављена је у секвенцу како би се обезбедило да се додела почетних вредности изврши пре него што програм уђе у *while* петље. Уколико би се при даљем развоју овог програма појавила потреба за увођењем неких нових променљивих или нових елемената на корисничком интерфејсу, довољно је у већ постојећу секвенцу за иницијализацију додати упис почетних вредности у локалну варијаблу новог елемента.

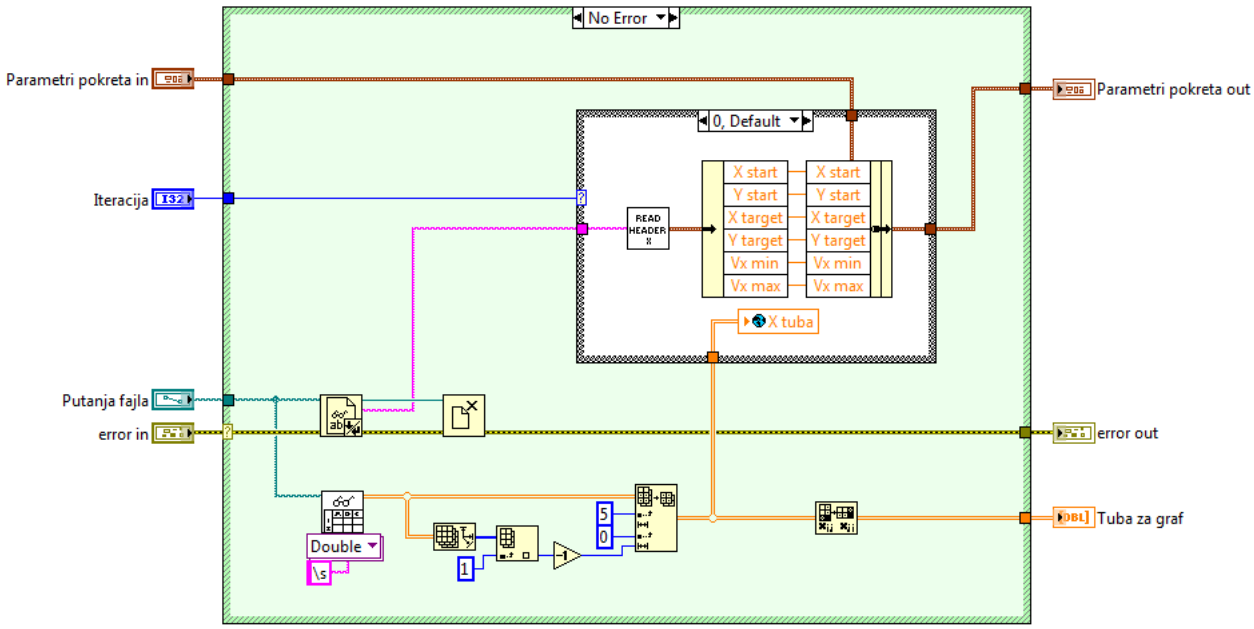
Друга функционална целина кода је *Event* структура. Ова структура користи се постављена унутар *while* петље како би се константно пратиле акције које корисник направи на корисничком интерфејсу. Када се региструје нека од корисничких акција која је предвиђена дефинисаним стањима *Event* структуре, тада се извршава програмски код који је смештен унутар тог стања структуре, и чека се на следећу акцију корисника. Стања *Event* структуре предвиђена реализованим програмским кодом су:

1. *Podesavanja* – када корисник ппритисне тастер *Podesavanja* отвара се конфигурацијски прозор приказан на слици XX. Вредности контрола након притиска тастера *U redu* уписаће се у глобалну променљиву *Podesavanja.*
2. *Ucitaj tubu* – ово стање *Event* структуре извршава се када корисник притисне контролни тастер за учитавање тубе.



Слика 31: *Event* при учитавању тубе

Након што се изабере фајл за X тубу, у подпрограму *Ucitavanje fajla.vi* врши се читање и обрада садржаја фајла.

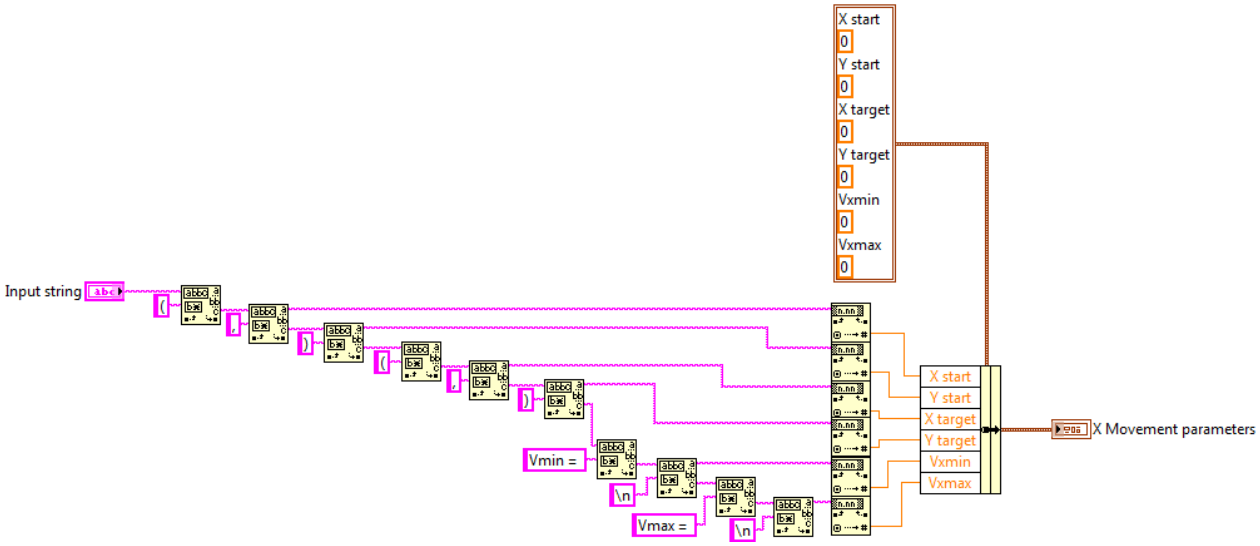


Слика 32: Учитавање и обрада фајла са тубама

Читање фајла састоји се од очитавања вредности и очитавања заглавља фајла. Вредности се очитавају помоћу функције *Read From Spreadsheet File.vi* где је размак (\s) дефинисан као делимитатор вредности, за тип излазних података изабрани су реални бројеви (Double). На излазу ове функције добија се 2D низ реалних бројева, а како је формат фајлова са тубама познат, за матрицу вредности узима се 2D низ који не садржи првих 5 редова оригиналног низа (у тим редовима уписано је заглавље фајла). Тако очитане вредности уписују се у глобалну променљиву *X tuba* која ће се касније користити приликом процеса израчунавања момената за моторе. Добијени 2D низ вредности се такође и транспонује и као такав шаље на излаз овог подпрограма како би се у главном програму користио за исцртавање тубе на графику корисничког интерфејса. Заглавље фајла чита се помоћу функције *Read from Text File*, која читав фајл прочита као један стринг (низ карактера) који се потом парсира у подпрограму *Citanje zaglavlja za X pokret.vi.* Парсирање фајла прлиагођено је текстуалном фајлу који има заглавље облика:

Start = (0.27357,0.0120633)
Target = (0.0658043,0.17349)
Vmin = -0.722668
Vmax = 0.137645

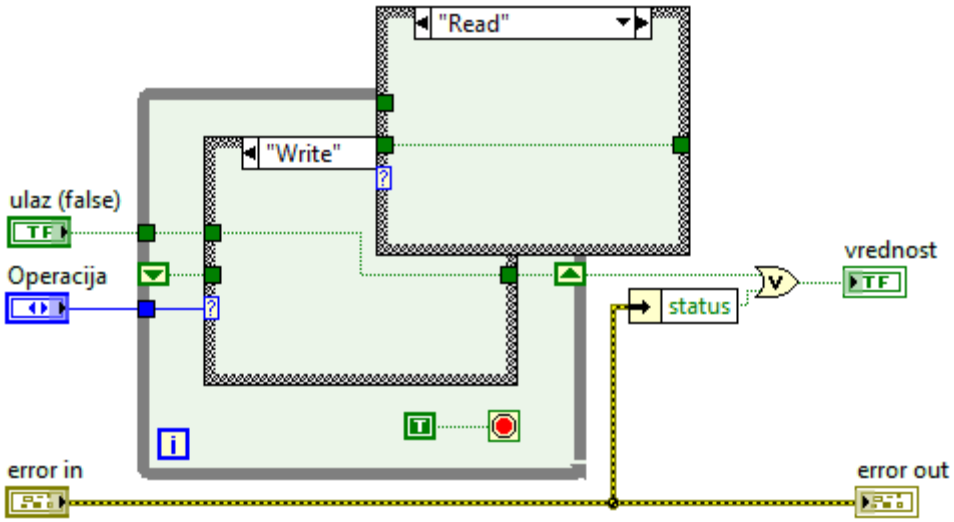
и своди се на тражење симбола отварања заграде, зареза и затварања заграде, и очитавање бројева између њих. Пошто је формат заглавља фајла познат, тачно се зна који број одговара ком параметру па се тако очитани бројеви уписују се у адекватни елемент кластера у којем се чувају параметри покрета.



Слика 33: Учитавање заглавља фајла

Након што су учитане вредности и заглавље фајла са тубама за X покрет, апсолутно идентична радња понавља се и за учитавање фајла за Y покрет у другој итерацији FOR петље. Како би се параметри покрета за X осу добијени у првој итерацији фор петље објединили са Y параметрима добијеним у наредној итерацији, за пренос података кроз извршавање FOR петље користи се шифт регистар из кога ће се на крају тако обједињени параметри уписати у глобалну променљиву *Parametri pokreta.*

3. *Nulovanje* – корисник се обавештава да је потребно поставити ручку планарног манипуландума у почетни положај како би се урадило израчунавање офсета. Уколико се акција потврди, мења се вредност локалне варијабле која дефинише стање аквизиционе петље (*State*) и додељује јој се вредност *Set null.*
4. *Start* – локална варијабла *State* добија вредност *Run*
5. *Stop* – локална варијабла *State* добија вредност *Stop*
6. *Izlaz* –ово стање се извршава при затварању програма. У функцијску глобалну варијаблу *FGV Stop* уписује се вредност TRUE и зауставља се *while* петља у којој се налази *event* структура. Како се иста фукнцијска глобална варијабла чита на крају сваке итерације *while* петљи за аквизицију и за ModBus комуникацију, доћи ће до заустављања и ових петљи, и програм ће прекинути извршавање.

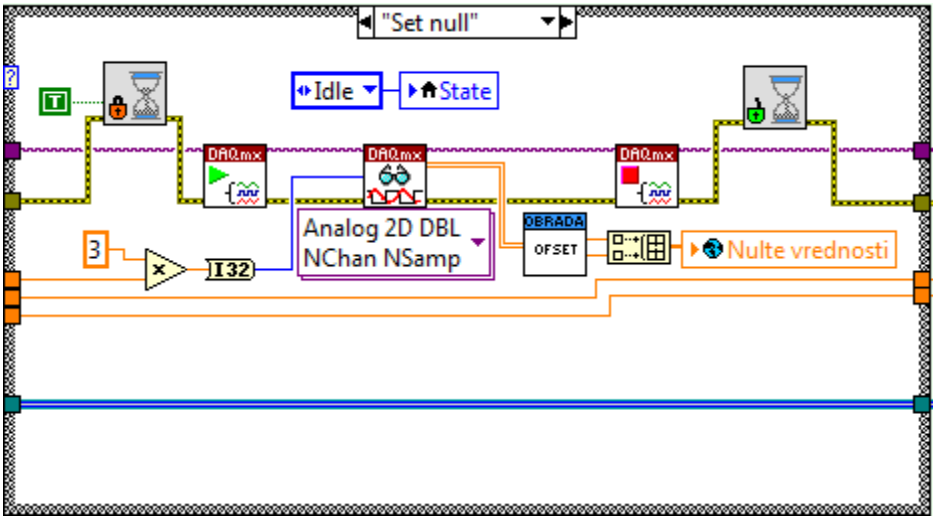


Слика 34: Блок дијаграм функцијске глобалне варијабле *Stop*

7. *Timeout* – ово стање служи само да би након дефинисаног времена (200ms) прочитало вредност функционалне глобалне варијабле Stop и да изађе из петље уколико је у међувремену у неком другом делу програма у њу уписана вредност TRUE.

Наредни део кода јесте *while* петља у којој се врши аквизиција података са сензора положаја, њихово процесирање ради израчунавања референци за моменте мотора и слање добијених вредности у комуникациону петљу. Како се у овој петљи „производе“ вредности које се после користе у комуникационој петљи, ова петља у произвођач/потрошач архитектури апликације представља произвођачку петљу. Сама петља реализована је као један вид *State* машине, где се у свакој итерацији петље прво прочита вредности контроле State па се у зависности од тога извршава један од случајева *Case* структуре:

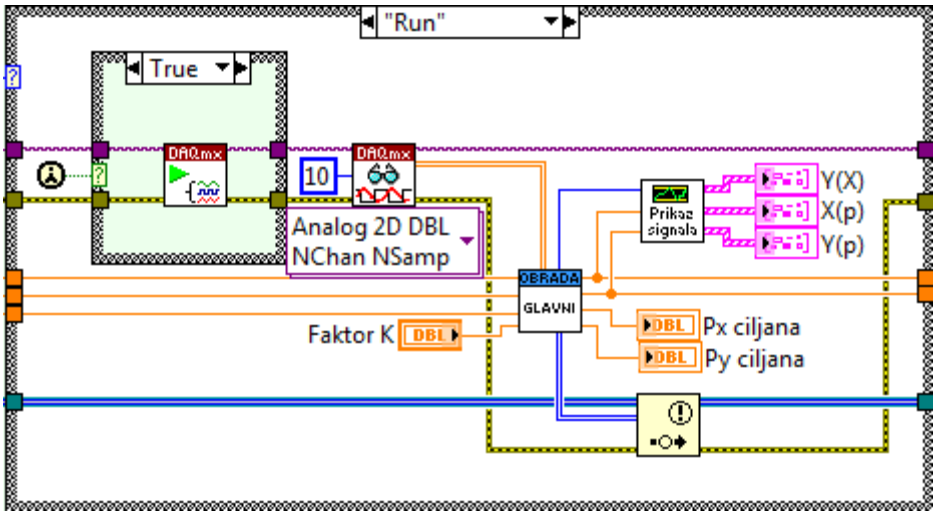
1. *Idle* – овај случај се извршава у стању мировања и у њему се само извршава чекање од 200ms како се не би користило непотребно много ресурса са CPU-а
2. *Set null* – у овом стању се врши израчунавање офсета при постављању ручке планарног манипуландума у почетни положај (0,0)



Слика 35: Стање за израчунавање офсета

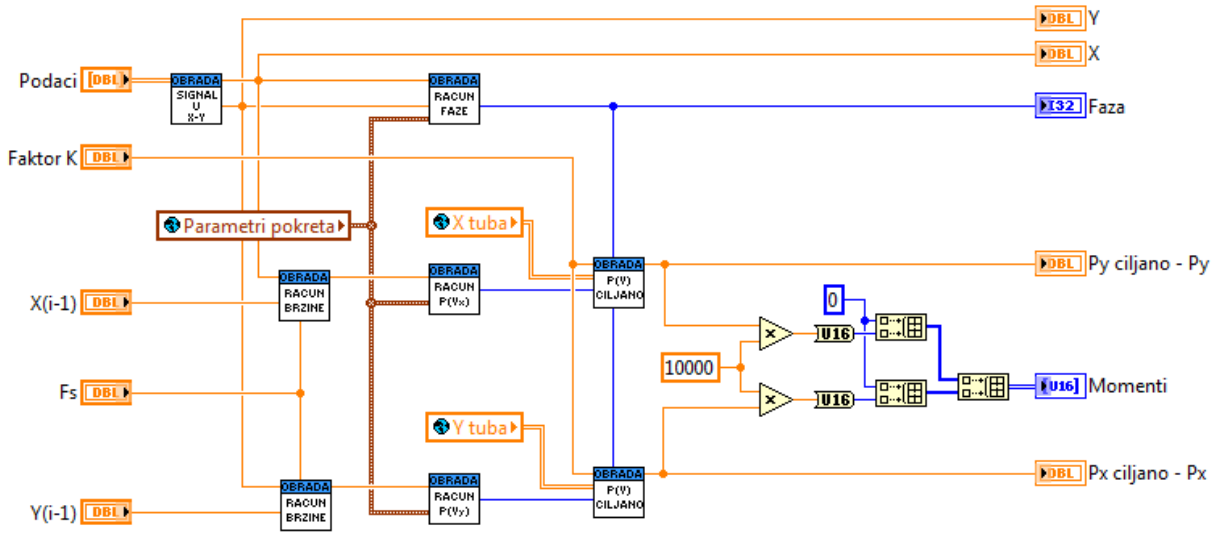
Офсет се израчунава тако што се током три секунде врши очитавање сензорског сигнала. Дужина трајања аквизиције у овом случају одређена је тиме што се на улаз *DAQmx Read* функције који дефинише број узорака који се узимају по једном каналу аквизиције, доведе вредност 3**Fs*, где *Fs* представља учестаност одабирања. Када се узме потребни број узорака, аквизиција се зауставља извршавањем DAQmx Stop функције, а офсет се прорачунава за сваки канал (сензор) посебно, једноставним усрењавањем аквизираних вредности након чега се резултат у облику низа од два елемента уписује у глобалну варијаблу *Nulte vrednosti*. Функције за закључавање и откључавање курсора миша користе се како би се кориснику онемогућило да током ове три секунде притисне неко друго контролно дугме (нпр. *Start*). У овом стању се уписом у локалну варијаблу *State* као наредно стање дефинише *Idle.*

3. *Run* – ово је главно стање петље у коме се врши аквизиција, процесирање и слање вредности момената у модул за ModBus комуникацију. Само у првој итерацији након притиска *Start* дугмета извршиће се DAQmx Start функција која покреће аквизицију. Када је аквизиција покренута, у свакој итерацији DAQmx Read функција аквизира по 10 одбирака на оба канала. Дводимензионални низ вредности добијен на излазу ове функције, заједно са вредношћу учестаности одабирања, параметра „K“ и вредностима координата X и Y из претходне итерације, улази у подпрограм *Procesiranje.vi* у коме се врши процесирање и анализа сигнала. Као резултат овог подпрограма добијају се тренутне вредности X и Y координате и фазе покрета, као и референце за моменте мотора у виду дводимензионалног низа.



Слика 36: *Run* стање произвођачке петље

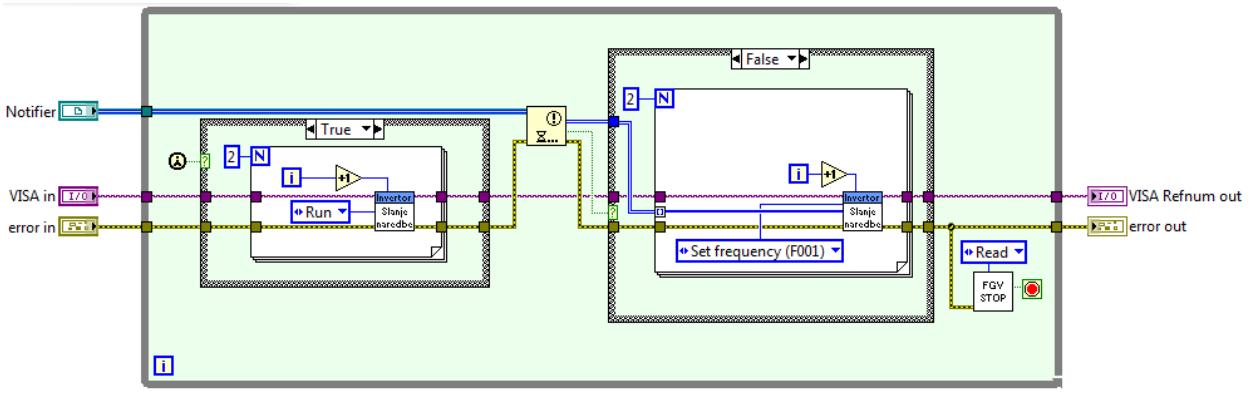
Помоћу информација о позицији и фази се у подпрограму *Prikaz signala.vi* праве кластери који ће омогућити да се на графицима на корисничком интерфејсу приказују зависности Y(X), X(p) и Y(p). Вредности добијене за моменте мотора шаљу се функцијом *Send notification* у комуникациони модул где ће бити форматиране у ModBus поруке и послате инверторима.



Слика 37: Блок дијаграм подпрограма *Procesiranje.vi*

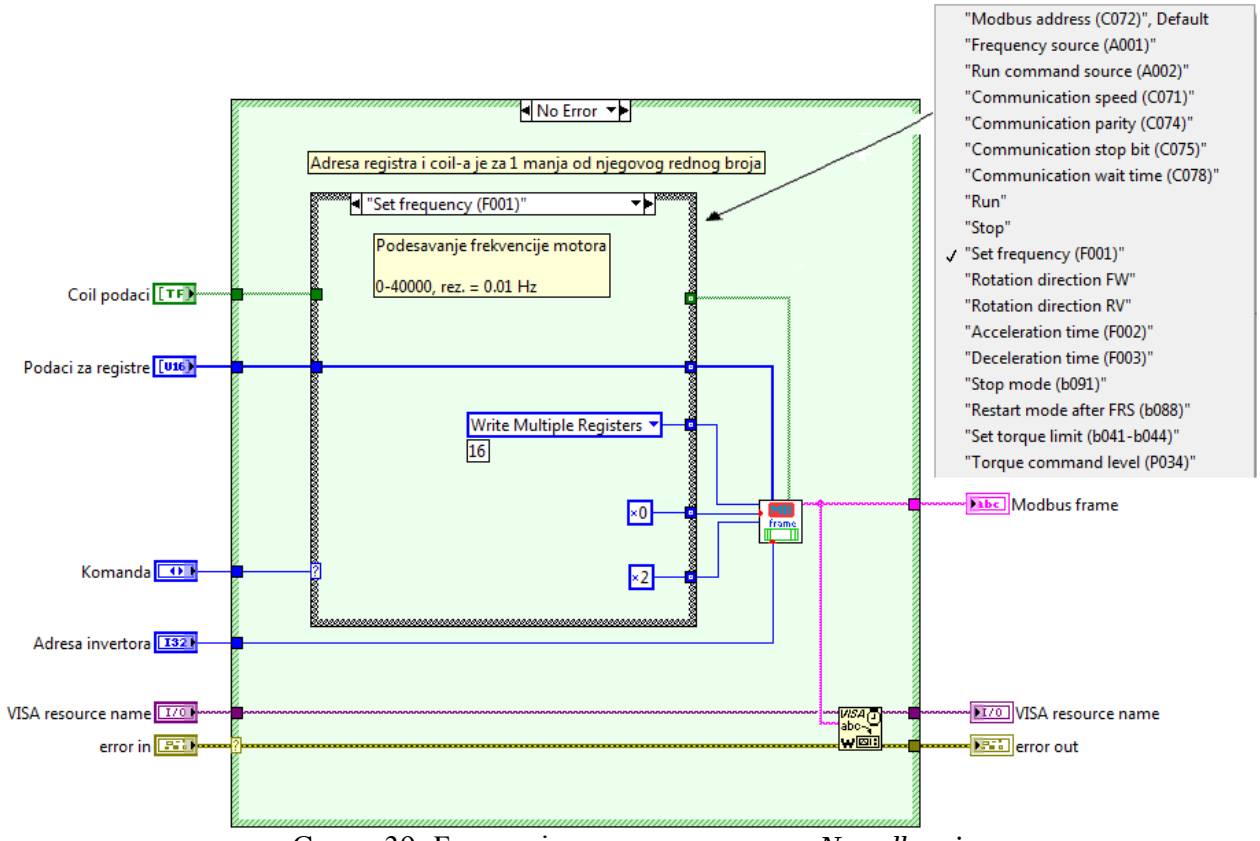
4. *Stop* – зауставља се аквизиција и прелази у стање *Idle*

Пријем добијених вредности референци за моменте мотора, прављење ModBus порука и њихово слање инверторима врши се у комуникационом модулу чији је блок дијаграм приказан на доњој слици. Извршавање овог модула почиње одмах након иницијализације, и траје све док се из глобалне варијабле *FGV Stop* не прочита вредност TRUE, тј. док се не прекине извршавање програма корисничком акцијом или појавом неке програмске грешке.



Слика 38: Блок дијаграм комуникацијског модула

Рад ове *while* петље синхронизован је помоћу функције *Obtain notifier* са радом аквизиционе петље. На приказаном блок дијаграму можемо видети да се поруке инверторима шаљу кроз FOR петље, што значи да се прво шаље порука једном инвертору, па одмах затим и другом. У првој итерацији након притиска контролног тастера *Start* инверторима се шаље команда RUN која сигнализира инвертору да почне са управљањем мотора, након чега се поруке инвертору шаљу сваки пут када *Obtain Notifier* функција прими нове податке из аквизиционе петље. Креирање и слање свих ModBus порука инверторима обавља се у подпрограму *Naredba.vi*.



Слика 39: Блок дијаграм подпрограма *Naredba.vi*

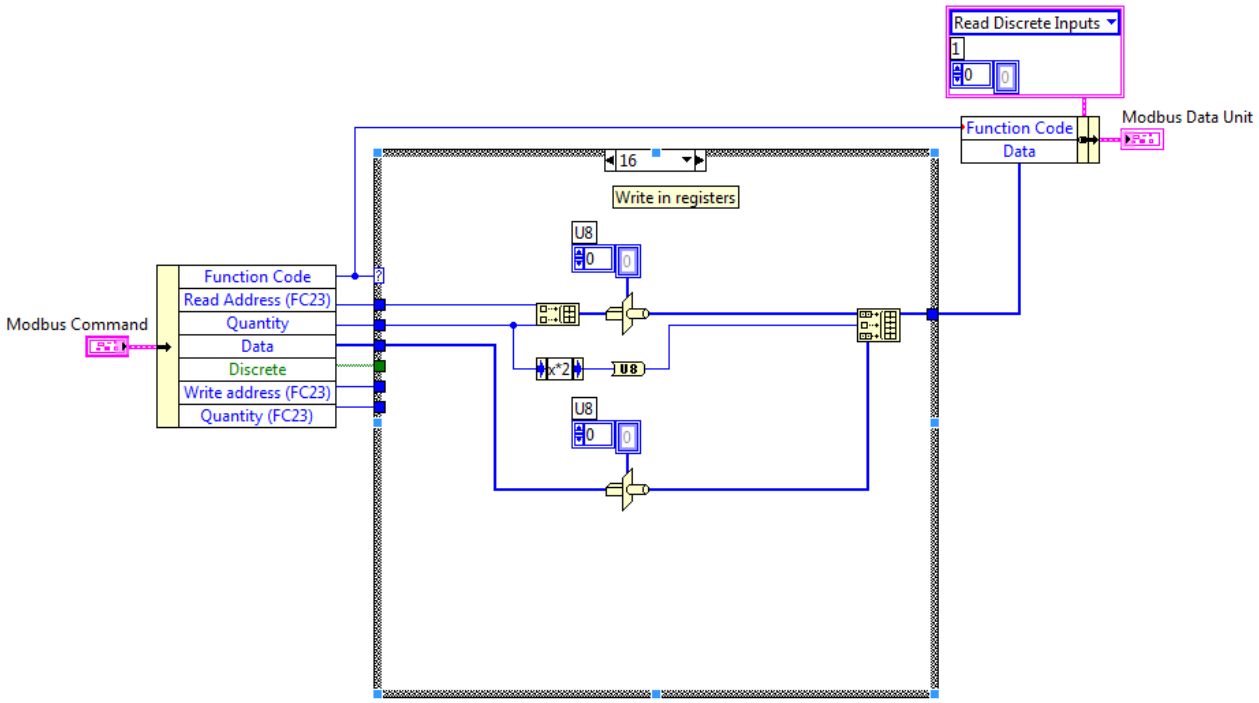
Овај подпрограм састоји се од три функционалне целине:

1. *case* структуре – у којој су предвиђена стања за све наредбе које се могу послати инвертору. Вредношћу улаза *Komanda* дефинисано је које стање ће се извршити. У сваком од стања дефинише се ModBus акција која се обавља, вредности адресе и количина регистара/*coil* променљивих над којима се врши акција, и уколико је потребно, подаци за упис у регистре или *coil* променљиве.
2. подпрограма *MB Frame.vi* у којем се врши креирање ModBus поруке за слање
3. функције *VISA Write* која ће добијену поруку послати инвертору преко виртуелног COM порта.

Овакав начин реализације програма за слање наредби омогућава врло једноставно додавање додатних порука које се могу послати инвертору. Све што је потребно урадити је да се у *TypeDef* контролу *Komanda* дода име нове поруке, након чега се десним кликом на *Case* структуру и избором опције *Add case for every value* ствара нови случај који ће се извршавати при слању нове поруке. У том новом случају потребно је дефинисати акцију која се извршава, адресу и количину променљивих над којима се врши акција (регистри,*coil*) и по потреби обезбедити податке за упис. Извршавање подпрограма *MB Frame.vi* и функције *VISA Write* обавља се на исти начин независно од тога која команда је изабрана, па ту није потребно уносити никакве модификације.

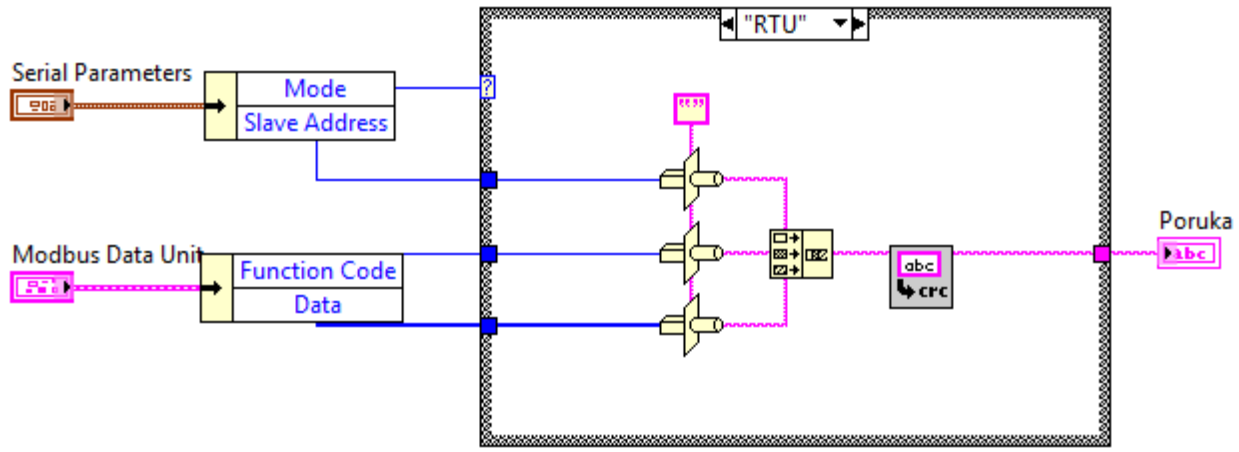
Подпрограм *MB Frame.vi* на основу улазних параметара (адресе инвертора, изабране акције, почетне адресе и количине променљивих над којима се врши акција, и података за упис) треба да креира ModBus поруку формата приказаног на слици XX. Креирање поруке обавља се у два корака. Прво се у подпрограму *MB Format.vi* добијају вредности за поља функцијског кода и поља са подацима. Након тога се у подпрограму *MB Format String.vi* од ових вредности и адресе инвертора формира један стринг. Потом се рачуна вредност CRC поља тако добијеног стринга, и формира се јединствени стринг који представља ModBus поруку која ће бити послата инвертору.

Вредност адресе инвертора је позната, а вредност функцијског кода добија се директно бирањем изабране команде. Једини проблем остаје одредити вредност поља са подацима. Као што се може видети на слици на којој је приказан блок дијаграм подпрограма *MB Format.vi* у којем се овај прорачун и ради, алгоритам за добијање вредности поља зависи од врсте акције која се извршава. Примењени алгоритми добијени су имплементацијом алгоритама датих у корисничком упутству инвертора (додатак 3 – *ModBus Network Communication,* стр. 268-275), а неки од њих приказани су и у овом раду (стране 20 и 21).



Слика 40: Блок дијаграм подпрограма *MB Format.vi*

На доњој слици приказан је изглед блок дијаграма подпрограма *MB Format String.vi* у којем се врши склапање добијених вредности поља ModBus поруке у јединствени стринг и додавање CRC поља. Оцако добијена вредност излазног параметра *Poruka* доводи се на улаз функције *VISA Write* одакле се шаље инвертору.



Слика 41: Блок дијаграм подпрограма *MB Format String.vi*

ЗАКЉУЧАК

У овом раду дат је опис система за управљање планарним рехабилитационим роботом. Приоритет рада био је развој апликације у LabVIEW програмском окружењу која би на основу аквизиране информације о положају ручке манипуландума, и алгоритама развијених у претходним фазама пројекта, одредила референтне вредности момената мотора, а затим помоћу ModBus комуникације остварене са инвертором, управљала моторима на основу тих вредности.

При тестирању оваквог система утврђено је да апликација може успешно да врши управљање моторима планарног манипуландума на основу аквизираних сензорских сигнала, и то у реалном времену, без приметног кашњења.

Током развоја апликације вођено је рачуна да она има висок степен скалабилности и модуларности како при будућем раду и даљем равоју пројекта, чак и корисник који се први пут среће са кодом не би имао већих проблема при даљем развоју апликације и додавању нових функционалности.

ЛИТЕРАТУРА

[1] Popović D.B., *Predavanja iz neuralnih proteza*, Elektrotehnički fakultet u Beogradu, 2007
[2] Schmidt R A, Lee T D. *Motor control and learning: A behavioral emphasis*. Human Kinetics Publishers: 2005
[3] Kollen B, Kwakkel G, Lindeman E. Functional recovery after stroke: a review of current developments in stroke rehabilitation research. *Reviews on recent clinical trials* 2006; **1:** 75-80
[4] Kwakkel G, Kollen B, Lindeman E. Understanding the pattern of functional recovery after stroke: facts and theories. *Restorative neurology and neuroscience* 2004; **22:** 281-300
[5] Kwakkel G. Intensity of practice after stroke: More is better. *power* 2009; **7:** 24
[6] Popović M, Popovic D, Sinkjaer T, Stefanovic A, Schwirtlich L. Clinical evaluation of functional electrical therapy in acute hemiplegic subjects. *Journal of rehabilitation research and development* 2003; **40:** 443-454
[7] http://en.wikipedia.org/wiki/Functional_training
[8] Robertson J V G, Roby-Brami A. Augmented feedback, virtual reality and robotics for designing new rehabilitation methods. In: *Rethinking physical and rehabilitation medicine* Springer: Paris, 2010; 223-245
[9] Kwakkel G, Kollen B J, Krebs H I. Effects of robot-assisted therapy on upper limb recovery after stroke: a systematic review. *Neurorehabilitation and Neural Repair* 2008; **22:** 111-121
[10] Burgar C G, Lum P S, Shor P C, Van der Loos H F M. Development of robots for rehabilitation therapy: The Palo Alto VA/Stanford experience. *Journal of rehabilitation research and development* 2000; **37:** 663-674
[11] Casadio M, Sanguineti V, Morasso P G, Arrichiello V. Braccio di Ferro: a new haptic workstation for neuromotor rehabilitation. *Technology and Health Care* 2006; **14:** 123-142
[12] InMotion Arm Robot, Interactive Motion Technologies, Massachusetts, USA. http://interactive-motion.com/
[13] Armeo Power, Hocoma AG, Switzerland, http://www.hocoma.ch/en/products/armeo/].
[14] Miloš D. Kostić, Mirjana B. Popović, Dejan B. Popović, "Control of robot assistant for rehabilitation of upper extremities", *34th IEEE Conference of the Engineering in Medicine and Biology Society 2012, San Diego USA, submitted*
[15] Miloš D. Kostić, Dejan B. Popović, "Action representation of point to point movements: Classification with probability tube", 19th Telecommunications Forum TELFOR, p. 43-46, 22-24.11.2011, Belgrade, Serbia DOI: 10.1109/TELFOR.2011.6143888
[16] Miloš D. Kostić, Mirjana Popović, Dejan B. Popović, "A Method for Assessing the Arm Movement Performance: Probability Tube" Medical & Biological Engineering & Computing, submitted