Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Martin Šimac

Vmesnik za vodenje sistema Lego Mindstorms v razvojnem okolju Siemens Step7

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana 2013

Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00370/2013
Datum: 05.03.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MARTIN ŠIMAC**

Naslov: **VMESNIK ZA VODENJE SISTEMA LEGO MINDSTORMS V RAZVOJNEM OKOLJU SIEMENS STEP7**

CONTROLLING LEGO MINDSTROMS SYSTEM FROM SIEMENS STEP7 DEVELOPMENT ENVIRONMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Sistem Lego Mindstorms s programirljivo kocko, senzorji in motorji z vgrajenimi dajalniki impulzov predstavlja zanimivo platformo za učenje na področju avtomatizacije procesov. Zaradi precej različnih standardov vmesnikov v zabavni elektroniki in industrijskih komunikacijah, povezava sistema z industrijskim krmilnikom ni enostavna. Predlagajte in izdelajte rešitev, ki bo omogočala programiranje sistema Lego Minstrorms v zelo razširjenem razvojnem okolju Siemens Step7. Rešitev preizkusite na problemu sledenja črti z regulatorjem PID.

Mentor:
izr. prof. dr.Uroš Lotrič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Martin Šimac, z vpisno številko **63060294**, sem avtor diplomskega dela z naslovom:

Vmesnik za vodenje sistema Lego Mindstorms v razvojnem okolju Siemens Step7

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. junija 2013 Podpis avtorja:

Zahvaljujem se vsem, ki so mi v času študija kakorkoli pomagali, še posebej pa družini za potrpežljivost in podporo. Zahvaljujem se Primožu, Nejc, Blažu, Miranu in Jerneju za nasvete in komentarje ter Meti, za potrpežljivo lektoriranje. Izr. prof. dr. Uroš Lotrič si zasluži posebno pohvalo, saj je odličen mentor in pedagog; hvala za vso pomoč v času študija.

Diplomsko delo posvečam staršema, ker nista izgubila upanja :)

Kazalo

Slike

SLIKE

Tabele

Povzetek

Za preizkušanje programske kode v praksi navadno potrebujemo primerno strojno opremo. Ker je v avtomatiki zagotavljanje testnega okolja še posebej težavno, smo v sklopu diplomskega dela razvili programsko opremo, ki nam omogoča preizkus programske kode, razvite v okolju Simatic Step7 na platformi Lego Mindstorms. Programska koda se izvaja na simulatorju krmilnika PLCSIM, programska oprema pa s pomočjo direktnih ukazov akcije prenaša na Mindstorms. Pri razvoju smo uporabili Siemensovo knjižnico S7ProSim za povezavo s simulatorjem PLCSIM in knjižnico Mindsqualls za povezavo z Lego Mindstorms. Realizirali smo oba načina povezovanja, tako preko povezave Bluetooth kot s kablom USB. Za razvoj aplikacije smo uporabili programski jezik C#. Aplikacija omogoča uporabo vseh zmogljivosti Mindstorms v razvojnem okolju Simatic Step7. S sestavnimi deli iz kompleta Mindstorms smo sestavili vozilce, ki ga preko povezave Bluetooth upravlja koda, ki se izvaja v PLCSIM. Za predstavitveno aplikacijo, razvito z uporabo lestvičnih diagramov, smo si izbrali problem sledenja črti.

Ključne besede:

vmesnik, simulator, testiranje, krmilnik, razhroščevanje, poučevanje avtomatike, Bluetooth, USB

Abstract

Debugging of the code usually requires appropriate hardware. In the field of automation it is especially challenging to provide a testing environment. For the purpose of this thesis an application has been developed, which allows us to debug or test our source code developed in Simatic Step7 tool on Lego Mindstorms. Our application transfers actions on Mindstorms using direct commands while the code is executed on PLCSIM. The application has been developed using Siemens S7ProSim library for the implementation of interface with PLCSIM and Mindsqualls library for the connection with Lego Mindstorms. Both means of connectivity were implemented, Bluetooth connection and USB cable. The source code of our application was written using C# programming language. The application allows us to use all functions of Mindstorms from Simatic Step7 environment. Three wheeled vehicle was constructed using parts from Mindstorms kit which is managed with code executed in PLCSIM via Bluetooth connection. The applicability of software was demonstrated on line following problem, programmed using ladder logic.

Key words:

interface, simulator, testing, controller, debugging, teaching automation, Bluetooth, USB

Poglavje 1

Uvod

Vsak pedagog ali študent na področju avtomatike je postavljen pred dejstvo, da je za kakovostno podajanje ali pridobivanje znanja potrebna izredno draga programska in strojna oprema svetovno uveljavljenih proizvajalcev, kot je denimo Siemens. Cenejša oprema sicer obstaja, a z njo študentje ne dobijo pravega vpogleda v realne industrijske sisteme. Za študenta računalništva sama kakovost in zanesljivost strojne opreme niti ni tako pomembna. Pomembno je le, da svojo kodo razvija v razvojnem okolju, ki mu nudi vse, kar bo potreboval za delo na realnem projektu, ter da lahko uporabi senzorje in opazuje izvedbo na aktuatorjih. Profesionalna razvojna okolja vsebujejo simulator, v katerem lahko razhroščujemo kodo, še preden jo naložimo na pravi krmilnik, zato bi lahko pri poučevaju uporabljali le simulator. Kdo si želi opazovati vrednosti v registrih? Priznajmo, pravi študent avtomatike želi slišati piskajoč glas servomotorja. Ker ima platforma Lego Mindstorm tako senzorje kot motorje, ki omogočajo vse, kar omogoča profesionalna oprema, je ena od možnosti zamenjava dragega senzorsko-aktuatorskega nivoja z Mindstormom. Potrebna je le programska oprema, ki bi omogočala povezavo med Siemensovim razvojnim okoljem in Lego Mindstormom in ravno z razvojem te programske opreme se bomo ukvarjali v tem diplomskem delu.

V prvem delu bomo predstavili opremo, ki se trenutno uporablja pri predmetu Procesna avtomatika, opisali bomo razvojno okolje Simatic Step7 in

simulator, ki je del razvojnega okolja.

V drugem delu se bomo dotaknili platforme Mindstorms ter podrobneje pogledali, na kakšne načine lahko z napravo komuniciramo, kakšne senzorje lahko uporabimo ter kaj omogočajo motorji. Podrobneje bomo predstavili tudi knjižnico, s katero smo si pomagali pri razvoju aplikacije.

Tretji del diplomskega dela bo posvečen programski opremi, ki poskrbi da naša ideja lahko zaživi. Predstavili bomo zasnovo ter si pogledali, kaj vse nam tak pristop omogoča in kakšne so njegove omejitve.

Zadnje poglavje bo namenjeno demonstraciji delovanja in uporabe. Opisali bomo, s kakšnimi omejitvami smo se pri uporabi srečali, kako smo jih rešil kaj smo se pri tem naučili in kaj bi bilo dobro spremeniti pri zasnovi programske opreme.

Poglavje 2

Industrijski krmilnik in Simatic Step7

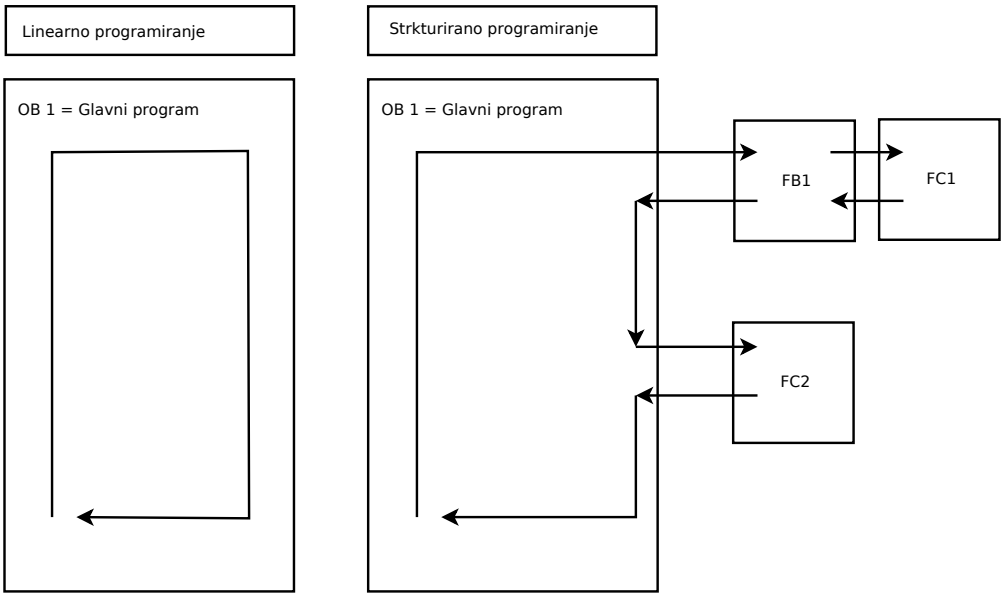
Trenutno na vajah študentje uporabljajo Siemensov krmilnik S7-300, ki je zelo pogost v industriji zaradi modularne zasnove in pregovorne zanesljivosti, z digitalnimi vhodi in izhodi v kombinaciji s tremi različicami kompletov fischertechnik. Izbirajo lahko med 3D robotsko roko in dvema maketama proizvodne linije. Svoje projekte študentje razvijajo v razvojnem okolju Simatic Step7, nato programsko kodo naložijo na krmilnik. Razvojno okolje Simatic Step7 se uporablja za vso družino krmilnikov S7-xxx in združuje več uporabnih orodij, med drugim tudi PLCSIM, simulator krmilnika, na katerega lahko naložimo kodo in jo testno izvajamo na primer za potrebe razhroščevanja. Slabost trenutnih kompletov je omejitev na samo digitalne vhode in izhode. Tako je lahko motor na maketi proizvodne linije samo aktiven ali neaktiven, brez nastavitve hitrosti, ramp, zavore... Vhod je lahko le aktiven ali neaktiven, kar nas prikrajša za meritev oddaljenosti, temperature, svetlobe, pozicije motorja in tako naprej. Take vrednosti bi morali študenti simulirati, simulacije pa bi morali uporabljati tudi pri vajah iz regulacije.

Kot smo omenili že v uvodu, je naš cilj aplikacija, ki bi povezala simulator in Lego Mindstrom, za kar pa potrebujejmo dostop do podatkov s katerimi simulator PLCSIM operira. V nadaljevanju si bomo ogledali knjižnico, s katero

lahko upravljamo s simulatorjem ter beremo in pišemo podatke v njegovem spominu. Dostop do stanj pomnilniških lokacij je osnova za realizacijo našega cilja. Preden pa se spustimo v podrobnosti, si na kratko oglejmo, kako se izvaja program na krmilniku, iz kakšnih gradnikov je sestavljen, kateri so najpogostejši podatkovni tipi ter na kakšen način Siemensov krmilnik naslavlja podatke v pomnilniku.

2.1 Pregled lastnosti

Dobra programerska praksa nas uči, da mora biti programska koda logično strukturirana in dobro komentirana. K prvemu nas usmerja že sam Siemensov pristop, saj omogoča, da je programska koda porazdeljena med več bloki. Program se ne glede na način programiranja na programabilnem logičnem krmilniku (PLK) izvaja ciklično (Slika 2.1) kadar je stikalo v RUN načinu.



Slika 2.1: Shema izvajanja. [1]

Pri Siemensu so nam na voljo naslednji programski bloki:

- organizacijski bloki (OB),
- funkcije(FC),
- podatkovni bloki(DB),
- funkcijski bloki(FB).

Organizacijski bloki določajo strukturo programske kode in so označeni z nespremenljivo identifikacijsko številko v razponu 0 do 65535 (Tabela 2.1). Omenimo OB1, ki služi kot vmesnik med operacijskim sistemom in uporabniškim programom. Funkcije so primerne za pogosto ponavljajoče se dele programa, poleg tega je z uporabo funkcij razhroščevanje enostavnejše.

Podatkovne bloke strukturiramo sami, poznamo globalne in take, ki pripadajo funkcijskim blokom. Pogosta primera uporabe sta hranjenje podatkov in izmenjava podatkov izven uporabniškega programa.

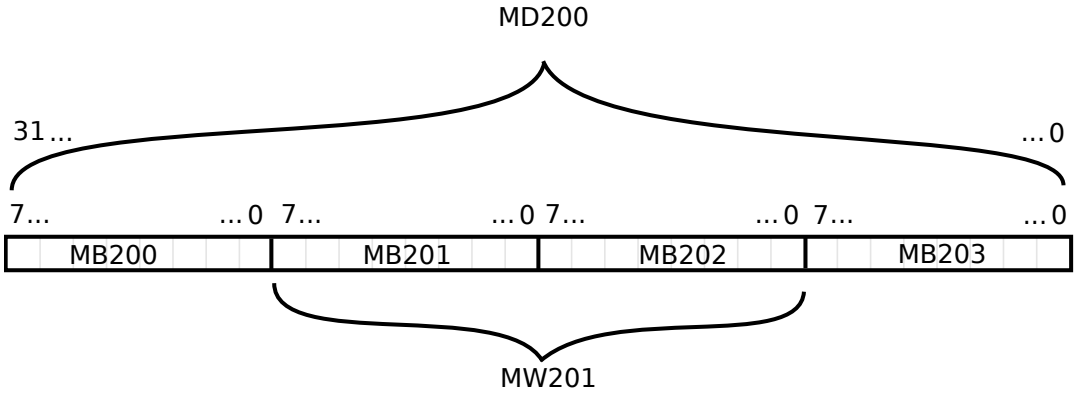
Funkcijski bloki so deli uporabniškega programa ki jih lahko kličemo večkrat, vsak ima lahko svoj podatkovni blok. Vendar podatkovni bloki niso najpogostejši način hranjenja podatkov. Tako na krmilniku S7-300 kot na simulatorju poznamo podatkovne tipe, naštete v tabeli 2.1.

Vhodi, izhodi ali spominske lokacije so lahko biti, bajti, besede ali dvojne besede. Da se med seboj razlikujejo, bajtom, besedam in dvojnim besedam dodamo črke B, W in D; na primer M, MB, MW, MD. Vhodi in izhodi perifernih naprav so lahko le bajti, besede ali dvojne besede, torej so vsi označeni z B, D ali W.

Podatki v pomnilniku so naslovljeni po pravilu debelega konca, kar pomeni da je naslov operanda enak naslovu tistega dela, v katerem je njegov najpomembnejši del (Slika 2.2). Pomnilniške lokacije lahko naslavljamo simbolno ali absolutno, razliko si bomo podrobneje ogledlai v poglavju 2.2. Iz slike 2.2 so razvidne razlike med MB - naslavljamo bajt, MW - naslavljamo besedo in MD - naslavljamo dvojno besedo, ter različne možnosti dostopanja do pomnilniških lokacij. Z MB201 na primer naslovimo bajt 201, z MW201

Tabela 2.1: Naslovni prostor. [1]

Simbol	**Opis**
I	vhodi
Q	izhodi
M	spominske lokacije
PI	vhodi perifernih naprav
PQ	izhodi perifernih naprav
T	časovnik
C	števec
FB	funkcijski blok
OB	organizacijski blok
DB	podatkovni blok
FC	funkcija
SFB	sistemski funkcijski blok
SFC	sistemska funkcija
VAT	tabela spremenljivk
UST	podatkovni blok ki ga definira uporabnik



Slika 2.2: Označevanje naslovnega prostora.

pa lokaciji MB201 in MB202, ki skupaj tvorita besedo 201. Dostopamo lahko tudi do posameznega bita, kar naredimo z M201.1.

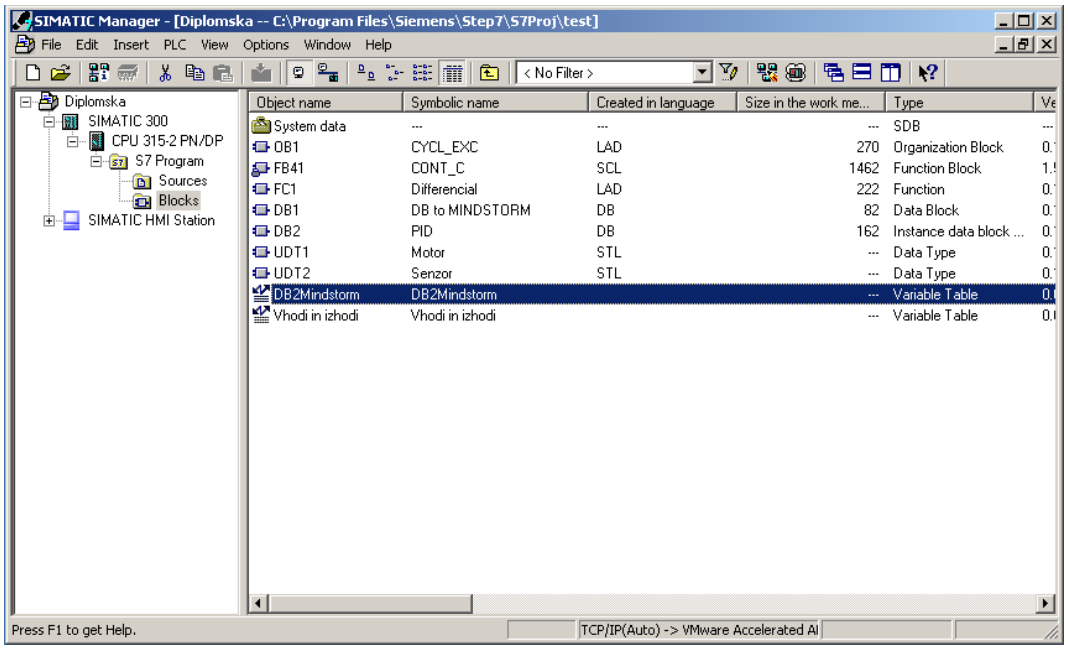
2.2 Simatic Step7

Okolje Step7 je standardno okolje za programiranje in upravljanje nastavitev krmilnikov družine Simatic, ker nam nudi vse, kar potrebujemo za realizacijo projekta na področju avtomatike. Programski jeziki, vključeni v okolje, so skladni z standardom EN61131-3, ki predpisuje tako simbole kot pravila programiranja. Paket je sestavljen iz več orodij:

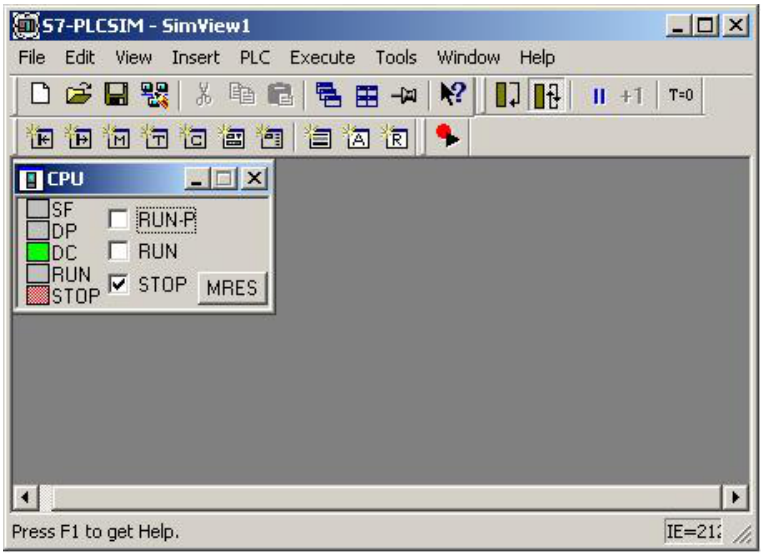
- Simatic Manager,
- Symbol editor,
- NetPro,
- PlcSim.

Simatic Manager (Slika 2.3) je orodje za upravljanje s projektom. V njem lahko odpremo nov projekt, urejamo že obstoječ projekt, zaženemo orodja za upravljanje s spremenljivkami, orodje za razhroščevanje, diagnostiko ter orodja za upravljanje nastavitev PLK. Tako imamo vsa orodja zbrana na enem mestu. Uporaba orodja Symbol editor nam omogoča upravljanje s spremenljivkami, vhodi, izhodi, časovniki, funkcijami ter ostalimi programskimi gradniki. Omogoča nam tudi pregled prostih pomnilniških naslovov, simbolno poimenovanje ter kratek komentar, kar nam olajša razhroščevanje tudi, če nismo avtor programske kode. Simbolno poimenovanje nam poenostavi programiranje, na primer, namesto naslova I1.0 uporabimo "DelovanjeMotorja". Program nam omogoča tudi uvoz in izvoz iz nekaterih drugih programov operacijskega sistema Windows. Z orodjem NetPro upravljamo z omrežnimi povezavami. Orodje PLCSIM (Slika 2.4) je simulator PLK, s katerim lahko opazujemo izvajanje programske kode in odpravljamo morebitne napake. Razvojno okolje vsebuje še orodje za konfiguracijo sistemskih

nastavitev krmilnika, urejevalnik programske kode, orodje za opazovanje in spreminaje vrednosti spremenljivk in tako dalje.



Slika 2.3: Simatic Manager.



Slika 2.4: Orodje PLCSIM.

2.3 Knjižnica S7ProSim

V diplomskem delu smo S7ProSim V5.3 Siemensovo knjižnico, ki skrbi za upravljanje s simulatorjem PLCSIM ter za dostop do podatkov v pomnilniku simuliranega krmilnika. Sama implementacija v okoljih .NET je zelo preprosta, saj je uporabljena tehnologija Microsoftov komponentni objektni model (Component Object Model, COM) [3], ki služi ponovni uporabi programskih modulov in povezovanju različnih aplikacij med seboj. Knjižnica implementira številne funkcije in metode, a smo uporabili le peščico, saj te zadostujejo našim potrebam. V nadaljevanju bomo predstavili, katere smo uporabili ter kako delujejo.

```
1  using S7PROSIMLib; //Uporabljali bomo  Siemensovo S7ProSim knjižnico
2  S7ProSim ps; //Deklariramo objekt ps tipa S7ProSim
3  ps = new S7ProSim(); //Kličemo konstruktor objekta ps
4  ps.Connect(); //vzpostavimo povezavo z aplikacijo PLCSIM
5
6  System.Object sendObject = new System.Object();
7  //deklariramo objekt v katerega bomo shranili podatek za pošiljanje
8  sendObject = (short)fromPLC.motorA.stopinje;
9  //podatek ki ga želimo poslati pretvorimo v primeren podatkovni tip
10 ps.WriteDataBlockValue(1, 6, 0, ref sendObject);
11 //kotne stopinje motorja A pošljemo v DB1, v bajt 6, od bita 0 naprej.
12
13 System.Object receiveObject = new System.Object();
14 //deklariramo in incialiriramo objekt v katerega bomo shranili prebrani podatek
15 ps.ReadDataBlockValue(1, 2, 0, PointDataTypeConstants.S7_Word, ref receiveObject);
16 //podatek beremo iz DB1, bajta 2, od 0 bita naprej.
17 fromPLC.motorA.hitrost = (short)receiveObject;
18 //prebrano vrednost shranimo v hitrost motorja A
19
20 ps.Disconnect(); //prekinemo povezavo z PLCSIM
```

Slika 2.5: Uporaba knjižnice S7 ProSim. [2]

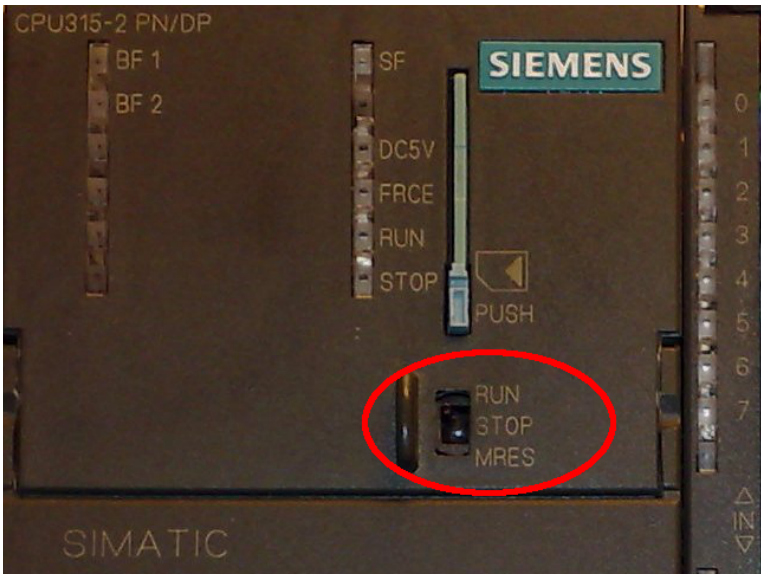
Metoda Connect vzpostavi povezavo s simulatorjem PLCSIM, kličemo jo brez argumentov (Slika 2.5, vrstica 4). V primeru napake vrne izjemo in kodo napake, ki jo nato ulovimo in ustrezno nadaljujemo z izvajanjem aplikacije.

Metoda Disconnect prekine povezavo s simulatorjem PLCSIM, način delovanja je enak kot pri Connect (Slika 2.5, vrstica 20).

Metodi SetScanMode kot argument podamo željen način delovanja simulatorja PLCSIM. Možna sta dva načina delovanja - kontinuirano izvajanje

programskih ciklov ali izvedba samo enega cikla. Način delovanja podamo kot argument metode: 0 pomeni izvedba enega cikla, 1 pa kontinuirano izvajanje programa.

Funkcija GetState vrne stanje stikala na simulatorju PLCSIM. Gre za virtualno stikalo, (Slika 2.6) ki simulira stikalo na pravem krmilniku, s katerim lahko postavimo krmilnik v način STOP - stoj, s katerim prenehamo z izvajanjem programa, RUN - delaj, v katerem se program izvaja, ter RUN-P - delaj-P, torej izvajanje z možnostjo spreminjanja programa in spremenljivk med izvajanjem.[1] Funkcija vrne enega izmed nizov znakov RUN, RUN-P ali STOP.



Slika 2.6: Stikalo za izbiro načina delovanja.

S klicem funkcije SetState postavimo virtualno stikalo v željeno pozicijo, to pa podamo kot argument funkciji. Kot argument podamo niz z željenim novim stanjem stikala, na primer: RUN.

Metoda WriteDataBlockValue nam omogoča zapis bita, bajta, besede ali dvojne besede v željen podatkovni blok v pomnilniški prostor S7-PLCSIM. Kot argument moramo podati podatek, ki ga želimo zapisati, podatkovni

[1] Način RUN-P je možen na PLCSIM

blok v katerega želimo pisati, v kateri bajt in kateri bit (Slika 2.5, vrstica 10).

Z metodo ReadDatalockValue preberemo željen bit, bajt, besedo ali dvojno besedo iz podatkovnega bloka. Argumente navedemo enako kot pri metodi WriteDataBlockValue, le da s to metodo podatka ne zapišemo temveč ga preberemo (Slika 2.5, vrstica 15).

Poglavje 3

Lego Mindstorms

Lego Mindstorms NXT2.0 je izredno zanimiva platforma. Komplet sestavlja programabilna kocka, na katero lahko priključimo tri servomotorje z vgrajenim tachometrom in zavoro ter do štiri senzorje. Omogoča nam veliko načinov upravljanja in programiranja in nam kot običajne lego kocke omogoča, da se prepustimo domišliji in izdelamo samosvojo kreacijo.

Slika 3.1: Komplet lego Mindstorms NXT. [4]

Lego Mindstorms nas je pritegnil, ker način uporabe motorjev močno

spominja na profesionalne motorje v industriji in ker lahko tudi s senzorji rokujemo na podoben način. Poleg programa, ki se izvaja na sami kocki, nam Mindstorms omogoča uporabo neposrednih ukazov, s katerimi lahko beremo vrednost senzorjev ali pa upravljamo z motorji. Z osebnim računalnikom kocka komunicira serijsko, preko kabla USB pa povezave Bluetooth. Neposredni ukazi in možnost povezave z osebnim računalnikom nam ponujajo osnovo, na kateri bomo zgradili našo aplikacijo.

3.1 Knjižnica Mindsqualls

Mindsqualls je odprtokodna knjižnico, ki implementira veliko objektov za upravljanje z Mindstormom in nam tako prihrani precej dela. Če povzamemo spletno stran avtorja: "Mindsqualls je .NET knjižnica za daljinsko upravljanje lego NXT ali lego NXT2.0 robota iz računalnika preko povezave USB ali povezave BlueTooth" [5]. Knjižnica vključuje vse neposredne ukaze, ki jih kocka podpira, zato je za naš cilj zelo primerna. Knjižnica je brezplačna in odprtokodna, zato lahko kdorkoli dopiše objekte za nove senzorje, izboljša delovanje ali odpravi morebitno skrito napako. Poleg tega je knjižnica dobro dokumentirana.

Knjižnico moramo najprej uvoziti v razvojno okolje, po uvozu pa moramo kreirati objekt, katerega metode in funkcije nam nato služijo za upravljanje kocke. Kot omenjeno v uvodu tega poglavja, se lahko s kocko povezujemo preko vmesnika USB ali povezave Bluetooth, kar specificiramo že ob kreiranju objekta. Nato se z to podrobnostjo ne ukvarjamo več, saj vse funkcije in metode delujejo enako ne glede na tip povezave, upoštevati moramo le dejstvo, da je brezžična povezava počasnejša. Knjižnica Mindsqualls ni nič drugega, kot skupek razredov, ki poskrbijo za kreiranje sporočil, ki jih nato pošlje kocki preko serijskega kanala. Gre za uporabo ukazov LsRead in LsWrite, s pomočjo katerih si s kocko izmenjujemo sporočila [6]. Glede na vsebino sporočila, nato kocka vrne vrednost senzorja ali začne vrteti izbrani motor.

3.2 Motor

Programabilna kocka iz kompleta Mindstorms omogoča priklop do treh servomotorjev na enakovredne izhode označene s črkami A, B in C. Motor krmilimo z nastavitvijo moči v razponu -100/100, minus pomeni obratno smer vrtenja. Nastavimo lahko zavoro kar pomeni, da motor poskuša ohraniti pozicijo v kateri smo ga ustavili. Če zavore nimamo nastavljene, se motor izteče, oziroma lahko os brez večje sile obračamo. Motor nam s pomočjo vgrajenega tahometra vrača kotne stopinje z natančnostjo ene kotne stopinje.

3.3 Senzorji

Istočasno lahko v kocko priklopimo do štiri senzorje, v vhode 1, 2, 3 in 4. Trenutno so na voljo senzor dotika, glasnosti, oddaljenosti, svetlobni senzor, kompas in še nekateri drugi. Nabor je pester, sproti pa se na trgu pojavljajo vedno novi senzorji, ki jih lahko uporabljamo z Mindstormom. Senzorji delujejo na različne načine; večinoma so pasivni svetlobni senzor pa ima lučko, s katero si osvetli opazovani predmet v slabših svetlobnih pogojih.

Poglavje 4

Programski vmesnik

Jedro diplomskega dela predstavlja aplikacija, ki smo jo poimenovali Sima(ti)c Mindstorms. Na strani osebnega računalnika, kjer imamo nameščeno Siemensovo razvojno okolje, v sklopu katerega lahko poženemo PLCSIM. Za uspešno uporabo naše aplikacije smo potrebovali za ta namen sprogramiran podatkovni blok. Izbrali smo DB1 in ga poimenovali 'DB to Mindstorms', kar v angleščini in v pomeni 'podatkovni blok za Mindstorms'. Vsaka koda, ki jo bomo želeli uporabljati z našo aplikacijo, bo morala vsebovati ta podatkovni blok, saj je vezni člen med PLCSIM in Sima(ti)c Mindstorms aplikacijo. Koda med svojim izvajanjem vrednosti vpisuje v podatkovni blok DB1 (Slika 4.1), zato lahko aplikacija Sima(ti)c Mindstorms do njih vedno dostopa ne da bi morali spreminjati nastavitve ali programsko kodo.

V Simatic Step7 je pomembno, da imamo zagnan PLCSIM in nanj naložen program, za vse ostalo lahko poskrbimo z uporabo aplikacije. V grobem smo aplikacijo razdelili na tri sklope razrede; S7SIMPLC LegoNXT in Form. S7SIMPLC vsebuje vse potrebno za delo in komunikacijo s simulatorjem PLCSIM. Poskrbi za vzpostavitev povezave, prekinitev povezave, branje in pisanje v podatkovni blok ter lovljenje morebitnih napak. Naslednji del je osnovna aplikacija z uporabniškim vmesnikom, formo na kateri so vsi potrebni gumbi za delo ter okno v katerem se izpisujejo vsi pomembnejši dogodki znotraj aplikacije (Slika 4.2). Poleg tega se v osnovni aplikaciji ciklično

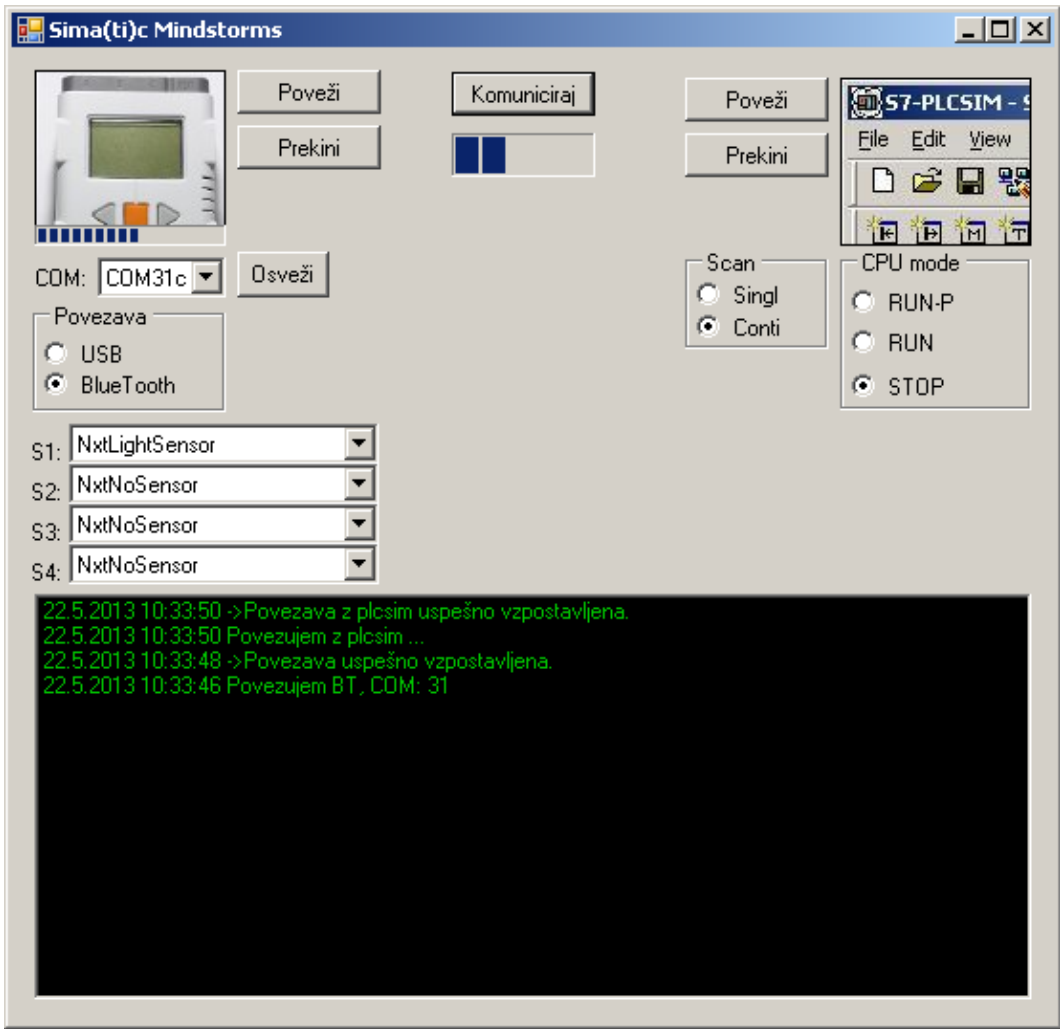
Address	Name	Type	Initial value	Actual value	Comment
0.0	MotorA.Delovanje	BOOL	FALSE	FALSE	
2.0	MotorA.Hitrost	INT	0	0	
4.0	MotorA.Zavora	BOOL	TRUE	TRUE	
6.0	MotorA.Stopinje	INT	0	0	
8.0	MotorA.Referencir	BOOL	FALSE	FALSE	
10.0	MotorB.Delovanje	BOOL	FALSE	FALSE	
12.0	MotorB.Hitrost	INT	0	0	
14.0	MotorB.Zavora	BOOL	TRUE	TRUE	
16.0	MotorB.Stopinje	INT	0	0	
18.0	MotorB.Referencir	BOOL	FALSE	FALSE	
20.0	MotorC.Delovanje	BOOL	FALSE	FALSE	
22.0	MotorC.Hitrost	INT	0	0	
24.0	MotorC.Zavora	BOOL	TRUE	TRUE	
26.0	MotorC.Stopinje	INT	0	0	
28.0	MotorC.Referencir	BOOL	FALSE	FALSE	
30.0	Senzor1.Vrednost	INT	0	0	
32.0	Senzor1.Luc	BOOL	FALSE	FALSE	
34.0	Senzor2.Vrednost	INT	0	0	
36.0	Senzor2.Luc	BOOL	FALSE	FALSE	
38.0	Senzor3.Vrednost	INT	0	0	
40.0	Senzor3.Luc	BOOL	FALSE	FALSE	
42.0	Senzor4.Vrednost	INT	0	0	
44.0	Senzor4.Luc	BOOL	FALSE	FALSE	

Slika 4.1: DB1 podatkovni blok za izmenjavo podatkov.

izvajajo branje podatkov iz PLCSIM, obdelava podatkov, priprava direktnih ukazov namenjenih kocki, klici direktnih ukazov in pisanje iz kocke pridobljenih vrednosti v PLCSIM. Naslednji pomemben razred je LegoNXT, ki podobno funkcijo kot S7SIMPLC na simulatorski strani opravlja na kockini strani. Glede na priklopljene motorje in senzorje ter na izbran tip komunikacije prilagodi obliko sporočil, ki jih pošilja kocki. Za uporabo aplikacije potrebujemo še povezavo Bluetooth, ali pa kabel USB, na eni strani priklopljen v računalnik, na dugi strani pa v kocko. Aplikacija generira sporočila in jih preko povezave pošlje do kocke. Nato jih programska oprema na kocki interpretira ter pretvori v dejanja, kar je tudi zadnji člen v verigi.

4.1 Razred S7SIMPLC

Knjižnico za povezavo s PLCSIM smo si ogledali že v poglavju 2.3 in na sliki 2.5, kjer smo opisali njene najpomembnejše funkcije, uporabljene pri razvoju aplikacije. Za hranjenje podatkov, ki jih izmenjujemo s PLCSIM, smo v aplikaciji spisali razred DB. Podobno kot v okolju Simatic Step7 uporabljamo podatkovni blok DB1, v aplikaciji uporabljamo razred DB. Tudi sama struk-



Slika 4.2: Aplikacija Sima(ti)c Mindstorms.

tura je podobna tisti iz DB1. Razred DB inicializiramo kot objekt fromPLC in vse podatke, ki jih preberemo iz PLCSIM, shranimo v ta objekt. Podatke, kot so izmerjene vrednosti senzorjev ali kotne stopinje iz motorjev, najprej zapišemo v fromPLC, šele nato jih pošljemo v PLCSIM. Na ta način smo poskrbeli, da so podatki hranjeni samo na enem mestu in so vedno ažurni, kadarkoli jih katerakoli funkcija znotraj aplikacije potrebuje. Da povezave s kocko ne obremenjujemo s sporočili, ko se na primer hitrost motorja ne spremeni, smo implementirali preverjanje vrednosti, ki jih prebiramo iz PLCSIM. V kolikor se nobena od vrednosti motorja ali senzorja v PLCSIM ni spremenila označimo podatke kot nespremenjene in jih v osnovni aplikaciji ignoriramo. V tem primeru kocki ne pošljemo nobenega sporočila.

4.2 Razred LegoNXT

LegNXT je razred, ki vsebuje vse potrebno za delo z kocko. Knjižnico smo opisali že v poglavju 3.1 zato je na tem mestu ne bi ponovno predstavljali. Ker ima vsak senzor svoj razred in funkcije, uporabnika pa nismo želeli omejevati, smo morali poskrbeti, da je lahko na vsakem vhodu priklopljen katerikoli senzor. Zato imamo na osnovni formi spustne menije za vsak senzorski vhod, kjer izberemo, kateri senzor je priklopljen, oziroma da na tem vhodu senzorja ni. Na sliki 4.2, nad oknom za izpis pomembnejših informacij vidimo spustne menije poleg 'Sx:' oznak, kjer x pomeni številko senzorskega vhoda na kocki. Kateri senzor je priklopljen na kateri vhod moramo izbrati še pred vzpostavitvijo povezave s kocko, saj na podlagi teh nastavitev aplikacija kliče primeren konstruktor. Da iz senzorja vrednosti beremo s pravilnimi funkcijami, pred branjem iz senzorskega vhoda preverimo, kateri tip senzorja je priklopljen nanj. Na podoben način na osnovni formi določimo tudi tip komunikacije ter v primeru povezave Bluetooth tudi, na katerih serijskih vratih je vzpostavljena serijska povezava z kocko. Nad spustnimi meniji za izbiro tipa senzorja na sliki 4.2 vidimo izbirni meni za tip povezave in spustni meni za izbiro vrat. Poleg že omenjenih funkcij naša aplikacija omogoča tudi pre-

verjanje stanja baterije, s prikazom na osnovni formi in prižiganje luči na svetlobnem senzorju, v kolikor je ta priklopljen na kocko. Ali je luč prižgana ali ne, določimo v kodi, ki teče na PLCSIM. Aplikacija omogoča uporabo večih senzorjev iste vrste. Če želimo, lahko priklopimo tudi štiri svetlobne senzorje.

Poglavje 5

Uporaba v praksi

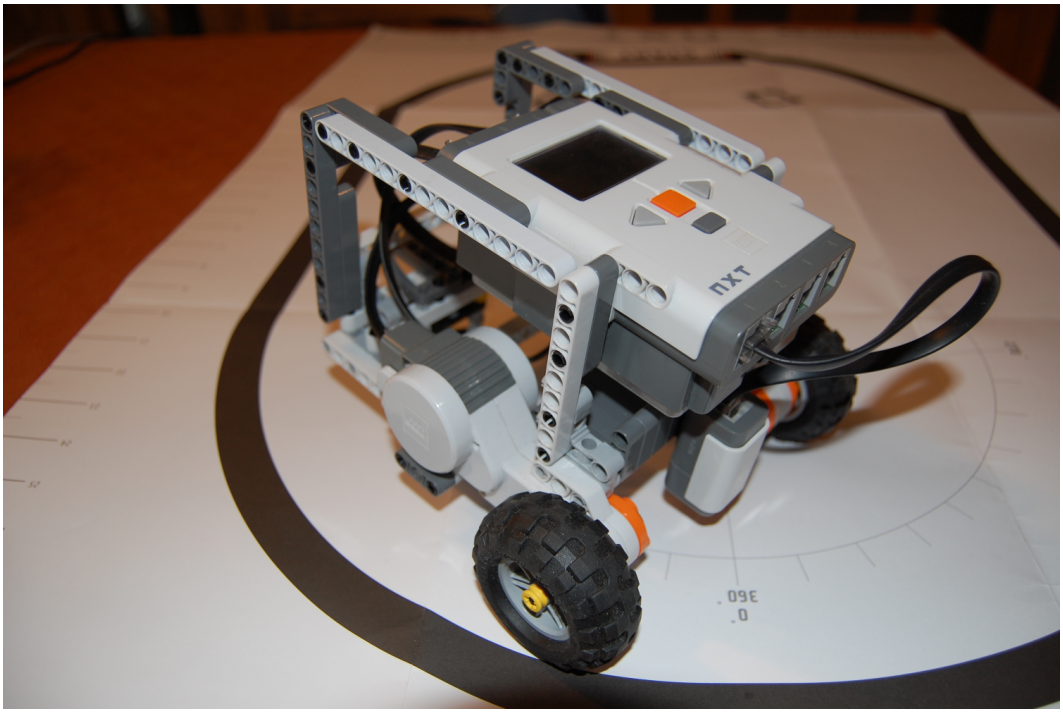
Kot primer uporabe v praksi smo si izbrali aplikacijo sledenja robu črte. Idejo smo našli objavljeno na internetu [7]. Izdelali smo vozilce s svetlobnim senzorjem in tremi kolesi, od katerih dve sta priterjeni vsaka na svoj motor (Slika 5.1), tretje kolo je prosto vpeto in vozilu omogoča spreminjanje smeri glede na vrtenje gnanih koles (Sliko 5.2).

Programska koda za sledenje črti je sprogramirana v Simatic Step7 in se izvaja samo na PLCSIM. Naša aplikacija skrbi za izvedbo akcij na vozilu in za branje vrednosti iz svetlobnega senzorja.

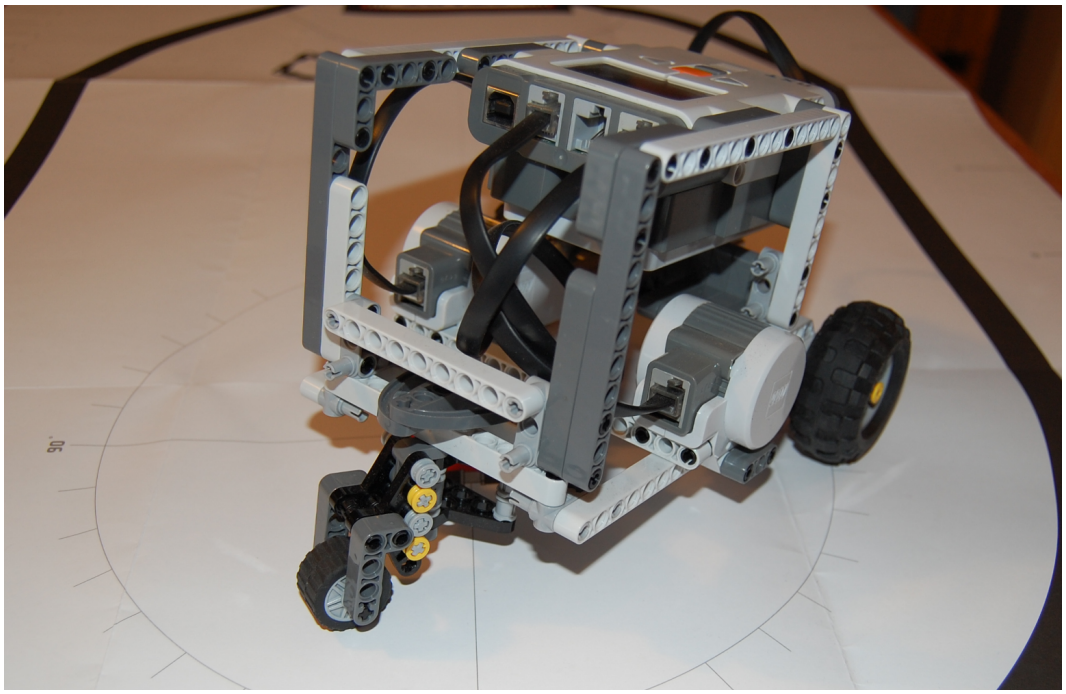
Uporabili smo funkcijski blok FB41, ki omogoča kontinuirano regulacijo PID, s pomočjo katere izračunavamo razliko med hitrostjo levega in desnega kolesa. Glede na izračunano razliko lahko vozilce spremeni smer in tako sledi črti. Čeprav podrobnosti PID regulacije niso predmet tega diplomskega dela, moramo omeniti, da je glede na svetlobne pogoje v prostoru in podlago v regulator potrebno predhodno vnesti željeno vrednost (ang. set point - SP). Vrednost izračunamo s pomočjo preproste enačbe

$$SP = crta + (podlaga - crta), \tag{5.1}$$

iz vrednosti svetlobnega senzorja na podlagi in na črti. Vpliv svetlobnih pogojev okolice zmanjšamo tako, da je ves čas prižgana pomožna lučka na senzorju.



Slika 5.1: Vozilce z gnanima kolesoma in svetlobnim senzorjem.



Slika 5.2: Vozilce s pomožnim kolescem.

S pomočjo enačbe

$$Kp = (0 - moc)/(0 - razlika) \tag{5.2}$$

smo izračunali parameter Kp in ga v FB41 vnesli s pomočjo orodja PID Control parameter Assignment, ki je tudi del orodja Simatic Step7.

V enačbi za izračun Kp spremenljivka 'moč' predstavlja želeno hitrost obeh motorjev pri vožnji naravnost, spremenljivka 'razlika' pa polovično vrednost razlike med odčitkom senzorja na črti in na podlagi.

Med testiranjem se je pokazalo, da je povezava Bluetooth prepočasna za uporabo pri reševanju našega problema. Vozilce je črti lahko sledilo le pri nizkih hitrostih. Ko smo poizkušali z večanjem hitrosti je zaradi počasnosti povezave Bluetooth hitro izgubilo črto. Dodatne težave je povzročala omejena natančnost svetlobnih senzorjev, odvisno od podlage in svetlobnih pogojev se izmerjena vrednost med belo podlago in črno črto lahko razlikuje le za 10 enot (+/- 5). Torej je med vožnjo naravnost in najbolj ostrim zavijanjem le 5

stopenj, kar pa je občutno premalo. Ker operiramo na tako ozkem področju merjenih vrednosti, pri višjih hitrostih vozilce prej izgubi črto, kot regulator uspe popraviti smer vožnje. V primeru, ko se programska koda, ki krmili vozilce, izvaja neposredno na programabilni kocki, vozilce lahko dosega tudi desetkrat višje hitrosti, saj ni zakasnitve zaradi povezave Bluetooth.

Poglavje 6

Zaključek

Motiv za razvoj aplikacije je bila želja po raziskavi možnosti uporabe kompleta Lego Mindstorms pri poučevanju procesne avtomatike. Ker nam orodje PLCSIM s knjižnico S7PROSIM ponuja že SIEMENS, je na začetku največjo težavo predstavljala odločitev ali uporabiti knjižnico Mindsqualls in se morda odreči delu svobode pri razvoju aplikacije, ali pa samostojno razviti vso potrebno kodo za komunikacijo s kocko. Zaradi same širine diplomskega dela smo se odločili za uporabo knjižnice Mindsqualls, ki se je izkazala za zelo primeno. Med samim razvojem smo se srečevali z raznimi težavami zaradi nepoznavanja razvojnega okolja Visual Studio Express in programskega jezika C#, ko pa smo ju osvojili, večjih težav ni bilo. Nekaj težav je povzročila znana napaka v .NET okolju, saj za serijske naprave na povezavi Bluetooth funkcija GetPortNames, na koncu imena vrat pripne črko [8].

Cilj je bil izkoristiti čim več možnosti ki nam jih Mindstorms ponuja in kljub nekaterim kompromisom smo cilj v veliki meri dosegli. Med pripravo predstavitvene aplikacije se je pokazalo nekaj pomanjkljivosti uporabe Mindstorms za ta namen, predvsem zaradi počasnosti povezave Bluetooth. Verjetno bi z optimiziacijo kode lahko deloma omiliti težave počasne povezave. Kadar je hitrost povezave pomembna se je bolje poslužiti povezave USB. Ker je bilo vozilce avtonomno, je bila uporaba USB v našem primeru žal nemogoča.

Da bi bila aplikacija primerna za uporabo na fakutleti, bi bilo potrebno dodati še preverjanje ustreznosti vsebine vseh vnosnih polj ter aplikacijo ponuditi v uporabo testni skupini uporabnikov. Preverjanje vnosnih polj je nujno, da preprečimo uporabnikom vnos vrednosti, ki bi lahko ustavile izvajanje aplikacije.

Možnosti za izboljšave je še veliko, kaj je smiselno in kaj ne pa bi se najbolje pokazalo z izčrpnim testiranjem in množično uporabo.

Literatura

[1] Siemens, 'Programing with STEP 7'. Dostopno na:
https://www.automation.siemens.com/doconweb/pdf/
SINUMERIK_SINAMICS_02_2012_E/S7P.pdf

[2] Siemens, 'S7ProSim V5.3 incl SP1 COM Object'. Dostopno na:
http://cache.automation.siemens.com/dnl/jkxNzkzAAAA_21170043_HB/
ProsimActive_e.pdf

[3] Microsoft, 'COM: Component Object Technologies'. Dostopno na:
http://www.microsoft.com/com/default.mspx

[4] Lego, 'Lego Mindstorms NXT'. Dostopno na:
http://mindstorms.lego.com/en-us/history/default.aspx

[5] Niels K. Handest, 'Mindsqualls'. Dostopno na:
http://www.mindsqualls.net/Default.aspx

[6] Lego, 'Lego Mindstorms NXT Bluetooth Developer Kit'.

[7] J. Sluka, 'A PID Controller For Lego Mindstorms Robots'. Dostopno na:
http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html

[8] 'Problem Serialport Class GetPortNames'. Dostopno na:
http://msdn.microsoft.com/en-us/library/system.io.ports.serialport.
getportnames(v=vs.90).aspx

[9] Siemens, 'Run/Run - P in S7 - 400'. Dostopno na: https://www.automation.siemens.com/WW/forum/guests/PostShow.aspx?HTTPS=REDIR&PageIndex=1&PostID=238659&Language=en

[10] Philippe E. Hurbain 'NXT motor internals'. Dostopno na: http://www.philohome.com/nxtmotor/nxtmotor.htm

[11] Uroš Lotrič 'Prosojnice predavanja Procesna avtomatika 2010'.