šolski center ptuj
elektro in računalniška šola

Volkmerjeva cesta 19, 2250 Ptuj

IZDELAVA ROBOSOCCER ROBOTA ZA SVETOVNO PRVENSTVO

Mentor: mag. Marjan Bezjak, *univ. dipl. inž. el.*

Avtor: Aljaž Nahberger, 4. ce

Program: SSI, elektrotehnik

Ptuj, marec 2017

Zahvala

Za pomoč se zahvaljujem vsem, ki so mi pomagali pri izdelavi robota in tudi pri pisnem delu naloge. Predvsem bi se rad zahvalil mentorju mag. Marjanu Bezjaku, ki mi je bil pri izdelavi naloge v veliko pomoč s svojimi nasveti in predlogi ter članom svoje ekipe, ki so mi ves čas stali ob strani ter pomagali uresničiti zastavljene cilje.

POVZETEK

Glavna tema moje naloge je bila izdelava robota za RoboCup Junior tekmovanje. Sam sem tekmoval v kategoriji RoboCup Junior soccer s svojo izbrano ekipo. Izdelava robota je velikokrat s seboj prinesla številne komplikacije in na videz nerešljive situacije. Ampak z veliko mero truda in vztrajnosti sem to odpravil in prišel do izdelka, na katerega sem zelo ponosen.

Ob izdelavi robota sem se naučil veliko tehničnih zadev, kot so programiranje v C programskem jeziku, risanje načrtov v orodjih: Sprint Layout, Altium Designer ter EaysEDA, uporaba CNC LPKF ProtoMat C30 rezkalnika ter mnoge druge, ki mi bodo prišle prav v nadaljnjem izobraževanju in življenju. Pridobil sem vrednoto, ki mi bo za celotno življenje koristila, in to je vztrajnost.

Robot, ki sem ga izdelal, se bo pomeril na mnogih tekmovanjih po svetu. Sama sestava robota je bila zelo zahtevna, prav tako pa sem veliko časa vložil v programiranje robota. Velikokrat sem presedel ure in ure za računalnikom in robotom. Zelo sem vesel, da mi je bila ponujena priložnost izdelave takšnega robota.

Ključne besede: RoboCup Junior, soccer, Sprint Layout, Altium Designer, EasyEDA, CNC LPKF ProtoMat C30 rezkalnik.

SUMMARY

The main theme of my project work was to construct an autonomous robot that plays football. It was used for RoboCup Junior competitions. Personally, I competed in RoboCup Junior soccer category, with a team I chose. The robot consists of a driving part, mechanical part, electronics and software. Its task is to find the ball on the playground, which is intended for playing robot soccer, and bring it into the opponent's goal. During the construction of the robot I have faced many complications and often seemingly insurmountable complications. But a lot of effort and hard work led me to the product I am proud of.

Making a robot has taught me many technical things, like programming in C Language, using drawing tools Sprint Layout, AltiumDesigner and EasyEDA, the use of CNC LPKF ProtoMat C30 router, along with other skills that will be useful in my further education. I have also learned the value of perseverance.

The robot I built will enter many competitions around the world. To build a robot was very demanding and I have also invested a lot of time in programming the robot. I often spent hours behind my computer and the robot, looking for solutions. I am very happy I was given an opportunity to create this kind of robot.

Key words: RoboCup Junior,soccer, Sprint Layout, AltiumDesigner, EasyEDA, CNC LPKF ProtoMat C30 router.

KAZALO

KAZALO SLIK:

KAZALO PRILOG

1 UVOD

Povod za izdelavo tega izdelka so bila RoboCup Junior tekmovanja, ki sem se jih začel udeleževati v drugem letniku. V 3. letniku sem prešel na kategorijo RoboCup Junior Soccer Light, kar v slovenščini pomeni robotski nogomet lahke kategorije. V ta namen sem s pomočjo Marcela Kaisesbergerja (lani je zaključil izobraževanje na ERŠ), ki je bil zaslužen za programski del, izdelal avtonomnega robota za igranje nogometa.

Takrat sem si zadal cilj, izdelati konkurenčnega robota ter dokazati, da lastna iznajdljivost ter kreativnost in skrbno načrtovane poteze pripomorejo k zmagam na mednarodnih ter tudi svetovnih tekmovanjih, kar je dokaz, da smo na pravi poti do uspeha. Največja vrednota, ki pa jo skozi proces načrtovanja in gradnje pridobimo, je znanje, kar nam v današnjem svetu zagotavlja službo ter poveča možnosti sodelovanja pri raznih drugih projektih.

Robot je sestavljen iz pogonskega dela, mehanskega dela, elektronike in programskega dela. Njegova naloga je poiskati žogico na igrišču, ki je namenjeno igranju robotskega nogometa. Ta oddaja infrardeči signal, ki ga robot zazna in žogico pelje ali izstreli na nasprotnikov gol. Za gradnjo robota sem veliko predhodnega znanja pridobil iz prejšnjih generacij dijakov, sam pa sem to podkrepil z lastnim. Robota sem kot celoto izboljšal. Po mojih načrtih je bilo narejeno ohišje robota, vsa elektronska vezja ter gonilniki za motorje, ki so se izkazali kot zelo kvalitetni in zanesljivi.

V tej maturitetni nalogi bom predstavil delovanje tega robota, opisal postopke izdelave ter prikazal vsa znanja ter izkušnje, ki sem jih ob izdelavi takšnega robota pridobil.

2 ROBOCUP JUNIOR TEKMOVANJE

RoboCup je mednarodna znanstvena organizacija, ustanovljena leta 1997 s ciljem, da spodbudijo različne generacije k uporabi najsodobnejših inteligentnih robotskih tehnologij. V sklopu te organizacije se organizirajo tekmovanja, v katere se vključujejo osnovnošolci, srednješolci, študentje in odrasli. Tekmovalci tekmujejo v različnih kategorijah: RoboCup Soccer, RoboCup Rescue, Robocup Industrial, RoboCup @Home in RoboCup Junior. Idejna zasnova organizacije je razvoj robotskih tehnologij, ki bodo do leta 2050 premagali ekipo aktualnih svetovnih nogometnih prvakov.

Del tekmovanja, imenovan Robo Cup Junior, je namenjen mladim, ki še niso dopolnili 19. leta starosti. RoboCup Junior je namenjen osnovnim (primary) in srednjim (secondary) šolam in ima izzive: nogomet (soccer), ples (on stage) in reševanje (rescue). Glavni namen tega tekmovanja je, da bi se mladi čim več naučili o uporabi robotske tehnologije ter o razvoju le-te. Za razliko od ostalih kategorij tekmovanja je princip robota iz leta v leto enak. Stopnjuje se le zahtevnost robota, da bi se približali zastavljenim ciljem, ki si jih je organizacija zadala do leta 2050. Mladi si pri nadgrajevanju pridobimo vsako leto nova znanja in neprecenljive izkušnje

Slika 1: RoboCup Junior

Vir: http://rcj.robocup.org/images/banner.jpg (marec, 2017)

2.1 ROBOCUP JUNIOR SOCCER TEKMOVANJE

V RoboCup Junior soccer (nogomet) tekmovanju ekipi popolnoma avtonomnih robotov tekmujeta med seboj. Na igrišču sta dva robota od vsake ekipe. RoboCup Junior soccer (nogomet) se odvija na posebej prirejenem igrišču, ki je v skladu z RoboCup pravili. Na Slika 2 so mere igrišča. Podlaga igrišča je tkanina podobna običajni preprogi. Pet t. i. nevtralnih točk je označenih s črno piko, ki so uporabljene za postavitev žoge, če se roboti nehajo odzivati ali če žoga zaide iz igrišča. Krog, ki je narisan na sredini igrišča, služi za postavitev robotov na začetku igre.

Slika 2: Slika igrišča

Vir: RoboCup Junior Soccer Rules 2017. Dostopno na: http://rcj.robocup.org/soccer.html (marec, 2017)

Roboti z ustreznimi senzorji iščejo po igrišču žogo, ki oddaja IR žarke. Robot mora žogo pripeljati do gola in zadeti gol, to je glavni cilj igre. Na tekmovanju sta dve kategoriji RoboCup Junior soccer – open oz. težka kategorija in lightweight oz. lahka kategorija. Teža, velikost in napetost baterije robota mora ustrezati RoboCup pravilom. Napetost baterije, velikost in teža robota se preveri pred tekmovanjem. Na Slika 3 je primer tabele, ki jo sodniki izpolnijo za vsakega robota. Konkreten primer je iz svetovnega prvenstva v Leipzigu. Cilj je, da robot med igro deluje povsem avtonomno. Pri končnem izidu ni vse odvisno od pravilnega delovanja robota, ampak tudi od pravil igre. Pravila določajo potek igre (npr. kaj se zgodi v primeru, da robot zadane gol, robot zapelje iz igrišča, pride do okvare, itd). Tekmo sestavljata dva polčasa po 10 minut, vmes pa so 5 minutni premori. Ob igrišču je sodnik, ki skrbi za pravilen potek igre. Ekipa ima prav tako kapetana in pomočnika, ki sta ob igrišču.

TEAM/ROBOTS INSPECTION SHEET

DATE	[ ]30th June, Thu [ ]1st July, Fri [ ]2nd July, Sat [ ]3rd July, Sun		
ROUND			
CATEGORY	[ ]Lightweight Primary [ ]Lightweight Secondary [ ]Open		
TEAM NAME		TEAM CODE	

Basic: !! Before EVERY game, REFEREE check AGAIN !!

1.SIZE (spread all moving part then ≤22.0cm, HANDLE is not included)			[ ]OK
2.WEIGHT (including battery ≤2.4kg Open,≤1.1 Light) [g] [g]			[ ]OK
3.BALL CAPTURE ZONE (<3cm)	[ ]OK	4.Top Marker	[ ]OK
5.BATTERY VOLTAGE (≤15V O, ≤12V L) [V] [V]	[ ]P ower pump used		[ ]OK
6.KICKER POWER	[ ] Electric [ ]Air Power[ ]		[ ]OK
7.EMITTING LIGHT, BLUE and YELLOW colored parts (or other equipments disturbing any sensors)	OK [ ] COMMENTS		
8.DANGER EQUIPMENT (damage the field, ball, other robots and referees!)	OK [ ] COMMENTS		

Need Special Interview/TC discussion

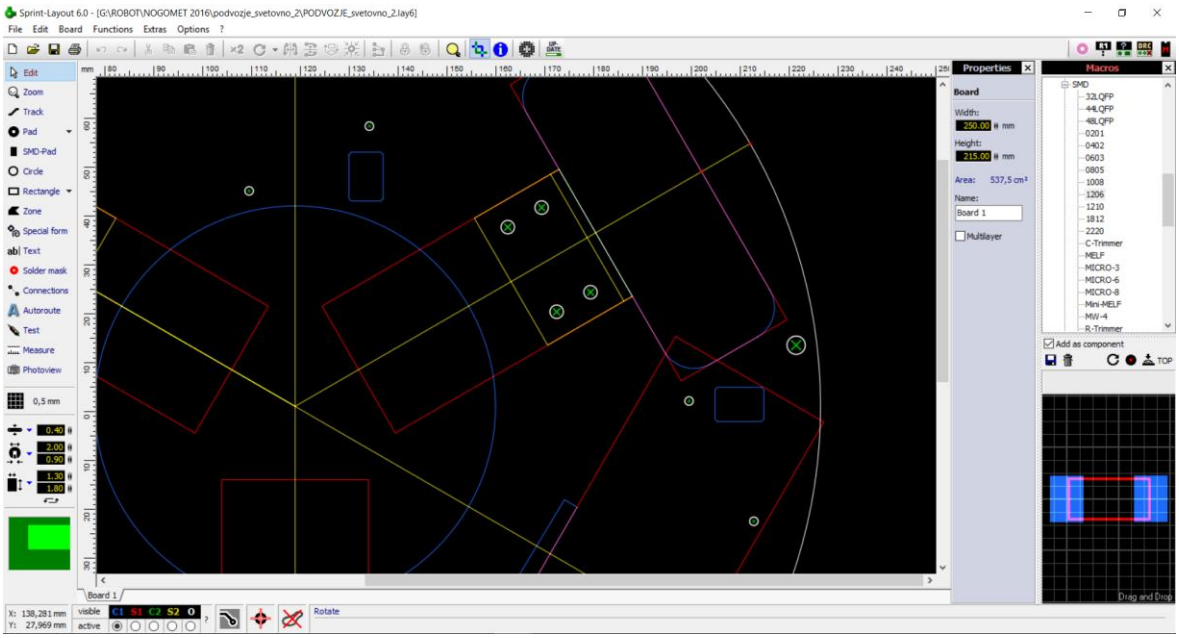
9.Check if you think need special interview [ ] Sensor issues (i.e. IR distance sensor) [ ] Battery Voltage issues [ ] Kicker Power issues [ ] Construction issues (i.e. danger equipments, commercial kit etc)	10.COMMENTS TO TC/Interviewers [ ] Special Interview [ ] TC discus sion sign.....

Slika 3: Razpredelnica za preverjanje ustrenosti robotov

Vir: RoboCup Junior Soccer Rules 2017. Dostopno na: http://rcj.robocup.org/soccer.html (marec, 2017)

3 NAČRTOVANJE TER KONSTRUKCIJA ROBOTA

Mehansko konstrukcijo robota sem zasnoval in zgradil sam. Celotno konstrukcijo sem narisal z orodjem Sprint Layout, kot prikazuje Slika 4. Skozi celoten proces risanja sem se ves čas držal pravil, ki so objavljena na uradni strani Roboup Junior ter temu primerno tudi zasnoval robota.

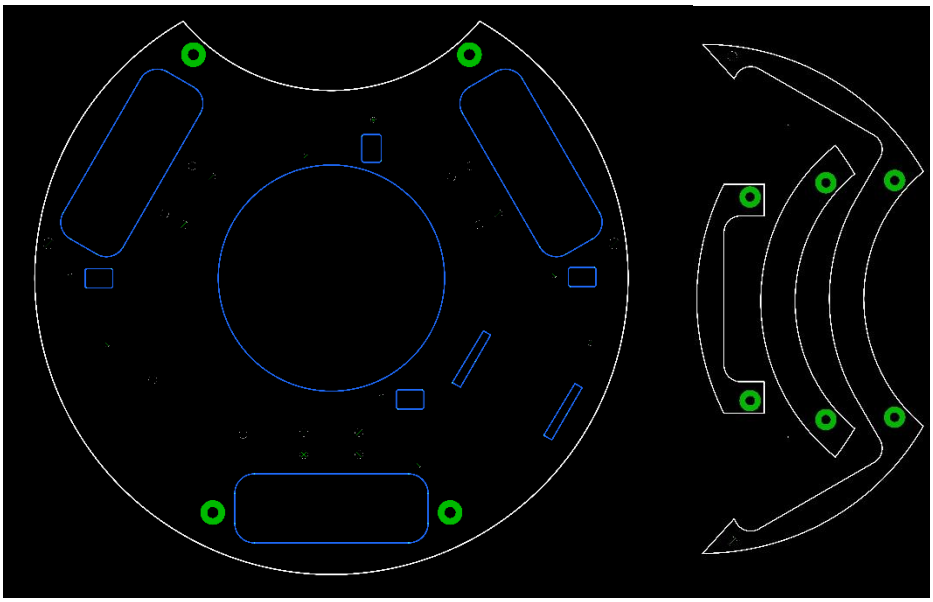


Slika 4: Orodje Sprint Layout

Vir: Lasten vir

3.1 OHIŠJE

Ohišje sem narisal z orodjem Sprint Layout in je sestavljeno iz dveh posebej zasnovanih plošč ter podpornikov. Zgornja služi za električni povezavi vseh komponent na robotu in ima premer 200 mm. Na spodnji plošči so ustrezni izrezi za namestitev baterije, barvne senzorje, nosilce, odprtine za kolesa ter izrez za pridržanje in zajem žogice. Premer te plošče je 215 mm. Robovi teh plošč so tudi skrajni robovi celotnega robota, tako da robot ustreza dovoljenemu premeru 220 mm. Spodnja plošča z vsemi podporniki je prikazana na Slika 5. Za povezovanje plošč, eno nad drugo, smo uporabili štiri navojne palice. Z njimi lahko prilagajamo razmak med spodnjo in zgornjo ploščo glede na situacijo. Po končanem načrtu so bile plošče izrezani s CNC rezkalnim strojem iz vitro plošče.

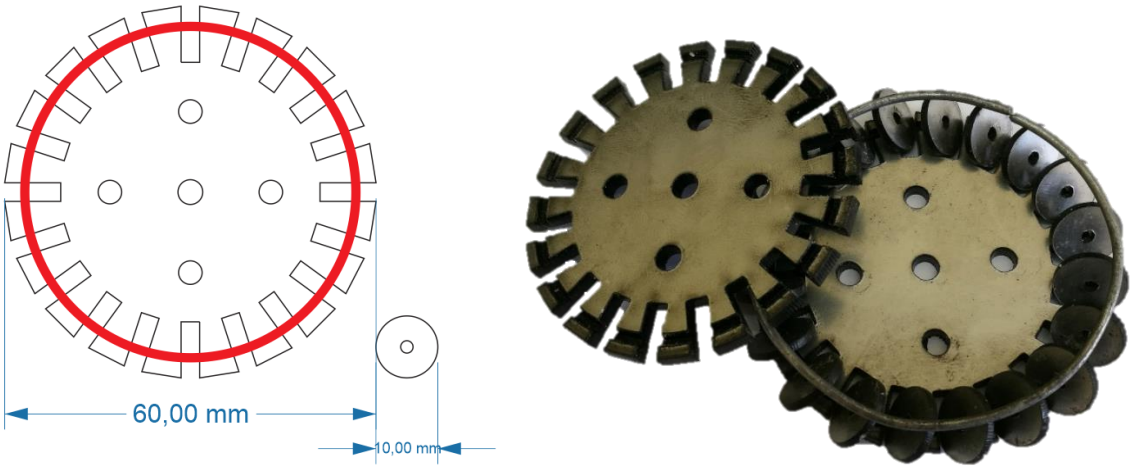


Slika 5: Izrez spodnje plošče in podpornikov

Vir: Lasten vir

3.2 VSESTRANSKO KOLO

Sam sem zasnoval vsesmerno kolo oz. angleško omniwheel po vzoru koles drugih ekip. Posebnost pri tem kolesu je, da je oprijem samo vzdolž kolesa, prečno pa nima trenja, ker je po robu kolesa prečno pritrjenih 20 manjših vrtljivih koles, ki se pri bočnem premikanju kolesa vrtijo. Celotno kolo sestavljata dve večji polovici, ki se skupaj privijeta, vmes pa se stisnejo manjša kolesa, skozi katere poteka žica in poseben nosilec, ki služi za pritrditev kolesa na os. Polovice koles so bile načrtovane v programu Corel in nato lasersko izrezane vključno s prirobnicami. Zagotavljati morajo odličen oprijem na podlago igralne površine in neovirano gibanje robota v vse smeri. Vsesmerna kolesa se razlikujejo od navadnih koles. Njihova prednost je ta, da se lahko gibljejo v vsakem trenutku v želeno smer.

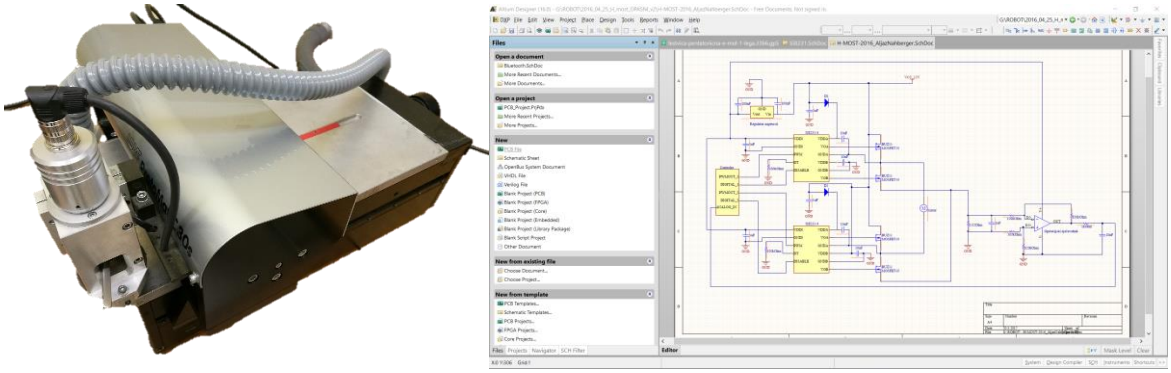


Slika 6: Na levi izris v Corelu, na desni pa končni izdelek

Vir: Lasten vir

4 ELEKTRONSKE KOMPONENTE

Vse elektronske sheme sem narisal v programih EasyEDA ter Altium designer. Elektronska vezja pa sem načrtoval s programom SprintLayout 6. Vezja sem potem izdelal s šolskim CNC LPKF ProtoMat C30 rezkalnikom.

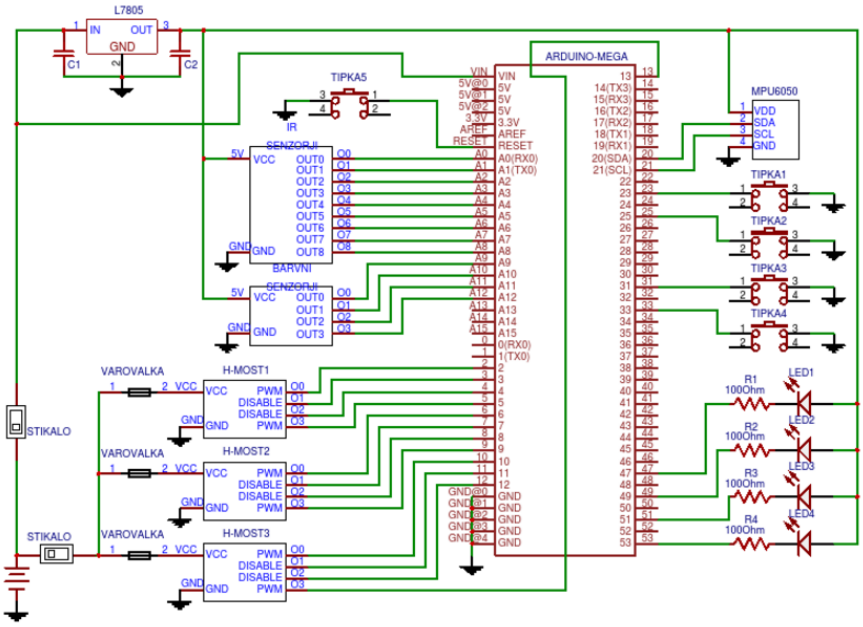


Slika 7: Slika CNC rezkalnika ter orodja Altium Designer

Vir: Lasten vir

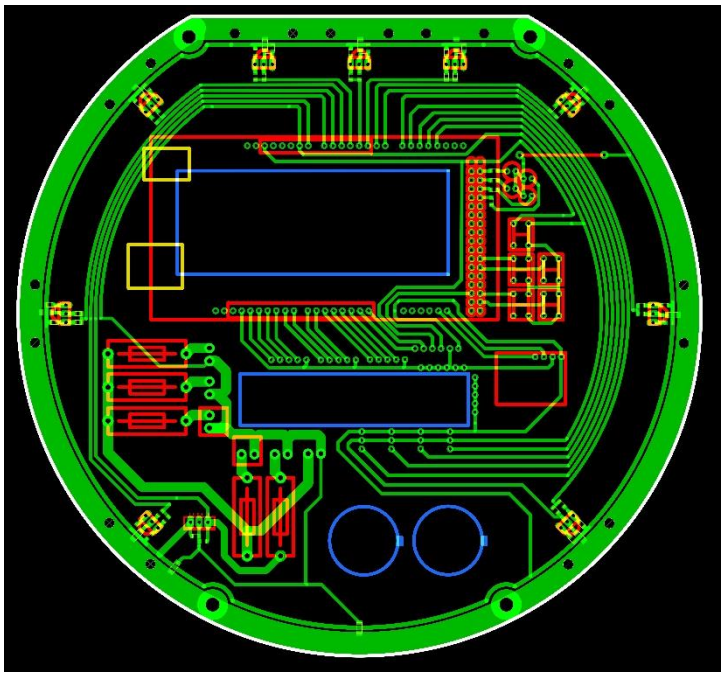
4.1 OSREDNJA PLOŠČA

Je največje vezje na robotu in povezuje vse aktuatorje ter senzorje z Arduino Mega 2560 razvojno ploščo. Na plošči je tudi glavno stikalo, regulator napetosti, tipke, priključki za senzorje, H-mostiče, žiroskop in priključek za baterijo. Električna shema te plošče je na Sliki 8, načrt vezja za rezkanje pa je na Sliki 9. Pri tem je najpomembneje, da sem se naučil, kako izdelati tiskanino dovolj robustno, da med igro ne pride do prekinitev oziroma razpok na tiskanem vezju. Največje težave so bile pri IR-senzorjih, saj so na robu plošč izpostavljeni neposrednim udarcem nasprotnih robotov. Težavo smo odpravili s tem, da smo jih zamaknili navznoter in konkretno odebelili zunanje vezi tiskanega vezja.



Slika 8: Električna shema vezja glavne plošče

Vir: Lasten vir

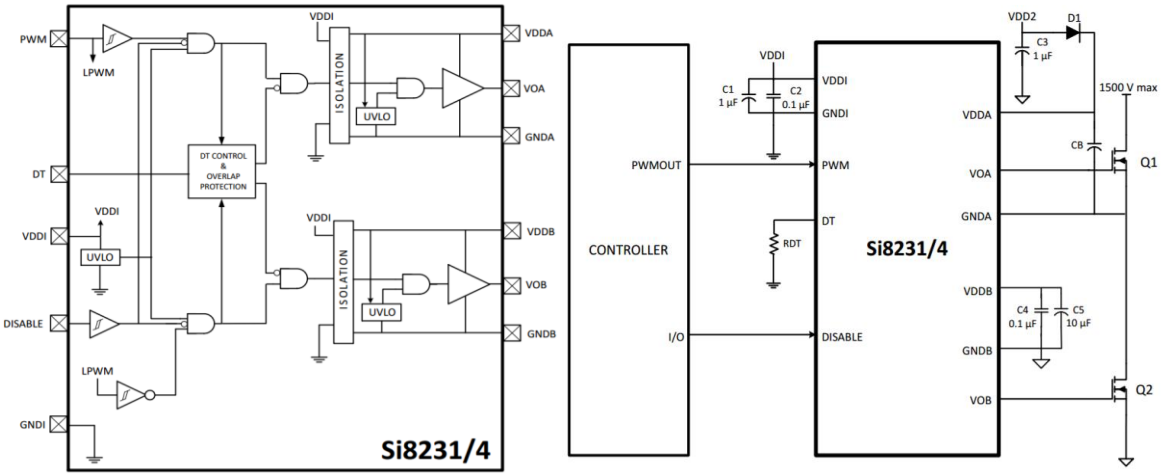


Slika 9: Zgornja plošča robota izdelana v Sprint Layout-u

Vir: Lasten vir

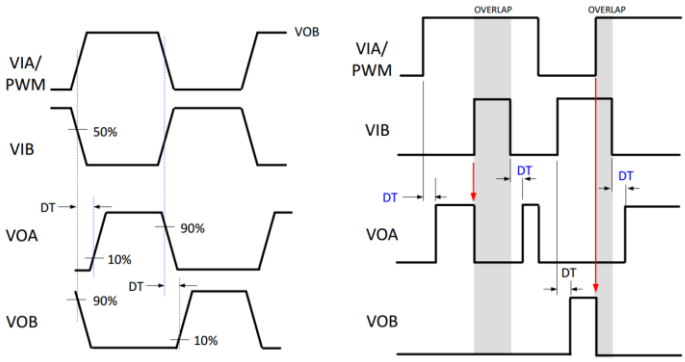
4.2 GONILNIK ZA MOTORJE (H – MOST)

V H-mostičnem vezju je pri krmiljenju posameznih tranzistorjev potrebno upoštevati zakasnitve vklopov in izklopov. V primeru, da ne krmilimo pravilno, se ob preklopih gornjega in spodnjega tranzistorja v posamezni polmostični vezavi zgodijo velike tokovne konice, ki ustvarijo velike toplotne izgube in posledično tudi uničenje stikalnih elementov. Za pravilno krmiljenje tranzistorjev sem se odločili poiskati prožilna integrirana vezja, ki imajo vgrajene zakasnilne čase ali pa imajo možnost nastavitve le-teh. Odločil sem se za uporabo Si8230/4 integriranega vezja v SOIC-16 WIDE ohišju, katerega proizvajalec je podjetje Silicon Labs. Z enim čipom lahko vklapljam polmostično vezje (eno vejo, torej en zgornji in en spodnji tranzistor). Pri izdelavi polnomostičnega vezja sem si pomagal s polmostično vezavo, ki je bila podana v tehničnih podatkih proizvajalca. Električna shema polnomostičnega vezja z vsemi komponentami je na Slika 11.



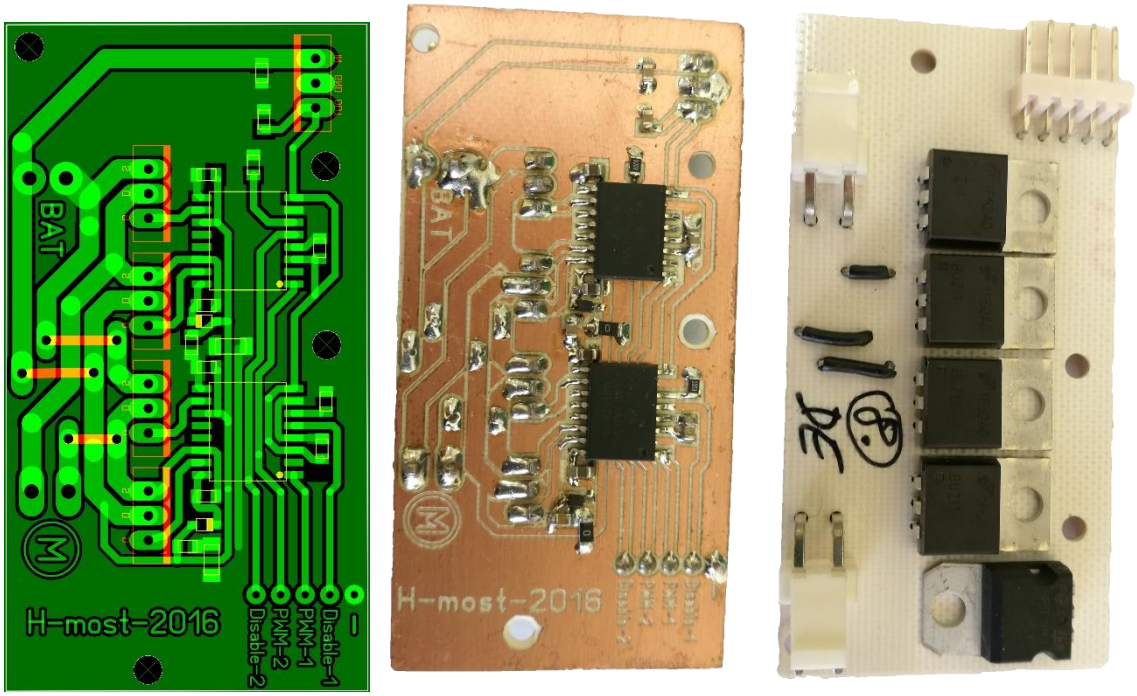
Slika 8: Blokovni diagram Si8231/4 ter polmostična vezava

Vir: https://www.silabs.com/documents/public/data-sheets/Si823x.pdf (marec, 2017)



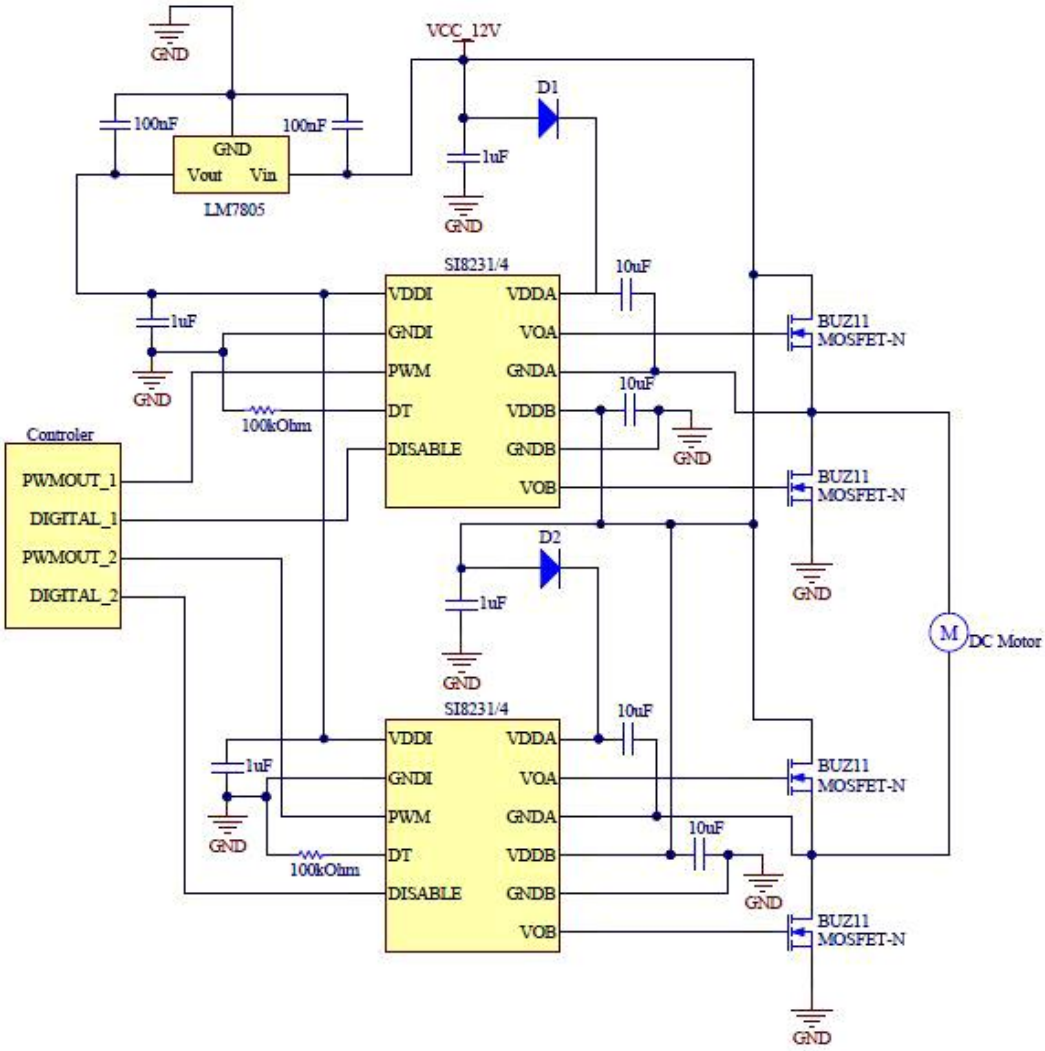
Slika 9: Prikaz mrtvih časov pri preklopih

Vir: https://www.silabs.com/documents/public/data-sheets/Si823x.pdf (marec, 2017)



Slika 10: Električno vezje narejeno v Sprint Layout-u ter končni izdelek

Vir: Lasten vir



Slika 11: Električna shema h-mosta narisana v Altium Designer-ju

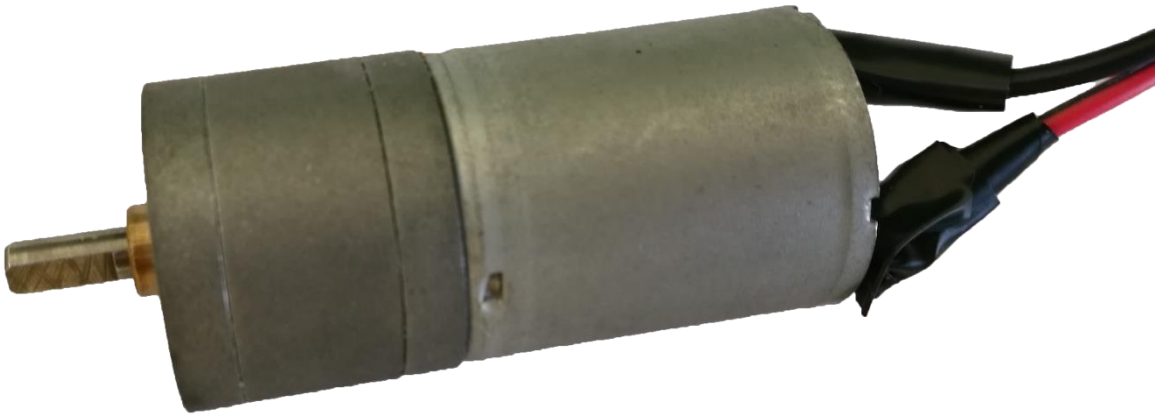
Vir: Lasten vir

4.3 ELEKTROMOTORJI

Pogon robota sestavljajo tri motorna gonila, ki so sestavljena iz 12 V enosmernega motorja ter reduktorja s prestavnim razmerjem 20.4:1. Nameščena so na spodnji plošči robota. Od središča plošče so enako oddaljena in med seboj zamaknjena za 120°. Motorno gonilo je valjaste oblike in ima premer 25 mm. Izhodna gred D-oblike ima premer 4 mm. Teža robota v lahki kategoriji ne sme presegati 1,1 kg, zaradi tega sem pri izbiri motorjev bil še posebej pazljiv pri teži. Vsi motorji skupaj ne presegajo teže 270 g. Te motorje sem izbral zaradi nizke teže, visoke moči in preprostega krmiljenja smeri.

Specifikacije motorja:

- nazivna napetost: 12 V;
- tok v prostem teku: 300 mA;
- tok pri polni obremenjenosti: 5,6 A;
- število vrtljajev na minuto pri nazivni napetosti: 500;
- navor: 6 kg-cm;
- teža: 85 g.

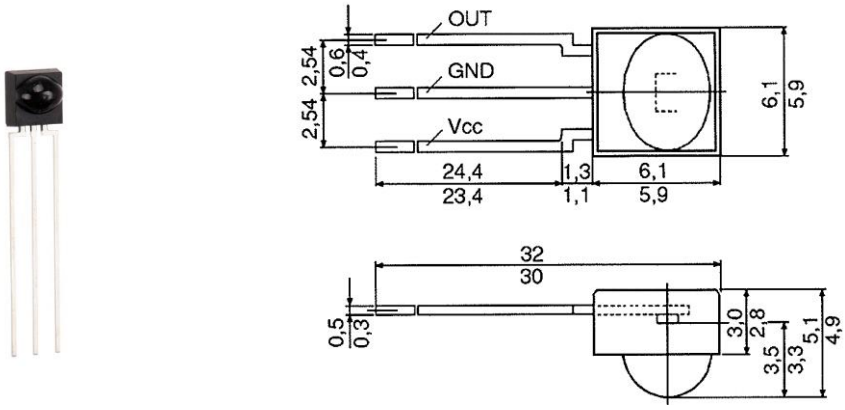


Slika 12: Uporabljen motor

Vir: Lasten vir

4.4 SENZOR ZA ZAZNAVANJE IR ŽOGICE

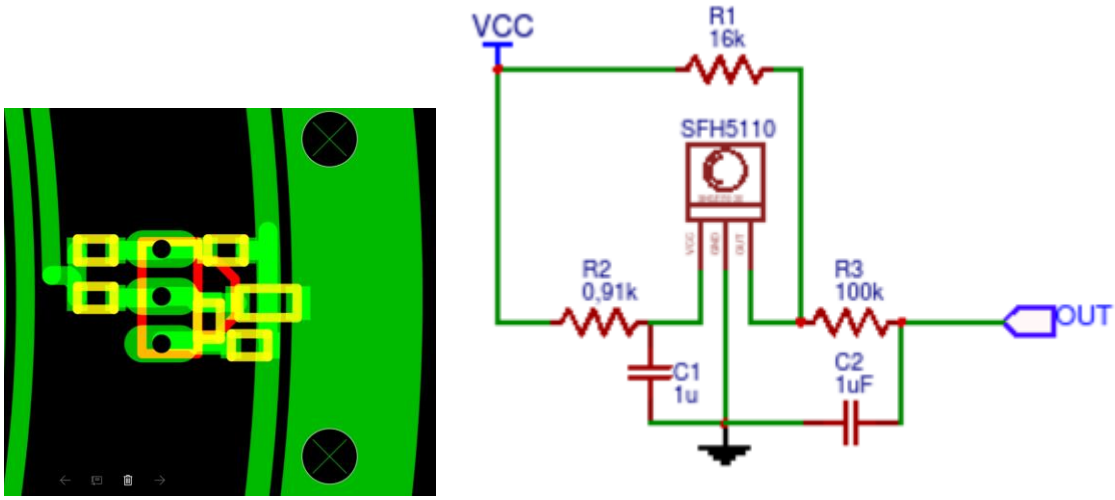
Najpomembnejša naloga robota za igranje nogometa je detekcija elektronske žoge, ki oddaja svetlobo v infrardečem spektru s frekvenco 1200 Hz. Uporabil sem sprejemno IR-diodo SFH5110-38 (Slika 13), saj se je ta v praksi izkazala kot najbolj zanesljiva ter nam je prikazala dokaj linearne vrednosti glede na oddaljenost žogice in zagotavlja detekcijo žoge pri razdalji več metrov.



Slika 13: Sprejemna IR SFH5110.38 dioda

Vir: http://cdn.pollin.de/article/xtrabig/X120821.1.JPG (marec, 2017), https://www.soselectronic.com/productdata/72/90/7290/Zbo_7290_02.jpg (marec, 2017)

Pri risanju električne sheme sem si pomagal s priporočeno vezavo s strani proizvajalca ter jo nekoliko dodelal. Na izhode sprejemnikov sem vezal RC-filtre, da sem stabiliziral napetost ter s tem zagotovil zanesljive analogne signale. Napetost na izhodu je analogne oblike in pada, s tem ko se žoga približuje senzorju. Na robotu je takšnih senzorjev devet, tako da robot vsepovsod okrog sebe vidi žogico. Smer žogice v programu določim tako, da preberem vse analogne vhode, na katerih so IR senzorji. Iz tega dobim niz podatkov, pri katerem poiščem najmanjšo vrednost. Ko program najde najmanjšo vrednost in ugotovi, kateremu senzorju je žogica najbližja, mu s tem določim, v katero smer se bo robot premaknil. Določi se tudi meja zaznave, da se lahko ugotovi, če robot žoge ne vidi.



Slika 14: Električna vezava ter shema IR senzorja

Vir: Lasten vir

4.5 BARVNI SENZOR

Če zapelje robot izven črt, mora zapustiti igro. Robot zazna črto s senzorji za črte. Program je napisan tako, da ne bo prevozil črte. Robot ima nameščene takšne štiri senzorje na dnu (spredaj, zadaj, levo in desno). Senzor ima nameščeni dve LED diodi, ki osvetljujeta podlago in fotoupor, ki zaznava svetilnost. Črte so bele barve, kar senzor zazna svetleje kot podlago igrišča, ki je zelene barve. Napetost pri izhodu senzorja je analogne oblike.

Predupor za LED diodo sem izračunal z enačbo:

$$R = \frac{U}{I}$$

kjer je:

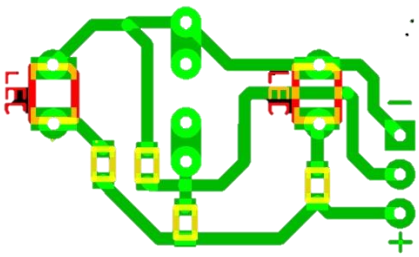
- R (Ω) - upornost predupora za LED diodo.
- U (V) - padec napetosti čez predupor.
- I (A) - tok skozi predupor in LED diodo.

Iz proizvajalčevih specifikacij za diodo sem vzel podatke za nazivni tok in napetost ter jih izstavil v enačbo in izračunal upornost predupora:

$$R = \frac{U}{I} = \frac{3\ V}{0{,}02\ A} = 150\ \Omega$$

Slika 15: Foto upor

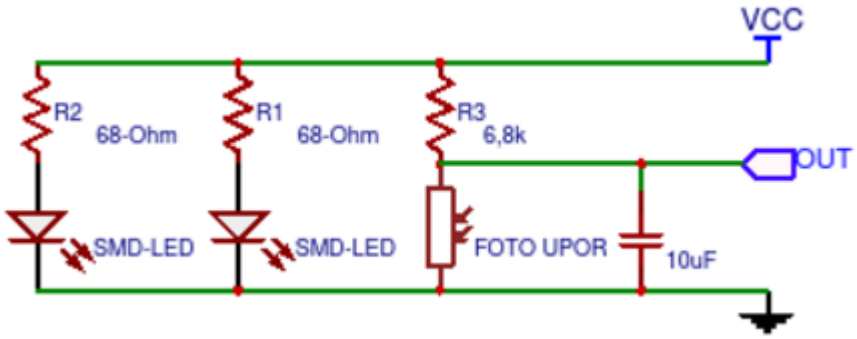
Vir: https://image.jimcdn.com/app/cms/image/transf/none/path/s56188d12bcb62751/image/id9d1f8cbb66a5ecf/version/1480783655/image.jpg (marec, 2017)



Slika 16: Električno vezje barvanega senzorja

Vir: Lasten vir

Senzor na izhodu daje analogno napetost, zato sem najprej to vrednost prebral in nato z izbranimi vrednostmi v programu določil mejo med zeleno in belo barvo. Program ob zagonu robota prebere vrednosti na vseh štirih analognih vhodih, kjer so priklopljeni senzorji črte. Robot si jih zapiše ter si tako določi referenco, pri kateri vrednosti ni črte oz. je senzor nad zeleno podlago. Vsi senzorji na robotu morajo biti ob vklopu nad zeleno površino igrišča. Belo črto zazna, če preveč odstopa vrednost na analognem vhodu od referenčne.

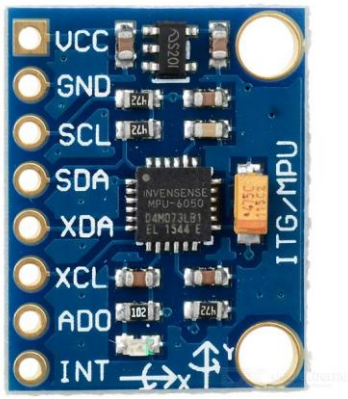


Slika 17: Električna shema barvnega senzorja

Vir: Lasten vir

4.7 ŽIROSKOP

Za zaznavanje rotacije sem uporabil žiroskop in pospeškomer MPU–6050 (Slika 18). Senzor meri rotiranje in pospeške po vseh treh oseh (X, Y in Z). S tem senzorjem se lahko zaznava samo rotacija v prostoru, ne moremo pa določiti položaja v koordinatnem sistemu. Z zaznavanjem rotacije si robot določi, v kateri smeri je nasprotnikov gol. Komunikacija s senzorjem poteka preko I2C vodila, in sicer ima registre, ki vsebujejo podatke o trenutnem premiku oz. pospešku na vseh treh oseh. Za delovanje senzorja sem uporabil Arduino kodo iz proizvajalčeve spletne strani. Kodo sem spremenil tako, da žiroskop bere samo informacijo o rotiranju okrog Z osi. Problem je nastal, ko je bilo potrebno podatek iz senzorja neprestano brati, ker se osveži vsako milisekundo in ne kaže odmika od neke želene začetne pozicije. Program na krmilniku prišteva oz. odšteva te spremembe rotacije od neke začetne pozicije. Program s časoma postane nenatančen, kar se na roboti vidi z neporavnanostjo med igro.



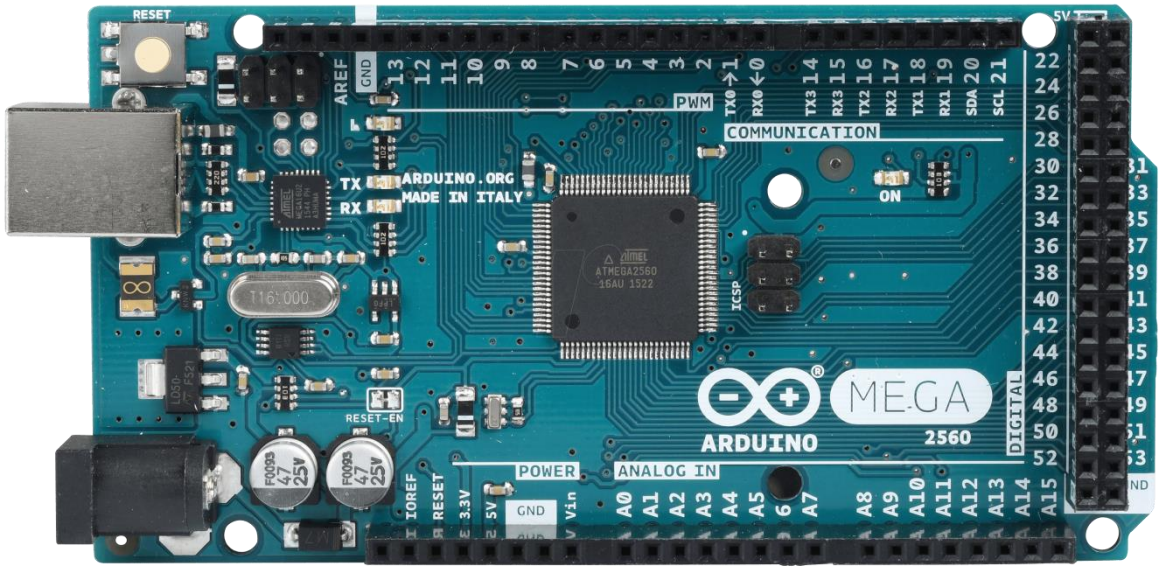
Slika 18: Žiroskop MPU6050

Vir:

https://www.google.si/search?q=mpu6050&espv=2&source=lnms&tbm=isch&sa=X&ved=0ahUKEwi6wrvfieHSAhUiAZoKHUNfBcIQ_AUICCgB&biw=1536&bih=735#imgrc=qziqgqM31Z92EM (marec, 2017)

4.8 ARDUINO MEGA RAZVOJNAPLOŠČICA

Za nalogo sem uporabljal razvojno platformo Arduino MEGA 2560, ki je namenjena učenju programiranja. Arduino je odprtokodna elektronska platforma, ki jo proizvajajo v Italiji. Arduino razvojna plošča je uporabna pri eksperimentiranju in gradnji raznih projektov z vključenim elektronskim krmilnikom. Uporabil sem ga, ker ponuja veliko število vhodov in izhodov, spremjemljiva pa je tudi cena. Na ploščici je štiriinpetdeset digitalnih vhodov in izhodov ter šestnajst analognih vhodov. Vgrajen mikrokrmilnik pa je 8 bitni Atmel ATmega2560. Procesor je 16 MHz, kar pomeni, da je sposoben opraviti 16.000.000 ukazov v eni sekundi. Na ploščico Arduino lahko priklopimo različne senzorje in aktuatorje, s programsko opremo Arduino IDE pa imamo možnost, da mikrokrmilnik na ploščici sprogramiramo, da se naprava obnaša po naših željah. Programski jezik je podoben C-ju oz. C++. Ko program napišemo, ga preko USB vmesnika prenesemo na ploščico.



Slika 19: Arduino MEGA 2560 razvojna platforma

Vir: https://cdn-reichelt.de/bilder/web/xxl_ws/B300/ARDUINO_MEGA_A03.png (marec, 2017)

4.9 NAPAJANJE

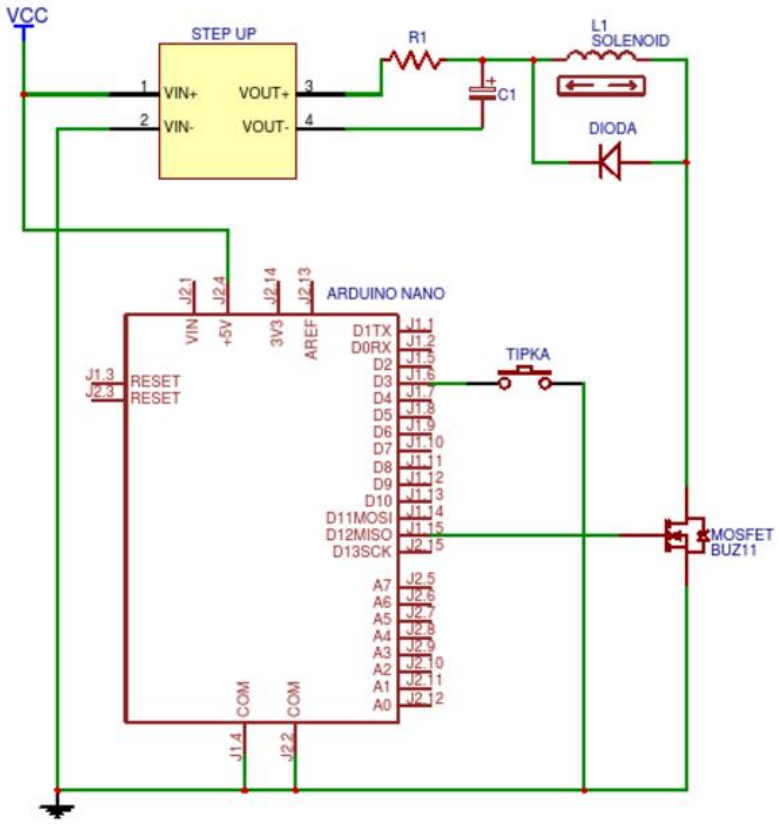
Pravila za to kategorijo dovoljujejo baterijo maksimalne napetosti 12 V, zato sem izbral tricelično Lipo baterijo velikosti 1,3 Ah. Na vezju imamo dva napetostna nivoja. Za napajanje gonilnikov motorja ter razvojno ploščico Arduino Mega sem uporabil 12 V napetost. Za barvne in IR senzorje, LED diode ter žiroskop pa sem uporabil 5 V napetost, ki sem jo dobil s pomočjo regulatorja napetosti LM7805.

Slika 20: Uporabljena baterija na robotu ter regulator napetosti LM7805

Vir: http://www.roboinventors.com/wp-content/uploads/2016/12/L7805-LM7805-7805-Voltage-Regulator-300x300.jpg (marec, 2017)
http://www.hangarone.co.nz/images/GA3S1300-E.jpg (marec, 2017)

4.10 MEHANIZEM ZA UDARJANJE ŽOGE

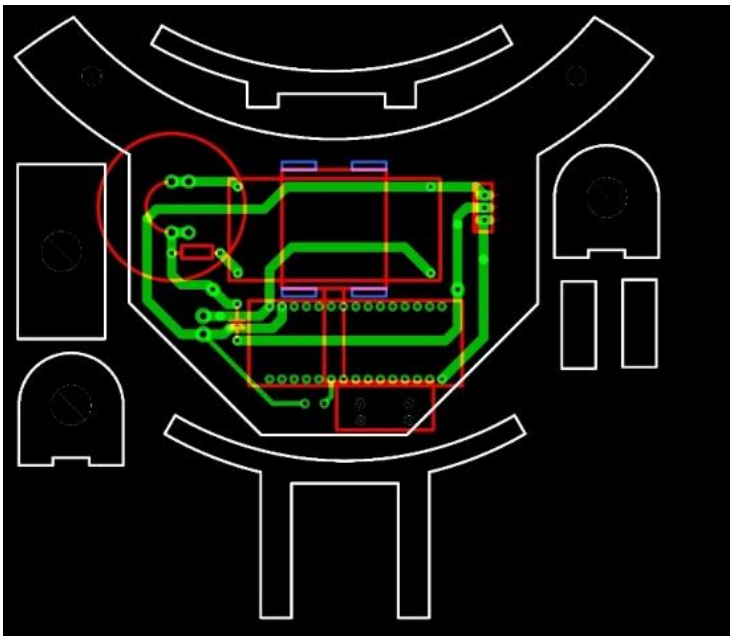
Zaradi konkurence na tekmovanjih sem se odločil narediti mehanizem za udarjanje žogice. Pri načrtovanju le-tega sem se zgledoval po teh, ki so jih imele druge ekipe na svojih robotih. Večina se je tega lotila s pomočjo elektromagneta z jekleno osjo, zato sem tudi jaz ubral to pot. Za vezje sem potreboval elektromagnet z jekleno osjo, N-kanalni mosfet, elektrolitski kondenzator, diodo, razvojno ploščico Arduino Nano, tipko, DC-DC pretvornik napetosti ter 12 V vir napajanja. Električna shema tega mehanizma je na Slika 21, vezje s konstrukcijo pa na Slika 22.



Slika 21: Električna shema mehanizma za udarjanje žoge

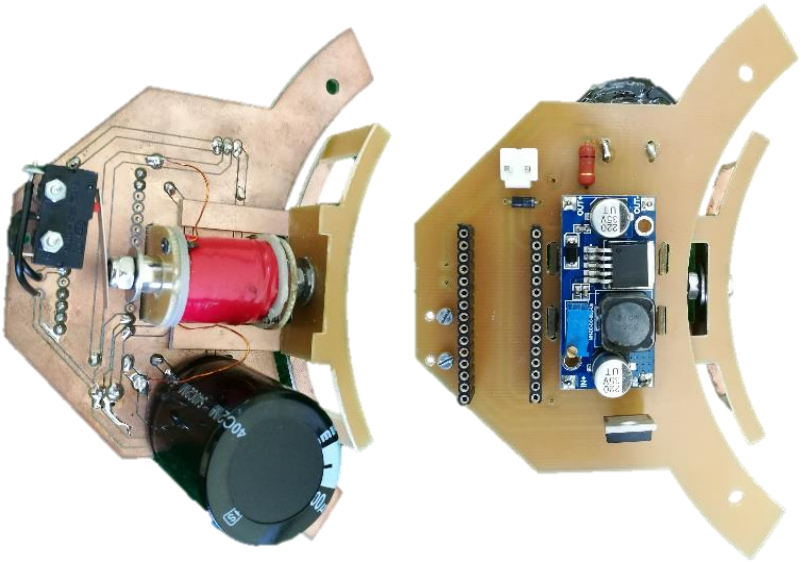
Vir: Lasten vir

Na končanem robotu tega mehanizma žal nisem uporabil. Bil je pretežak ter ni dosegel željene moči udarca. Tehtal je 134 g, kar je bilo kljub manjši bateriji 20 g preveč.



Slika 22: Električno vezje mehanizma za udarjanje žogice s konstrukcijo

Vir: Lasten vir



Slika 23: Mehanizem za udarjanje žogice

Vir: Lasten vir

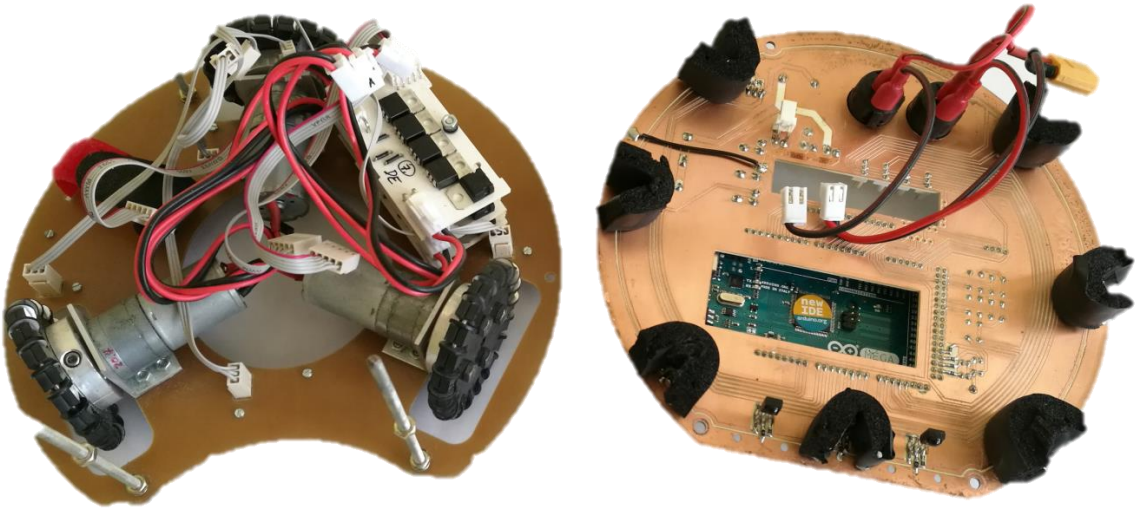
5 OPIS POSTOPKA IZDELAVE ROBOTA

Najprej sem s CNC strojem izrezal vse elemente, ki so potrebni za izdelavo konstrukcije robota.

Slika 24: Izrezani deli s pomočjo CNC rezkalnika

Vir: Lasten vir

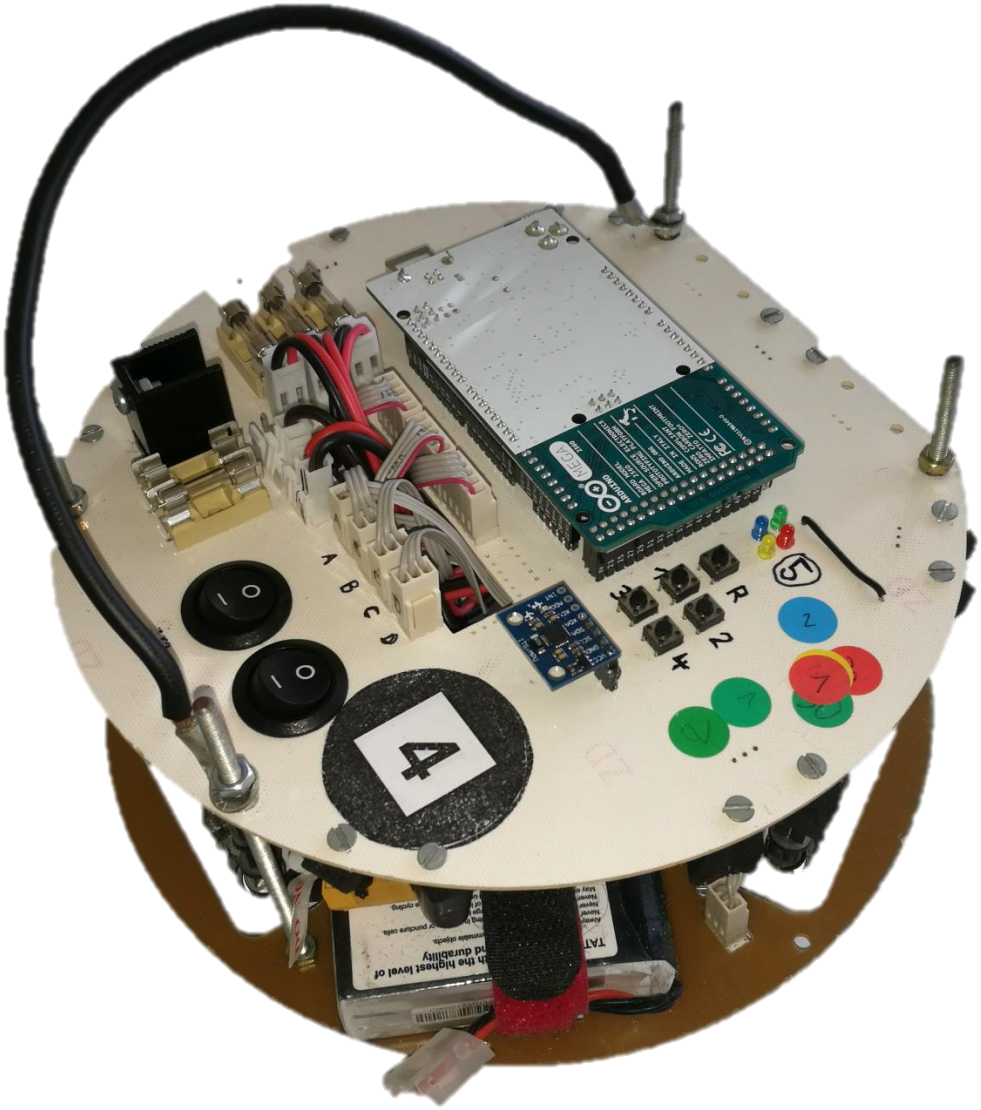
Nato sem spajkal zgornjo ploščo, barvne senzorje ter h-mostiče. Kasneje sem sestavil vsesmerna kolesa ter jih namestil na spodnjo ploščo, na katero sem tudi pritrdil barvne senzorje, baterijo ter h-mostiče. Naredil sem tudi konektorje za povezavo barvnih senzorjev z zgornjo ploščo ter konektorje za povezavo motorjev s h-mostiči ter napajanjem.



Slika 25: Dokončana zgornja in spodnja plošča

Vir: Lasten vir

Razrezal sem M3 navojno palico na štiri kose ter z njimi med seboj povezal zgornjo in spodnjo ploščo. Senzorje in h-mostiče sem preko vodil povezal na glavno ploščo. Pritrdil sem Arduino Mega 2560 razvojno ploščico ter priklopil baterijo.



Slika 26: Končni izdelek

Vir: Lasten vir

Robot je tako bil sestavljen. Težak je 1090 g, torej ustreza RoboCup pravilom za lahko oz. *light* kategorijo po teži in velikosti.

6 ZAKLJUČEK

S končanim robotom smo osvojili prvo mesto na RoboCup tekmovanju na Portugalskem, maja 2016 ter smo aktualni državni prvaki. Na svetovnem prvenstvu v Leipzigu pa smo uspeli premagati ekipe ZDA, Kanade, Južne Koreje ter Brazilije. S tem rezultatom sem dokazal konkurenčnost robota. Kljub dobrim rezultatom ima robot ogromno možnosti za izboljšavo. Največ bi pridobil, če bi uspel prepoznati pozicijo robota na igrišču. Za naslednje svetovno tekmovanje, ki bo potekalo na Japonskem, sem se odločil sestaviti novo ekipo, saj sem ugotovil, da delo poteka hitreje, če se porazdeli na več ljudi in sestaviti dva nova robota, na katerih bosta mehanizma za udarjanje in pridržanje žogice.

7 VIRI IN LITERATURA

1. RoboCupJunior Soccer Technical Committee. RoboCupJunior Soccer Rules 2015. RoboCup (online). (Citirano 10. 3. 2017). Dostopno na internetnem naslovu: http://rcj.robocup.org/rcj2017/soccer_2017.pdf
2. Arduino SA. Arduino Mega 2560. Arduino (online). (Citirano 9. 3. 2016). http://arduino.cc/en/Main/ArduinoBoardMega2560
3. Jeff Rowberg. MPU6050 (online). (Citirano 21. 2. 2015). Dostopno na naslovu: https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
4. Arduino SA. MPU6050. Arduino (online). (Citirano 1. 3. 2017). Dostopno na naslovu: http://playground.arduino.cc/Main/MPU-6050#short
5. Silicon Labs. Si823x Data Sheet. Si8231/4 (online). (Citirano 5. 3. 2017). Dostopno na naslovu: https://www.silabs.com/documents/public/data-sheets/Si823x.pdf
6. Polulu. 20.4:1 Metal Gearmotor 25Dx50L mm HP 12V (online). (Citirano 4. 3. 2017). Dostopno na naslovu: https://www.pololu.com/product/3203

Priloga 1: Programska koda

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include <Wire.h> //I2C Arduino Library
#include "MPU6050.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
///////////////MOTORJI//////////////////////////
#define mdfd 2 // motor desni forward disable
#define mdf 4
#define mdb 3
#define mdbd 5
#define mlfd 9 // motor levi forward disable
#define mlb 8
#define mlf 7
#define mlbd 6
#define mzbd 10 // motor zadni back disable
#define mzb 11
#define mzf 12
#define mzfd 13
/////////////////LED//////////////////////////////
#define led1 53
#define led2 47
#define led3 51
#define led4 49


//////////////////////IR//////////////////////////
#define irZD A8 // ZADAJ DESNO
#define irL A1 // LEVO
#define irLN A2 // LEVO NAPRE
#define irLS A3 // LEVO SREDINA
#define irS A4 // NASREDI
#define irZL A0 // ZADAJ LEVO
#define irDN A6 // DESNO NAPRE
#define irDS A5 // DESNO SREDINA
#define irD A7 // DESNO
////////////////////TIPKE/////////////////////
#define t1 33 // zgori leva
#define t2 31 // zgori desna
#define t3 25 // spodi leva
#define t4 23 // spodi desna
///////////////SENZORJI ZA ČRTO/////////////////
#define scL A11 // LEVO
#define scP A10 // SPREDNJI
#define scD A9 // DESNI
#define scZ A12 // ZADNJI
////////////ŽIROSKOP ///////////////////////////
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL

bool blinkState = false;
bool dmpReady = false;
uint8_t mpuIntStatus;
```

```
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];

Quaternion q;          // [w, x, y, z]        quaternion container
VectorInt16 aa;        // [x, y, z]           accel sensor measurements
VectorInt16 aaReal;    // [x, y, z]           gravity-free accel sensor measurements
VectorInt16 aaWorld;   // [x, y, z]           world-frame accel sensor measurements
VectorFloat gravity;   // [x, y, z]           gravity vector
float euler[3];        // [psi, theta, phi]    Euler angle container
float ypr[3];          // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

uint8_t teapotPacket[14] = { '$', 0x02, 0, 0, 0, 0, 0, 0, 0, 0, 0x00, 0x00, '\r', '\n' };

// ================================================================
// ===                MAIN PROGRAM LOOP                 ===
// ================================================================
volatile bool mpuInterrupt = true;     // indicates whether MPU interrupt pin has gone high
int alpha2, korekcija;

int kot() {
  mpuInterrupt = true;
  if (!dmpReady) return 0;

  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();

  fifoCount = mpu.getFIFOCount();

  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {

    mpu.resetFIFO();
  } else if (mpuIntStatus & 0x02) {

    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    fifoCount -= packetSize;

#ifdef OUTPUT_READABLE_YAWPITCHROLL
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
#endif
  }

  alpha2 = ypr[0] * 180 / M_PI;

  if (alpha2 <= (-180 + korekcija)) {
    alpha2 = alpha2 + (360 - korekcija);
  }
  else {
    alpha2 = alpha2 - korekcija;
  }
  return alpha2;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////
int vrednostL;
int vrednostP;
int vrednostD;
int vrednostZ;
int odstopanjebela = 100; // analogna vrednost, razlika med zeleno in belo barvo
int odstopanjecrna = 100; // analogna vrednost, razlika med zeleno in črno barvo
////////////////////////////////////////////////////////////////////////////////////////////
// napoved podprogramov
void naprej();
void nazaj();
void levo();
void desno();
void pocivaj();
void obrni_levo();
void obrni_desno();
void crta();
void kalibracija();
void napadalec();
void spilajnapadalec();
void spilajstoper();
void crtastoper();
////////////////////////////////////////////////////////////////////////////////////////////

void setup() {
  Serial.begin(9600);
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz). Comment this line if having compilation difficulties with TWBR.
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  mpu.initialize();

  // load and configure the DMP
  devStatus = mpu.dmpInitialize();
  // supply your own gyro offsets here, scaled for min sensitivity
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
    // turn on the DMP, now that it's ready
    mpu.setDMPEnabled(true);
    // enable Arduino interrupt detection
    //attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();
    // set our DMP Ready flag so the main loop() function knows it's okay to use it
    dmpReady = true;

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
```

```
 } else {
   // ERROR!
   // 1 = initial memory load failed
   // 2 = DMP configuration updates failed
   // (if it's going to break, usually the code will be 1)

   int k = 1; //za spilajnapadalec()
 }
 pinMode(2, OUTPUT);   //motorji
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
 pinMode(6, OUTPUT);
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(9, OUTPUT);
 pinMode(10, OUTPUT);
 pinMode(11, OUTPUT);
 pinMode(12, OUTPUT);
 pinMode(13, OUTPUT);

 pinMode(33, INPUT_PULLUP); // tipke
 pinMode(31, INPUT_PULLUP);
 pinMode(25, INPUT_PULLUP);
 pinMode(23, INPUT_PULLUP);

 pinMode(A9, INPUT); //senzorji za črto
 pinMode(A10, INPUT);
 pinMode(A11, INPUT);
 pinMode(A12, INPUT);

 pinMode(47, OUTPUT); // led
 pinMode(49, OUTPUT);
 pinMode(51, OUTPUT);
 pinMode(53, OUTPUT);

 pinMode(A0, INPUT);
 pinMode(A1, INPUT);// ir
 pinMode(A2, INPUT);
 pinMode(A3, INPUT);
 pinMode(A4, INPUT);
 pinMode(A5, INPUT);
 pinMode(A6, INPUT);
 pinMode(A7, INPUT);
 pinMode(A8, INPUT);

}
////////////////////////////////////////////////////////////////////////////////////////////
int one_time = 1;
void loop() {

 digitalWrite(led1, HIGH);
 digitalWrite(led2, HIGH);
 digitalWrite(led3, HIGH);
 digitalWrite(led4, HIGH);
```

```
 unsigned long tkot = millis();
 if (one_time == 1) {
  do {
   kot();
   one_time = 0;
  } while ((millis() - tkot) < 15 * 1000);   // 15 sekund počaka da se umiri stanje na gyrotu
 }
 digitalWrite(led1, LOW);   // vklop led da vemo kdaj lahko vklopimo program
 //kot();

 if ( digitalRead(t1) == LOW)
 {
  korekcija = kot();
  napadalec();
 }

 if ( digitalRead(t2) == LOW)
 {
  korekcija = kot();
  stoper();
 }
}




////////////////////////////////////////////////
int alpha = 0;
void napadalec() {
 kalibracija();  //za črto si prebere vrednosti na zeleni podlagi

 digitalWrite(led1, HIGH);
 digitalWrite(led2, HIGH);
 digitalWrite(led3, LOW);
 digitalWrite(led4, HIGH);

 while (1) {
  if (digitalRead(t2) == LOW) {
   stoper();
  }
  alpha = kot();
  if ( -11 < alpha && alpha < 11) {   // če je robot med -9 in 9 stopinjami, normalno igra
   spilajnapadalec();
  }
  else if ( alpha <= -11) {  // drugače se poravna do 6 stopnj
   obrni_desno();
  }
  else if (alpha >= 11) {
   obrni_levo();
  }

  crta();          //podprogram za branje črte

 }
}
////////////////////////////////////////////////////////////////////////////////////////
```

```
void stoper() {
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, LOW);
  while (1) {
    pocivaj();
    if (digitalRead(t1) == LOW) {
      napadalec();
    }
  }
}
////////////////////////////////////////////////////////////////////////////////////////////
void spilajnapadalec() {
  int meja = 900;   // meja senzorjev
  int stanje[10];

  int minStanje = 1023;
  int minPin;

  stanje[0] = analogRead(irZL);
  stanje[1] = analogRead(irL);
  stanje[2] = (analogRead(irLN) - 20);
  stanje[3] = analogRead(irLS);
  stanje[4] = analogRead(irS);
  stanje[5] = analogRead(irDS);
  stanje[6] = (analogRead(irDN) - 20);
  stanje[7] = analogRead(irD);
  stanje[8] = analogRead(irZD);
  stanje[9] = 1000;

  for (int i = 0; i < 10; i++) {   // 9x gre skozi senzorje in vidi kateri je najmanjši
    if (stanje[i] < minStanje) {  // če je določeno stanje manjšo od prejšnega
      minStanje = stanje[i];    // najmanjšo stanje
      minPin = i;               //minimalni pin
    }
  }
  switch (minPin)
  {
    case 0: //zadi levo
      if(minStanje>420)
      {
      nazaj();
      }
      else{
        desnonazajposevno();
      }
      break;
    case 1: //levi
      if(minStanje<250){
        nazaj();
      }
      else{
      levonazajposevno();
      }
      break;
```

```
   case 2: //levo naprej
    if (minStanje > 400)
    {
     naprej();
    }
    else {
     levo();
    }
    break;
   case 6: //desno naprej
    if (minStanje > 400)
    {
     naprej();
    }
    else {
     desno();
    }
    break;
   case 7: //desno
    if(minStanje<250){
     nazaj();
    }
    else{
    desnonazajposevno();
    }
    break;
   case 8: //zadi desno
    if (minStanje > 420)
    {
     nazaj();
    }
    else {
     levonazajposevno();
    }
    break;
   case 9:
    pocivaj();
    break;
   default: // 3, 4 in 5 // naprej
    naprej();
  }
}
////////////////////////////////////////////////////////////////////////
void spilajstoper() {
}
void naprej()   // podprogram za naprej
{
  analogWrite(mzf, 0);
  analogWrite(mzb, 0);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 200);
  analogWrite(mlb, 0);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 200);
```

```
  analogWrite(mdb, 0);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
void nazaj()
{
  analogWrite(mzf, 0);
  analogWrite(mzb, 0);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 0);
  analogWrite(mlb, 200);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 0);
  analogWrite(mdb, 200);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
void levo()
{
  analogWrite(mzf, 0);
  analogWrite(mzb, 180);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);

  analogWrite(mlf, 0);
  analogWrite(mlb, 90);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 95);
  analogWrite(mdb, 0);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
void desno()
{
  analogWrite(mzf, 180);
  analogWrite(mzb, 0);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 90);
  analogWrite(mlb, 0);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 0);
  analogWrite(mdb, 95);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
void pocivaj()
{

  analogWrite(mzf, 0);
  analogWrite(mzb, 0);
  digitalWrite(mzfd, LOW);
```

```
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 0);
  analogWrite(mlb, 0);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 0);
  analogWrite(mdb, 0);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
void obrni_levo()
{
  analogWrite(mzf, 60);
  analogWrite(mzb, 0);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 0);
  analogWrite(mlb, 60);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 60);
  analogWrite(mdb, 0);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);

}
void obrni_desno()
{
  analogWrite(mzf, 0);
  analogWrite(mzb, 60);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 60);
  analogWrite(mlb, 0);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 0);
  analogWrite(mdb, 60);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
void desnonazajposevno() // 30*
{
  analogWrite(mzf, 90);
  analogWrite(mzb, 0);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 0);
  analogWrite(mlb, 90);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 0);
  analogWrite(mdb, 180);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
```

```
void levonazajposevno() // 30*
{
  analogWrite(mzf, 0);
  analogWrite(mzb, 90);
  digitalWrite(mzfd, LOW);
  digitalWrite(mzbd, LOW);
  analogWrite(mlf, 0);
  analogWrite(mlb, 180);
  digitalWrite(mlfd, LOW);
  digitalWrite(mlbd, LOW);
  analogWrite(mdf, 0);
  analogWrite(mdb, 90);
  digitalWrite(mdfd, LOW);
  digitalWrite(mdbd, LOW);
}
void kalibracija()   //prebere vrednosti in si jih zapomni (na začetku programa v setup)
{
  vrednostZ = analogRead(scZ);
  vrednostL = analogRead(scL);
  vrednostD = analogRead(scD);
  vrednostP = analogRead(scP);
}
void crta()
{
  int zamik = 400; // kako dolgo gre bek od črte  (400 je kul)

  unsigned long timer;

  if (analogRead(scZ)  < (vrednostZ - odstopanjebela))   // če je prebrana vrednost manjša od kalibriranje in
odstanje za belo barvo
  {
    timer = millis();
    do {
      naprej();
    } while ((millis() - timer) < zamik); // toliko časa gre v določeno smer
  }

  else if (analogRead(scL) < (vrednostL - odstopanjebela))
  {
    timer = millis();
    do {
      desno();
    } while ((millis() - timer) < zamik);

  }
  else if (analogRead(scD) < (vrednostD - odstopanjebela))
  {
    timer = millis();
    do {
      levo();
    } while ((millis() - timer) < zamik);
  }
  else if (analogRead(scP) < (vrednostP - odstopanjebela))
  {
    timer = millis();
    do {
```

```
    nazaj();
   } while ((millis() - timer) < zamik);
  }
}
void crtastoper() {
  int zamik = 400; // kako dolgo gre bek od črte  (400 je kul)
  unsigned long timer;
  if (analogRead(scZ)  < (vrednostZ + odstopanjecrna))   // če je prebrana vrednost manjša od kalibriranje in
odstanje za belo barvo
  {
    timer = millis();
    do {
      naprej();
    } while ((millis() - timer) < zamik); // toliko časa gre v določeno smer
  }
  else if (analogRead(scL) > (vrednostL + odstopanjecrna))
  {
    timer = millis();
    do {
      desno();
    } while ((millis() - timer) < zamik);
  }
  else if (analogRead(scD) > (vrednostD + odstopanjecrna))
  {
    timer = millis();
    do {
      levo();
    } while ((millis() - timer) < zamik);
  }
  else if (analogRead(scP) > (vrednostP + odstopanjecrna))
  {
    timer = millis();
    do {
      nazaj();
    } while ((millis() - timer) < zamik);
  }
}
```