Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Medved

Vodenje modelov Lego Mindstorms NXT z industrijskimi krmilniki Siemens

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo: Vodenje modelov Lego Mindstorms NXT z industrijskimi krmilniki Siemens

Tematika naloge:

Sistem Lego Mindstorms NXT s programirljivo kocko, senzorji in motorji predstavlja zanimivo platformo za praktični pouk procesnega vodenja. Zaradi precej različnih standardov vmesnikov v zabavni elektroniki in industrijskih komunikacijah, povezava sistema z industrijskim krmilnikom ni enostavna. Predlagajte rešitev, ki bo omogočala vodenje sistema Lego Minstrorms NXT z industrijskimi krmilniki Siemens. Ustreznost rešitve potrdite in demonstrirajte na problemu regulacije nivoja snovi v maketi industrijskega silosa.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Blaž Medved, z vpisno številko **63070204**, sem avtor diplomskega dela z naslovom:

Vodenje modelov Lego Mindstorms NXT z industrijskimi krmilniki Siemens

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Uroša Lotriča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 14. november 2014 Podpis avtorja:

Zahvaljujem se mentorju, izr. prof. dr. Urošu Lotriču, za strokovno pomoč, usmeritve ter napotke pri izvedbi naloge. Zahvaljujem se tudi družini, ki mi je študij omogočila in me skozi vsa leta študija podpirala, punci Demi Finc za njeno potrpežljivost in podporo ter prijateljem za druženje tekom študija.

Diplomo posvečam svoji družini in Demi.

Kazalo

Seznam uporabljenih kratic

bps (*angl. bits per second*)**, baud** Število bitov na sekundo.

EEPROM (*angl. Electrically Eraseable Programmable Read-Only Memory*) Električno izbrisljiv programirljiv bralni pomnilnik.

HEX (*angl. Hexadecimal*) Šestnajstiški številski sistem.

HMI (*angl. Human Machine Interface*) Vmesnik človek stroj.

IDE (*angl. Integrated Development Environment*) Integrirano razvojno okolje.

IP (*angl. Internet Protocol*) Internetni protokol.

IPv4 (*angl. Internet Protocol version 4*) Četrta generacija internetnega protokola (IP).

ISO (*angl. International Organization for Standardization*) Mednarodna organizacija za standardizacijo.

MAC (*angl. Media Access Control address*) Naslov nadzora dostopa do medija.

OS (*angl. Operating System*) Operacijski sistem.

PDU (*angl. Protocol Data Unit*) Enota protokolnih podatkov.

PLK Programirljiv logični krmilnik.

SCADA (*angl. Supervisory Control And Data Acqusition*) je skupno ime za sisteme, ki so namenjeni nadzorovanju in krmiljenju različnih tehnoloških procesov z računalnikom.

SPI (*angl. Serial Peripheral Interface Bus*) je standard za sinhrono serijsko podatkovno povezavo elektronskih naprav (običajno integriranih vezij), ki deluje v dvosmernem načinu.

SRAM (*angl. Static Random-Access Memory*) Statični spomin z direktnim dostopom.

TCP (*angl. Transmission Control Protoco*) Protokol za nadzor prenosa.

TCP/IP (*angl. Transmission Control Protocol/Internet Protocol*) Protokol za nadzor prenosa in internetni protokol.

UART, USART (*angl. Universal Asynchronous Receiver-Transmitter*) Univerzalni asinhroni sprejemnik in oddajnik.

UDP (*angl. User Datagram Protocol*) Je nepovezovalni protokol za prenašanje paketov.

USB (*angl. Universal Serial Bus*) Vsestransko zaporedno vodilo.

Povzetek

Namen diplomskega dela je predstavitev splošne rešitve za vzpostavitev povezave med industrijskim krmilnikom Siemens in modeli Lego Mindstorms. Krmilniki Siemens omogočajo več tipov komunikacije s perifernimi napravami, med drugim tudi komunikacijo Ethernet po protokolu TCP/IP, medtem ko imajo modeli Lego Mindstorms starejših generacij bolj omejene možnosti komunikacije, saj omogočajo komunikacijo le preko vmesnika USB ali Bluetooth. V okviru diplomske naloge smo izdelali komunikacijski vmesnik, ki omogoča fizično vzpostavitev medsebojne komunikacije med krmilnikom Siemens in modeli Lego Mindstorms NXT in razvili potrebno programsko kodo za integracijo obeh sistemov. Osnovo komunikacijskega vmesnika predstavlja mikrokrmilnik Arduino.

Ključne besede: industrijski krmilnik Siemens, Lego Mindstorms, komunikacijski vmesnik.

Abstract

The purpose of the thesis is to present a general solution for establishing connection between Siemens industrial controller and Lego Mindstorm models. Siemens industrial controllers allow several types of communication with peripheral devices, among those is also Ethernet communication over TCP/IP protocol, while older generation Lego Mindstorm models have more limited possibilities, because they allow communication only over USB or Bluetooth. In the context of the thesis we developed an communication interface, which allows physical establishment of communication between the Siemens industrial controller and Lego Mindstorms. The basis of the communication interface represents the Arduino microcontroller, where also program code was developed.

Key words: Siemens industrial controller, Lego Mindstorms, communication interface.

Poglavje 1

Uvod

Industrijski krmilniki so danes postali nepogrešljivi gradniki sodobne industrijske avtomatizacije in vodenja industrijskih procesov. Te procese lahko srečamo v industrijskih okoljih, kot so proizvodne linje ali visokoregalna skladišča, najdemo pa jih tudi v vsakdanjem življenju, na primer pri pametnih hišah, parkiriščih. Današnji industrijski krmilniki običajno omogočajo tudi komunikacijo z več deset perifernimi napravami [2].

Modeli Lego Mindstorms NXT so programirljivi roboti, ki se jih uporablja predvsem za učne in pedagoške namene. Ti modeli imajo v nasprotju z industrijskimi krmilniki Siemens precej omejene možnosti za komunikacijo s perifernimi napravami. Ponujajo nam možnost komunikacije preko vmesnikov USB in Bluetooth. Vmesnik USB ima prednost pri hitrosti komunikacije in je namenjen predvsem komunikaciji z osebnim računalnikom. Vmesnik Bluetooth pa je počasnejši, a je zato bolj fleksibilen, saj omogoča brezžično komunikacijo z osebnim računalnikom in drugimi Lego Mindstorms modeli ter napravami. Upravljanje modela Lego Mindstorms NXT lahko poteka preko vnaprej napisanega programa ali pa s pošiljanjem neposrednih ukazov v realnem času.

Ker krmilniki Siemens in modeli Lego Mindstorms v osnovi ne ponujajo skupnega načina komunikacije, smo izdelali komunikacijski vmesnik, ki bo to povezavo najprej fizično omogočal, nato pa s programsko kodo vmesnik tudi

nadgradili. Ker je bil vmesnik zasnovan splošno, nam to omogoča integracijo z drugimi sistemi.

V nadaljevanju diplomske naloge bomo v 2. poglavju predstavili vso strojno in programsko opremo, ki smo jo potrebovali za izdelavo komunikacijskega vmesnika. Sledi 3. poglavje, v katerem bomo podrobneje predstavili vzpostavitev komunikacije med krmilnikom Siemens in modeli Lego Mindstorms NXT. V 4. poglavju bomo predstavili izvedbo simulacije industrijskega silosa ter analizo meritev ultrazvočnega senzorja in uporabljenih regulacij. Zadnje, 5. poglavje pa je namenjeno sklepnim ugotovitvam.

Poglavje 2

Uporabljena strojna in programska oprema

2.1 Strojna oprema

2.1.1 Siemens 314C-2 PN/DP

Industrijski krmilniki ali PLK so krmilniki, ki upravljajo z diskretnimi procesi in uporabljajo shranjene ukaze ter programirljiv pomnilnik za uporabo logičnih, sekvenčnih in matematičnih funkcij. Omogočajo nam proceduralno upravljanje strojev in/ali usklajevanje nadzornih sistemov [1]. Industrijski krmilniki danes omogočajo priključitev velikega števila perifernih naprav, zelo gosta ožičenja, enostavno sestavljanje in montažo, vsebujejo binarne in analogne vhode, ki podpirajo standardne napetostne in tokovne nivoje, krmilnike pa lahko nameščamo tudi v bližino sistema [2]. Uporabljeni industrijski krmilnik Siemens 314C-2 PN/DP spada v družino Siemens Simatic S7-300. Krmilnik je visoko procesorsko zmogljiv, saj podpira tehnologijo detekcije frekvenc, pulzno modulacijo in pozicioniranje. Vgrajene ima tako analogne kot digitalne vhode in izhode, ki omogočajo zajem podatkov iz senzorjev in upravljanje aktuatorjev. Krmilnikov polnilnik je razdeljen na dva dela, in sicer na 192 kB delovnega in 8 MB zunanjega, na slednjem pa

se nahajajo shranjeni programi [7]. Poleg tega ima tudi kombinirano vodilo MPI/Profibus DP in dva vmesnika Profinet, ki sta skupaj povezana v omrežno stikalo. Slednji vmesnik smo tako lahko uporabili za omrežno povezavo z mikrokrmilnikom Arduino in vmesnikom človek – stroj (HMI).

Slika 2.1: Industrijski krmilnik Siemens CPU 314C-2 PN/DP.

Z razcvetom računalniško vodenih procesov in razvojem informacijskih tehnologij, so se pojavile tudi potrebe po komunikaciji znotraj industrijskih okolij za namene spremljanja (angl. monitoring), nadzora in medsebojnega povezovanja naprav v industrijsko omrežje. To omrežje se nekoliko razlikuje od pisarniškega, predvsem zaradi specifičnih zahtev okolja in samih operacij. Kljub funkcionalnim razlikam med njima, lahko opazimo velik porast medsebojne integracije. Nivoji v industrijskih okoljih so med seboj povezani z različnimi vodili, glede na to kakšne so karakteristike prenosa podatkov, ali so vodila izpostavljena neprijaznim razmeram, kakšne razdalje morajo prepotovati in kakšen je način komunikacije (angl. master - slave, peer to peer). Tako lahko pri povezovanju področnih naprav opazimo predvsem vodili Profibus in Profinet [2].

Medsebojno povezovanje odprtih sistemov, je standardizirano v referenčnem modelu OSI. Iz tega modela je nastal tudi protokol TCP/IP, ki sedaj postaja standard za vse časovno nekritične povezave v industriji. Profinet je

mednarodni standard pod okriljem organizacije PNO in je nastal kot nadgradnja Ethernet protola, namenjen industrijskemu okolju. Poznamo dve različici Profineta. Profinet IO je namenjen povezovanju področnih naprav na vodilo. Druga različica je Profinet CBA, ki je namenjen porazdeljenim sistemom vodenja [2].

2.1.2 Lego Mindstorms NXT 2.0

Lego Mindstorms NXT 2.0 je programirljiv robot podjetja Lego. Na tržišču se je pojavil leta 2006 kot naslednik prve generacije Lego Mindstorms, vendar je tudi ta že dobil svojega naslednika, poimenovanega EV3. Vgrajen ima 32 bitni procesor ARM7TDMI z 256 KB bliskovnega in 64 KB bralno pisalnega pomnilnika. Vsebuje tudi 8 bitni krmilnik Atmel AVR ATmega48 s 4 KB bliskovnega in 512 KB bralno pisalnega pomnilnika [5]. Programski jeziki, ki so podprti s strani podjetja Lego, so NXT-G Code in ROBOLAB. Modeli Lego Mindstorms NXT pa poznajo tudi več neuradnih programirljivih jezikov, kot so NXC, LeJOS NXJ, NBC in RobotC. Vsaka generacija Lego Mindstorms s seboj prinaša tudi nove možnosti razvoja programske opreme. Modeli Lego Mindstorms NXT imajo 3 izhodna in 4 vhodna vrata, ki omogočajo priključitev servomotorjev, luči, različnih senzorjev podjetja Lego in drugih senzorjev I2C. Za medsebojno povezavo komponent uporablja RJ12 telefonski priključek. Servomotorji imajo vgrajene reduktorje, ki lahko s pomočjo notranjih optičnih dajalnikov zaznavajo rotacije do stopinje natančno. Ultrazvočni senzor je namenjen meritvam oddaljenosti objektov. Območje, v katerem senzor deluje, je med 3 in 250 cm in najboljše meritve zagotavlja takrat, ko je merjena površina na objektu ravna. Lego Mindstorms omogoča komunikacijo preko vmesnika USB 2.0 in Bluetooth. Upravljanje modela Lego Mindstorms NXT lahko poteka preko vnaprej napisanih programov, kateri se izvajajo na sami kocki, ponuja pa nam tudi pošiljanje neposrednih ukazov, s katerimi lahko upravljamo motorje in zajemamo vrednosti senzorjev.

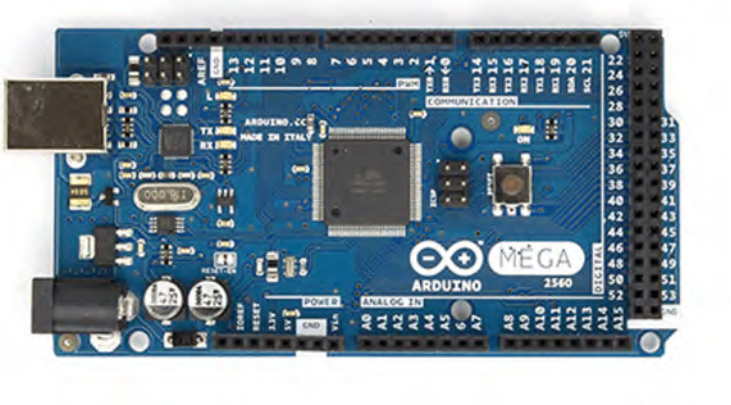
Slika 2.2: Lego Mindstorms NXT 2.0.

2.1.3 Arduino Mega 2560 R3

Arduino je odprtokodna platforma, ki je bila zasnovana kot nizkocenovna in enostavno uporabna rešitev za snovanje naprav ter nam s pomočjo senzorjev in aktuatorjev omogoča interakcijo z okoljem. Mikrokrmilniki Arduino so se na tržišču prvič pojavili v letu 2005, danes pa jih lahko zasledimo že v devetnajstih različicah. Omogočajo nam nadgradnjo osnovne plošče z velikim številom raznovrstnih modulov in dodatkov. Za namene diplomske naloge smo uporabili Arduino Mega 2560 R3, ki bazira na 8 bitnem mikrokrmilniku Amtel ATmega 2560 AVR z 256 KB bliskovnega, 8 KB SRAM, 4 KB pomnilnika EEPROM in operativno napetostjo 5 V. Vgrajenih ima 54 digitalnih vhodov in izhodov, 16 analognih vhodov, 4 UART vrata (RX/TX), 16 MHz kristalni oscilator in povezavo USB [3].

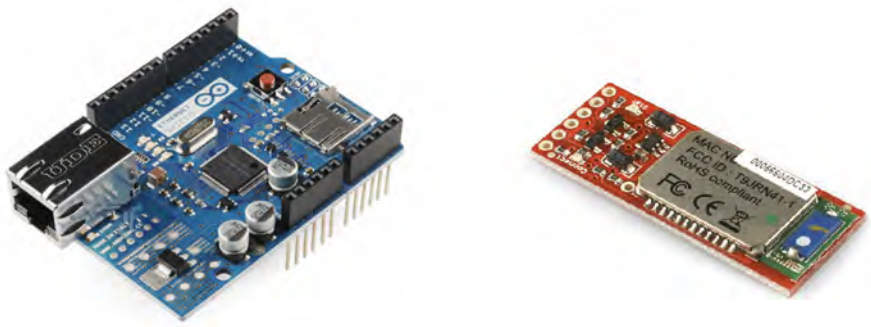
2.1.4 Arduino Ethernet Shield R3 in Bluetooth Sparkfun BlueSMiRF Gold.

Naš model mikrokrmilnika Arduino smo morali nadgraditi z dodatnimi moduli. Za potrebe komunikacije Ethernet med industrijskim krmilnikom Siemens in krmilnikom Arduino, smo le-tega nadgradili z modulom Arduino Ethernet Shield R3. Vgrajen ima krmilnik Ethernet W5100 s 16 KB in-



Slika 2.3: Arduino Mega 2560 R3.

ternega pomnilnika, ki omogoča hitrosti do 10/100 Mb/s in podpira tako protokol TCP kot UDP [3]. Mikrokrmilnik Arduino smo na koncu nadgradili še z modulom Bluetooth Sparkfun BlueSMiRF Gold, s katerim smo lahko v realnem času pošiljali neposredne ukaze na model Lego Mindstroms. Model z mikrokrmilnikom Arduino komunicira preko RX/TX serijskih vhodov in izhodov. Serijski tok podatkov lahko potuje med 2400 bps in 115200 bps, njegova operativna napetost pa je med 3.3 V in 6 V [8].



Slika 2.4: Arduino Ethernet Shield R3 (levo) in Bluetooth Sparkfun BlueSMiRF Gold (desno).

2.2 Programska oprema

2.2.1 Siemens Simatic Step7 V5.5

Simatic Step7 je programsko okolje, ki je namenjeno programiranju, diagnosticiranju, servisiranju, dokumentiranju, testiranju, konfiguriranju in povezovanju strojne opreme krmilnikov Siemens. Program se na industrijskem krmilniku izvaja ciklično. Število teh blokov je nespremenljivo in je odvisno od modela krmilnika. Programsko okolje Siemens Step7 nam omogoča programiranje v več programskih jezikih [2]:

- lestvični diagram (LAD) je grafični jezik, ki omogoča enostaven prehod iz električnih shem v programiranje in je najbolj uporabljen jezik za industrijske krmilnike,
- funkcijski načrt (FBD) je eden od standardnih grafičnih jezikov, ki nazorno prikazuje medsebojno povezanost funkcij in funkcijskih blokov. Podobni so električnim in blokovnim shemam iz analogne in digitalne tehnike,
- diagram poteka (Siemens Graph) opisuje zaporedje operacij in interakcij med vzporednimi procesi, ki temeljijo na Petrijevih mrežah,
- lista ukazov (STL) je nizkonivojski programski jezik, podoben zbirniku in je namenjen predvsem izkušenim programerjem za izdelavo učinkovitejše programske kode. V ta jezik se prevedejo vsi višjenivojski programski jeziki,
- strukturirani tekst (SCL) je visoko nivojski tekstovni programski jezik, zelo podoben običajnim programskim jezikom. Primeren je za kompleksne obdelave podatkov in pisanje blokov.

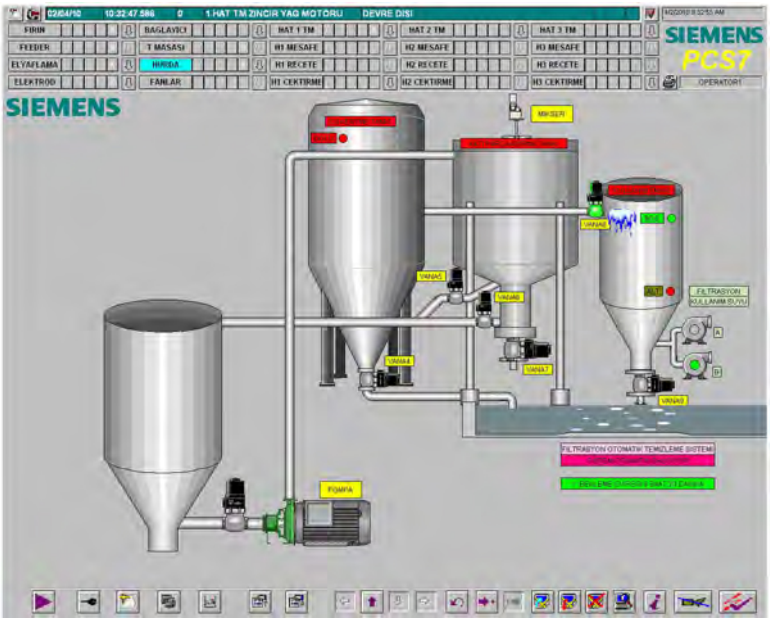
Uporabniški programi so sestavljeni iz programskih blokov, ki prinašajo nekaj prednosti, kot so lažje branje napisanih programov, standardizacijo

individualnih odsekov, enostavno programsko organizacijo in enostavno uveljavljanje sprememb programa. Odseke strukturiranega uporabniškega programa, ki ustrezajo tem nalogam, imenujemo uporabniški bloki. Poznamo več vrst uporabniških blokov [2]:

- organizacijski bloki (OB), so vmesniki med operacijskim sistemom in uporabniškim programom. Tu ima vsak blok svoj pomen in svojo nespremenljivo identifikacijsko številko,
- funkcije (FC) so pogosto ponavljajoči deli programa. Lastnosti jim določamo s klicnimi parametri, vračajo pa lahko funkcijsko vrednost, kot tudi dodatne izhodne parametre,
- podatkovni bloki (DB) vsebujejo podatke o uporabniškem programu, njihovo obliko pa lahko določamo sami,
- funkcijski bloki (FB), so del uporabniškega programa. Statično lahko hranijo lokalne spremenljivke v dodeljenem podatkovnem bloku. Funkcijski blok lahko v programu kličemo večkrat in vsak program ima lahko svoj podatkovni blok.

2.2.2 Siemens Simatic WinCC flexible

Siemens Simatic WinCC flexible je programsko okolje, ki omogoča izdelavo vmesnika za interakcijo človek – stroj (HMI). Ti vmesniki so se razvili kot uporabniku prijazen način prikaza delovanja sistema vodenja. Sistem SCADA zbira informacije iz krmilnikov in jih prikazuje na enostaven in pregleden način. Operaterju omogoča poseganje v proces in je pomemben element pri generiranju alarmov, ki pritegnejo pozornost operaterja in s tem omogočijo hitre reakcije. Nastopa lahko kot celosten sistem vodenja, a v tem primeru služi kot dodatek, saj nastopa kot vezni člen med procesom, operaterjem in višjimi sistemi vodenja [2].



Slika 2.5: Primer sistema SCADA, izdelanega z orodjem Siemens Simatic WinCC flexible.

2.2.3 Knjižnice Arduino

Kot večina platform, tudi okolje Arduino omogoča razširitev njegove funkcionalnosti z uporabo knjižnic, ki so napisane v programskem jeziku C ali C++. Veliko teh knjižnic dobimo že z integriranim razvojnim okoljem (IDE), najdemo jih tudi na spletni knjižnici Arduino, lahko pa jih razvijemo tudi sami. Pri izdelavi diplomske naloge smo uporabili knjižnice Serial, Ethernet in Settimino. Vse knjižnice moramo v našo programsko kodo vključiti in jih v funkciji `setup()` tudi inicializirati. Ta funkcija se požene ob zagonu mikrokrmilnika Arduino in določa začetno stanje sistema.

Knjižnica Serial

Knjižnica Serial je uporabljena za komunikacijo med mikrokrmilnikom Arduino in računalnikom ali drugimi napravami, ki podpirajo serijsko komunikacijo. Vsak mikrokrmilnik Arduino ima vsaj ena serijska vrata, ki so poznana

tudi pod imenom UART ali USART. Komunikacija lahko poteka preko digitalnih vhodov in izhodov ali preko vmesnika USB. Model mikrokrmilnika Arduino, ki smo ga izbrali, ima štiri serijska vrata. Tako moramo določiti, na katerih serijskih vratih je naš vmesnik Bluetooth Sparkfun BlueSMiRF Gold, jih v programsko kodo inicializirati in določiti hitrost (baud), s katero bodo delovala. Knjižnico Serial lahko najdemo že v integriranem razvojnem okolju Arduino.

Knjižnica Ethernet in Settimino

V kombinaciji z modulom Arduino Ethernet Shield, nam knjižnica Ethernet služi kot programska realizacija komunikacije preko protokola TCP/IP ali UDP. Mikrokrmilnik Arduino z modulom komunicira preko SPI vodila, ki se nahaja na digitalnih vhodih in izhodih. Knjižnica je že vključena v integrirano razvojno okolje Arduino in jo moramo v našo programsko kodo vključiti in inicializirati tako, da določimo omrežni naslov IP in naslov omrežne kartice MAC.

Knjižnica Settimino je nadgrajena odprtokodna knjižnica Ethernet za povezovanje industrijskih krmilnikov Siemens S7 in mikrokrmilnikov Arduino. Glavne prednosti te knjižnice so [6]:

- neposreden dostop do vseh podatkovnih blokov DB, ki se nahajajo na industrijskem krmilniku,
- neodvisnost od enote protokolnega podatka (PDU), kar pomeni, da je velikost prenesenih podatkov odvisna le od pomnilnika na mikrokrmilniku Arduino,
- uporabne funkcije za pretvorbo zapisa podatkov, saj so podatki na industrijskem krmilniku Siemens zapisani po pravilu debelega konca, na mikrokrmilniku Arduino po pravilu tankega konca,
- da potrebuje 3 ms za branje in pisanje ene enote protokolnega podatka (PDU) v notranji medpomnilnik in 24 ms za zapis 1024 bitov na zunanji

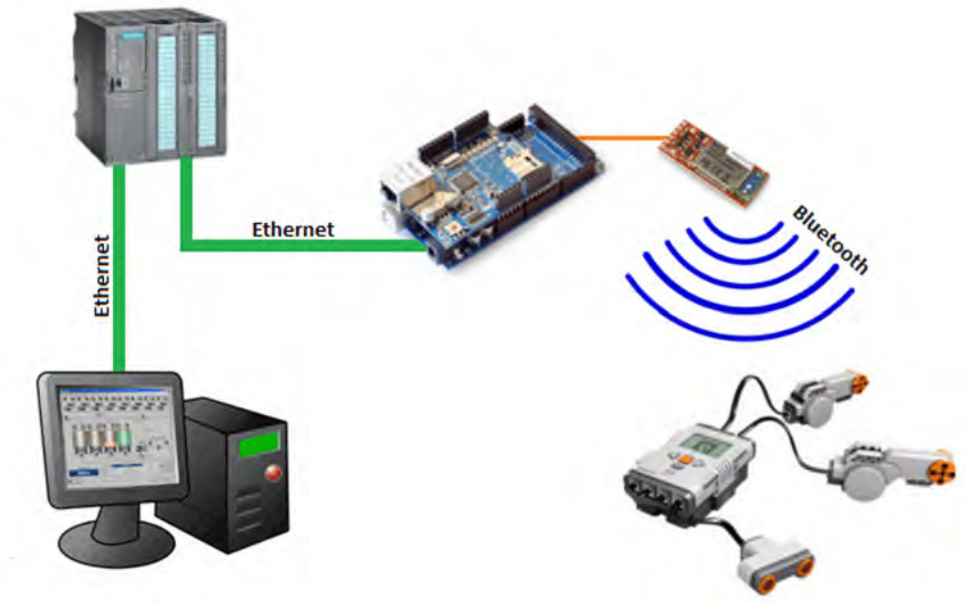
pomnilnik,

- da uporablja standardno knjižnico Arduino Ethernet in nam tako omogoča, da lahko vzporedno tečejo tudi drugi strežniki ali odjemalci, ki uporabljajo to knjižnico,
- da pozna tri spominske module, ki nam omogočajo optimizacijo porabe virov na mikrokrmilniku Arduino.

Poglavje 3

Izvedba komunikacije med industrijskim krmilnikom Siemens in Lego Mindstorms NXT

Izvedbo komunikacije med industrijskim krmilnikom Siemens in Lego Mindstorms NXT lahko vidimo na sliki 3.1. Model Lego Mindstorms NXT je v kombinaciji z izdelanim komunikacijskim vmesnikom, pasivna naprava in čaka na ukaze mikrokrmilnika Arduino Mega 2560 R3, ki z njim upravlja s pošiljanjem neposrednih ukazov preko Bluetooth vmesnika. Arduino Mega 2560 R3 predstavlja osnovo za komunikacijski vmesnik, ki povezuje industrijski krmilnik Siemens in model Lego Mindstorms NXT. Da smo to dosegli, smo morali mikrokrmilnik Arduino nadgraditi z dodatnima moduloma Arduino Ethernet in Bluetooth. Na komunikacijskem vmesniku smo razvili tudi potrebno programsko kodo za integracijo obeh sistemov. S tem smo razvili splošno namenski komunikacijski vmesnik, ki ga lahko integriramo v druge aplikacije.

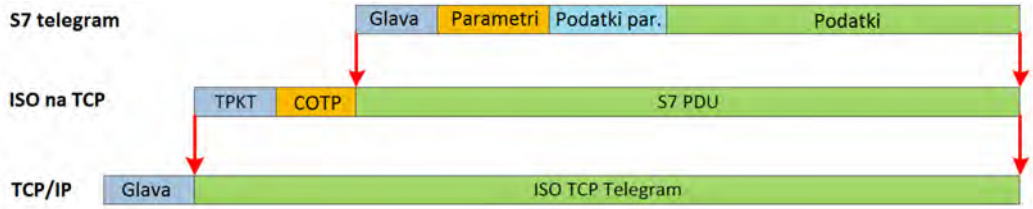


Slika 3.1: Povezave med uporabljenimi komponentami.

3.1 Povezava med industrijskim krmilnikom Siemens in mikrokrmilnikom Arduino

3.1.1 Protokol Siemens S7 in knjižnica Settimino

Protokol Siemens S7 je hrbtenica komunikacije Siemens in je funkcijsko ali ukazno orientiran. Njegova implementacija temelji na ISO TCP (RFC1006), ki je blokovno orientirana. Vsak blok se imenuje enota protokolnih podatkov ali PDU in predstavlja maksimalno dolžino prenesenega bloka. Velikost teh blokov je lahko med 240 in 960 biti. V primeru, da podatki presegajo velikost PDU enote, se ta razdeli čez več PDU enot. Vsak ukaz je sestavljen iz glave, zaporedja parametrov, podatkov o parametrih in podatkovnega okvirja. Protokoli S7, ISO TCP in TCP/IP sledijo dobro poznani enkapsulaciji podatkov. Enkapsulacija je povezava med protokoli, ki delujejo na različnih plasteh modela OSI. Ta deluje tako, da različne plasti prevzamejo pakete iz višjih plasti kot podatkovni del in jim dodajo svojo glavo.



Slika 3.2: Enkapsulacija protokolov.

Funkcije v knjižnici Settimo, so razdeljene v kategorije za podatkovno branje/pisanje, ciklično podatkovno branje/pisanje, v informacije o mapah, v sistemske informacije, premikanje podatkovnih blokov, nadzor industrijskega krmilnika, varnost, programiranje, čas in datum.

Slika 3.3: Kodiranje 16 bitne napake.

Knjižnica Settimino omogoča optimizacijo porabe virov mikrokrmilnika Arduino z uporabo spominskih modulov. Pozna majhen, normalen in razširjen spominski modul. Vsak našteti, razširja modul pred njim z dodatnimi funkcijami, katerega bomo uporabili, pa določimo v sami knjižnici Settimino. V našem komunikacijskem vmesniku smo uporabili normalen spominski modul, ki vsebuje razred S7Helper. Razred S7Helper nam omogoča ekstrakcijo podatkov iz industrijskega krmilnika v pravilnem formatu. Glavni razred knjižnice Settimino je S7Client. To je objekt, ki se lahko povezuje in prenaša podatke med industrijskim krmilnikom Siemens in mikrokrmilnikom Arduino. Čip W5100 Ethernet, ki se nahaja v Arduino Ethernet Shieldu, je omejen na največ štiri instance objektivov EthernetClient. Vsak klient se nahaja na svoji vtičnici, kljub temu pa tega na mikrokrmilniku Arduino ne moremo izkoristiti, saj okolje ni večopravilno.

Memonik	HEX koda	Pomen
errTCPConnectionFailed	0x0001	Napaka v povezavi TCP
errTCPConnectionReset	0x0002	Povezava ponastavljena s strani partnerja
errTCPDataRecvTout	0x0003	Potek časovne omejitve ob čakanju na odgovor
errTCPDataSend	0x0004	Gonilnik Ethernet je vrnil napako ob pošiljanju podatkov
errTCPDataRecv	0x0005	Gonilnik Ethernet je vrnil napako ob prejemanju podatkov
errISOConnectionFailed	0x0006	Neuspešna ISO povezava
errISONegotiatingPDU	0x0007	Neuspešna ISO PDU pogajanja
errISOInvalidPDU	0x0008	Sprejet poškodovan PDU

Tabela 3.1: Tabela napak protokola TCP/IP

Memonik	HEX koda	Pomen
errTCPConnectionFailed	0x0100	Sprejet neveljaven PDU
errTCPConnectionReset	0x0200	Napaka pri pošiljanju PDU
errTCPDataRecvTout	0x0300	Napaka med branjem podatkov
errTCPDataSend	0x0400	Napaka med pisanjem podatkov
errTCPDataRecv	0x0500	Industrijski krmilnik je vrnil napako klicane funkcije
errISOInvalidPDU	0x0600	Medpomnilnik premajhen za prejete podatke

Tabela 3.2: Tabela napak protokola S7

Večina funkcij, ki jih uporablja knjižnica Settimino, vračajo parametre napak, če bi se le-te zgodile. Rezultat napake je 16 bitna koda, katere sestava je prikazana na sliki 3.3. Prvih osem bitov oz. prvi bajt predstavlja napako,

ki jo je povzročil protokol TCP/IP, opisi vseh možnih napak so prikazani v tabeli 3.1. Drugi bajt predstavlja napako v S7 protokolu, opisi vseh možnih napak so prikazani v tabeli 3.2.

3.1.2 Povezava med industrijskim krmilnikom Siemens in mikrokrmilnikom Arduino

Povezava med industrijskim krmilnikom Siemens in mikrokrmilnikom Arduino je programsko realizirana na samem mikrokrmilniku. Z uporabo knjižnice Settimino lahko iz mikrokrmilnika Arudino neposredno dostopamo do vseh podatkovnih blokov (DB) na industrijskem krmilniku Siemens. Na sliki 3.4 lahko vidimo sestavo uporabniško določene strukture UDT1. Ker mora komunikacijski vmesnik vnaprej vedeti, s katerim podatkovnim blokom bo izmenjeval podatke, smo v ta namen uporabili podatkovni blok DB9, katerega smo definirali z uporabniško določeno strukturo UDT1. Podatkovni blok uporabljamo za hranjenje in medsebojno izmenjavo podatkov med industrijskim krmilnikom Siemens in mikrokrmilnikom Arduino.

UDT1 -- "NXTNode" -- CommandNXTviaArduino\SIMATIC 300(1)\CPU 314C

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	NODE_ID	INT	0
+2.0	STAUS	INT	0
+4.0	SERVO_1_STATUS	INT	0
+6.0	SERVO_1_VALUE	WORD	W#16#0
+8.0	SERVO_2_STATUS	INT	0
+10.0	SERVO_2_VALUE	WORD	W#16#0
+12.0	SERVO_3_STATUS	INT	0
+14.0	SERVO_3_VALUE	WORD	W#16#0
+16.0	ULTRASONIC_VALUE	REAL	0.000000e+000
+20.0	USER_VALUE_1	INT	1
+22.0	USER_VALUE_2	INT	5
=24.0		END_STRUCT	

Slika 3.4: Sestava uporabniško določene strukture UDT1.

Informacije, ki jih hranimo v uporabniško določeni strukturi:

- NODE_ID – v tej spremenljivki se hrani identifikacijska številka modela Lego Mindstorms NXT, s katerim se bo ali je povezan mikrokrmilnik Arduino preko povezave Bluetooth,
- STATUS – ta spremenljivka hrani trenutni status povezave. Nahaja se lahko v statusu 0, ki nam pove, da je celoten komunikacijski vmesnik trenutno izklopljen. Lahko se nahaja tudi v statusu 2, ta nam pove, da je komunikacijski vmesnik vklopljen, a trenutno ni vzpostavljene nobene aktivne povezave Bluetooth med mikrokrmilnikom Arduino in Lego Mindstorms NXT. Če je spemenljivka v statusu 1, se povezava z izbranim Lego Mindstorms NXT vzpostavlja in preide v status 3, ko je povezava uspešno vzpostavljena,
- SERVO_1_STATUS – je statusna spremenljivka, ki nam pove ali naj se servomotor, ki je priključen na prva izhodna vrata Lego Mindstorms NXT, vklopi in v katero smer naj se vrti,
- SERVO_1_VALUE – določa koliko časa naj se servomotor na prvih izhodnih vratih vrti od vklopa (število sekund),
- SERVO_2_STATUS - je statusna spremenljivka, ki nam pove ali naj se servomotor, ki je priključen na druga izhodna vrata Lego Mindstorms NXT, vklopi in v katero smer naj se vrti,
- SERVO_2_VALUE – določa koliko časa naj se servomotor na drugih izhodnih vratih vrti od vklopa (število sekund),
- SERVO_3_STATUS - je statusna spremenljivka, ki nam pove ali naj se servomotor, ki je priključen na tretja izhodna vrata Lego Mindstorms NXT, vklopi in v katero smer naj se vrti,
- SERVO_3_VALUE – določa koliko časa naj se servomotor na tretjih izhodnih vratih vrti od vklopa (število sekund),

- ULTRASONIC_VALUE – predstavlja zadnjo zajeto vrednost iz ultrazvočnega senzorja (v centimetrih). Ultrazvočni senzor mora biti priključen na prvi vhodna vrata Lego Mindstorms NXT. Spremenljivka je zgolj izhodna,
- USER_VALUE_1 – je dodatna vhodno/izhodna spremenljivka,
- USER_VALUE_2 – je dodatna vhodno/izhodna spremenljivka.

Vrednosti, ki jih lahko zavzame posamezna SERVO_X_STATUS spremenljivka so lahko med 0 in 2 (0 – izklopljen, 1 – vrtenje v smeri urinega kazalca in 2 – vrtenje v nasprotni smeri urinega kazalca). Naš komunikacijski vmesnik omogoča tudi uporabo časovnika za posamezen servomotor. Čas vrtenja posameznega servomotorja določimo s spremenljivko SERVO_X_VALUE. V primeru, da je vrednost v spremenljivkah SERVO_X_VALUE enaka 0, potem bo servomotor vklopljen oziroma izklopljen do spremembe statusne spremenljivke posameznega motorja, kar pomeni, da se časovnik v tem primeru ne uporablja.

Naš program na komunikacijskem vmesniku ciklično preverja, ali je povezava trenutno vzpostavljena s funkcijo Settimino `Connected()`. Program bo v primeru, da je odziv funkcije negativen, poskušal vzpostaviti povezavo. Za vzpostavitev povezave z industrijskim krmilnikom Siemens, smo uporabili funkcijo Settimino `ConnectTo()`. Funkcija vzpostavi povezavo med odjemalcem Arduino in industrijskim krmilnikom:

- deklaracija: `int ConnectTo (IPAddress Address, uint16_t Rack, uint16_t Slot)`,
- vhodni parametri:

	Vrsta	**Tip**	**Namen**
`Address`	`IPAddress razred`	Vhodni	PLK IPv4 omrežni naslov
`Rack`	`uint16_t`	Vhodni	Številka PLK držala
`Slot`	`uint16_t`	Vhodni	Številka PLK reže

- funkcija vrača vrednost 0, v primeru uspešno vzpostavljene povezave, ostale vrednosti pa predstavljajo napake definirane v tabeli 3.1.

Vhodna parametra `Rack` (0..7) in `Slot` (1..31) predstavljata nastavitve strojne opreme na industrijskem krmilniku. Krmilniki iz družine S7 300 CPU uporabljajo Rack 0 in Slot 2. Vhodni parameter `Address` predstavlja omrežni naslov naprave, s katero želimo vzpostaviti povezavo.

Ko je povezava uspešno vzpostavljena, lahko pričnemo z neposrednim zajemanjem podatkov iz industrijskega krmilnika Siemens s funkcijo Settimino `ReadArea()`. Funkcija nam omogoča tudi zajemanje podatkov iz vhodov, izhodov, markerjev, časovnikov ter števcev krmilnika Siemens.

- deklaracija: `int ReadArea (int Area, uint16_t DBNumber, uint16_t Start, uint16_t Amount, void *pUsrData)`,
- vhodni parametri:

	Vrsta	**Tip**	**Namen**
`Area`	`int`	Vhodni	Območni identifikator
`DBNumber`	`uint16_t`	Vhodni	Številka DB, ki ga želimo brati
`Start`	`uint16_t`	Vhodni	Odmik od začetka
`Amount`	`uint16_t`	Vhodni	Količina besed, ki jih želimo prebrati
`pUsrData`	`Kazalec`	Vhodni	Kazalec na pomnilniško lokacijo

- funkcija vrača vrednost 0, v primeru uspešnega prenosa podatkov, ostale vrednosti pa predstavljajo napake definirane v tabeli 3.2.

Vsaka izmenjava podatkov z industrijskim krmilnikom mora ustrezati velikosti PDU, čigar velikost je vnaprej določena z 240 biti. V primeru, da je paket večji, se ta razdeli na več PDU enot. Ker podatkovni blok DB9, ki bo uporabljal uporabniško definirano strukturo UDT1, s 192 biti ne presegal velikosti ene enote PDU, lahko ta parameter nastavimo kot `NULL`, preneseni podatki pa se bodo shranili v medpomnilniško lokacijo PDU. Območni

	Vrednost	**Namen**
`S7AreaPE`	`0x81`	Procesni vhodi
`S7AreaPA`	`0x82`	Procesni izhodi
`S7AreaMK`	`0x83`	Spominski biti
`S7AreaDB`	`0x84`	Podatkovni bloki
`S7AreaCT`	`0x1C`	Števci
`S7AreaTM`	`0x1D`	Časovniki

Tabela 3.3: Tabela območnih identifikatorjev

identifikator nam pove, katere vrste podatkov naj funkcija zajame. Možne vrednosti območnega identifikatorja lahko vidimo v tabeli 3.3.

S poznavanjem sestave zajetega podatkovnega bloka, lahko iz medpomnilnika PDU razberemo posamezne spremenljivke in jih lokalno shranimo. Za te potrebe smo uporabili funkciji Settimino `IntegerAt()` in `BitAt()`.

- deklaracija: `integer IntegerAt(int Index)`,
- vhodni parametri:

	Vrsta	**Tip**	**Namen**
`Index`	`int`	Vhodni	Pozicija začetnega bajta

- funkcija vrne Integer podatkovni tip (16 bitov) za dano pozicijo iz medpomnilnika PDU.

Ker so v medpomnilniku PDU podatki zapisani po pravilu debelega konca, jih navedene funkcije pred zapisom v lokalne spremenljivke pretvorijo v podatke, ki so zapisani po pravilu tankega konca.

- deklaracija: `bool BitAt(int ByteIndex, byte BitIndex)`,

- vhodni parametri:

	Vrsta	**Tip**	**Namen**
`ByteIndex`	`int`	Vhodni	Pozicija začetnega bajta
`BitIndex`	`Byte`	Vhodni	Bit znotraj prebranega bajta

- funkcija vrne bit na želeni lokaciji v prebranem bajtu.

Nekatere spremenljivke so vhodne/izhodne, nekatere pa samo izhodne, kar pomeni, da moramo poskrbeti, da se podatki zapišejo tudi nazaj v podatkovni blok (DB9). Spremenljivka ULTRASONIC_VALUEi je izhodnega tipa in hrani zadnjo vrednost, ki je bila zajeta iz ultrazvočnega senzorja. Spremenljivke statusov servomotorjev pa so vhodno/izhodne, saj mora v primeru uporabe časovnika, komunikacijski vmesnik poskrbeti tudi za izklop servomotorja in nastaviti pravilno vrednost statusa servomotorja. Za te potrebe smo uporabili funkcijo Settimino `WriteArea()`:

- deklaracija: `int WriteArea(int Area, uint16_t DBNumber, uint16_t Start, int Amount, void *pUsrData)`,
- vhodni parametri:

	Vrsta	**Tip**	**Namen**
`Area`	`int`	Vhodni	Območni identifikator 3.3
`DBNumber`	`uint16_t`	Vhodni	Številka DB, v katerega želimo pisati
`Start`	`uint16_t`	Vhodni	Odmik od začetka, začetna pozicija
`Amount`	`uint16_t`	Vhodni	Količina besed, ki jih želimo zapisati
`pUsrData`	`Kazalec`	Vhodni	Kazalec na podatke, ki jih želimo zapisati

- je glavna funkcija za zapis podatkov nazaj na industrijski krmilnik. Funkcija je komplementarna funkciji `ReadArea()`.

V primeru, da hočemo na industrijski krmilnik Siemens zapisati več kot en bit, moramo nad podatki opraviti pretvorbo, da podatke zapisane po pravilu tankega konca, zapišemo s podatki zapisanimi po pravilu debelega konca. Funkcije za tako pretvorbo, knjižnica Settimino ne vsebuje in smo jo zato morali izdelati sami.

3.2 Povezava med mikrokrmilnikom Arduino in Lego Mindstorms NXT

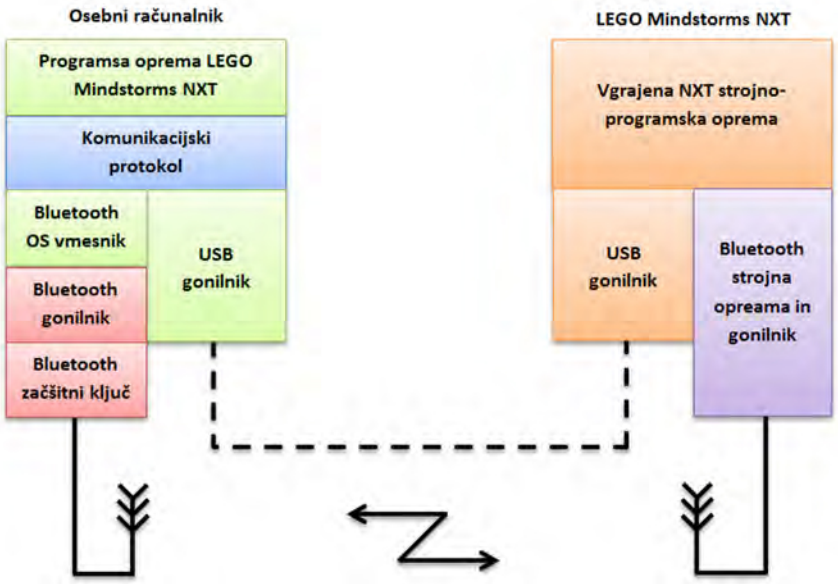
3.2.1 Komunikacijski protokol Lego Mindstorms NXT

Komunikacijski protokol Lego Mindstorms NXT omogoča komunikacijo preko Bluetooth ali USB. Poleg teh dveh glavnih komunikacijskih protokolov pozna še dva komunikacijska vmesnika, ki sta v osnovi namenjena komunikaciji s perifernimi napravami:

- ena digitalna komunikacijska vrata, s komunikacijsko hitrostjo do 1 Mb/s (uporaba teh vrat omogoča komunikacije s perifernimi napravami, ki zahtevajo visoke komunikacijske hitrosti),
- štiri digitalna komunikacijska vrata, s komunikacijsko hitrostjo do 9600 b/s (vrata uporabljajo komunikacijski standard I2C).

Slika 3.5 prikazuje glavne komunikacijske plasti med modelom Lego Mindstorms NXT in programsko opremo na osebnem računalniku. Omogočen je tudi neposreden dostop do komunikacijskega medpomnilnika, v katerega lahko pišemo ali beremo. Vmesnik Bluetooth je na modelu Lego Mindstorms NXT samostojen čip CSR BlueCore4 z 8 Mb bliskovnega pomnilnika. Ukaze prejema skozi vmesnik UART, ki je povezan s procesorjem ARM7. Vmesnik Bluetooth komunicira preko serijskih vrat, ki pa prinašajo dve težavi. Prva težava je pri večjem podatkovnem paketu, ki bo s strani procesorja ARM prispel z manjšimi časovnimi razlikami, druga pa je, ko mora čip BlueCore

preklopiti med načinom za sprejemanje in razpošiljanje podatkov, pri tem pa nastane časovna kazen (približno 30 ms).



Slika 3.5: Komunikacijski blok.

Za realizacijo komunikacije med mikrokrmilnikom Arduino in modelom Lego Mindstorms NXT smo uporabili komunikacijo Bluetooth, ki podatke izmenjuje s telegrami. Telegram vsebuje podatke o ukazu in njegovih parametrih. Ukaze, ki jih Lego Mindstorms uporablja, razdelimo na sistemske in neposredne. Sistemske uporabljamo za administracijo modelov Lego Mindstorms, saj nam omogočajo upravljanje z datotekami in nastavitve komunikacije, medtem ko neposredne ukaze uporabljamo za upravljanje s senzorji in aktuatorji Lego. Vsi ukazi omogočajo način z odzivom. Ko pri komunikaciji Bluetooth uporabljamo takšne ukaze, mora med odzivom in naslednjim ukazom preteči 30 ms [4]. Sestavo telegrama Bluetooth lahko vidimo na sliki 3.6.

Slika 3.6: Sestava telegrama Bluetooth.

Prva dva bajta v telegramu uporabljamo za določanje dolžine telegrama, tretji bajt pa je namenjen določanju tipa ukaza. Lego Mindstorms NXT pozna štiri vrste ukazov. Poznamo neposredne ukaze z odzivom (0x00), sistemske ukaze z odzivom (0x01), neposredne ukaze brez odziva (0x80) in sistemske ukaze brez odziva (0x81). Četrti bajt predstavlja sam ukaz, ki ga želimo poslati v izvedbo. Od petega bajta naprej pa nastopajo parametri posameznega ukaza. Največja možna dolžina telegrama Bluetooth, ki ga lahko pošljemo, je 64 bajtov [4].

3.2.2 Vzpostavitev povezave med mikrokrmilnikom Arduino in Lego Mindstorms NXT

Vzpostavitev povezave med mikrokrmilnikom Arduino in Lego Mindstorms NXT smo realizirali preko komunikacije Bluetooth. Naš komunikacijski vmesnik deluje tako, da pred vsako vzpostavitvijo, preveri trenutni status povezave. V primeru, da je povezava že vzpostavljena z drugim modelom Lego Mindstorms NXT, mora naš komunikacijski vmesnik najprej poskrbeti za prekinitev povezave s trenutnim modulom. Bluetooth, ki smo ga uporabili mi, lahko deluje v ukaznem ali podatkovnem načinu. Za potrebe vzpostavitve, nastavitve in prekinitve povezave, moramo najprej vstopiti v ukazni način. Primer vzpostavitve povezave, v programski kodi, lahko vidimo na sliki 3.7. Na vsak ukaz, ki ga pošljemo v ukaznem načinu, nam modul Bluetooth tudi odgovori in ker so te informacije pomembne za nadaljnjo izvajanje programa na komunikacijskem vmesniku, moramo na odgovor tudi počakati. Komunikacijski vmesnik ima omejeno število poskusov vzpostavitve Bluetooth, saj se lahko zgodi, da želen Lego Mindstorms NXT ni v dosegu.

```
// start serial communication for bluetooth on Serial1 pins( RX1(ardui
while(!readStringFromBt().equals("CMD")){  // wait to enter command mo
  bluetoothSerial.print("$$$");
  delay(btCommandDelay);
}

while(true){  // attempt to connect to NXT
  bluetoothSerial.println("GK");  // get status of the connection 0->d
  delay(btCommandDelay);
  while(true){  // wait for bluetooth to respond to GK command
    connectionStatus = readStringFromBt();
    if(connectionStatus.equals("CONNECT failed")
                || connectionStatus.equals("0")
                || connectionStatus.equals("4")){
      break;
    }
    else{
      delay(btCommandDelay);
    }
  }
  if(connectionStatus.equals("0")){ //try establish connection to NXT
    bluetoothSerial.println(currNode); //MAC address of NXT -> located
    delay(btCommandDelay);
  }
  else if(connectionStatus.equals("4")){  // exit of already connected
    break;
  }
}

delay(btCommandDelay);
while(!readStringFromBt().equals("END")){ //exit command mode -> wait
  bluetoothSerial.println("---");
  delay(btCommandDelay);
}
```

Slika 3.7: Del programske kode za vzpostavitev povezave Bluetooth.

Po uspešni vzpostavitvi povezave Bluetooth, je Lego Mindstorms pripravljen na sprejemanje neposrednih ukazov. Izdelan komunikacijski vmesnik vključuje možnost krmiljenja do treh servomotorjev in možnost zajema podatkov iz enega ultrazvočnega senzorja. Primer telegramov za krmiljenje servomotorja lahko vidimo na sliki 3.8. Ta nam prikazuje tri vrste telegramov. Pri prvem telegram se servomotor vklopi in začne vrteti v smeri urinega kazalca, pri drugem se servomotor prav tako vklopi in začne vrteti v nasprotni smeri urinega kazalca. Tretji telegram pa vrtenje servomotorja vstavi. Prvi telegram, ki sproži vrtenje servomotorja v smeri urinega kazalca, je sestavljen iz zaporedja bajtov:

- `0x0C 0x00`, ki predstavljata dolžino telegrama,
- `0x80` nam pove, da od prejemnika ne zahtevamo odgovora. V našem primeru odgovora na premik servomotorja ne uporabljamo, saj bi s tem poslabšali odziv sistema,
- `0x04` nam pove tip ukaza, ki v tem primeru prestavlja neposreden ukaz za servomotorje,
- `0x00` predstavlja številko vrat, na katera je priključen naš servomotor,
- `0xCE` s tem bajtom nastavimo izhodno moč v odstotkih maksimalne moči in smeri servomotorja. V tem primeru je to 75 % moči servomotorja, v smeri urinega kazalca,
- `0x07` definira način delovanja motorja. Izbrani način omogoča neposredno ustavitev rotacije servomotorja in s tem bolj natančno rotacijo servomotorja,
- `0x00` predstavlja regulacijski bajt, ki nastavi način, kako naj NXT strojna – programska oprema (angl. firmware) samodejno prilagaja hitrost, z dodajanjem hitrosti, glede na zunanjo bremenitev servomotorja, da bi s tem dosegli želeno hitrost vrtenja,

- `0x20` definira rotacijsko stopnjevanje moči. S tem bajtom nastavimo, da se moč servomotorja ob zagonu stopnjuje do želene izhodne moči. Mi te funkcije nismo uporabili,
- `0x00 0x00 0x00 0x00` v kombinaciji z bajtom za rotacijsko stopnjevanje moči definira, čez kakšno časovno periodo naj se stopnjevanje izhodne moči izvede.

```
// Servo one on port A
byte ServoOnetCW[14] =   {0x0C, 0x00, 0x80, 0x04, 0x00,
                          0xCE, 0x07, 0x00, 0x00, 0x20,
                          0x00, 0x00, 0x00, 0x00};
byte ServoOneCCW[14] =   {0x0C, 0x00, 0x80, 0x04, 0x00,
                          0x32, 0x07, 0x00, 0x00, 0x20,
                          0x00, 0x00, 0x00, 0x00};
byte ServoOneStop[14] =  {0x0C, 0x00, 0x80, 0x04, 0x00,
                          0x32, 0x07, 0x00, 0x00, 0x00,
                          0x00, 0x00, 0x00, 0x00};
```

Slika 3.8: Primer telegrama za neposredno upravljanje servomotorja Lego.

Poizvedovanje po podatkih iz ultrazvočnega senzorja je nekoliko kompleksnejše, saj moramo senzor pred uporabo nastaviti na željen način delovanja, nato pa lahko pričnemo z zajemom podatkov. Senzor omogoča več načinov delovanja. Prvi način je enkratna meritev. Omogoča nam, da ob vsaki poizvedbi vrne vrednost prve meritve, ki jo je senzor izvedel v tem načinu delovanja. Drugi način je neprekinjena meritev. Ta ob vsaki poizvedbi izvede novo meritev in nam le-to posreduje. Tretja možnost je lovljenje dogodka. Ta način čaka na našo poizvedbo in ko jo prejme, začne z zajemom. Odziv na našo poizvedbo dobimo takrat, ko senzor v svojem dometu zazna neko spremembo. Naš komunikacijski vmesnik uporablja neprekinjen način meritve. Modelu Lego Mindstorms moramo, poleg načina meritve, sporočiti tudi na katera vrata je senzor priključen. Ko je senzor nastavljen, lahko začnemo s poizvedbo po podatkih. Ta poteka v dveh fazah. V prvi fazi moramo poslati ukaz, ki od Lego Mindstorms zahteva ultrazvočno meritev. Lego Mindstorms

to meritev izvede in podatke odloži na izhodno medpomnilniško lokacijo. Ko so podatki pripravljeni nam pošlje odgovor. Tukaj se lahko začne druga faza, kateri pošljemo ukaz za zajem podatkov na medpomnilniški lokaciji. Lego Mindstorms pa nam preko telegrama te podatke posreduje. Naša programska koda na komunikacijskem vmesniku skrbi tudi za filtriranje slabih telegramov in meritev. Zgodi se namreč lahko, da prejmemo telegram, ki ni popoln ali da Lego Mindstorms iz ultrazvočnega senzorja zajame negativno meritev in zato takih podatkov ne smemo uporabiti, saj lahko škodijo sistemu, ki integrira ta splošni komunikacijski vmesnik. S tem zagotovimo, da so vrednosti, ki se zapišejo v izhodno spremenljivko ULTRASONIC_VALUE, v merilnem območju ultrazvočnega senzorja.

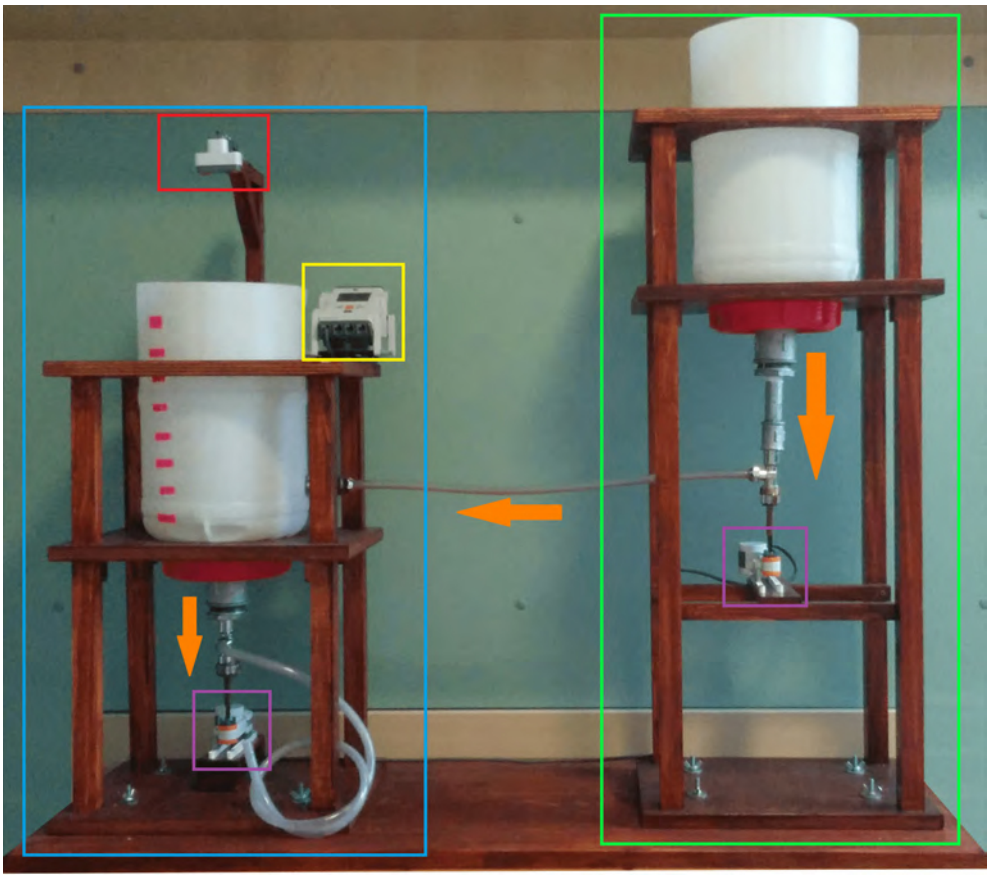
```
bluetoothSerial.write(ultrasonic1CommandRead, sizeof(ultrasonic1CommandRead));
while((dataLenght = bluetoothSerial.available()) < 5){
  delay(5);
}

canRead[dataLenght];
bluetoothSerial.readBytes(canRead, dataLenght);

// read only if read response positive
if ((int)canRead[4] == 0){
  bluetoothSerial.write(ultrasonic1Read, sizeof(ultrasonic1Read));
  readRequest = true;
}

// wait for data to arive from ultrasonic sensor
while(readRequest){
  if((dataLenght = bluetoothSerial.available()) > ultrasonicBtTelegramSize-1){
    readed[dataLenght];
    bluetoothSerial.readBytes(readed, dataLenght);

    if((int)readed[6] <= 0) {
      // skip bad measurements
      readRequest = false;
      break;
    }
    else{
      //check if recieved measurment is in Ultrasonic range

      tmp_S7Ultrasonic = (int)readed[6];
```

Slika 3.9: Del programske kode za poizvedbo po ultrazvočni meritvi.

Poglavje 4

Simulacija industrijskega silosa in analiza rezultatov

Komunikacijski vmesnik za povezavo med industrijskim krmilnikom Siemens Lego Mindstorms NXT smo na koncu nadgradili s smiselno simulacijo industrijskega silosa. Za simulacijo industrijskega silosa smo se odločili na podlagi meritev časa odzivnosti in natančnosti ultrazvočnega senzorja, saj so meritve pokazale, da se v primeru visoke dinamike merjenih objektov, senzor ne odzove najbolje. Simulacijo industrijskega silosa, smo realizirali z izdelavo makete (slika 4.1). Na koncu smo uporabili še Siemens Simatic WinCC, s pomočjo katerega smo izdelali vmesnik za interakcijo človek – stroj (HMI). Vmesnik je uporabniku omogočil spremljanje in določanje količine tekočine v silosu, povezovanje na različne modele Lego Mindstorms ter izbiro med dvotočkovno regulacijo in regulacijo PID.



Slika 4.1: Maketa industrijskega silosa.

4.1 Simulacija industrijskega silosa

Maketa industrijskega silosa, ki jo lahko vidimo na sliki 4.1, je razdeljena na več komponent. Osrednji del makete sta dva silosa. Silos na levi strani (označen z modro barvo) predstavlja industrijski silos, medtem ko desni silos (označen z zeleno barvo) služi za simulacijo pritiska. Ta omogoča pretok tekočine v levi silos. Meritve in regulacijo nad industrijskim silosom smo izvajali z modelom Lego Mindstorms NXT 2.0 (označen z rumeno barvo). Za potrebe simulacije industrijskega silosa smo potrebovali ultrazvočni senzor (označen z rdečo barvo), ki smo ga namestili nad silos in z njim izva-

jali meritve oddaljenosti gladine. Potrebovali pa smo tudi dva servomotorja (označena z vijolično barvo), ki sta nam omogočala zvezno upravljanje vodnih ventilov. Senzorje in aktuatorje na maketi upravljamo z modelom Lego Mindstorms NXT. Z Lego Mindstorms neposredno upravlja Arduino MEGA 2560, ki preko povezave Bluetooth pošilja ukaze za zajem podatkov iz ultrazvočnega senzorja, lahko pa tudi neposredno upravlja s servomotorjema. Povezave med posameznimi komponentami, lahko vidimo na sliki 3.1. Meritve in spremembe regulacije lahko uporabnik spremlja in upravlja preko vmesnika človek – stroj. Vmesnik omogoča tudi povezovanje na druge modele Lego Mindstorms, ki so mu v dosegu. Vse potrebne informacije o silosu se hranijo na industrijskem krmilniku Siemens, v podatkovnem bloku DB9. Posamezne spremenljivke smo uporabili v naslednje namene:

- NODE_ID – identifikacijska številka modela Lego Mindstorms NXT,
- STATUS – ta spremenljivka hrani trenutni status povezave,
- SERVO_1_STATUS – nam pove ali je ventil za pretok tekočine iz desnega v levi silos odprt,
- SERVO_1_VALUE – hrani trenutno količino pretoka v primeru odprtega vhodnega ventila,
- SERVO_2_STATUS – nam pove ali je ventil za odtok tekočine iz industrijskega silosa trenutno odprt,
- SERVO_2_VALUE – hrani trenutno količino pretoka v primeru odprtega odtočnega ventila,
- SERVO_3_STATUS – tretji servomotor na maketi ni prisoten, zato te spremenljivke ne uporabljamo,
- SERVO_3_VALUE – tretji servomotor na maketi ni prisoten, zato te spremenljivke ne uporabljamo,

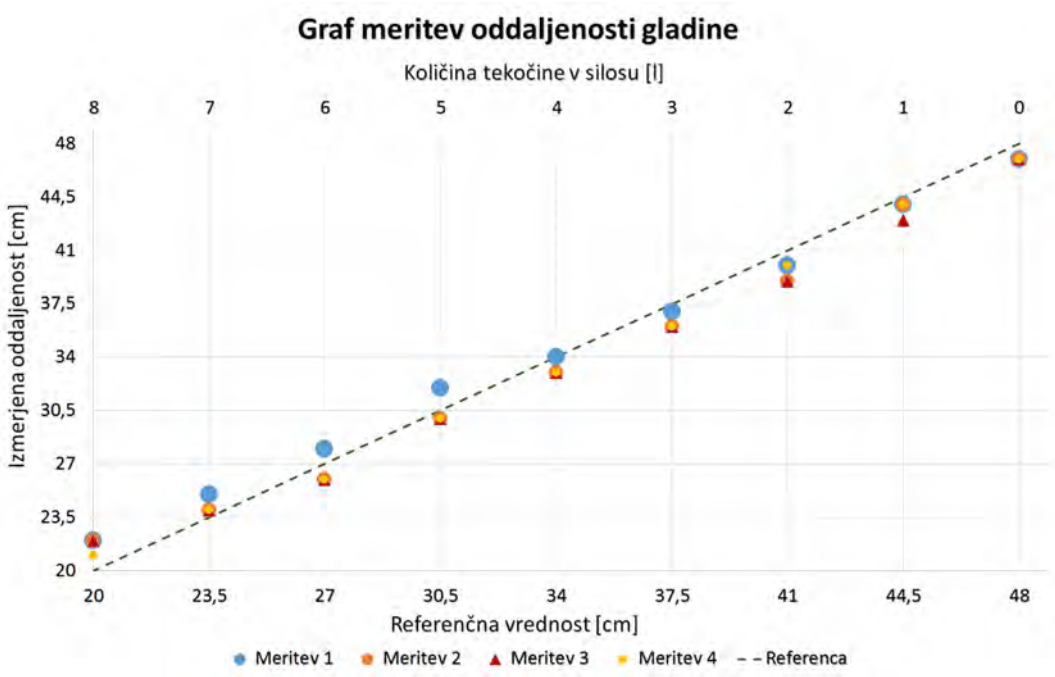
- ULTRASONIC_VALUE – predstavlja oddaljenost gladine tekočine v silosu,
- USER_VALUE_1 – nam pove trenutno vrsto regulacije, ki se uporablja na industrijskem silosu (dvotočkovna regulacija ali regulacija PID),
- USER_VALUE_2 – hrani referenčno vrednost količine tekočine v silosu, ki jo želimo regulirati.

Ob zagonu sistema moramo tako poskrbeti za vzpostavitev začetnega stanja sistema. Poskrbeti moramo, da so ventili v začetnem stanju in poenostaviti spremenljivko stanja povezave. Ker neposredni ukazi ne omogočajo pošiljanja števila želenih rotacij ali stopinj, smo zato poleg neskončnega vrtenja servomotorja, izdelali tudi časovno določeno vrtenje. Ali se servomotor ob vklopu vrti neprestano, določa posamezna spremenljivka servomotorja SERVO_X_VALUE. Ko je vrednost te spremenljivke 0, bo servomotor tekel neprekinjeno, v nasprotnem primeru pa spremenljivka predstavlja časovni zamik v sekundah, med vklopom in izklopom vrtenja servomotorja. Ker zahtevnejših operacij na industrijskem krmilniku Siemens nismo izvajali, smo zato uporabili programski jezik LAD. Za potrebe vzpostavitve začetnega stanja sistema, smo uporabili organizacijski blok OB100. Ta blok nastavi začetno stanje spremenljivke `STATUS`, saj povezava med mikrokrmilnikom Arduino in modelom Lego Mindstorms ob zagonu ni vzpostavljena.

4.2 Odzivnost in natančnost ultrazvočnega senzorja

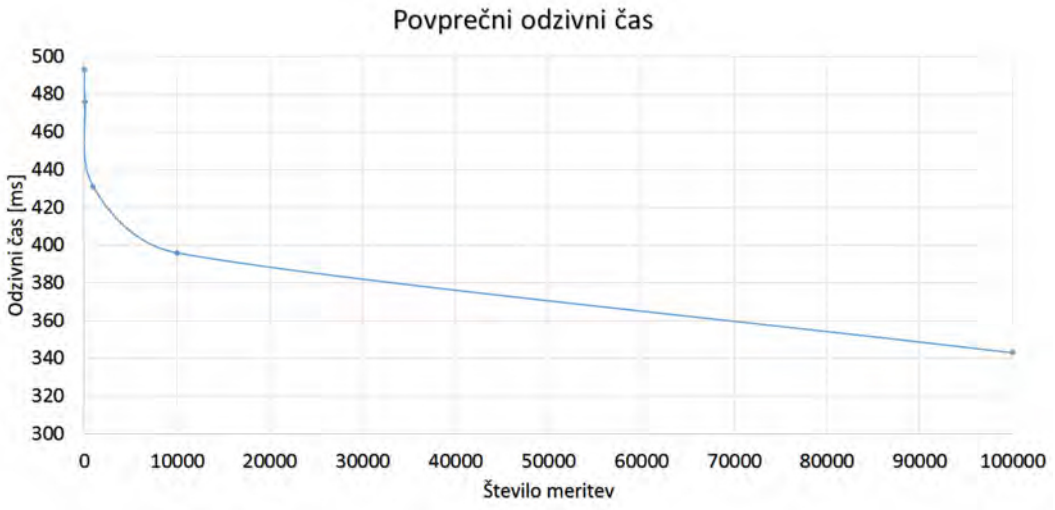
Območje delovanja ultrazvočnega senzorja je med 3 in 250 cm in ta nam zagotavlja natančnejše meritve v primeru, da je merjena površina na objektu ravna. V našem primeru je bila merjena površina nivo tekočine v silosu in njeno merjeno območje je bilo med 20 in 46,5 cm. Nivo tekočine se je dinamično spreminjal glede na izbrano referenčno vrednost tekočine, izbran tip

regulacije in porabo tekočine industrijskega silosa. Uporabljen ultrazvočni senzor omogoča meritev le na centimeter natančno, kar pomeni, da se vrednost pred pošiljanjem matematično zaokroži. Posamezne meritve so pokazale manjša odstopanja od referenčne vrednosti. Ponovitve meritev so tudi pokazale, da odstopanja niso konstantna. Odstopanja od referenčne vrednosti pri posameznih meritvah, so prikazane na sliki 4.2.



Slika 4.2: Meritve oddaljenosti od referenčne vrednosti.

Zaradi enostavnosti določanja enačbe za izračun trenutne količine v silosu in linearnega spreminjanja meritev, smo enačbo določili s pomočjo linearne interpolacije dveh robnih točk. Poleg nekonstantnega obnašanja meritev, tu nastopi tudi faktor odzivnosti meritve. Zaradi postopka pridobivanja podatkov o trenutni količini tekočine v silosu, kateremu moramo najprej poslati neposreden ukaz za zajem meritve, mora mikrokrmilnik Arduino počakati na odziv, da so podatki pripravljeni za branje iz medpomnilnika Lego Mindstorms. Šele ko so podatki pripravljeni, lahko pošljemo ukaz za zajem. Poleg



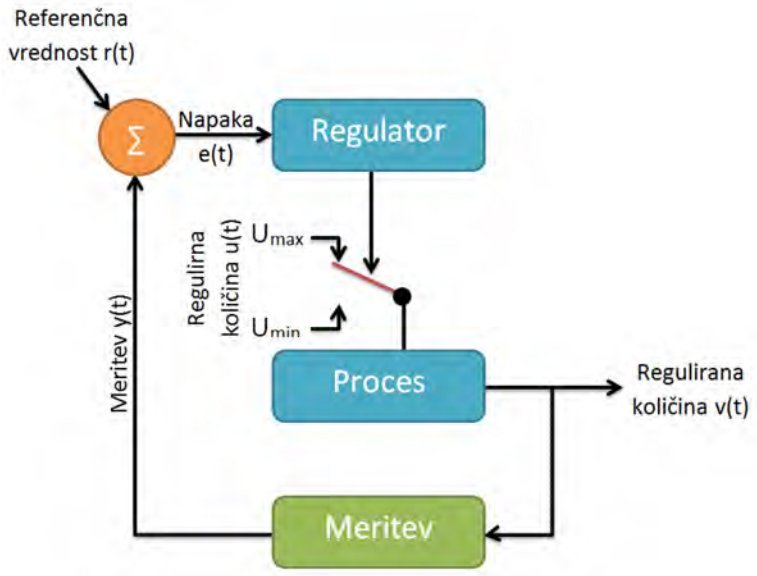
Slika 4.3: Odzivni čas zajema podatkov iz ultrazvočnega senzorja.

slednje zakasnitve v odzivnosti povzroči tudi nezanesljivost prejetih telegramov Bluetooth. Zgodi se namreč lahko, da mikrokrmilnik Arduino prejme popačene podatke. Takšne podatke smo morali filtrirati, saj so neuporabni. Na sliki 4.3 je prikazana odzivnost ultrazvočnega senzorja v primerjavi s številom meritev. Na sliki lahko opazimo, da z večanjem števila ponovitev nekoliko izboljšamo sam odziv. Meritve odzivnosti smo opravili z 10, 100, 1000, 10000 in 100000 ponovitvami zajema meritev iz ultrazvočnega senzorja. Na podlagi opravljenih meritev natančnosti in hitrosti odziva ultrazvočnega senzorja smo se odločili, da bomo realizirano povezavo med industrijskim krmilnikom Siemens, mikrokrmilnikom Arduino in Lego Mindstorms NXT uporabili za simulacijo industrijskega silosa, saj je v primeru visoke dinamike merjenih objektov odzivnost prepočasna in senzor premalo natančen.

4.3 Dvotočkovna regulacija in regulacija PID

4.3.1 Dvotočkovni regulator

Poznamo več vrst regulatorjev, ki jih v grobem delimo na standardne in tiste z modernim pristopom. Dvotočkovni regulator in regulator PID sta standardna regulatorja, ki zahtevata matematični model procesa, medtem ko moderni procesi svoj model ustvarjajo iz meritev. Primeri modernih procesov so nevronske mreže in mehka logika. Dvotočkovni regulator glede na vrednost napake preklaplja med dvema vrednostima regulirane količine. Če bo imel regulirani proces pozitivni odziv, bo visoka vrednost regulirne količine U_{max} povzročila povečanje regulirane količine, nizka vrednost regulirne količine U_{min} pa bo povzročila zmanjšanje regulirane količine. Težava teh regulatorjev je, da regulirna količina neprestano oscilira okrog referenčne vrednosti in povzroča obrabo preklopnih elementov [2]. Shemo dvotočkovne regulacije lahko vidimo na sliki 4.4.



Slika 4.4: Shema dvotočkovne regulacije.

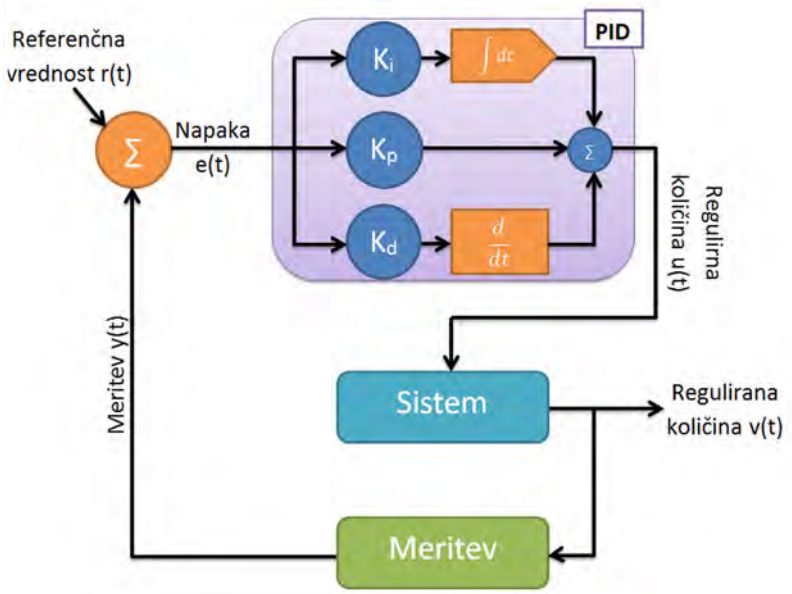
Dvotočkovni regulator smo razvili na mikrokrmilniku Arduinu, vendar

pri temu nismo uporabili dodatnih knjižnic Arduino. Algoritem deluje tako, da po vsaki uspešno prebrani ultrazvočni meritvi, preveri ali je trenutna vrednost v mejah uporabniško določene referenčne vrednosti. Mejo uporabnik določa preko vmesnika človek – stroj. Zaradi nihanj, ki nastajajo med posameznimi meritvami ultrazvočnega senzorja, smo se pri dvotočkovni regulaciji odločili za dovoljeno odstopanje $\Delta = \pm 0{,}5\ l$. To pomeni, da se bo silos začel polniti, ko bo nivo tekočine padel pod uporabnikovo določeno referenčno vrednost, zmanjšano za dovoljeno odstopanje Δ. Polnjenje silosa se bo ustavilo, ko bo nivo količine presegel uporabnikovo določeno referenčno vrednost, povečano za odstopanje Δ. Pri dvotočkovnem regulatorju lahko uporabnik določa pretok tekočine skozi posamezen ventil in s tem pospeši oz. upočasni hitrost polnjenja in praznjenja silosa. Z uporabo Δ smo se izognili nepotrebni obrabi elementov.

4.3.2 Regulator PID

Regulator PID je posplošitev dvotočkovnega regulatorja. V praksi se največkrat uporabljajo kombinacije PI, PD in PID, ki so simulirane preko mikrokrmilnikov. Uporablja tri člene: P (proporcionalni – je najpreprostejši zvezni regulator), I (integracijski – odziv ni vezan na trenutno napako temveč na vsoto vseh prejšnjih napak) in D (diferencialni člen – poskuša predvidevati dogajanje v prihodnosti). Regulator PID ima tako v kombinaciji z vsemi členi vso potrebno dinamiko. Večina današnjih regulatorjev je diskretnih, saj omogočajo hitro in enostavno spreminjanje strukture programa ali parametrov. Parametre lahko določamo z različnimi metodami. Osnovna metoda, je metoda s poskušanjem, kjer vse člene najprej postavimo na nič, nato pa jih postopno dodajamo. Poznamo pa tudi hevristične metode. S temi metodami določimo začetno nastavitev parametrov, nato pa jih s poskušanjem natančneje nastavimo. Primeri takšnih metod so: Ziegler - Nichols, Cohen - Coon, Chien - Hornes - Reswick [2].

Da bi celotno aplikacijo ohranili na mikrokrmilniku Arduino, smo se odločili regulator PID realizirati s pomočjo knjižnice Arduino PID, kljub



Slika 4.5: Regulator PID.

temu, da nam industrijski krmilnik Siemens ponuja v tem odlično podporo. S tem smo nekoliko pridobili tudi na sami odzivnosti sistema, saj nam ni potrebno posredovati vseh vhodnih in izhodnih spremenljivk regulatorja nazaj na industrijski krmilnik. Sam industrijski krmilnik smo nato uporabili za realizacijo nadzornega sistema, ki omogoča spremljanje in upravljanje posameznih servomotorjev in omogoča njihovo zaustavitev v primeru napake v sistemu.

Za potrebe regulacije dveh servomotorjev, smo potrebovali dva regulatorja PID. Ob zagonu programa moramo za posamezen regulator najprej definirati vhodne, izhodne spremenljivke in referenčno spremenljivko ter določiti začetne parametre PID. To postorimo z inicializacijo razreda `PID valveInPID( &InputPID, &OutputPID, &SetpointPID, aggKp, aggKi, aggKd, REVERSE)`. Med inicializacijo programa moramo poskrbeti tudi za vklop posameznega regulatorja PID, saj je ta privzeto izklopljen. Določiti moramo tudi izhodne meje regulirne količine. Slednje nastavimo s funkcijama `SetMode()`

in `SetOutputLimits()`. Meje izhodne regulirne količine smo za naše potrebe nastavili med 80 in 1000. Te vrednosti predstavljajo časovni zamik, s katerim mora biti servomotor odmaknjen od začetne pozicije. Naš program ob vsaki uspešni meritvi gladine izvede proceduro za nastavitev nove vrednosti meritve in sproži izračun PID s funkcijo `Compute()`. V naslednjem koraku preverimo ali se je regulirna količina od prej spremenila in če, kakšno je odstopanje. Za to odstopanje program nato spremeni trenutno pozicijo servomotorja s tem, da servomotor nekoliko obrne v smeri urinega kazalca ali pa ga obrne v nasprotni smeri urinega kazalca. V primeru, da pride do spremembe uporabnikove želene količine, se ta ustrezno nastavi v referenčno spremenljivko. Same parametre smo določili z metodo poskušanja, knjižnica pa nam omogoča tudi spremembo le-teh med samim izvajanjem regulacije. Tako lahko izberemo pristop, da ob veliki napaki uporabljamo bolj agresivno nastavljene parametre, ko pa se približamo želeni količini, te parametre nastavimo na bolj konzervativne. Del programske kode za regulator PID lahko vidimo na sliki 4.6.

```
// PID regulation mode
//---------------------------------------------------------------------
void PIDReguMode(float siloLevel){
    int writeResult;

    InputInPID = siloLevel;
    InputOutPID = siloLevel;

    valveInPID.Compute();
    valveOutPID.Compute();

        //filling silo - liquid level lower then the setpoint
        if (oldVavleInPosition < OutputInPID && SetpointPID < siloLevel){
            // open Valve in more
            bluetoothSerial.write(valveInOpen, sizeof(valveInOpen));
            delay(4 * abs(OutputInPID - oldVavleInPosition)); // 1 degre rotation is done in 4ms
            bluetoothSerial.write(valveInStop, sizeof(valveInStop));

            Serial.print("Open valve in for:");
            Serial.println(abs(OutputInPID - oldVavleInPosition));

            if(!S7ValveInStatus){
              S7ValveInStatus = true;
              // write new Valve in status
              writeResult = Client.WriteArea(S7AreaDB, 9, 4, 1, &S7ValveInStatus);
              if (writeResult != 0)
              {
                 CheckError(writeResult);
              }
            }

            oldVavleInPosition = OutputInPID;
        }
```

Slika 4.6: Del programske kode za realizacijo regulatorja PID.

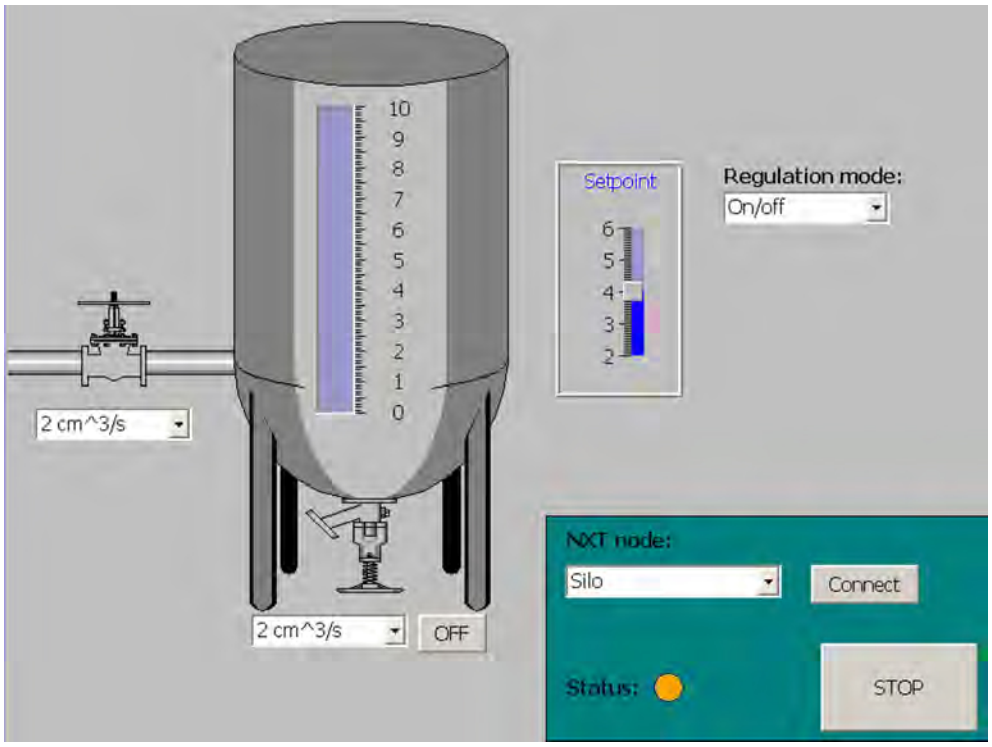
Za določanje vrednosti členov regulatorja PID smo uporabili metodo s poskušanjem. Metode se lotimo tako, da najprej vse člene (K_p, T_i in T_d) postavimo na nič in jih nato postopoma dodajamo. Najprej konstanto K_p postavimo na tako vrednost, da sistem oscilira. Nato konstanto zmanjšujemo za faktor od 8 do 10 dokler oscilacije ne izzvenijo. Na koncu lahko še z manjšimi spremembami (faktor 2) poiščemo všečen odziv. Pri integralnem členu najprej začnemo z majhnimi vrednostmi T_i, na primer 1000, nato pa s faktorjem 10 vrednost zmanjšujemo dokler nam odziv ni všeč. Diferencialni člen T_d najprej nastavimo tako, da sistem oscilira, nato pa ga zmanjšamo za faktor 10. Nadaljnje popravke delamo za faktor 2 navzgor ali navzdol. Tako smo s poskušanjem prišli do vrednosti členov $K_p = 50$, $T_i = 30$ in T_d

Slika 4.7: Odziv regulatorja PID.

= 90. Odziv regulatorja PID lahko vidimo na sliki 4.7. Med samo simulacijo industrijskega silosa z uporabo regulatorja PID, smo velikokrat naleteli na nihaj v regulaciji, ki ga je povzročila napačna meritev ultrazvočnega senzorja. Ta nihaj lahko opazimo tudi na sliki 4.7, med 5 in 6 minuto.

4.4 Nadzorni sistem

Uporabnik za potrebe upravljanja z industrijskim silosom uporablja nadzorni sistem SCADA. Nadzorni sistem v realnem času prikazuje trenutno količino tekočine v silosu, trenuten status povezave ter uporabniku omogoča neposredno poseganje v sam proces. Nadzorni sistem je bil načrtovan in izdelan tako, da omogoča čim boljši pregled delovanja sistema in njegovo enostavno uporabo. Sestavljen je iz enega glavnega zaslona, ki ga lahko vidimo na sliki 4.8.



Slika 4.8: Zaslonska slika nadzornega sistema SCADA.

Glavni zaslon je razdeljen na dva dela. V desnem spodnjem kotu se nahaja del za upravljanje povezav z različnimi modeli Lego Mindstorms in zasilno zaustavitev sistema z gumbom `STOP`. V primeru zasilne zaustavitve bo sistem avtomatsko poskrbel za zaprtje vseh odprtih ventilov. Tu lahko najdemo tudi indikator statusa povezave, ki v primeru, da sistem miruje, sveti rdeče, v primeru vzpostavljanja povezave, utripa oranžno in v primeru vzpostavljene povezave, zeleno. Osrednji del glavnega zaslona zavzema silos, na katerem se prikazuje informacija o trenutnem nivoju tekočine. Nadzorni sistem omogoča uporabniku tudi neposredno poseganje v proces, saj je le temu omogočeno spreminjanje količine pretoka in nivoja količine v silosu ter vrsta regulacije. Uporabnik ima možnost izbire med dvema vrstama regulacije. V primeru, da uporablja dvotočkovno, ima prosto izbiro pri velikosti pretoka posameznega ventila, medtem ko v regulaciji PID to ni mogoče, saj s pretokom upravlja

regulacija sama. Uporabnik ima v obeh regulacijah možnost neposredno odpreti ali zapreti odtočni ventil z gumbom `ON/OFF` in nastavljati želeni nivo tekočine v silosu z drsnikom `Setpoint`. Vse podatke SCADA zajema in shranjuje v podatkovni blok DB9, ki nam ga prikazuje slika 3.4.

Poglavje 5

Sklepne ugotovitve

Cilj diplomske naloge je predstavitev splošne rešitve za vzpostavitev komunikacije med industrijskim krmilnikom Siemens in modeli Lego Mindstorms NXT. Rešitev naj bi tako omogočala integracijo splošnega komunikacijskega vmesnika v druge sisteme. Za vzpostavitev te povezave smo izbrali mikrokrmilnik Arduino, ki je enostavno modularno nadgradljiv in podpira tako komunikacijo Ethernet, kot komunikacijo Bluetooth. V nalogi smo na kratko predstavili posamezno uporabljeno strojno opremo in nekoliko podrobneje spoznali komunikacijski protokol Ethernet, ki ga uporablja industrijski krmilnik Siemens ter protokol Bluetooth, katerega uporablja Lego Mindstorms NXT. Ob poznavanju potrebnih protokolov, smo lahko izdelali splošno namenski komunikacijski vmesnik, s katerim lahko poljubno upravljamo do tri servomotorje in zajemamo podatke iz ultrazvočnega senzorja.

Modeli Lego Mindstorms nam z uporabo komunikacije Bluetooth in s sprejemanjem neposrednih ukazov, omogočajo neposredno upravljanje. Tako smo morali na mikrokrmilniku Arduino razviti programsko kodo, ki je bila na zahtevo sistema, ki uporablja splošni komunikacijski vmesnik, sposobna vzpostaviti povezavo z različnimi modeli Lego Mindstorms. Po uspešni vzpostavitvi povezave, je sistem lahko neposredno upravljal s servomotorji in uporabljal meritve, zajete iz ultrazvočnega senzorja. Komunikacijski vmesnik, je moral omogočiti tudi komunikacijo Ethernet z industrijskim krmilnikom

Siemens, saj smo komunikacijo zasnovali tako, da smo na krmilniku hranili vse potrebne informacije in parametre, ki jih je Arduino potreboval za upravljanje.

Ker je bilo na koncu smiselno, da naš izdelan komunikacijski vmesnik uporabimo tudi na smiselni aplikaciji, smo se na podlagi meritev odzivnosti in natančnosti ultrazvočnega senzorja odločili, da z uporabo servomotorjev in ultrazvočnega senzorja Lego, izdelamo maketo, ki bo simulirala industrijski silos. Našo simulacijo smo na koncu nadgradili še z izdelavo nadzornega sistema, ki uporabniku omogoča povezovanje z različnimi modeli Lego Mindstroms. Simulacija prikazuje trenuten nivo tekočine v silosu, omogoča izbiro dveh vrst regulacij, omogoča pa tudi spreminjanje želenega nivoja tekočine.

Simulacija industrijskega silosa je primerna za nadgradnjo v večih smereh. Skozi meritve oddaljenosti gladine smo opazili, da lahko natančnost meritev izboljšamo že s samim pozicioniranjem ultrazvočnega senzorja. Na tržišču pa smo opazili tudi druge modele ultrazvočnih senzorjev, ki omogočajo boljšo odzivnost in natančnejše meritve. Programsko kodo bi lahko nadgradili s periodičnim preverjanjem komunikacije Bluetooth, saj se ta, v primeru da model Lego Midstorms ni več v dosegu, prekine s strani modula Arduino Bluetooth, vendar naša programska koda tega ne preverja.

Komunikacijski vmesnik je bil izdelan v nadaljnje pedagoške namene. Ker je bil vmesnik zasnovan splošno, študentom omogoča precej enostavno uporabo in proste roke pri izbiri projektov, saj v kombinaciji z Lego Mindstorms NXT, ponuja širok spekter možnosti za simulacijo realnih sistemov. Sam komunikacijski vmesnik ponuja več možnosti nadgradnje. Trenutno vmesnik podpira le uporabo servomotorjev in ultrazvočnega senzorja, vendar Lego Mindstorms NXT pozna še druge aktuatorje in senzorje, kot so luči, stikala, zvočni senzorji in svetlobni senzor. Pri implementaciji dodatnih aktuatorjev in senzorjev, bi bilo smiselno še dodatno posplošiti uporabniško določeno strukturo UDT in omogočiti uporabniku definicijo posameznih vhodno/izhodnih vrat Lego Mindstorms NXT.

Literatura

[1] D. E. Kandray, "Programmable automation technologies: An introduction to CNC, robotics and PLCs", New York : Industrial Press, cop. 2010

[2] U. Lotrič, "Prosojnice predavanja Procesna avtomatika", Univerza v Ljubljani, 2010

[3] (2014) "Arduino Mega 2560 R3", Dostopno na: http://arduino.cc/en/Main/arduinoBoardMega2560

[4] (2006) "Lego MINDSTORMS NXT Direct commands", Dostopno na: http://es.scribd.com/doc/44386755/Appendix-2-Lego-Mind-Storms-Nxt-Direct-Commands

[5] (2006) "NXT User Guide", Dostopno na: http://cache.lego.com/downloads/education/9797_LME_UserGuide_US_low.pdf

[6] (2014) "Settimino Arduino Ethernet communication library for S7 Siemens PLC", Dostopno na: http://settimino.sourceforge.net/

[7] (2011) "Siemens Simatic CPU 31xC and CPU 31x: Technical specifications manual", Dostopno na: http://cache.automation.siemens.com/dnl/Tk/Tk1MDQwMQAA_12996906_HB/s7300_cpu_31xc_and_cpu_31x_manual_en-US_en-US.pdf

[8] (2013) “Sparkfun BlueSMIRF Gold”, Dostopno na:
http://www.sparkfun.com/products/12582