OSNOVNA ŠOLA MIHE PINTARJA TOLEDA
KIDRIČEVA 21, 3320 VELENJE
MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

LEGO NXT IN SUDOKU

Tematsko področje: ROBOTIKA

Avtor:
Luka Jevšenak, 9. razred

Mentor
Dejan Zupanc, prof.

Velenje, 2014

Raziskovalna naloga je bila opravljena na osnovni šoli Mihe Pintarja Toleda Velenje.

Mentor: Dejan Zupanc, prof.

Datum predstavitve:

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD OŠ Mihe Pintarja Toleda, šolsko leto 2013/2014
KG lego nxt/sudoku/programski jezik NXC/
AV Jevšenak, Luka
SA ZUPANC, Dejan
KZ 3320 Velenje, SLO
ZA OŠ Mihe Pintarja Toleda Velenje
LI 2014
IN LEGO NXT IN SUDOKU
TD Raziskovalna naloga
OP
IJ SL
JI sl/en
AI Na spletu lahko najdemo posnetke robotov, zgrajenih s kompletom Lego NXT, ki rešujejo različne naloge: vožnjo skozi labirint, reševanje sudokujev in celo sestavljanje rubikove kocke. Ker sem ljubitelj logike, a samo začetnik v računalniškem programiranju, sem se odločil za kompromisni izziv: naredil bom robota, ki bo rešil sudoku 4x4, a namesto števil od 1 do 4 bo uporabljal kroglice štirih različnih barv. Senzorji v Lego NXT kompletu ne omogočajo prepoznavanja števil (za kaj takega je potrebno obsežno računalniško znanje), barvni senzor pa prepozna 6 barv in za vsako vrne ustrezno številko. Sudoku polje predstavlja črno pobarvana deska, v katero je zvrtanih 16 lukenj. Robot pokrije površino polja s premiki koles naprej–nazaj in vrtenjem roke levo–desno. Na roki so barvni senzor, klešče za premikanje kroglic in motor, ki jih odpira. Program je napisan v posebej prirejenem programskem jeziku C (imenuje se NXC), ki vsebuje tudi ukaze za krmiljenje NXT robotov. Program ima tri faze: 1. branje začetne situacije, 2. reševanje, 3. postavitev manjkajočih kroglic na prava mesta. Za uspešno izvedbo naloge morajo biti premiki robota milimetersko natančni. Zato ima robot zobniške prenose z motorja na osi, sami motorji pa se krmilijo s kotom zasuka. Uspešnost pri branju je praktično 100 %, pri polaganju kroglic v luknjice pa več kot 90 %. Robot manjkajoče kroglice pobira s podajalca, kjer so kroglice razvrščene v naslednjem vrstnem redu: 1. modra, 2. zelena, 3. rumena in 4. rdeča. Preden se robot požene, je potrebno napraviti tri stvari: na sudoku polju postaviti začetno situacijo, preostale kroglice zložiti na podajalec in barvni senzor robota usmeriti na prvo mesto polja (levo zgoraj). Potem se požene program in robot v približno sedmih minutah opravi vse ostalo. Na koncu je vseh 16 mest zapolnjenih s kroglico ustrezne barve.

KEY WORDS DOCUMENTATION

ND OŠ Mihe Pintarja Toleda, 2013/2014
CX
AU Jevšenak, Luka
AA ZUPANC, Dejan
PP 3320 Velenje, SLO,
PB OŠ Mihe Pintarja Toleda Velenje
PY 2014
TI LEGO NXT AND SUDOKU
DT RESEARCH WORK
NO
LA SL
AL sl/en

AB On YouTube we are able to find Lego NXT robots, solving different problems like sudoku and rubik's cube. But dealing with that kind of problems requires a great deal of experience and knowledge about computer programming. I have just started to learn the C programming language, which enables me to guide NXT robots. So the problem in my research work should have been appropriate to my knowledge. The decision was to solve sudoku 4x4, but instead of reading and writing numbers (that's a very complex problem with Lego NXT sensors) I used balls of different colours. The colour sensor returns different numbers for different colour. My task was to create and programme a robot, which could read the initial position of the balls on the sudoku plate, find the solution and fulfill the empty spots with missing balls. The final version of the robot moves forward – backward on wheels. The arm, with colour sensor and pliers for grabbing the balls, turns left – right. Moves of the robot must have been a millimetre precise. That was achieved by using gears and controlling motors by angle of rotation. The robot takes missing balls from the inclined pedestal, where the balls are sorted in the following order: blue, green, yellow, red. We have to do three things, before we start the robot: put the balls on the initial position on the sudoku plate, sort out the rest of the balls on the pedestal and direct the colour sensor to the first place (upper – left) on the sudoku plate. After starting the program, the robot needs approximately 7 minutes to complete the task. At the end there are all 16 balls in the right spots on the sudoku plate.

Kazalo vsebine

Kazalo slik

1 UVOD

V šoli pri izbirnem predmetu robotika nam je učitelj predstavil sistem Lego NXT. Nad njim sem se tako navdušil, da sem s pomočjo staršev kupil lastni komplet. Ko sem iskal nove ideje pri sestavljanju in programiranju, sem na spletnem portalu YouTube videl robote, ki rešujejo različne logične naloge, od sudokujev pa vse do sestavljanja rubikove kocke. Da bi tudi sam pripravil robota do česa podobnega, mi je predstavljalo velik izziv. Ker pa sem začetnik na področju računalniškega programiranja, sem moral najti sebi primeren problem. Tako je počasi v pogovorih s starši in mentorjem dozorela ideja, da izdelam robota, ki bo reševal sudoku 4x4, vendar bi namesto števil uporabil kroglice štirih različnih barv. Kroglice modre, zelene, rumene in rdeče barve so del kompleta Lego NXT. Na ta način se poenostavi branje začetne situacije v sudoku polju, saj barvni senzor v kompletu prepozna barvo kroglice in vrne ustrezno številko. Branje in prepoznavanje samih števil s svetlobnim senzorjem bi predstavljalo bistveno večji programerski zalogaj. Po ugotovitvi začetne pozicije kroglic v polju, robot poišče rešitev in v prazna polja postavi manjkajoče kroglice. Na spletu (vsaj zaenkrat) še nisem našel posnetka, kjer bi robot rešil tako zastavljeno nalogo. Pri izdelavi robota sem se zanašal na svojo inovativnost in veliko veselje do sestavljanja, računal pa sem tudi na strojniške izkušnje, ki sem jih dobil s kompletom Fischer Technic. Programerski del naloge mi je na začetku vlival več spoštovanja, saj sem se s programskim jezikom C šele spoznaval in vse moje izkušnje so bile par kratkih programčkov za premikanje robota. Ker pa rad rešujem logične naloge in ker me zanima računalništvo, sem se odločil, da se preizkusim tudi v tem.

HIPOTEZE:

1. Komplet Lego NXT omogoča izgradnjo robota, ki samostojno, brez vmesnih posredovanj, reši sudoku 4x4 z barvnimi kroglicami namesto števil (branje začetne situacije, rešitev problema, postavitev manjkajočih kroglic na prazna mesta).
2. Sam bom uspel sestaviti in sprogramirati takega robota, čeprav sem v računalniškem programiranju začetnik.

2 PREGLED OBJAV

2.1 Sudoku

Sudoku je logična naloga pri kateri mora reševalec zapolniti vsa polja v kvadratu, tako da se v vsaki vrsti, stolpcu in manjših kvadratih ne ponovi nobena od številk. Najpogostejši od sudokujev je 9x9, poznamo pa tudi sudokuje 6x6, 4x4 in barvne sudokuje. Številske uganke so se pojavile takrat, ko so francoski sestavljalci ugank v 19 stol., začeli eksperimentirati z odstranjevanjem števil iz magičnega kvadrata. Čeprav je ime japonsko, je izum sudokuja zagotovo ameriški. Sestavil ga je Howard Garns in ga prvič objavil v reviji leta 1979 pod imenom Number place (1).

	2	4	
			2
3			
	1	3	

Slika 1: Sudoku 4 x 4

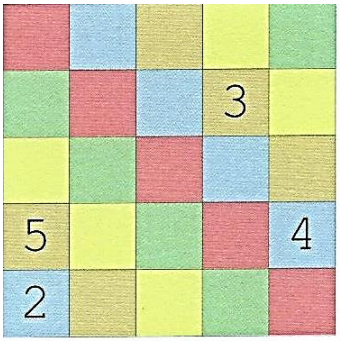
			1		6
6		4			
1		2			
			5		1
			6		3
5		6			

Slika 2: Sudoku 6 x 6

			1		5		6	8
						7		1
9		1					3	
		7		2	6			
5								3
			8	7		4		
	3					8		5
1		5						
7	9		4		1			

Slika 3: Sudoku 9 x 9

(Vir: http://www.dailysudoku.com/sudoku/examples.shtml, 1.12. 2013)



Slika 4: Barvni sudoku

(Vir: Logika in razvedrilna matematika: Barvni sudoku 5x5. sedemnajsti letnik, 2007-2008, 3 zvezek, stran 3)

2.2 Lego NXT 2.0

Za reševanje sem uporabil Lego robota iz programa Mindstorms NXT 2.0. Na eno "pametno opeko" se lahko priključi tri servo-motorje in štiri senzorje. Za nalogo je poleg motorjev najpomembnejši barvni senzor. Ta prepozna 6 različnih barv, za vsako pa vrne svojo številko: 1. črna, 2. modra, 3. zelena, 4. rumena, 5. rdeča in 6. bela. Del kompleta je tudi programska oprema LabVIEW, ki omogoča programiranje robota za opravljanje različnih nalog. Tako imenovano programiranje "s sličicami" omogoča enostavno premikanje vozila in vključevanje senzorjev v odločitve, če si le–te sledijo ena za drugo. Za malo kompleksnejše naloge pa program postane hitro preobsežen in nepregleden. Po priporočilu gospoda Petra Vrčkovnika s Srednje elektro in računalniške šole Šolskega centra Velenje sem za programiranje robota uporabil prirejeno verzijo jezika C.

2.3 Bricx Command Center

Je okolje, v katerem programiramo Lego robote v programskem jeziku NXC. To je prirejena verzija jezika C z dodanimi funkcijami za upravljanje motorjev in senzorjev. NXC je za razliko od programa C precej omejen s stavki za kontrolo toka. Tako sem probleme reševal s ***for*** in ***do while*** zanko in ***if*** stavki (2). NXC ne pozna stavka ***switch*** in tudi ne večdimenzionalnih polj, kar mi je otežilo programiranje in podaljšalo program.

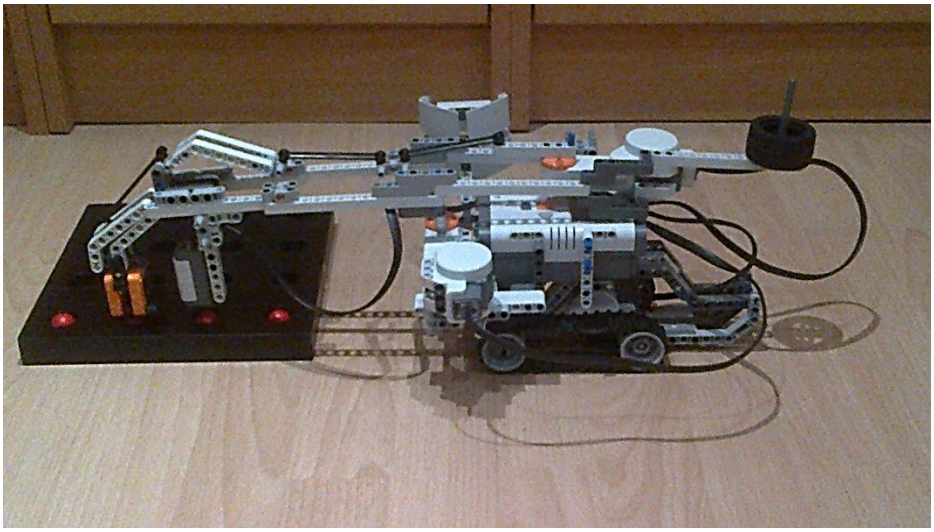
Slika 5: komplet Lego Mindstorms 2.0

3 METODE DELA

3.1 Robot

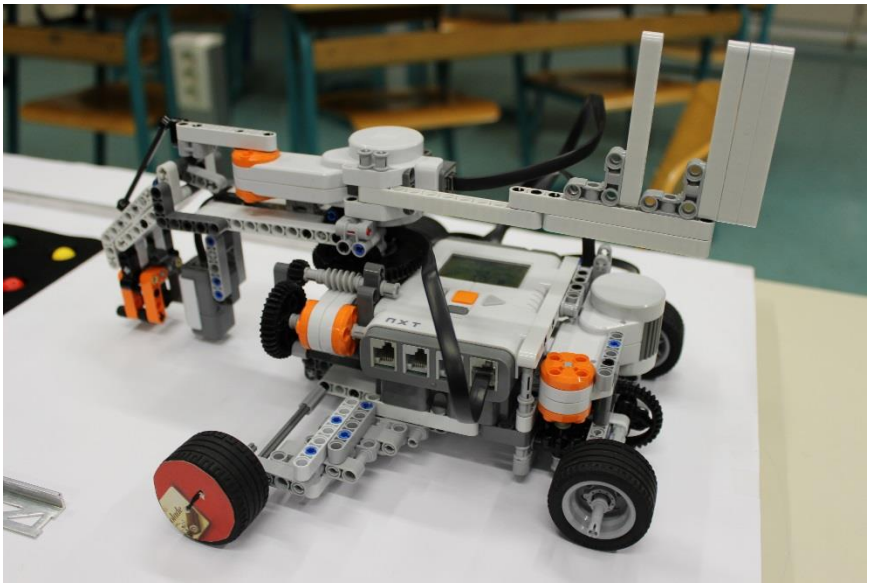
Robota sem sestavil s svojim kompletom Lego mindstorms NXT 2.0. Komplet me je omejeval na uporabo treh servo-motorjev. En motor potrebujem za premik robota naprej – nazaj in še enega za premik roke levo–desno. Tako robot pokrije ravnino sudoku polja. Preostali motor sem uporabil za odpiranje in zapiranje klešč na roki, s katero robot postavlja kroglice. To je avtomatično pomenilo, da bo robotu na koncu potrebno na nek standardiziran način ponuditi kroglice za dopolnitev polja.

Prvi poskusi izdelave robota so se izkazali za preveč nenatančne. Roka je bila predolga, robot previsok in preozek in zato je precej nihal. Motorji so neposredno vrteli kolesa in roko. Rezultat je bil, da robot ni uspel uspešno prebrati niti enega stolpca, kaj šele celotnega polja.

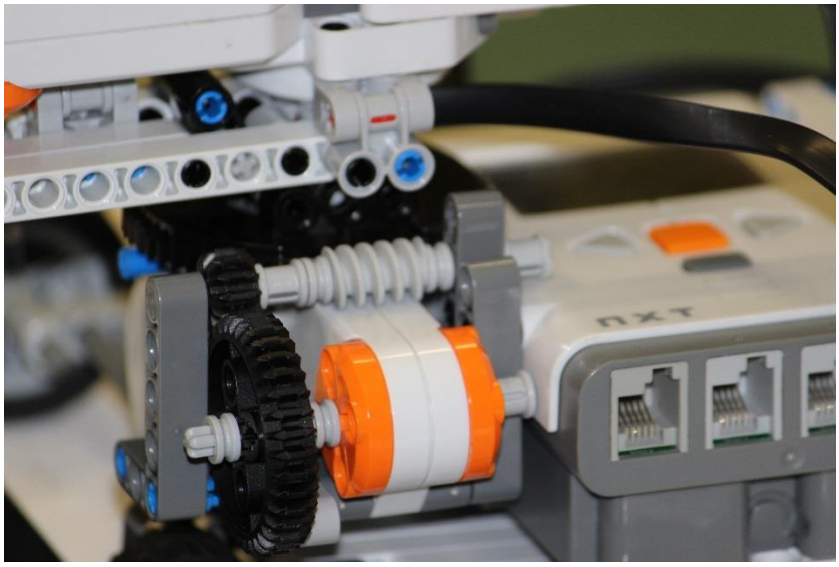


Slika 6: Prvi poskus izdelave in preizkušanje robota

Robot je nato doživel korenito prenovo. Naredil sem ga nižjega, širšega, mu skrajšal roko in za povečanje natančnosti premikov dodal zobniške prenose. Ker se je roka še vedno nagibala naprej, sem jo uravnotežil z repom na nasprotni strani osi.

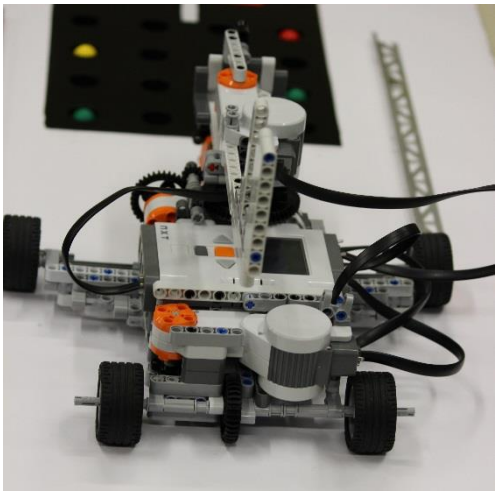


Slika 7: Končna izvedba robota



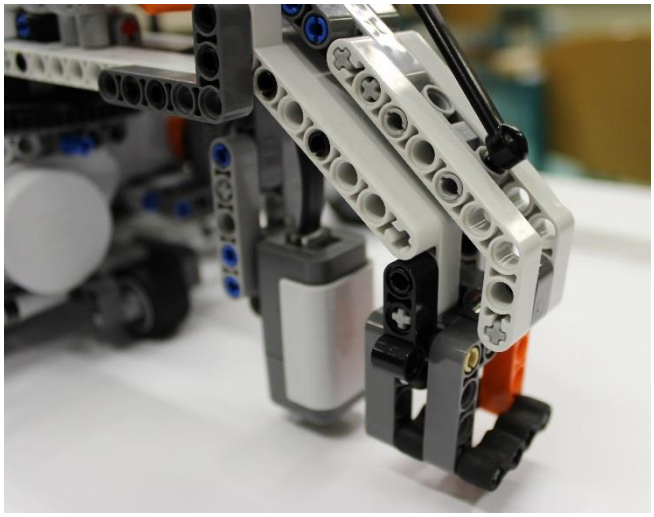
Slika 8: Motor in zobniški prenos za zasuk roke

Pri zobniškem prenosu z motorja na roko sem uporabil polžasti zobnik. Pričvrstitev motorjev ne dopušča popolne simetrije, zato os, okrog katere se vrti roka, ni na sredini robota. Tako je bilo potrebno v programu zasuk za vsak stolpec nastaviti posebej.



Slika 9: Zadnji, pogonski del

Pogonski motor preko zobnikov v razmerju 3:1 vrti vodoravno os, na kateri sta kolesi. To omogoča natančno premikanje robota naprej in nazaj.

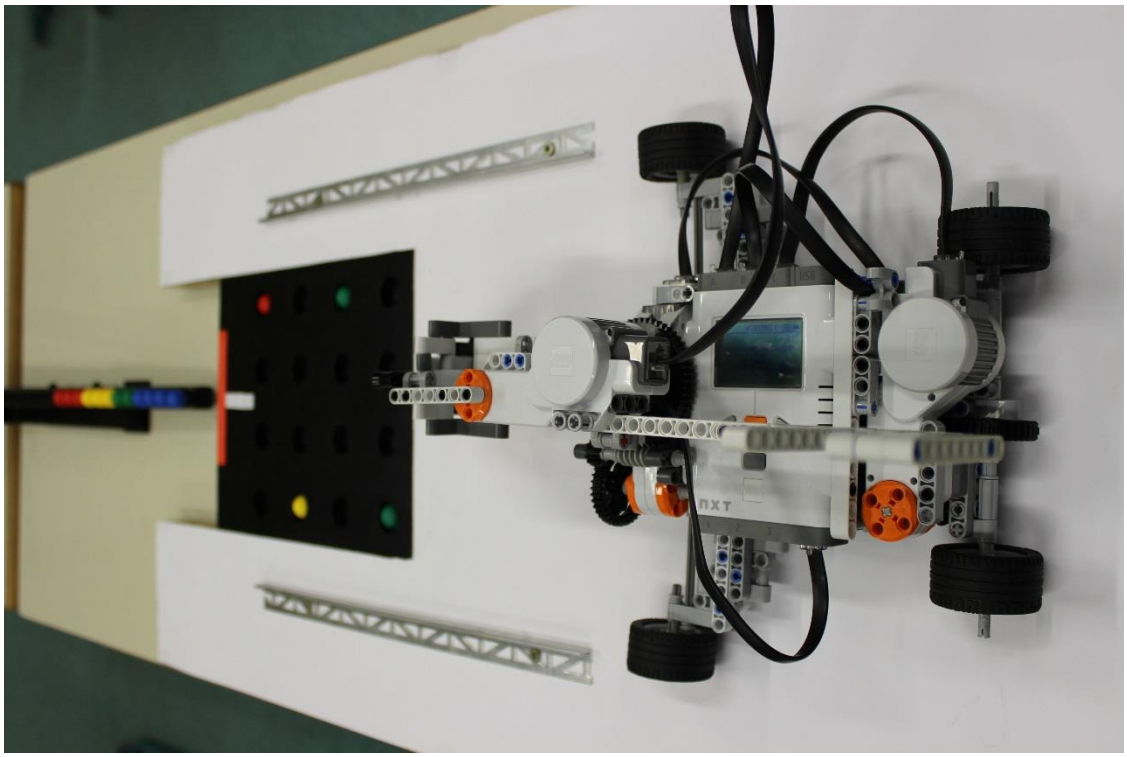


Slika 10: Roka s kleščami za prijem kroglic in barvnim senzorjem

Klešče drži skupaj gumica. Da se čeljusti razprejo, mora motor sunkovito potegniti ročico navzgor.

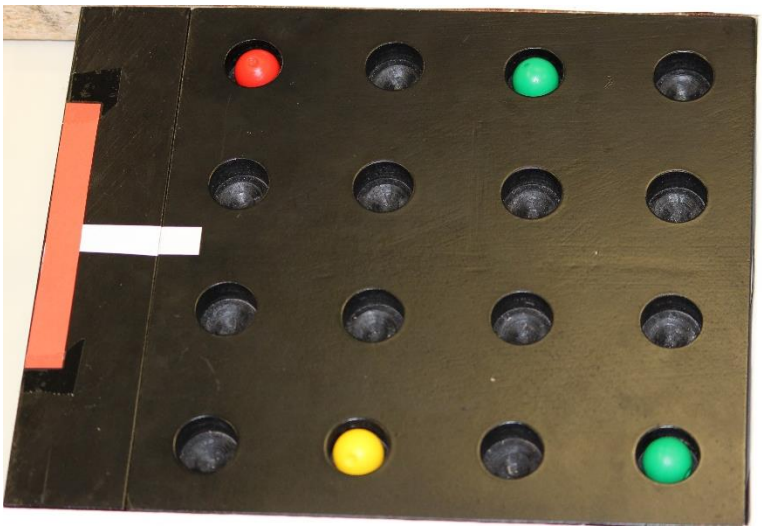
3.2 Dodatna oprema

V pomoč robotu sem naredil črno ploščo s sudoku poljem 4x4. Da je vse v isti višini, sem dodal še ploščo z vodili, po kateri se robot vozi. Na koncu sem se odločil dodati še podajalec za kroglice.



Slika 11: Robot in vsa ostala oprema

3.2.1 Plošča s sudoku poljem 4x4



Slika 12: Plošča s sudoku poljem

Plošča je kvadratna z stranico 20 cm. Ima 16 lukenj premera 20 mm, v katere postavimo kroglice (premera 16,5 mm) za začetno postavitev. Modre kroglice pomenijo številko 1, zelene 2, rumene 3, rdeče pa 4. Odločil sem se za črno barvo plošče (lahko bi bila tudi bela), ker se mi je zdelo manj verjetno, da bi

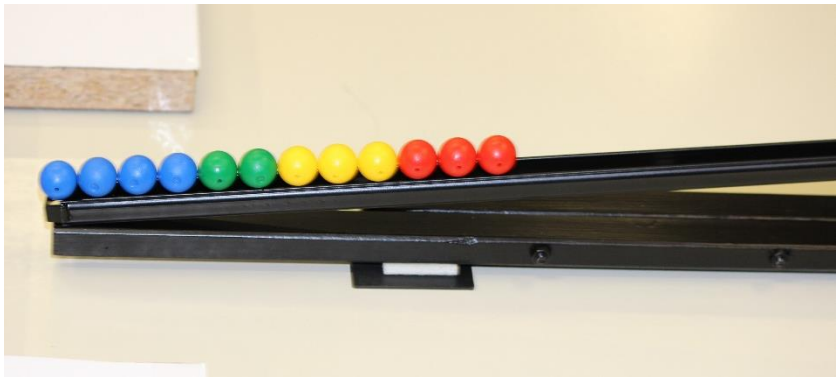
senzor na praznem mestu odčital napačno barvo. Plošča s sudokujem se prilega v večjo ploščo z vodili. Zraven plošče postavimo še manjšo ploščo z belo rdečim »T«. Ta T robot uporablja za iskanje podajalca kroglic. Tako robot prime kroglico vedno na istem mestu, tudi če ga med postavljanjem prejšnje kroglice kaj zmoti.

3.2.2 Vodila

Slika 13: Vodila

Vodila sem dodal, ko sem opazil, da robota obrača v levo v zadnji fazi pri postavljanju kroglic. Razlog je motor na roki, ki mora sunkovito odpirati klešče. Kot sem že omenil, postavitev motorja ne dopušča simetrije. Za učinkovito razpiranje čeljusti se motor vedno zavrti v isto smer, kar povzroča takšen sunek, da robot začne izgubljati orientacijo. Vodila na prednjih kolesih prečne sunke uspešno nevtralizirajo in robot ohranja smer.

3.2.3 Podajalec kroglic



Slika 14: Podajalec kroglic

Ker ima robot na razpolago samo tri motorje, ni bilo boljše možnosti kot narediti podajalec. Zamislil sem si ga kot klanec, s katerega robot jemlje kroglice, ki jih potem postavi na pravo mesto. Kroglice se na podajalec postavi vedno v tem vrstnem redu: modre, zelene, rumene in rdeče.

3.3 Programiranje

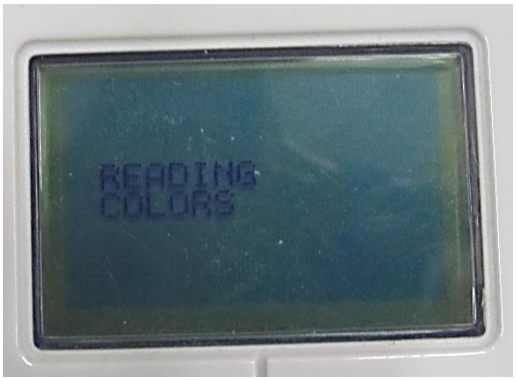
Programiral sem s programskim jezikom NXC v programu Bricx Command Center. Takoj sem naletel na težavo, da NXC ne prepozna dvodimenzionalnih polj. Uporabil sem enodimenzionalna polja, kjer sem uporabil pretvorbo a[x*4+y], x je številka vrstice in y številka stolpca. Tako je na primer element a[10] enodimenzionalnega polja enak elementu a[2][2], kar pomeni tretjo vrstico in tretji stolpec na sudoku polju (števci se začnejo z 0). Probleme sem imel tudi pri programiranju premikov. Sprva sem premike programiral na čas vrtenja motorja pri določeni hitrosti (kombinacija ukazov *OnFwd, Wait*), vendar je vsaka najmanjša sprememba (popravek zgradbe, napolnjenost baterij) vplivala na opravljeno pot. Po nenehnem ponastavljanju premikov sem prišel do sklepa, da tako ne bo šlo. Servo-motorji naj bi se vendar dali upravljati bolj natančno. Po sprehodu skozi pomoč (Help) NXC-ja sem končno našel ukaz *RotateMotor*, ki omogoča kontrolo kota zasuka motorja. To je izničilo napako zaradi spreminjajoče moči baterij in celo nekoliko skrajšalo program. Razdalja med središčema dveh sosednjih lukenj na sudoku polju je 5 cm, obseg pogonskega kolesa na robotu je 13,35 cm (premer je 4,25 cm), prenos z motorja na kolo pa je 3:1. Tako sem izračunal, da za premik od luknje do luknje zavrtim pogonski motor za 404°.

Program sem razdelil na tri dele:

- branje začetne postavitve
- reševanje sudokuja
- postavljanje kroglic.

3.3.1 Branje začetne postavitve

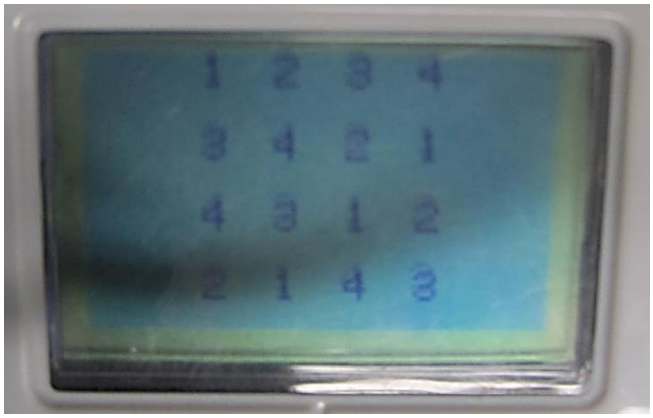
Robota se ročno nastavi na prvo mesto sudoku polja, a[0][0] ali drugače povedano, barvni senzor usmerimo v prvo luknjo. Nato se robot premika po stolpcih navzdol, a[1][0], a[2][0] in a[3][0]. Nad vsako luknjo se robot ustavi za sekundo in na polovici svojega postanka (500 ms) preveri barvo polja (kroglice). Ko pregleda cel stolpec, se premakne nazaj na prvo mesto v stolpcu in se prestavi na prvo mesto v naslednjem stolpcu, a[0][1]. To robot ponavlja dokler ne prebere vseh šestnajst mest. Nato se začne drugi del.



Slika 15: Izpis na ekranu med branjem

3.3.2 Reševanje sudokuja

Robot reši sudoku tako, da program poišče prvo prazno mesto v polju 4x4 in preveri barve vseh mest v tej vrstici, stolpcu in diagonalno mesto v lokalnem kvadratu 2x2, ki mu pripada. Če po tem ugotovi, da je na to mesto mogoče postaviti samo eno številko, pripiše ustreznemu elementu polja vrednost te številke in se premakne naprej. Če je trenutno možnih več številk, ki se jih da postaviti na to mesto, program to mesto pusti prazno in se premakne naprej. Če pa na obravnavano mesto ne more postaviti nobene številke, ker bi se pri tem ponovila, program sudoku prepozna kot nerešljiv in izpiše na ekranu robota ERROR. Ko pregleda vsa mesta in sudoku še ni rešen, ponovno začne nov krog pregledovanja. To ponavlja dokler sudoku ni rešen (vseh 16 elementov polja je različnih od 1). Če v določenem krogu na tak način ne postavi nobene številke, to pri 4x4 sudokuju pomeni, da ima več rešitev ali pa sploh ni rešljiv. Tudi v takem primeru robot izpiše ERROR. Sicer pa se na ekranu izpiše rešitev in začne se tretja faza. Program rešuje sudoku s števili od 2 do 5, ker so to številke barv, ki jih vrača barvni senzor. Pri izpisu na ekranu sem od elementov polja odštel število 1, tako da je rešitev podana bolj standardno s števili od 1 do 4. Del programa, ki reši sudoku, je jedro celotnega programa in je v prilogi A.



Slika 16: Izpis rešitve na ekranu robota

3.3.3 Postavljanje kroglic

Najprej robot s pomočjo belo-rdečega T najde podajalec in pobere prvo modro kroglico. Nato preveri, kateri je prvi stolpec brez modre kroglice in na katero mesto mora postaviti to kroglico. Odnese jo tja, spusti jo v luknjico in se vrne po novo kroglico. Ko postavi vse modre, se loti zelenih itd.

Priloga B na DVD-ju je film, kjer robot opravi vse zgoraj opisane faze. Film je zmontiran, sestavljen je iz več kadrov in slik.

4 REZULTATI IN DISKUSIJA

Robot na poligonu s programom, ki je opisan v točki 3, opravi zastavljeno nalogo – reševanje sudokuja 4x4. Delo opravi popolnoma samostojno od branja začetne postavitve, reševanja in dopolnitve praznih mest z barvnimi kroglicami. Sposoben je prepoznati tudi nerešljive sudokuje in take z več rešitvami. Pri rešljivem sudokuju z najmanjšim možnim številom kroglic v začetni postavitvi (štiri kroglice) robot potrebuje približno sedem minut za celotno opravilo (priloga C na DVD-ju: film, kjer robot opravi nalogo od začetka do konca brez menjave kadrov). Z izpolnitvijo cilja sem potrdil obe postavljeni hipotezi. Končna verzija programa (priloga D na DVD-ju) obsega 891 vrstic in pri njegovem snovanju sem pridobil dragocene programerske izkušnje.

Pri premikih robota prihaja do majhnih odstopanj. Začetna postavitev robota, nastavimo ga ročno, vedno za kak milimeter odstopa od prejšnje začetne postavitve. Izkušnje kažejo, da so tudi premiki robota za kak milimeter nenatančni. Vsi ti milimetri pa se včasih tako seštejejo, da se kakšna kroglica odbije od roba luknje. To napako bi lahko verjetno odpravil, če bi luknje še povečal ali vsaj dobro posnel rob. Vendar bi potem naloga morda izgledala prelahka, zanesljivost pa je že sedaj dovolj velika, da se mi to ni zdelo potrebno. Po moji oceni je natančnost večja kot 90%.

5 ZAKLJUČEK

Zadal sem si praktično nalogo: z Lego NXT kompletom sestaviti in sprogramirati robota, ki bo rešil sudoku 4x4. Nalogo sem uspešno rešil. Moj robot je sposoben samostojno prebrati, rešiti in dokončati sudoku, ki namesto števil od 1 do 4 uporablja kroglice štirih različnih barv. Prav tako sem tudi sam napisal program, ki robota vodi med nalogo. Za končnim izdelkom se skrivajo ure in ure testiranj in izboljšav, tako zasnove robota kot programa, ter natančnosti premikov. Predvsem sem vesel novega računalniškega znanja. Mislim, da bi se sedaj upal lotiti reševanja večjih sudokujev 6x6 in 9x9. Način, ki sem ga uporabil za reševanje sudokuja 4x4, pri večjih sudokujih ne bi deloval. To pa zato, ker se pri njih možnosti izključujejo še na druge načine, ne samo po stolpcih, vrsticah in lokalnih kvadratih(9x9) ali pravokotnikih(6x6). Večjega sudokuja bi se lotil tako, da ko program ne bi mogel zagotovo postaviti nobene številke, bi v izbrano polje eno od možnih števil postavil sam. Če bi se ta možnost pokazala kot protislovna, bi se vrnil nazaj in nadaljeval z naslednjo možnostjo. Večji sudokuji s kroglicami se mi ne zdijo smiselni, zato bi moral robot preko barvnega ali svetlobnega senzorja znati prepoznati zapisana števila. Tu leži jedro problema in zaenkrat še nisem zares razmišljal, kako bi se tega lotil.

6 POVZETEK

S sistemom Lego NXT sem se seznanil pri izbirnem predmetu robotika v 8. razredu osnovne šole. Takoj me je navdušil in s pomočjo staršev sem kmalu imel svoj komplet doma. Pri iskanju novih idej, kaj vse bi še lahko sestavil, sem brskal po spletu. Našel sem posnetke robotov, ki iščejo pot skozi labirinte in rešujejo celo logične naloge: sudoku in rubikovo kocko. To, da bi tudi moj robot opravil kakšno logično nalogo, mi je predstavljalo velik izziv. Tako sem se odločil za nalogo, ki bi ji kot začetnik v programiranju lahko bil kos: reševanje sudokuja 4x4, vendar bi namesto števil uporabil kroglice štirih različnih barv, ki so del Lego NXT kompleta. Za branje in prepoznavanje števil s priloženimi senzorji bi potreboval bistveno višji nivo računalniškega znanja. Barvo kroglice pa prepozna barvni senzor in vrne ustrezno številko. Rešil bi torej sudoku s števili, na koncu pa na posamezno mesto sudoku polja ne bi napisal številke, ampak bi tja položil kroglico ustrezne barve.

Za začetek sem se moral najprej poglobiti v programski jezik C. Uporabil sem okolje Brixc Command Center, kjer lahko Lego NXT robote programiramo z izvedenko tega jezika, ki se imenuje NXC. Ta je precej okrnjena, kar se tiče ukazov za kontrolo toka, zato pa omogoča precej enostavno upravljanje z roboti. Nalogo sem razdelil na tri dele: branje začetne situacije na sudoku polju, reševanje, dopolnitev polja z manjkajočimi kroglicami. Seveda pa je bilo potrebno sestaviti robota, ki je to sposoben narediti. Končna izvedba robota je na kolesih, s katerimi se premika naprej–nazaj. Na vrtljivi roki, ki se suče levo-desno, sta barvni senzor in klešče za prijemanje kroglic. Poligon, na katerem robot rešuje nalogo, ima več delov. Za sudoku polje sem uporabil leseno ploščo, v katero je zvrtanih 16 lukenj, v katere spuščamo kroglice. Ker so vsi razpoložljivi motorji že izkoriščeni, sem moral narediti podajalec, s katerega robot jemlje kroglice. Tega robot najde s pomočjo belo rdeče oznake T na dodatni plošči. Med postavljanjem kroglic robota suče v levo zaradi sunkovitega odpiranja klešč. Problem sem rešil z vodili na sprednjih kolesih. Za uspešno izvedbo naloge je potrebna milimetrska natančnost premikov. To mi je uspelo s krmiljenjem servo-motorjev na kot zasuka v stopinjah in prenosi pogonov z zobniki. Preden poženemo robota, moramo na sudoku polju postaviti kroglice v izbrano začetno situacijo. Pri tem upoštevamo, da modra pomeni številko

1, zelena 2, rumena 3 in rdeča 4. Preostale kroglice zložimo na podajalec v istem vrstnem redu. Nato centriramo barvni senzor na prvo mesto polja (levo zgoraj) in poženemo program. V najtežji situaciji, ko mora robot polje dopolniti z dvanajstimi kroglicami, rabi za celotno opravilo približno 7 minut. Program je sposoben prepoznati tudi nerešljive sudokuje, kar oznani z izpisom ERROR na ekranu. Menim, da sem nalogo v okviru možnosti uspešno rešil. Potrdil sem prvo hipotezo, da sistem Lego NXT omogoča izdelavo robota, ki samostojno reši sudoku 4x4 z uporabo barvnih kroglic. Potrdil pa sem tudi drugo hipotezo, da sem kaj takega sposoben narediti sam. Vesel sem, da je moje računalniško znanje tako napredovalo, da je vseh 891 vrstic programa izključno moje delo.

7 ZAHVALA

Rad bi se zahvalil svojemu mentorju za vso pomoč, predvsem pri izdelovanju dodatne opreme (sudoku poligona) in pisanju naloge.
Pri izdelavi poligona mi je pomagal z mizarskimi spretnostmi tudi dedek, za kar se mu zahvaljujem. Zahvala gre očetu, ki me je navdušil nad računalniškim programiranjem v programskem jeziku C. Zahvaljujem pa se tudi vsej svoji družini za podporo.

8 PRILOGE

A. Del programa, ki reši sudoku 4x4

```
task main()
{
 int a[16] = {1,5,1,1,1,4,1,1,1,1,4,1,1,1,2,1};  // začetna postavitev od leve proti desni od zgoraj navzdol
 int x,y,z,b,c,d,e,o,p,konec,er;
 er = 0;
 while(konec < 16)
 {
  konec = 0;
  if(er != 0)
  {
   TextOut(30,LCD_LINE4,"ERROR");     // prepoznavanje nerešljivega sudokuja
  }
  else
  er++;
  for(x = 0;x < 4;x++)                        // x – števec vrstic
  {
   for(y = 0;y < 4;y++)                       // y – števec stolpcev
   {
    if(a[x*4+y] == 1)
    {
     b = 0;
     c = 0;
     d = 0;
     e = 0;
     for(z = 0;z < 4;z++)
     {
      if(a[x*4+z] == 2)
      {
       b = 2;
      }
      else if(a[x*4+z] == 3)
      {
       c = 3;
      }
      else if(a[x*4+z] == 4)
      {
       d = 4;
      }
      else if(a[x*4+z] == 5)
      {
       e = 5;
      }
     }
     for(z = 0;z < 4;z++)
     {
      if(a[z*4+y] == 2)
      {
       b = 2;
      }
      else if(a[z*4+y] == 3)
      {
       c = 3;
      }
      else if(a[z*4+y] == 4)
```

```
{
   d = 4;
  }
  else if(a[z*4+y] == 5)
  {
   e = 5;
  }
 }
 if((x == 0)||(x == 2))                          // preverjanje diagonalnega člena v lokalnem kvadratu
 {
  if((y == 0)||(y == 2))
  {
   o = x + 1;
   p = y + 1;
  }
  else
  {
   o = x + 1;
   p = y - 1;
  }
 }
 else
 {
  if((y == 0)||(y == 2))
  {
   o = x - 1;
   p = y + 1;
  }
  else
  {
   o = x - 1;
   p = y - 1;
  }
 }
  if(a[o*4+p] == 2)
  {
   b = 2;
  }
  else if(a[o*4+p] == 3)
  {
   c = 3;
  }
  else if(a[o*4+p] == 4)
  {
   d = 4;
  }
  else if(a[o*4+p] == 5)
  {
   e = 5;
  }
 if((b == 0)&&(c != 0)&&(d != 0)&&(e != 0))          // ugotavljanje ali lahko na izbrano mesto postavi samo eno število
 {
  a[x*4+y] = 2;
  er = 0;
 }
 else if ((b != 0)&&(c == 0)&&(d != 0)&&(e != 0))
 {
  a[x*4+y] = 3;
  er = 0;
 }
```

```
  else if ((b != 0)&&(c != 0)&&(d == 0)&&(e != 0))
     {
      a[x*4+y] = 4;
      er = 0;
     }
     else if ((b != 0)&&(c != 0)&&(d != 0)&&(e == 0))
     {
      a[x*4+y] = 5;
      er = 0;
     }
     else if ((b != 0)&&(c != 0)&&(d != 0)&&(e != 0))
     {
      TextOut(30,LCD_LINE4,"ERROR");                    // prepoznavanje sudokuja, ki ni rešljiv ali pa ima več rešitev
     }
     else
    }
    else
    konec++;
   }
  }
 }
NumOut(25,LCD_LINE1,a[0]);              // izpis rešitev na zaslon
NumOut(40,LCD_LINE1,a[1]);
NumOut(55,LCD_LINE1,a[2]);
NumOut(70,LCD_LINE1,a[3]);
NumOut(25,LCD_LINE3,a[4]);
NumOut(40,LCD_LINE3,a[5]);
NumOut(55,LCD_LINE3,a[6]);
NumOut(70,LCD_LINE3,a[7]);
NumOut(25,LCD_LINE5,a[8]);
NumOut(40,LCD_LINE5,a[9]);
NumOut(55,LCD_LINE5,a[10]);
NumOut(70,LCD_LINE5,a[11]);
NumOut(25,LCD_LINE7,a[12]);
NumOut(40,LCD_LINE7,a[13]);
NumOut(55,LCD_LINE7,a[14]);
NumOut(70,LCD_LINE7,a[15]);
while(true)
{
 Wait(5000);
}
}
```

B. Film (4:27): sestavljeni predstavitveni video

C. Film (7:54): robot opravi celotno nalogo brez menjave kadrov

D. Program, s katerim robot opravi nalogo

9 VIRI IN LITERATURA

1. Anja Gligić, SUDOKU: seminarska naloga, 2011. Univerza v Ljubljani, Fakulteta za matematiko in fiziko.
2. Kernighan, B.W., Ritchie, D.M. 1991. Programski jezik C. Fakulteta za elektrotehniko in računalništvo, Ljubljana.