ŠOLSKI CENTER CELJE

Srednja šola za strojništvo, mehatroniko in medije

POLAVTONOMNI REŠEVALNI ROBOT

Raziskovalna naloga

Področje: Mehatronika in robotika

Avtorji:

Žan HEDŽET KOSTAJNŠEK, M-4. c

Jakob KOLENC, M-4. c

Jaša SAMEC, M-4. c

Mentor:

mag. Matej VEBER, univ. dipl. inž.

ZOTKS, Srečanje mladih raziskovalcev RS, Murska Sobota 2020

ZAHVALA

Zahvaljujemo se vsem, ki so nam pri izdelavi te raziskovalne naloge pomagali. Posebej se zahvaljujemo sošolcu Lenartu Marovtu, ki nam je pomagal pri grafičnem oblikovanju.

Iskrena hvala mentorju mag. Mateju Vebru, univ. dipl. inž., ki nam je raziskovanje omogočil, nam nudil teoretične in praktične nasvete ter nas vseskozi spodbujal. Hvala tudi prof. Brigiti Renner za jezikovni pregled naloge.

POVZETEK

V svetu, kjer je vse več naravnih katastrof, je tudi vedno večja potreba po sodobnih rešitvah, ki bi lahko pomagale reševalcem. Zato smo si zadali cilj, da ustvarimo robota, ki bi bil sposoben pri nekaterih nalogah nadomestiti reševalce. Odločili smo se, da razvijemo avtonomno vozečega robota, ki bi bil sposoben prevoziti prostor, ne da bi mu pri tem pomagal človek.

Skozi raziskovalno nalogo smo reševali tako programske, kot tudi strojne in elektronske probleme za razvoj takšnega robota. Poleg avtonomnega pa izdelujemo tudi tekmovalnega robota za tekmovanje RoboCup Rescue RMRC 2020, ki bo potekalo v Franciji. V nalogi smo ugotavljali načine za razvoj optimizirane elektronike, zasnovane na tiskanem vezju, in poskušali optimizirati topologijo konstrukcije robota. Razvoj konstrukcije in elektronike smo zasnovali na osnovi podobnega lanskoletnega reševalnega robota, pri čemer smo ga poskušali karseda izboljšati.

SUMMARY

In a world where there are more natural disasters than ever, there is a growing need for modern solutions. That is why we gave ourselves a goal to create a robot that will help rescue teams. We decided to create an autonomous robot that could traverse a room without any human intervention.

Through our paper, we describe how we resolved software as well as hardware problems. Alongside creating an autonomous robot, we also made a competition robot for the Robocup rescue RMRC competition that will take place in France. We also tried to find solutions to better our circuitry by using PCBs as well as optimize the robot's design. The basis for the idea and the electronics was last year's competition robot, that we tried to improve.

KAZALO VSEBINE

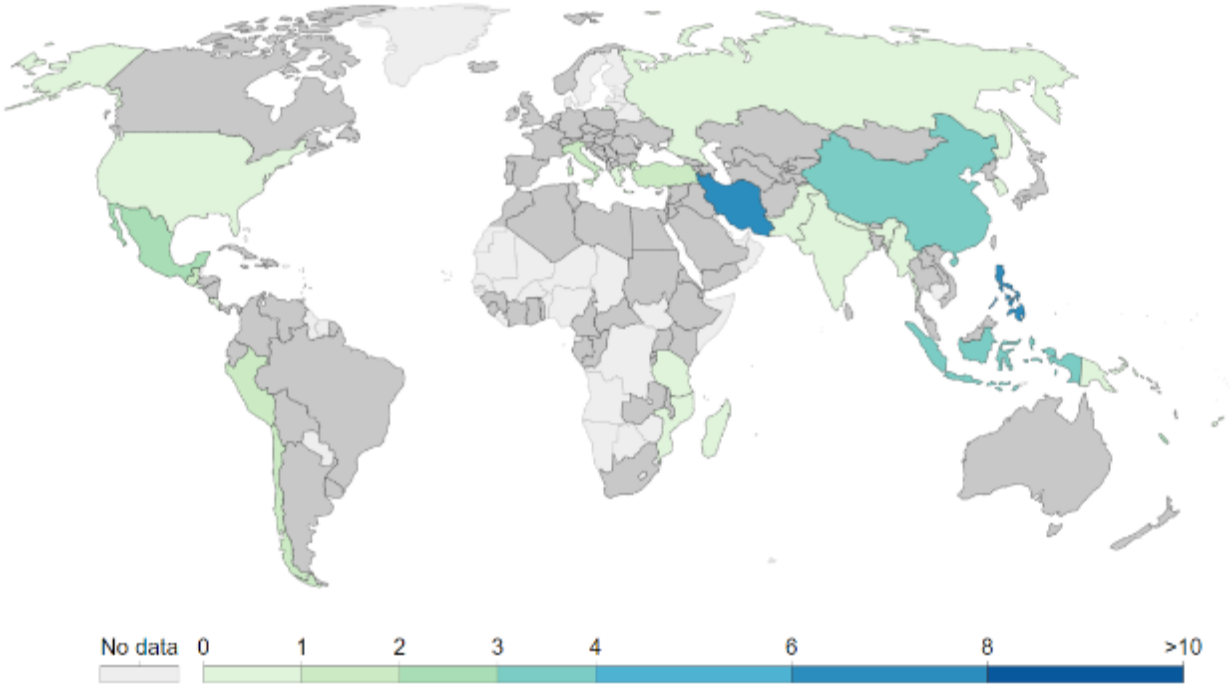
KAZALO SLIK

KAZALO TABEL

KAZALO GRAFOV

1 UVOD

V današnjem času je vse več naravnih katastrof. Večjo nevarnost predstavljajo potresi, spet druge težave pa pozročajo neurja in posledice podnebnih sprememb, ki so zadnje čase še posebej očitne. V zadnjih desetletjih so ravno potresi terjali največ smrtnih žrtev. Čeprav je večina manjših in jih skoraj ne čutimo, vsake toliko časa nastopijo večji s hudimi posledicami. Na žalost na njih nismo dovolj pripravljeni. [1]



Slika 1: Hujši potresi v letu 2017
(Vir: https://ourworldindata.org/natural-disasters)

Potresi tudi v Sloveniji predstavljajo večjo nevarnost. Vsako leto jih seizmografi zabeležijo okrog 100. Približno 65,3 % celotnega prebivalstva Slovenije živi na območju, kjer so mogoči potresi z intenziteto 7 po MSK-lestvici. [2]

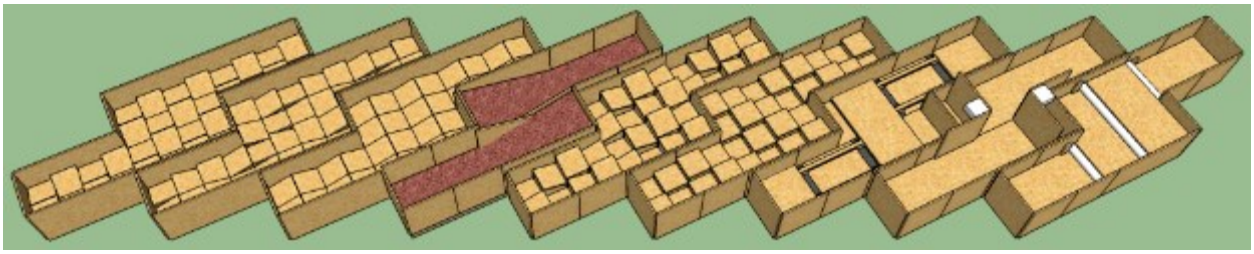
Slika 2: Karta potresne nevarnosti Slovenije

(Vir: http://gis.arso.gov.si/atlasokolja/profile.aspx?id=Atlas_Okolja_AXL@Arso)

Poleg samih potresov tudi popotresni sunki predstavljajo hudo nevarnost za reševalne ekipe. Območje potresa je lahko še nekaj časa po dogodku nevarno za ekipe, zato se lahko pogosto znajdejo v nevarnosti. Prav zaradi tega smo želeli razviti reševalnega robota, ki bi lahko sam prevozil prostor in reševalni ekipi narisal zemljevid okolja. Hkrati pa smo želeli nadgraditi že obstoječega reševalnega robota. V naši raziskovalni nalogi bomo predstavili, kako smo ustvarili avtonomnega reševalnega robota. Nadgradili smo aspekte reševalnega robota, ki so ga v lanskem šolskem letu izdelali naši dijaki.

1.1 SVETOVNO PRVENSTVO ROBOCUP RESCUE RMRC

Junija 2020 se bomo udeležili svetovnega tekmovanja RoboCup 2020 v Bordeauxu. Tekmovanja se bo udeležilo 3500 tekmovalcev iz 45 držav. Tekmovali bomo na različnih področjih in v različnih panogah. Naša panoga se imenuje RMRC (Rapidly Manufactured Robot Challenge). Cilj tekmovanja je izdelati robota za reševanje, ki je daljinsko voden. Robot mora opraviti tri naloge. Prva je prevoziti tekmovalno areno z raznimi ovirami. Operater med vožnjo ne sme videti robota in lahko gleda le na zaslon pred seboj.



Slika 3: Tekmovalne arene

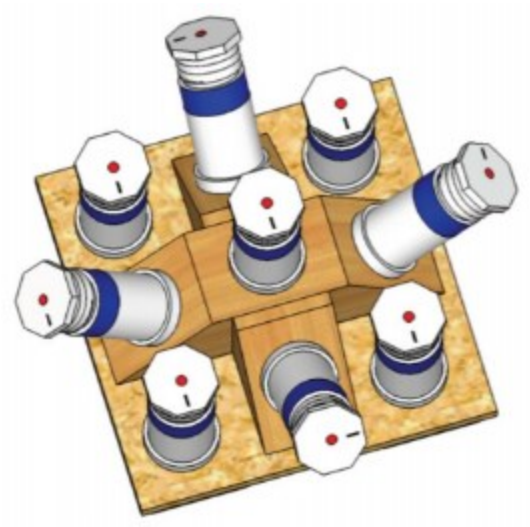
(Vir: http://oarkit.intelligentrobots.org)

V drugem delu tekmovanja mora robot prepoznati znake za nevarnost, kode QR, spremembo temperature in prisotnost CO_2.

Slika 4: Znaki na tekmovanju

(Vir: http://oarkit.intelligentrobots.org)

V tretjem delu tekmovanja pa je potrebno z robotsko roko odpreti posamezne ventile in pritiskati na tipke tipkovnice.



Slika 5: Ventili na tekmovanju
(Vir: http://oarkit.intelligentrobots.org)

Organizatorji so zadnja leta razmišljali, da bi ekipe nagradili z dodatnimi točkami, če bi robot naloge opravljal popolnoma sam. To nam je dalo še dodatno motivacijo, da raziščemo možnost avtonomnega robota.

1.2 PREDSTAVITEV PROBLEMA

V letošnjem letu smo si zadali cilj, da dodatno nadgradimo reševalnega robota in ustvarimo različico avtonomno vozečega robota. Spopadali smo se tako s težavami v programskem delu, kot tudi s tistimi v elektronskem in mehanskem. Pri raziskovanju smo naleteli na težave pri optimizaciji topologije robota, kasneje pa še pri risanju tiskanega vezja. V programskem delu raziskovanja nam je največ težav povzročalo programiranje avtonomne vožnje. Vse težave smo s pomočjo svetovnega spleta, nasvetov mentorja, testiranja in lastnega znanja, pridobljenega skozi vsa štiri leta našega izobraževanja, uspešno rešili.

1.3 HIPOTEZE

Cilj naše raziskovalne naloge je, da raziščemo možnost avtonomnega reševalnega robota in hkrati izboljšamo delovanje trenutnega tekmovalnega robota. Želeli smo, da bi bili naši algoritmi in ideje iztočnice za nadaljnje raziskovanje na področju avtonomije in reševanja z roboti. Naša raziskovalna naloga bo predstavila probleme, s katerimi smo se srečali, tako da se lahko v prihodnosti uporabi kot iztočnica pri podobnih podvigih.

V ta namen smo postavili naslednje hipoteze:

- H1 – Zaradi optimizacije topologije se bo potencialna energija pri padcu robota zmanjšala za 10 %.
- H2 – Robot bo sam sposoben prevoziti osnovno tekmovalno areno.
- H3 – Robot bo sposoben narediti zemljevid arene z maksimalno 5 % odstopanjem.
- H4 – Robot bo sposoben priti do določene točke v prej narejenem zemljevidu in odstopanje ne bo večje od 5 %.
- H5 – Z uporabo tiskanega vezja bomo dosegli, da bo letošnje vezje manjše od lanskega.

1.4 METODE RAZISKOVANJA

Pri raziskovanju smo uporabili naslednje metode:

- Metodo razčlenitve, ki temelji na osnovi razčlenitve neke celote. Na ta način smo si razdelili delo. Raziskovanje smo tako razdelili na strojni del ali konstrukcijo, elektronski del in tiskano vezje ter programski del, ki zajema velik del te raziskovalne naloge.
- Primerjalno metodo, ki temelji na primerjanju dveh ali več podobnih elementov. Ta metoda nam je pomagala pri problemih, kot sta izbira mikroračunalnika in izbira programske opreme za izdelavo tiskanega vezja.
- Testiranje v različnih okoliščinah avtonomne vožnje in reševanja, primerjava in ovrednotenje rezultatov.

2 PREDSTAVITEV POTEKA RAZISKOVALNE NALOGE

Naš glavni namen raziskovanja je bil, da razvijemo robota, ki je zmožen avtonomne vožnje. Že od samega začetka smo vedeli, da bo pot do avtonomije dolga in zahtevna, zato smo se reševanja problemov lotili pametno in analitično. Ko smo določili način, s katerim bomo zaznavali okolico, smo lahko začeli z načrtovanjem kode, pri čemer smo si pomagali z znanjem matematike in programiranja.

Kljub temu da je raziskovalna naloga namenjena predvsem avtonomni vožnji, bomo v njej predstavili tudi tekmovalnega robota, ki ga trenutno še razvijamo za tekmovanje RoboCup Rescue RMRC 2020, ki se bo odvijalo v Franciji v juniju 2020. Takšnega robota je delala že skupina vrstnikov v lanskem letu, letos pa smo se odločili, da ga med raziskovanjem še dodatno nadgradimo. Programsko smo ga poskušali izboljšati pri branju QR-kod. Pri mehanskih posodobitvah smo se posvetili optimizaciji topologije, s katero smo želeli narediti robota lažjega. Elektroniko pa smo med raziskovanjem posodobili z razvojem in uporabo tiskanega vezja.

3 AVTONOMIJA

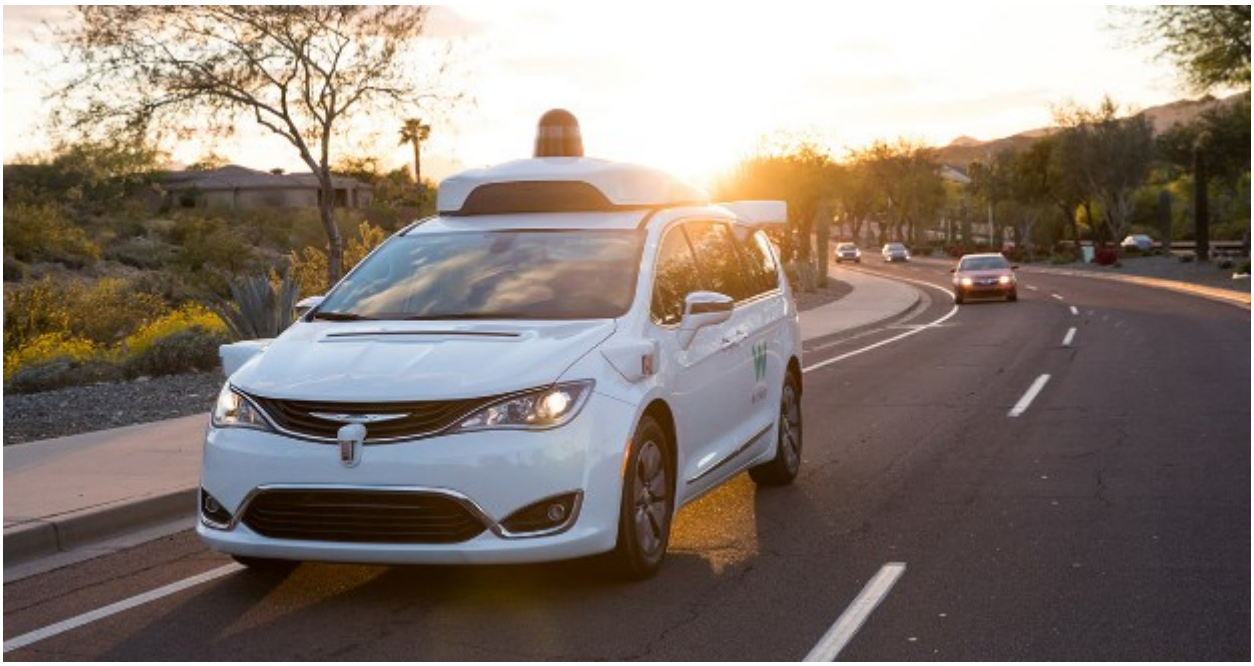
3.1 AVTONOMNA VOŽNJA

Dejstvo je, da ljudje nismo dobri vozniki. Razni faktorji prevečkrat vplivajo na naše odločitve, reakcije in pozornost. Ljudje se že mnogo let ukvarjajo z avtonomnimi avtomobili. Čeprav se sistemi razvijajo že nekaj let, to ne pomeni, da so že blizu rešitvi. [3]

Leta 2007 je na DARPA izzivu tekmovalo 11 avtonomnih vozil. Njihova naloga je bila, da prevozijo urbano okolje, ki so ga ustvarili za tekmovanje. Celotna proga je bila dolga 89 km in ekipa, ki je zmagala, jo je prevozila v malo več kot štirih urah. Poleg zmagovalne ekipe je progo uspelo prevoziti le še petim. [4]

Slika 6: Tekmovalni avto na DARPA izzivu
(Vir: https://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2007))

Po tekmovanju je veliko ljudi mislilo, da je problem avtonomne vožnje rešen. Vendar je od takrat minilo že več kot deset let in nismo veliko bližje avtonomnim taksijem. Danes ostaja v povezavi z avtonomno vožnjo še veliko nerešenih problemov. Mednje sodijo lokalizacija, izdelava zemljevida, zaznavanje okolja, nadzor vozila, načrtovanje poti ipd. Edina možnost, da so danes na cesti avtonomna vozila, je, če je za volanom tudi človek. Njegova naloga je, da prevzame nadzor v situacijah, ko vozilo ni prepričano, kako naj se odzove. [3]

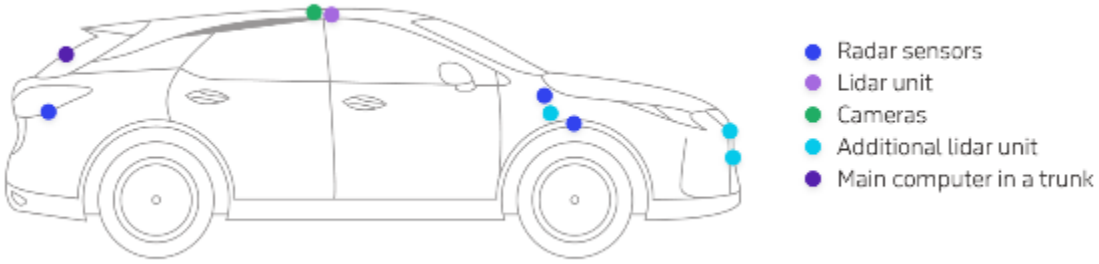


Slika 7: Avtonomni avtomobil podjetja Waymo

(Vir: https://waymo.com/)

3.2 SENZORJI PRI AVTONOMNI VOŽNJI

Da bi avtonomna vozila lahko vozila bolje kot ljudje, morajo najprej zaznavati okolje bolje kot ljudje. Da lahko to dosežemo, potrebujemo dovolj dobre senzorje. Trenutno se večina avtonomnih avtomobilov zanaša na tri različne senzorje. To so RADAR, kamere in LIDAR. Vsak izmed njih ima svoje prednosti in slabosti. [5]

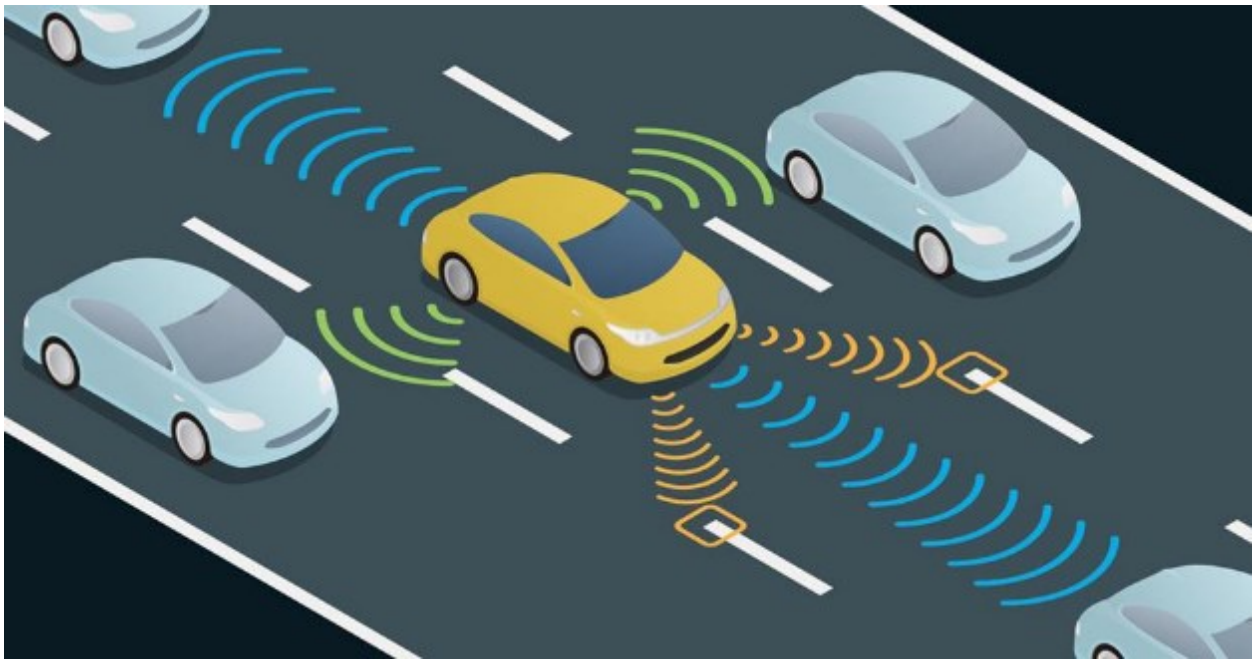


Slika 8: Senzorji na avtonomnih avtomobilih

(Vir: https://www.itransition.com/blog/autonomous-vehicle-sensors)

3.2.1 RADAR

RADAR je sistem, ki se zanaša na radijsko valovanje za zaznavanje objektov. Oddajnik oddaja pulze radijskih valov, ki se odbijejo od avtomobilov, pešcev in drugih ovir nazaj v senzor. Tako je mogoče izračunati oddaljenost in hitrost drugih udeležencev v prometu. RADAR za delovanje ne potrebuje svetlobe, tako da je primeren za vožnjo ponoči. Na žalost pa preko njega ni mogoče prepoznati objektov, določimo lahko le njihovo razdaljo in hitrost. [6]



Slika 9: Primer delovanja RADAR-ja

(Vir: https://www.crutchfield.com/S-TIOo9BJFp8i/learn/radar-detector-glossary.html)

3.2.2 LIDAR

LIDAR uporablja laser, s katerim osvetli okolje in nato preko odboja ustvari 3D sliko (podobno kot RADAR). [6]

Ta senzor omogoči, da računalnik dobi natančno 3D sliko. Prav tako kot RADAR deluje tudi ponoči. Senzor je zelo natančen in uporaben, vendar je zaenkrat še precej drag. [5]

Slika 10: Primer delovanja LIDAR-ja

(Vir: https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff?gi=84dabaa744d9)

3.2.3 Kamere

Kamere so najzanesljivejši način zaznavanja okolja. Na avtonomnih avtomobilih so postavljene tako, da računalnik dobi 360° sliko okolja. Z njimi lahko prepoznavamo druge udeležence v prometu, prometne znake, nevarnosti itd. Imajo pa dve veliki slabosti. Prva je, da z njimi težko določimo razdalje, druga pa je, da v razmerah, kjer je vidljivost slabša, težko prepoznajo objekte. [5]

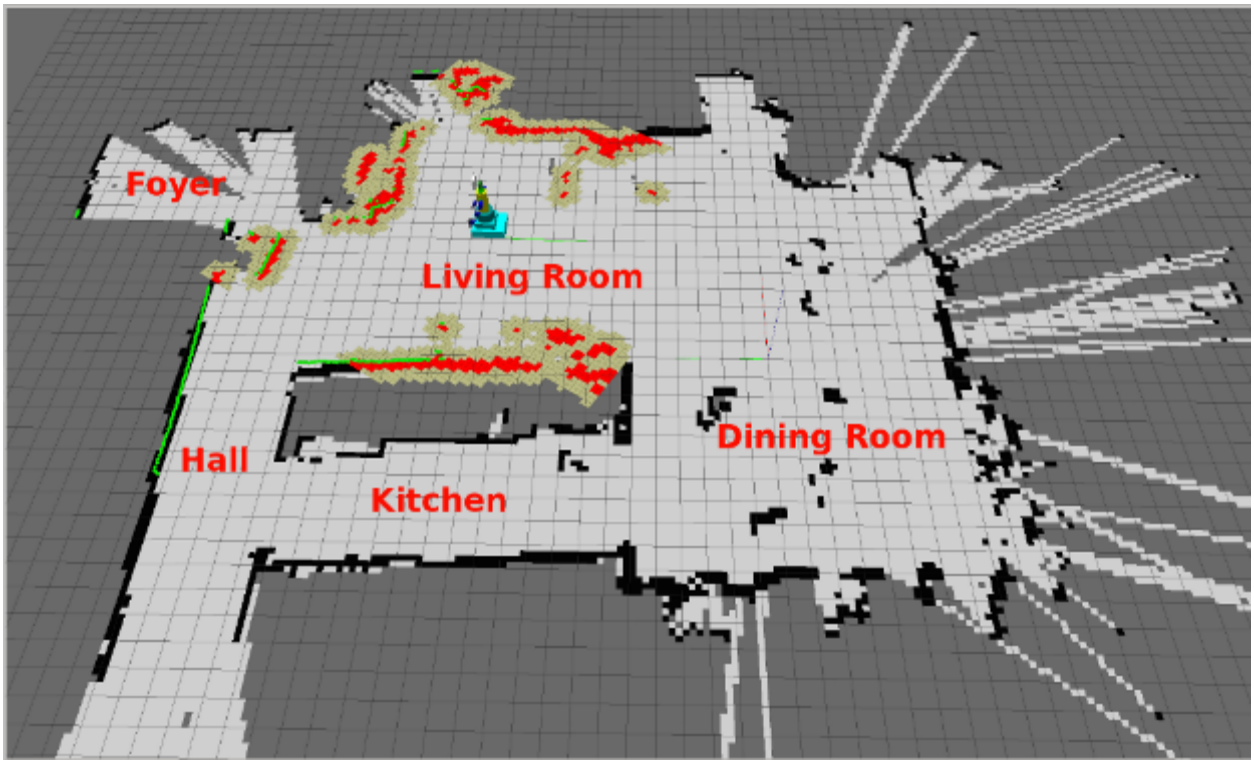
Nekateri strokovnjaki menijo, da bodo lahko v prihodnosti s stereokamerami dosegli visoko natančnost pri računanju razdalj in s tem izničili potrebo po dragem LIDAR-ju. [7]

Slika 11: Stereokamere

(Vir: https://www.therobotreport.com/researchers-back-teslas-non-lidar-approach-to-self-driving-cars/)

3.3 SLAM

Kadar v robotiki želimo ustvariti zemljevid neznanega prostora, najpogosteje uporabimo algoritme, ki omogočajo sočasno lokalizacijo in gradnjo zemljevida (SLAM, simultaneous localization and mapping). S temi algoritmi želimo istočasno slediti poziciji robota in ustvarjati tloris njegove okolice. Na prvi pogled ta težava nima rešitve, saj potrebujemo zemljevid za izračun položaja, položaj pa za sestavo zemljevida. Zaradi tega se uporabljajo različni senzorji za določanje lege in iz njih se izračuna povprečna pozicija. Tako se lahko ta sistem prilagodi senzorjem in uporabi. Ta fleksibilnost omogoča, da so SLAM uporabili na veliko področjih (npr. avtonomna vožnja, roboti na Marsu …). [8]

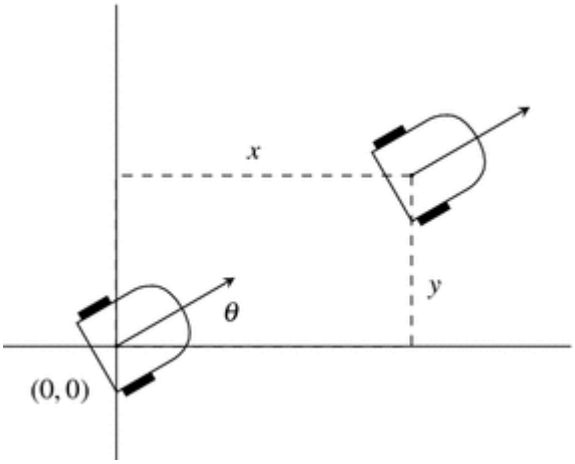


Slika 12: Zemljevid narejen z LIDAR senzorjem

(Vir: https://www.pirobot.org/blog/0015/)

3.3.1 Odometrija

Odometrija je uporaba senzorjev, ki zaznavajo premik za izračun lege robota. Ta način računanja pozicije je v primerjavi z ostalimi enostavnejši, vendar je zaradi tega tudi manj natančen. Najbolj očiten razlog je, da sistem ni sposoben zaznati zdrsa koles. [9]



Slika 13: Odometrija

(Vir: https://link.springer.com/chapter/10.1007/978-3-319-62533-1_5)

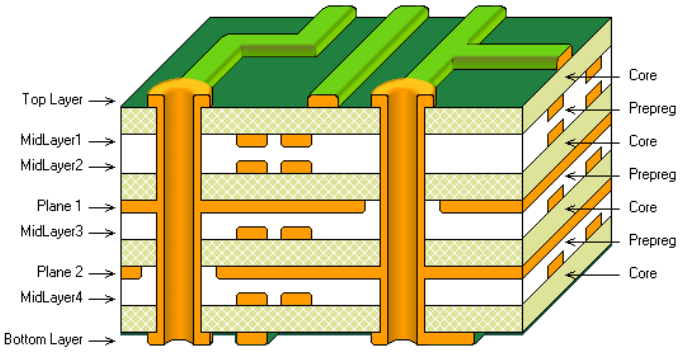
Zaradi enostavne uporabe se odometrija uporablja poleg drugih sistemov za izračun lege pri SLAM algoritmih. [8]

4 TISKANO VEZJE

Tiskano vezje ali tiskanina omogoča mehansko osnovo in električno povezanost elektronskih elementov. Sestavljeno je iz enega ali več plasti bakra, ki so oblikovane tako, da omogočajo različne vrste stikov z elektronskimi komponentami, plasti bakra pa so ločene s posebnimi prevodnimi materiali. Slednji so na tiskanino spojeni, s čimer omogočimo stabilen elektronski stik in mehansko pritrditev.

Tiskano vezje se je razvilo iz tehnologij "wire wrap" in "point-to-point construction", ki sta danes neuporabni in zamudni. Za izdelavo tiskanine je za razliko od prej omenjenih tehnologij sicer potrebno izdelati računalniški program, a sta proizvodnja in montaža na principu nove tehnologije mnogo hitrejši, poleg tega pa je omogočena uporaba manjših elektronskih komponent in vezi med njimi, kar močno vpliva na velikost vezja.

Za razvoj tiskanin se uporabljajo različni računalniški programi, ki omogočajo hitrejšo in enostavnejšo modeliranje zahtevnih vezij. Zahtevnost razvoja vezja se stopnjuje s številom plasti, ki ga vezje vsebuje, s tem pa se hkrati znižuje možnost za večjo gostoto elektronskih komponent na tiskanini in posledično zmanjšanje tiskanih vezij, kar je danes še kako pomembno. [10]



Slika 14: Prerez 8-plastne tiskanine

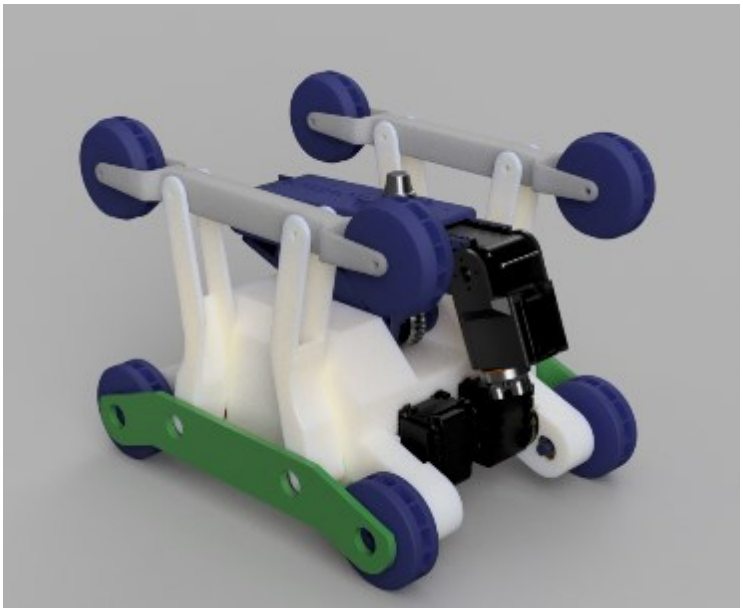
(Vir: https://www.allpcb.com/8_layer_pcb.html)

5 TEKMOVALNI ROBOT

Lanski tekmovalni robot se je med redkimi prebil čez vse ovire na tekmovanju. To mu je uspelo zaradi njegovih gosenic, ki so v osnovi lahko v dveh položajih: v obliki pravokotnika ali paralelograma.

Slika 15: Lanski tekmovalni robot
(Osebni vir)

Kadar so gosenice v obliki pravokotnika, omogočajo robotu hitro vožnjo in obračanje. Ko pride do ovire, lahko postavi gosenice v obliko paralelograma. To mu omogoča, da gosenice postavi na oviro in jo tako lažje premaga. Po našem mnenju je ideja z robotom vrhunska, zato konstrukcije nismo želeli preveč spreminjati.



Slika 16: Letošnji tekmovalni robot

(Osebni vir)

5.1 IZBOLJŠAVE TEKMOVALNEGA ROBOTA

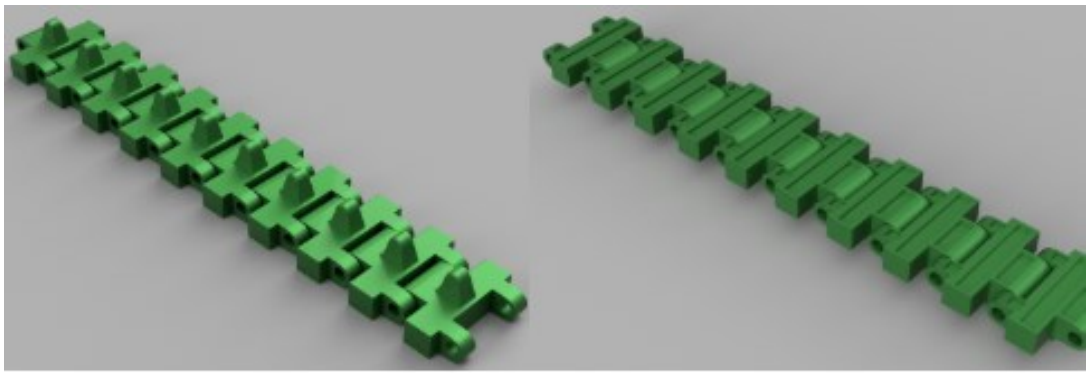
Med opazovanjem lanskoletnega robota na tekmovanju in predvsem z njegovim servisiranjem smo opazili določene pomanjkljivosti. Tako smo se odločili, da bomo naredili nekatere izboljšave.

Pomanjkljivosti, ki smo jih poskušali rešiti:

- gosenice se med vožnjo niso držale koles,
- motorjem so se zaradi velikih sil zvile osi,
- napetost med gosenicami je omejila hitrost motorjev,
- kamere niso dovolj dobro zaznale kode QR.

5.1.1 Gosenice

Stare gumijaste gosenice so s časom postale ohlapne in se snele s koles. Težavo smo rešili tako, da smo natisnili nove. Gosenice smo med seboj povezali z M3 vijaki. Čeprav smo s plastičnimi gosenicami dosegli zastavljeni cilj, smo opazili, da imajo pomanjkljivost. Plastika nima dovolj velikega koeficienta trenja, da bi jo lahko uporabili. Zaradi tega smo prišli do ideje, da bi na njih dodali silikon, kar je težavo rešilo.

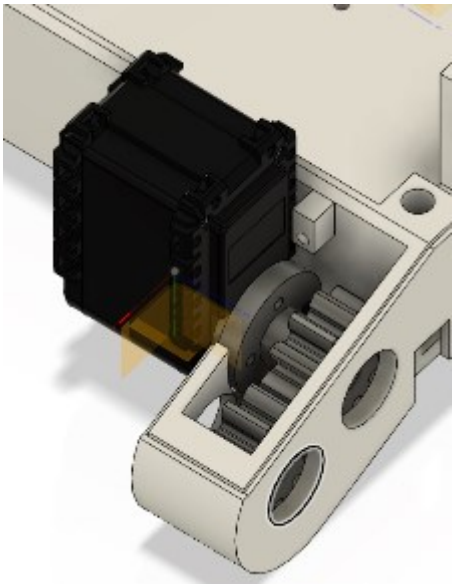


Slika 17: Nove gosenice

(Osebni vir)

5.1.2 Prenos sil na zobnike

Zaradi neposrednega prenosa pogona iz motorja na kolesa so se v motorju zvile osi. To smo popravili s preprostim zobniškim prenosom med motorjem in pogonskim kolesom. Za ta prenos smo uporabili dva zobnika z razmerjem 1 : 1 in tri ležaje.



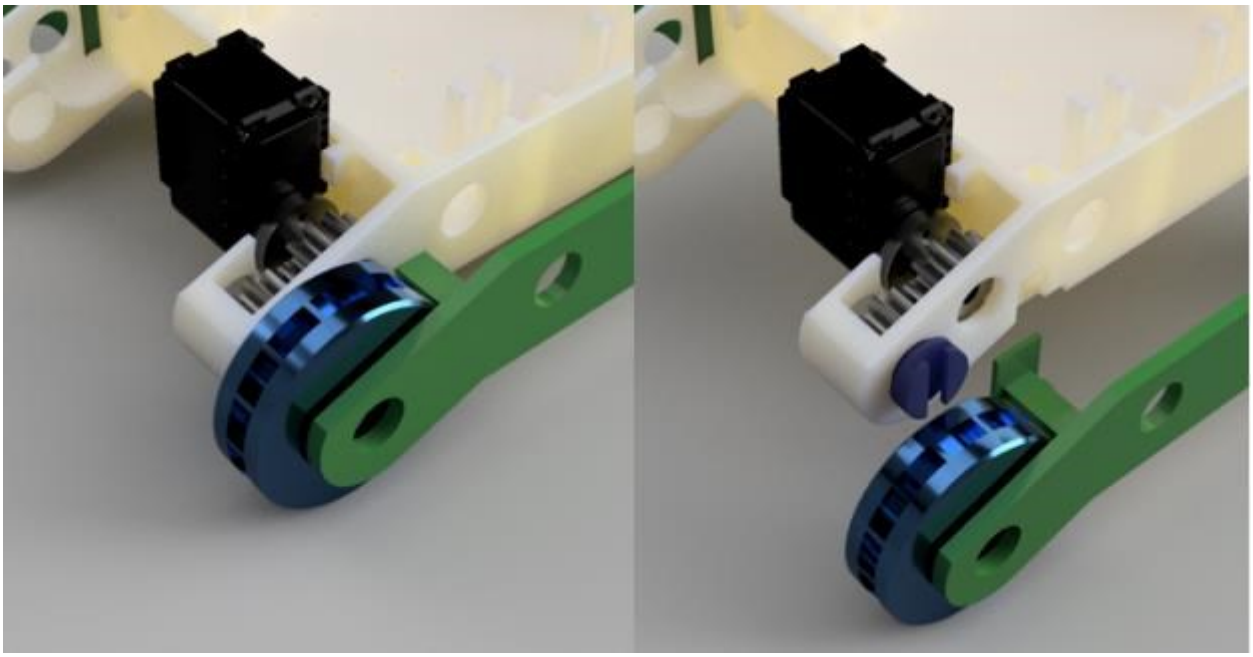
Slika 18: Zobniški prenos

(Osebni vir)

5.1.3 Možnost menjave gosenic za kolesa

Po opazovanju našega tekmovalnega robota in robotov konkurence smo ugotovili, da so za nekatere terene primernejša kolesa. Ta so imela prednost predvsem na ravnih površinah. Tam so kolesa lahko dosegla precej višje hitrosti v primerjavi z gosenicam. Vendar so na določenih ovirah roboti, ki so imeli kolesa, obtičali. Tako smo prišli do ideje, da bi lahko menjali kolesa in gosenice glede na vrsto proge.

To smo izvedli tako, da smo naredili dva para držal za kolesa. En par ima navadna kolesa, drugi pa ima kolesa, na katera se pritrdijo gosenice.



Slika 19: Mehanizem za menjavo koles
(Osebni vir)

Držalo (na sliki zelene barve) se pritrdi na ohišje z vijakom. Kolesa pa se povežejo na zobnike preko konektorja za hitro menjavo (na sliki vijolične barve). S tem mehanizmom je menjava enostavna in hitra.

5.1.4 Kamera zoom

Ena izmed nalog na tekmovanju je prebiranje QR-kod različnih velikosti. Na tekmovanju lanski ekipi ni uspelo prebrati vseh kod. Na kameri so morali ročno nastaviti lečo. Če so želeli prebrati najmanjšo kodo, je bila največja zamegljena in obratno. Med samo vožnjo pa niso smeli nastavljati leče.

Slika 20: QR-kode na tekmovanju

(Osebni vir)

Prišli smo do rešitve, da bi na kamero pritrdili zobnik in motor. Motor bi lahko med vožnjo vrteli in s tem spreminjali položaj gorišča leče. Odločili smo se, da za pogon uporabimo predelan servomotor. Njegova prednost je, da je majhen in dovolj močan. S tem mehanizmom nam je uspelo prebrati vse štiri QR-kode.

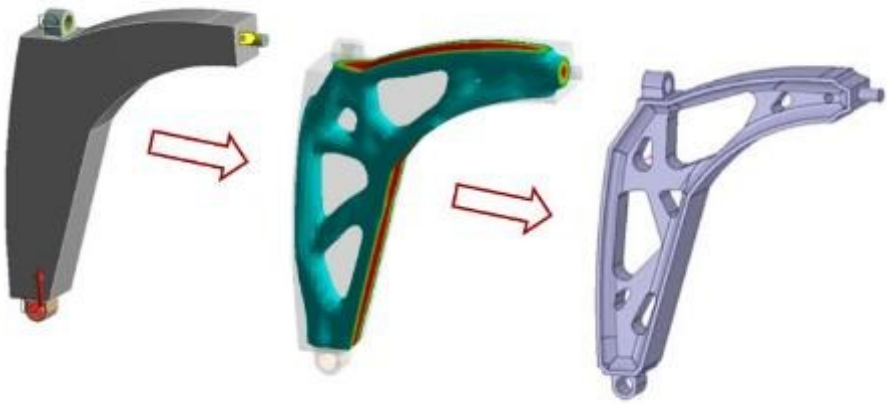
Slika 21: Mehanizem za kamero

(Osebni vir)

5.2 OPTIMIZACIJA TOPOLOGIJE ROBOTA

Na koncu tekmovanja ima vsaka ekipa možnost, da spusti robota z višine 150 cm. Po padcu se mora robot peljati, vendar je velika možnost, da se kakšen pomemben del robota zlomi. Ker ta kategorija prinese veliko točk, želimo letos z robotom opraviti tudi to dodatno nalogo. Najenostavnejši način, da zmanjšamo silo ob pristanku, je, da zmanjšamo potencilno energijo robota pred padcem. Potencialna energija je odvisna od mase, višine in gravitacijskega pospeška. Če zmanjšamo maso, lahko zmanjšamo potencialno energijo.

Ugotovili smo, da lahko največ mase privarčujemo pri 3D tiskanju robota, saj so ostale komponente nepogrešljive in jih ne moremo odstraniti. Ker smo hoteli narediti robota lažjega, smo pri konstruiranju poskusili uporabljati proces, ki ga program FUSION 360 imenuje topology optimization ali optimizacija topologije. Pri tem procesu je potrebno programu najprej podati osnovni model. V programu je nato potrebno podati sile, ki bodo delovale na elementu. Ko to opravimo, program pošlje model na strežnik, kjer izvede simulacijo obremenitev. Tam odstrani ves nepotrebni material. Tako dobimo lažji model, ki naj bi v teoriji prenesel enake sile.



Slika 22: Optimizacija topologije
(Vir: https://www.pinterest.ca/pin/774478467145651132/)

5.2.1 Rezultati optimizacije topologije

Masa vseh nepogrešljivih komponent je 1305 g (motorji, Raspberry Pi, senzorji, vezje …), medtem ko je masa vseh 3D tiskanih delov pred optimizacijo 945 g. Celota pred optimizacijo tehta 2250 g, kar pomeni, da bo imel robot pred padcem potencialno energijo 33.1 J (z višine 1,5 m). Po optimizaciji pa je bila masa kosov 889 g, za skupek 2194 g in potencialno energijo pred padcem 32.3 J. Energija po optimizaciji je za 2,4 % manjša, kar ni bilo skladno z našimi pričakovanji. Razloga za tako majhno spremembo sta dva. Prvi je, da je večina mase nepogrešljive, drugi pa je, da večine kosov ni mogoče optimizirati. Ali so kosi tako zasnovani, da jih ne smemo spreminjati, ali pa so premajhni.

6 STROJNE KOMPONENTE

6.1 ELEKTRONIKA

Elektronski del robota smo zasnovali na obstoječi elektroniki lanskega robota, pri čemer smo upoštevali pomanjkljivosti in napake ter jih poskušali izboljšati. Začeli smo z analizo lanske elektronike, ki smo jo izvedli na podlagi dokumentacije lanskega robota in pogovorov s člani lanske tekmovalne ekipe. S skrbnim pregledom njihovih električnih shem in testiranjem prej omenjenega robota smo najprej nadgradili znanje naših predhodnikov. Ekipo smo nato povprašali o slabostih elektronike in možnosti za nadgradnjo ter na podlagi informacij in lastnega znanja ustvarili seznam z željami za posodobitev električnega vezja.

Po temeljitem premisleku smo določili naslednje pomanjkljivosti:

- Med delovanjem robota nastaja velika verjetnost za nastanek kratkih stikov na vezju, kar ogroža delovanje celotnega robota in predstavlja nevarnost za uničenje dragih priključenih elektronskih naprav, kot je Raspberry Pi ipd.
- Vezje zaradi okornih povezav in velikih komponent zavzema veliko prostora v robotu, s tem pa je konstrukcija robota tudi bolj zapletena, kar dodatno onemogoča dostopnost do vezja v primeru popravila.
- Baterije za napajanje robota neučinkovito zavzemajo prostor.
- Mikroračunalnik Raspberry Pi se pregreva.

Potem, ko smo definirali vse pomanjkljivosti, smo se pozanimali o možnih rešitvah. Ugotovili smo, da je edina zares učinkovita rešitev za izboljšavo, uporaba tiskanega vezja oz. PCB (printed circuit board) za razliko od ročno izdelanega vezja, ki so ga uporabljali in izdelali naši predhodniki.

Začeli smo z natančnim spoznavanjem delovanja, razvoja in izdelave tiskanih vezij. Ugotovili smo, da potrebujemo za kakovostno izdelavo takšnega vezja ustrezno programsko opremo. Preučili smo dve glavni programski opremi za razvoj tiskanih vezij - Autodesk Eagle in Altium Designer. Ti smo primerjali na podlagi parametrov, prikazanih v spodnji tabeli.

	Altium Designer	**Eagle**
3D interaktivni vizualni prikaz tiskanega vezja	DA	NE
Maksimalno število plasti vezja	48	16
Neomejena velikost vezja	DA	DA

Tabela 1: Altium Designer proti Eagle

Po temeljitem premisleku smo prišli do zaključka, da uporabimo Altium Designer. Profesionalna programska oprema je težka za učenje, saj ponuja veliko možnosti, a z vajo in sprotnim učenjem smo lahko pričeli z razvojem lastnega vezja.

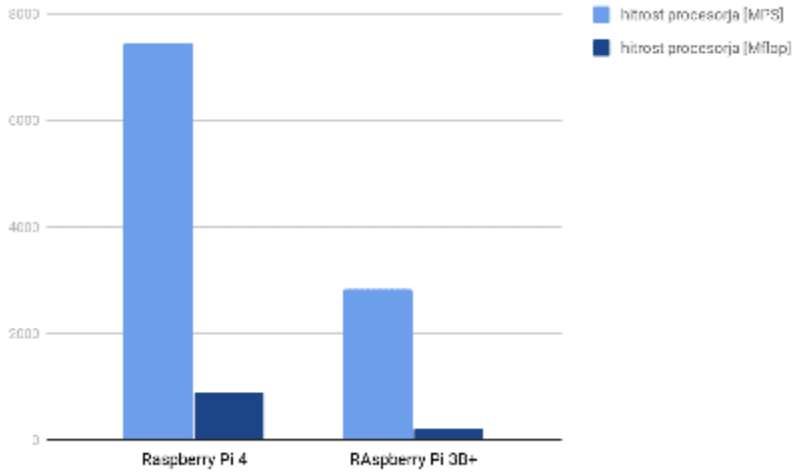
6.1.1 Mikroračunalnik

Naša naslednja naloga je bila izbira mikroračunalnika. Izbirali smo med Raspberry Pi 3 B+, Raspberry Pi 4 in Nvidia Jetson Nano. Ker se ploščice precej razlikujejo, smo jih primerjali glede na njihove specifikacije.

	Raspberry Pi 3 B+	Raspberry Pi 4	Nvidia Jetson Nano
Arhitektura procesorja	Broadcom BCM2837B0	Broadcom BCM2711	Neznano
Procesor	ARM Cortex-A53	ARM Cortex-A73	ARM Cortex-A57
Število jeder procesorja	4	4	4
Frekvenca procesorja	1,4 GHz	1.5 GHz	1,42 GHz
Grafična kartica	VideoCore IV	VideoCore VI	128-jedrni Maxwell
Velikost delovnega pomnilnika	1 GB	4 GB	4 GB
Tip delovnega pomnilnika	LPDDR2 SDRAM	LPDDR4	LPDDR4

Tabela 2: Primerjava mikroračunalnikov

Ploščico Jetson Nano smo, kljub neprimerljivo boljši grafični kartici, zaradi njene velikost kmalu izločili iz ožjega izbora. Poleg te pomanjkljivosti pa Raspberry Pi zaradi ogromnega števila uporabnikov omogoča mnogo enostavnejši razvoj kode.



Graf 1: Grafična primerjava procesorjev Raspberry Pi 3 B+ in Raspberry Pi 4

(Vir:https://www.techrepublic.com/article/raspberry-pi-4-model-b-review-this-board-really-can-replace-your-pc/)

S pomočjo zgornjega grafa smo primerjali hitrosti procesorjev na ploščicah Raspberry Pi 3 B+ in Raspberry Pi 4. Graf prikazuje štiri meritve procesorske moči, pri čemer sta procesorja obremenjena na dva različna načina. Na podlagi teh podatkov smo precenili, da bomo zaradi izrazite razlike v procesorski moči in velikost pomnilnika uporabili novejši model, tj. Raspberry Pi 4.

Slika 23: Raspberry Pi 4

(Vir: https://www.reichelt.com/de/en/raspberry-pi-4-b-4x-1-5-ghz-2-gb-ram-wlan-bt-rasp-pi-4-b-2gb-p259919.html)

6.1.2 Kamere

Eden od načinov zaznavanja okolice, ki smo ga uporabljali za tako avtonomno kot tudi neavtonomno vožnjo, so kamere. Na avtonomni robot smo namestili le eno kamero, na tekmovalnega pa tri.

Slika 24: Kamera

(Vir: https://www.amazon.com/180degree-Fisheye-Camera-usb-Android-Windows/dp/B00LQ854AG?fbclid=IwAR3xE01R1AC3Z19ThUmKuRe1z05RXYqpfaBTLXIamideVS22V8-tYJvpScc)

Pri tekmovalnem robotu je namen glavne kamere vožnja naprej, druga služi za vzvratno vožnjo, tretja pa nam omogoča boljše branje QR-kod.

6.1.3 LIDAR in zaznavanje okolja

6.1.3.1 Zaznavanje okolja

Za nalogo zaznavanja okolja smo izbrali LIDAR. Odločali smo se, da ga bomo uporabili skupaj s kamerami. Kamere bi nam omogočile zaznavanje ovir na več nivojih in s tem oblikovanje 3D slike okolja, vendar bi zahtevale močno grafično kartico za obdelovanje podatkov. Raspberry Pi 4 pa na žalost ni grafično najmočnejši, tako da smo uporabo kamere izločili. Medtem pa LIDAR ne zahteva veliko računalniške moči za procesiranje meritev, ampak nam poda le 2D sliko okolja.

6.1.3.2 Arduino LIDAR

Preden smo se dokončno odločili za nakup LIDAR-ja, smo se morali prepričati, ali lahko z njim izpolnimo zahteve, ki smo si jih zamislili, saj je senzor precej velik strošek. Tako smo naredili svoj LIDAR. Sestavljen je iz laserskega merilnika VL53L0X in servomotorja.

Odločili smo se, da bomo za pogon motorja in sprejemanje podatkov senzorja uporabili Arduino namesto Raspberry Pi. Za motor je potrebno uporabiti 5 V digitalne izhode, medtem ko ima Pi samo 3,3 V izhode.

Slika 25: Arduino LIDAR

(Osebni vir)

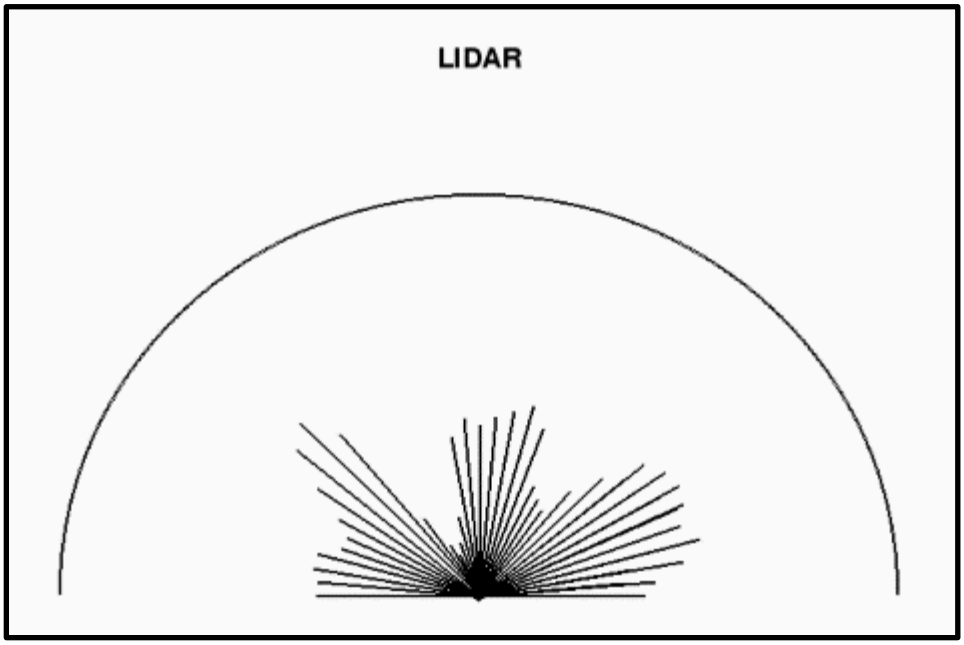
Vendar, ker smo želeli LIDAR postaviti na našega robota in ustvariti sliko okolja, smo uporabili USB za komunikacijo med Arduinom in Raspberry Pi-jem.

```
void loop() {
  //Serial.print nam samo pošlje podatke preko USB kabla
  //tako da jih lahko naš Raspberry Pi sprejme
  for (polozaj_lidarja = 0; polozaj_lidarja <= 180; polozaj_lidarja += 10)
  {
    myservo.write(polozaj_lidarja); //prestavi servo motor
    Serial.print("d");             //d kot distance pove Raspberry Pi-ju, da je to število meritev
    Serial.println(lidar_meritev());//meritev
    Serial.print("a");             //a kot angle, pove da je število kot
    Serial.println(polozaj_lidarja);//kot
    delay(5);
  }
  //ponovimo meritve še v drugo smer
  for (polozaj_lidarja = 180; polozaj_lidarja >= 0; polozaj_lidarja -= 10)
  {
    myservo.write(polozaj_lidarja);
    Serial.print("d");
    Serial.println(lidar_meritev());
    Serial.print("a");
    Serial.println(polozaj_lidarja);
    delay(5);
  }
}
```

Slika 26: Koda za Arduino LIDAR

(Osebni vir)

Za vsako razdaljo, ki jo je laserski merilnik posnel, smo si hkrati zapomnili tudi trenutni kot. Oba podatka smo nato poslali preko serijske povezave do Raspberry Pi-ja, ki ju je procesiral in meritev ponazoril kot črto na ekranu. To se je ponovilo za vsako meritev in tako je nastala spodnja slika.



Slika 27: Prikaz meritev Arduino LIDAR-ja

(Osebni vir)

Čeprav je bil naš eksperiment uspešen, saj smo dokazali, da lahko zanesljivo zaznamo okolje, te naprave nismo mogli uporabiti. Motor se je vrtel prepočasi, da bi lahko z njim zaznavali ovire ob premikanju robota in kabli so se prehitro obrabili. Vendar pa je bil naš cilj dosežen; dokazali smo, da lahko z LIDAR-jem zanesljivo zaznavamo ovire in enostavno obdelamo podatke.

6.1.3.3 LIDAR

Na robota smo postavili RPLIDAR A1. Za ta senzor smo se odločili zaradi njegove enostavne uporabe z Raspberry Pi-jem. Senzor je sposoben zajeti do 8000 točk v sekundi na maksimalni razdalji 12 m.

Slika 28: RPLIDAR

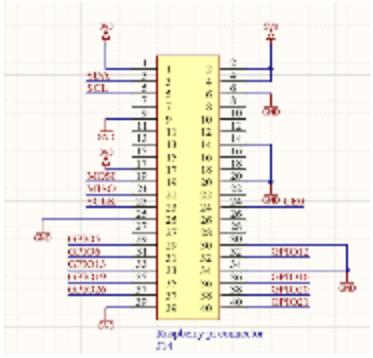
(Vir: https://www.adafruit.com/product/4010)

6.1.4 Tiskanina

Pred začetkom razvoja shem vezja smo za enostavnejši potek dela podsheme razdelili na tri imaginarna področja:

- logične komponente,
- aktuatorske komponente,
- napajalne komponente.

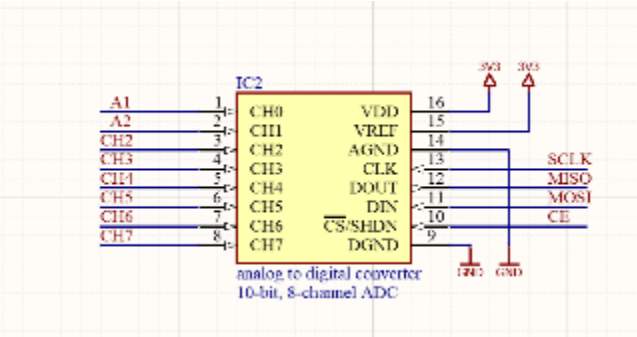
6.1.4.1 Logične komponente



Slika 29: Raspberry Pi konektor

(Osebni vir)

Mikroračunalnik Raspberry Pi 4 smo z vezjem povezali s pomočjo konektorja s 40 kontakti. Konektor nam omogoča, da mikroračunalnik pozicioniramo pod tiskanino in s tem zmanjšamo prostor, ki ga vezje zavzema v robotu.

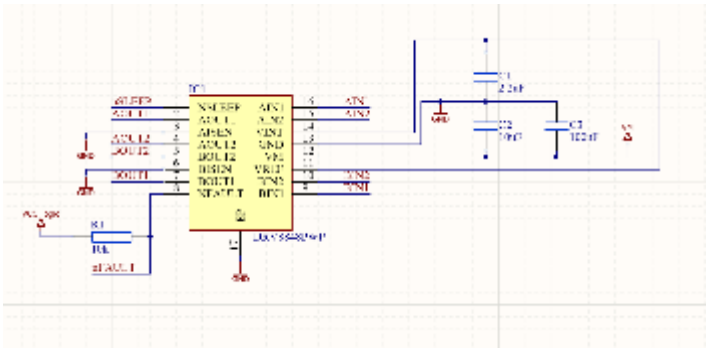


Slika 30: ADC

(Osebni vir)

Poleg digitalnih senzorjev smo uporabljali tudi analogne. Ker Raspberry Pi ne omogoča priklopa analognih senzorjev direktno na logično vezje, smo vse analogne signale pretvorili v digitalne. Za to smo izbrali čip MCP3004/3008, proizvajalca Microchip.

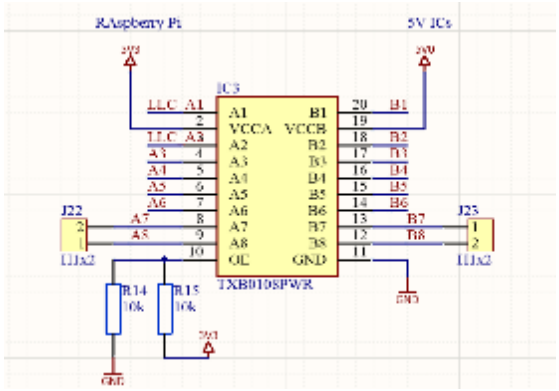
10-bitno integrirano vezje nam omogoča visoko resolucijo na vseh 8 analognih vratih, narejenih za pretvarjanje analognih signalov v digitalne in obratno. Ker je omogočeno napajanje od 2,7 V do 5,5 V, smo čip povezali s 3,3 V napajalno linijo. Z Raspberry Pi-jem pa smo vzpostavili komunikacijo preko 4 digitalnih vhodov in izhodov ter protokola serijske komunikacije SPI.



Slika 31: Integrirano vezje za pogon motorjev

(Osebni vir)

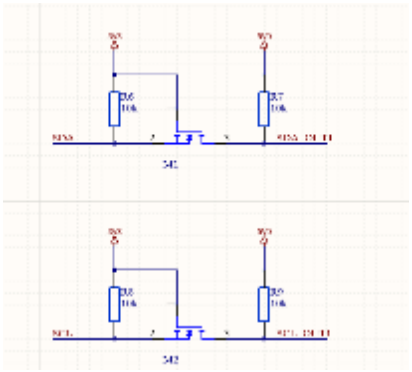
Za napajanje in usmerjanje motorjev na enosmerno napetost smo uporabili integrirano vezje DRV8848, proizvajalca Texas Instruments, ki v naši konfiguraciji zagotavlja priklop dveh takšnih motorjev, pri čemer pri 12 V vsakemu zagotavlja do 2 A toka. Logični del integriranega vezja smo povezali na nizkotokovno 5 V napajalno linijo, za napajanje motorjev pa smo uporabili 12 V visokotokovno napajalno linijo. Ker je komunikacijski protokol zasnovan na 5 V, smo za komunikacijo z mikroračunalnikom potrebovali vmesnik, ki med čipoma spremeni napetost. Za to smo uporabili čip TXB0108.



Slika 32: Obojesmerni regulator napetosti

(Osebni vir)

Integrirano vezje TXB0108, proizvajalca Texas Instruments, je 8-bitni obojesmerni regulator napetosti, ki dani komunikacijski povezavi omogoča povezavo naprav z različnima maksimalnima napetostma. V naši konfiguraciji omogoča povezavo mikroračunalnikovih digitalnih 3,3 V izhodov s prej opisanim integriranim vezjem DRV8848, ki deluje pri 5 V.



Slika 33: Regulator napetosti

(Osebni vir)

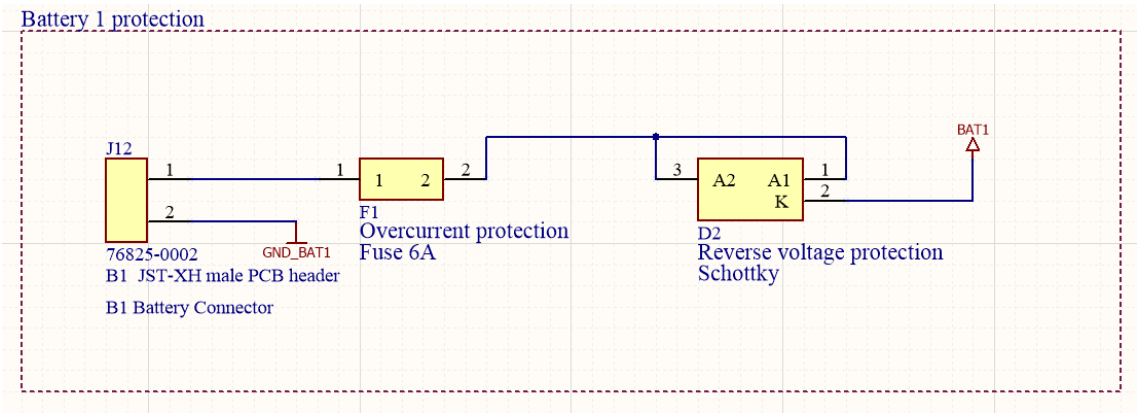
Slabost prej opisanega čipa TXB0108 je nezmožnost regulacije napetosti za komunikacijske linije s protokolom I2C, saj le-te vsebujejo upore za dvig napetosti z visoko upornostjo, kar onemogoči zmožnost, da čip zazna, katera stran linije je vhodna oz. daje signal. To določanje smeri signalov je nujno, saj brez njega čip ne more pravilno regulirati napetosti, zato smo za tak tip komunikacije, ki jo uporablja eden od senzorjev, uporabili drugačen način regulacije napetosti.
Za dve komunikacijski liniji, ki smo ju potrebovali za protokol I2C, smo uporabili dva mosfeta BSS138.

6.1.4.2 Napajalne komponente

Napajanje smo razdelili na 3 napajalne linije:

- 5 V napajalna linija,
- 3,3 V napajalna linija in
- 12 V napajalna linija.

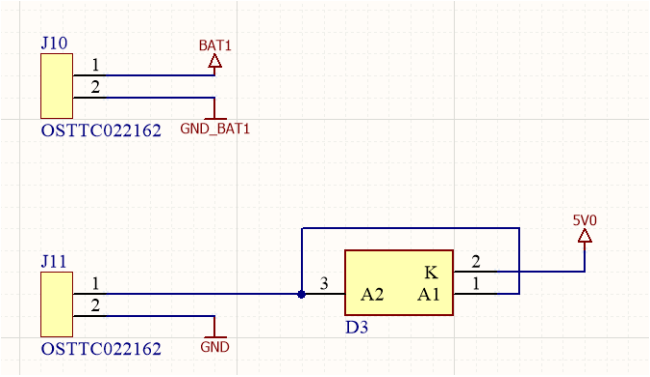
Prvi dve napajalni liniji sta namenjeni za napajanje logičnega dela vezja, tretja, 12 V linija pa je namenjena za napajanje aktuatorjev. Zaradi pogostih tokovnih špic, ki jih povzročijo servomotorji AX-18A, smo za napajanje aktuatorjev uporabili posebej baterijo, saj smo precenili, da bi takšne tokovne špice vplivale na padec napetosti na glavni bateriji in s tem onemogočale nemoteno delovanje logičnih komponent, kot je Raspberry Pi ipd.



Slika 34: Napajalno vezje 1

(Osebni vir)

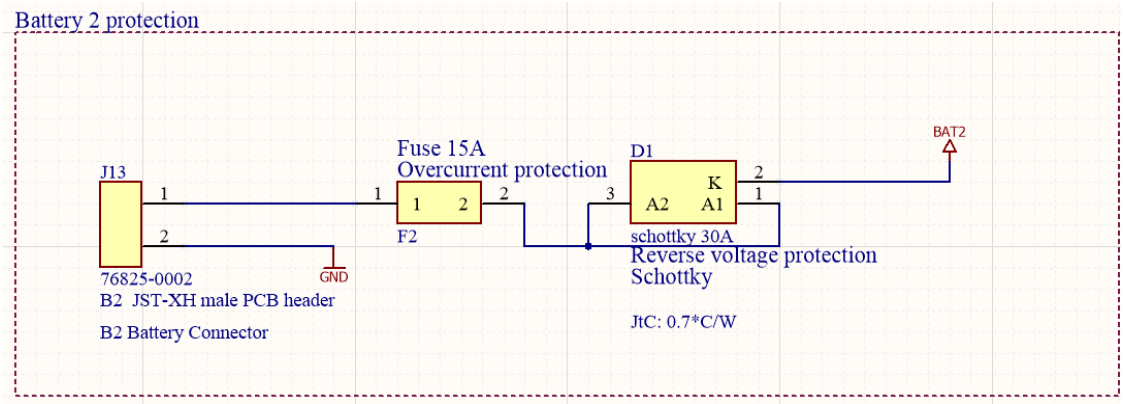
Za dodatno zavarovanje tiskanine in integriranih logičnih vezij v primeru napake smo na napajalno linijo dodali varovalko z maksimalnim tokom 6 A in schittky diodo, ki služi neprepuščanju toka v primeru napačnega priklopa baterije.



Slika 35: Zaščitno vezje

(Osebni vir)

Za uravnavanje napetosti smo se odločili uporabiti regulator CL6009 z zmožnostjo za zniževanje in zviševanje napetosti. Regulator omogoča 4 A izhodnega toka, kar zadošča za napajanje logičnega dela vezja. Za priklop regulatorja na vezje smo uporabili vijačne konektorje, ki smo jih v shemi povezali takoj za varovalko in schottky diodo. Ker priključitev regulatorja, v primeru napake uporabnika, dopušča nastanek negativnega napetostnega potenciala, smo za vhodnim konektorjem dodali dodatno schottky diodo, ki v prej opisanem primeru prekine tokokrog.



Slika 36: Napajalno vezje 2

(Osebni vir)

Varnostno vezje za 12 V napajalno linijo smo dimenzionirali podobno kot tisto za 5 V napajanje, z razliko, da smo pri tem uporabili 15 A varovalko in da pri tej napajalni liniji nismo uporabili regulatorja napetosti.

6.1.5 Aktuatorski člen

Slika 37: AX-18A motor

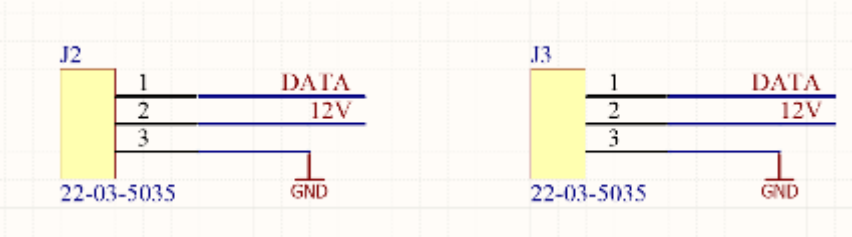
(Vir: https://www.trossenrobotics.com/dynamixel-ax-18A-robot-actuator.aspx)

Za pogon gosenic reševalnega robota smo uporabili servomotorje AX-18A, proizvajalca Dynamixel. Motorji delujejo na 12 V in od 50 do 2200 mA toka, pri čemer zagotavljajo vrtilno hitrost do 97 obratov na minuto in 1,8 Nm navora.

Za pogon koles avtonomnega robota smo uporabili podoben motor AX-12A, ki zagotavlja 59 obratov na minuto in 1,5 Nm navora. Slabši motor smo izbrali zaradi nižje cene in manjše porabe toka, kar nam omogoča daljši čas vožnje z eno baterijo. Dokazali smo tudi nižjo porabo toka, in sicer tako da smo motor napajali preko napajalnika, komunikacijo pa smo zagotovili preko mikroračunalnika Raspberry Pi 4. Motor smo nato obremenili s konstantno obremenitvijo in ga pri polnih obratih zagnali pri različnih vhodnih napetostih. Pri vsakem zagonu smo nato iz napajalnika razbrali maksimalni tok in vse skupaj zapisali v tabelo.

Vhodna napetost [V]	Vhodni tok [mA]
12	700
11	550
10	400
9	300

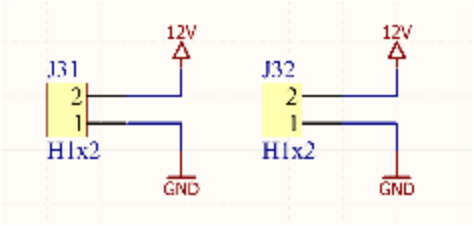
Tabela 3: Tok v primerjavi z napetostjo



Slika 38: Vezje za povezavo motorjev

(Osebni vir)

Motorje smo na vezje priključili s konektorji, preko katerih smo jim omogočili priključitev na 12 V napajalno linijo in komunikacijsko linijo. Komunikacijska linija je med konektorji povezana vzporedno, komunikacija z mikroračunalnikom pa je omogočena preko USB komunikacijskega vmesnika za serijsko komunikacijo, priključenega v enega od USB-vhodov mikroračunalnika.



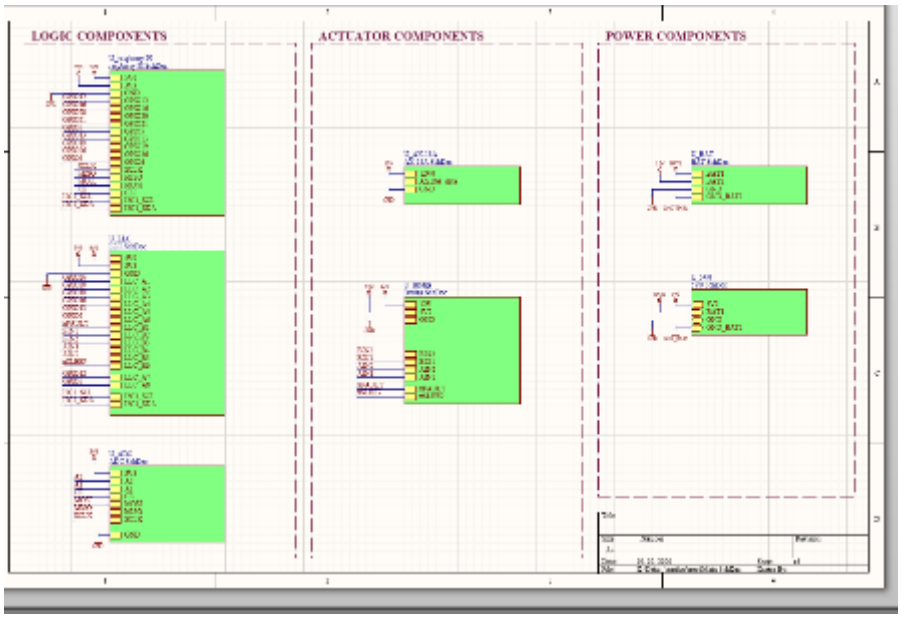
Slika 39: Vezje za povezavo ventilatorjev

(Osebni vir)

Za hlajenje mikroračunalnika Raspberry Pi smo uporabili 12 V ventilatorje z visokim pretokom zraka. Slednje smo na vezje priključili s pomočjo konektorjev, ki omogočajo priključitev na 12 V napajalno linijo.

6.1.6 Tiskano vezje

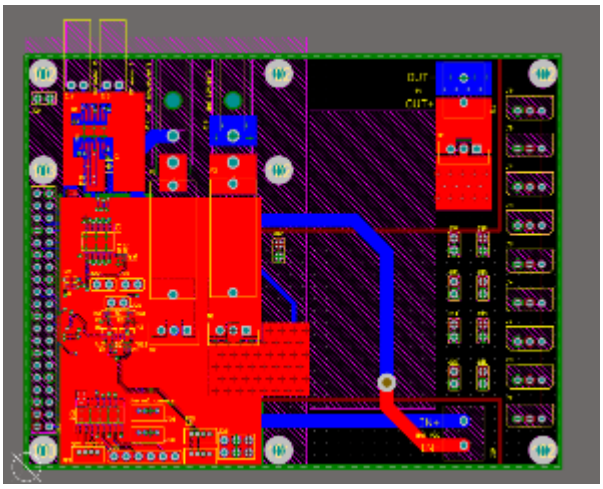
Za vsako integrirano vezje in pasivno komponento smo pri proizvajalcih elektronike, kot sta Texas Instruments in Microchip, poiskali ustrezne podatkovne liste, z njih razbrali parametre in na Altiumu kreirali 2D in 3D modele.



Slika 40: Glavna shema

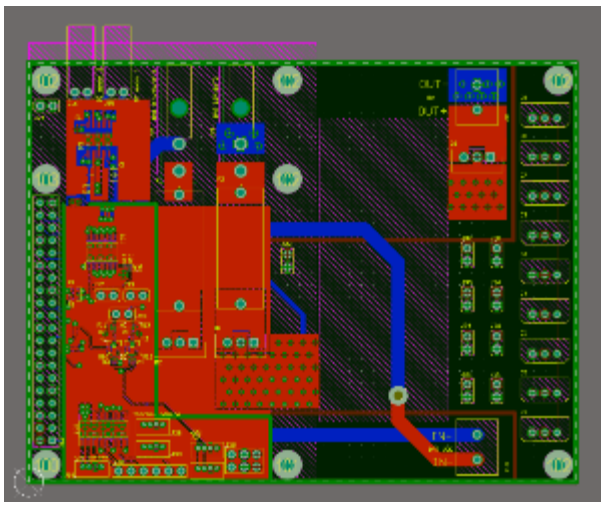
(Osebni vir)

Na koncu smo podsheme povezali v glavno shemo in na podlagi tega začeli s projektiranjem vezja. Najprej smo se osredotočili na to, da vezje naredimo manjše in bolj prilagojeno robotu. Da smo to dosegli, smo se odločili, da uporabimo štiriplastno tiskanino, ki omogoča večjo gostoto elektronskih komponent na vezju in s tem znatno zmanjša površino vezja. V programu smo vse komponente, definirane na shemi, uvozili na tiskanino in jih razporedili tako, da ustrezajo notranjosti robota.



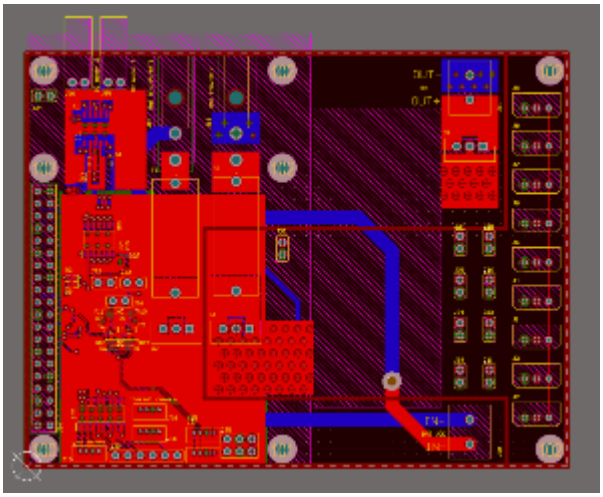
Slika 41: Prva plast tiskanega vezja

(Osebni vir)



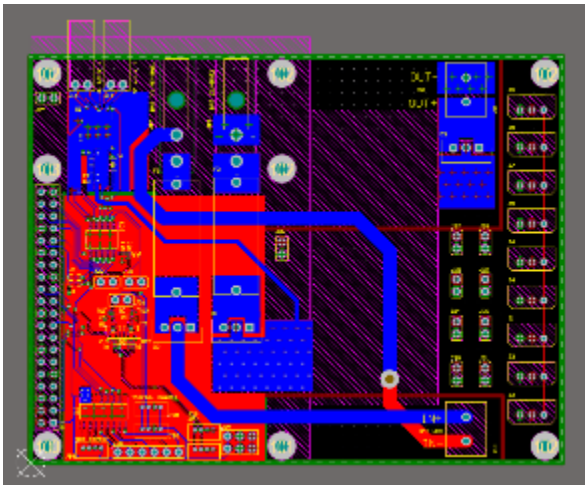
Slika 42: Druga plast tiskanega vezja

(Osebni vir)



Slika 43: Tretja plast tiskanega vezja

(Osebni vir)

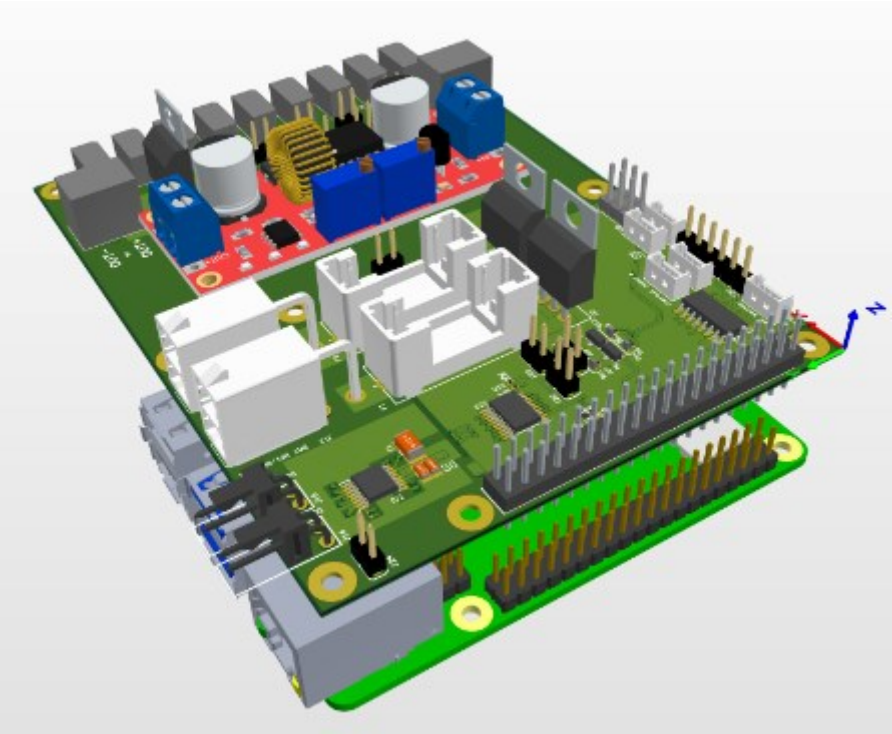


Slika 44: Četrta plast tiskanega vezja

(Osebni vir)

Zgornje slike prikazujejo vse štiri plasti tiskanega vezja. Na levo stran vezja smo postavili logični del vezja, na sredino komponente za napajanje, desno pa aktuatorski del. Raspberry Pi smo pozicionirali direktno pod vezje. Komponente na tiskanini smo nato ustrezno povezali, pri čemer smo si pomagali s shemami. Za preračun ustrezne debeline povezav smo uporabili brezplačni program Saturn PCB Design Toolkit Version 7.08.

Štiri plasti vezja smo uporabili tako, da smo prvo in četrto plast uporabili za povezave med komponentami, na drugo smo povezali zemljo oz. 0 V, tretjo pa smo uporabili za 5 V in 12 V povezave.



Slika 45: 3D pokaz dokončane tiskanine

(Osebni vir)

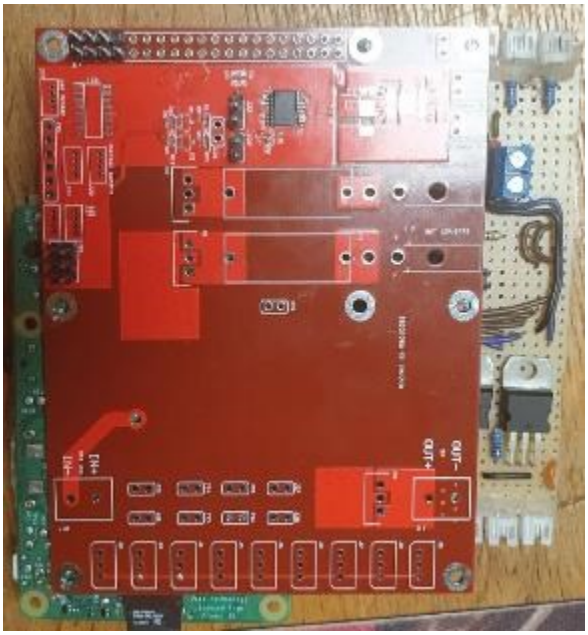
S pomočjo prej opisanih postopkov in komponent nam je uspelo doseči vse željene izboljšave elektronike, ki smo si jih zastavili takoj po njenem pregledu.

Z uporabo štiriplastne tiskanine in majhnih elektronskih komponent nam je uspelo ustvariti majhno tiskano vezje z dimenzijami 111 mm x 85 mm. Tiskano vezje poleg svoje majhnosti preprečuje nastanek nepredvidenih kratkih stikov.

Prostor, ki ga zavzema elektronika, smo zmanjšali tudi z menjavo oblike baterij. Uporabili smo namreč štirikotne in ovalne baterije, s katerimi smo učinkoviteje zapolnili prostor. Pregrevanje mikroračunalnika smo rešili z dodajanjem hladilne ploščice in nanjo pritrjenega ventilatorja.

6.1.7 Primerjava vezij

Velikost našega vezja smo primerjali z velikostjo lanskega in ugotovili, da smo s pomočjo tiskanega vezja uspešno zmanjšali njegovo velikost. Spodnja slika prikazuje letošnje in lansko vezje enega na drugem. Kot lahko vidimo, je letošnje vezje (rdeče) očitno manjše od lanskega.



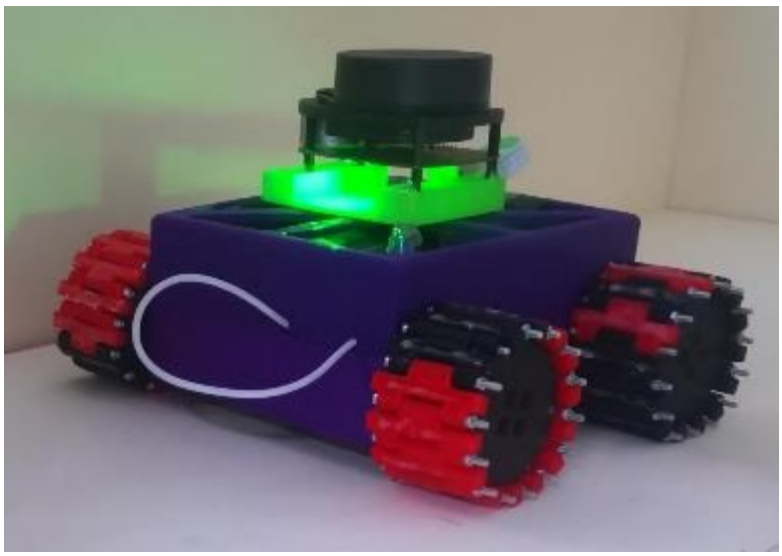
Slika 46: Primerjava vezij

(Osebni vir)

7 POLAVTONOMNI REŠEVALNI ROBOT

Naš tekmovalni robot zaradi svoje konstrukcije ne omogoča, da bi nanj postavili LIDAR, zato smo morali ustvariti novo konstrukcijo. Ker je ta robot samo prototip, smo ga zasnovali čim bolj preprosto. To nam je omogočilo hiter dostop do komponent in enostavno servisiranje.

V robotu so 3-celična 5000 mAh LI-PO baterija, tiskano vezje, Raspberry Pi 4 in AX-18A motorji. Nanj smo postavili LIDAR in kamero. To nam je omogočilo, da ima dva načina vožnje. Prvi način – vozi ga operater, ki spremlja dogajanje okoli robota s pomočjo kamere ali LIDAR-ja. Drugi način – robot sprejema podatke iz LIDAR-ja in se na podlagi teh sam odloča.



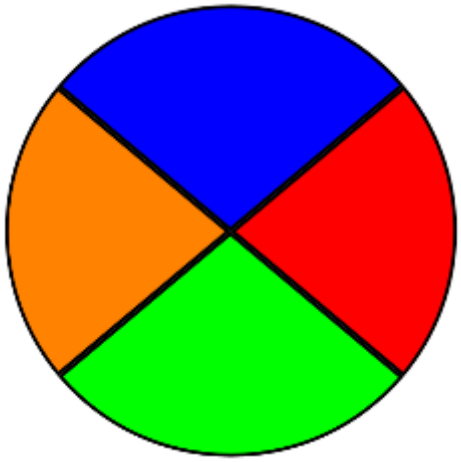
Slika 47: Polavtonomni robot
(Osebni vir)

8 VOŽNJA PO LABIRINTU

Čeprav je bil naš cilj, da robot prevozi enak labirint, kot ga bo na tekmovanju, smo hkrati želeli, da bo algoritem univerzalen. Hoteli smo doseči, da bi lahko robota postavili v katerokoli sobo in bi se še vedno lahko vozil po njej. Veliko enostavnejši in tudi hitrejši program bi lahko naredili, če bi robota krmilili (npr. ukazali, naj se najprej premakne naprej za določeno razdaljo, nato zavije levo, potem desno in cilj bo pred njim). Te pomike bi lahko računali preprosto preko rotacije koles in preko točk LIDAR-ja. Namesto tega smo si zadali cilj enostavnega, vendar univerzalnega programa.

Glavna težava pri tem je, da mora robot opravljati štiri naloge hkrati. Prva je vožnja, druga izračunavanje premika, tretja je sprejemanje podatkov LIDAR-ja za kasnejšo rekonstrukcijo in četrta zapisovanje ter shranjevanje podatkov.

Vedeli smo, da robot ne sme uporabiti preveč časa pri algoritmu za vožnjo, ker bi to preprečilo pravočasno izvajanje ostalih treh funkcij. Tako smo prišli do ideje štetja točk. V programu smo narisali krog s polmerom 300 mm in ga razdelili na kvadrante.

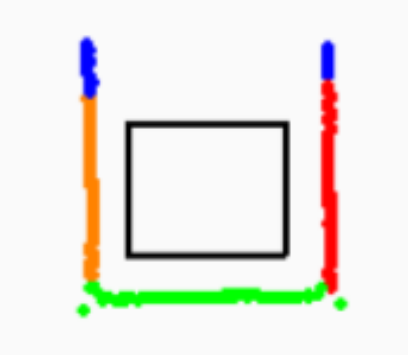


Slika 48: Kvadranti okoli robota

(Osebni vir)

Modri krožni izsek predstavlja območje pred robotom in meri 100° ter je skladen z območjem, obarvanim z zeleno barvo. Rdeči in oranžni izsek pa imata kot 80° ter predstavljata stranici robota.

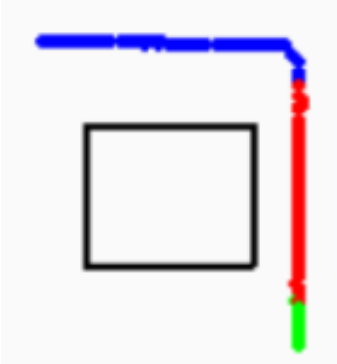
Med vožnjo robot spremlja le dogajanje znotraj modrega kvadranta. Ko v njem zazna oviro oziroma število točk v izseku, prekorači neko mejo, se ustavi. Takrat razširi svoj polmer na 6000 mm in preveri tudi stranska kvadranta, ki ju skrči na 10°. Nato ugotovi, kateri izmed njiju pokriva največjo zračno razdaljo do naslednje ovire oziroma kateri kvadrant mora več raziskati, nakar se obrne v smer izbranega kvadranta. Če je v obeh stranskih kvadrantih največja razdalja pod določeno mejo (npr. 200 mm), pomeni, da je na njegovih stranicah stena in se nato obrne za 180° ter nadaljuje vožnjo.



Slika 49: Robot v začetnem položaju

(Osebni vir)

Ko je robot v areni na začetni poziciji, zaznava okolje, kot prikazuje slika 49. Ker vidi, da je modrih pik premalo, da bi jih štel kot oviro, se odloči, da bo vozil naravnost.

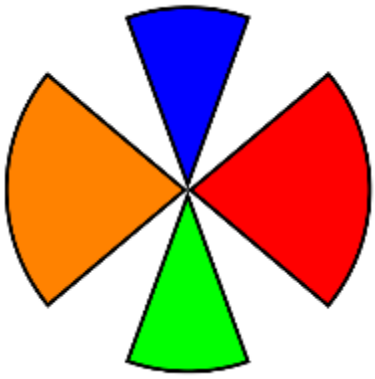


Slika 50: Robot v prvem kotu arene

(Osebni vir)

Takoj ko prešteje dovolj modrih pik, se ustavi in pogleda stranska kvadranta. S slike 50 je razvidno, da je na njegovi desni stena, tako da se odloči za rotacijo v levo. Po takšni logiki robot nadaljuje vožnjo, dokler ne izpolni pogojev za konec programa (npr. 25 meritev okolja).

Trenutni koti kvadrantov in število točk, potrebnih za zaznavanje ovire, so nastavljeni za vožnjo po areni. Ker smo vedeli, da se ne more zaleteti v tanek objekt (npr. nogo stola), smo mejo števila točk nastavili precej visoko. Hkrati smo tudi želeli, da stranska kvadranta ne zaznata stene pred robotom, zato smo njun kot nastavili na 80°. Če bi želeli, da se robot vozi po sobi, kjer je velika možnost, da nastopi tanka ovira, bi lahko zmanjšali mejo in kot na primernejši vrednosti, tako da bi slika kvadrantov okrog robota izgledala približno takole.



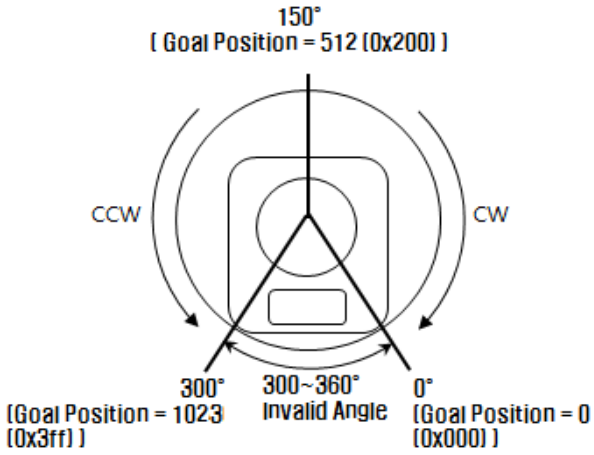
Slika 51: Kvadranti za vožnjo po sobi
(Osebni vir)

Ker je trenutno med kvadranti prostor, kjer ne zaznava ovir, bi lahko dodali tudi krožne izseke, povečali stranska kota ali pa jih preprosto pustili takšne, kot so. Nismo pa omejeni le na krožne izseke. Oblika polj, v katerih šteje točke, je lahko kakršnakoli. Pri tem pa lahko nastopi težava, da mora robot za vsako točko narediti veliko preračunov, da ugotovi, v katerem polju je, kar bi samo še bolj upočasnilo program.

8.1 ODOMETRIJA IN POZICIONIRANJE ROBOTA PREKO ROTACIJE MOTORJA

8.1.1 Merjenje premika robota

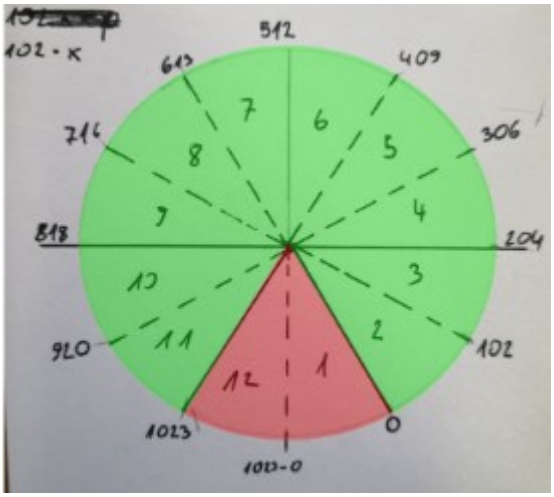
Odločili smo se, da bomo za pogon robota uporabili Dynamixel AX-18A motorje zaradi njihove enostavne uporabe in sorazmerno velike moči. Težava pa je nastopila, ko smo z njimi želeli meriti premik robota. Namesto enkoderja uporabljajo za pozicioniranje potenciometer, ki pa je sposoben konstantnega vrtenja. Motorji merijo trenutni položaj le v krožnem izseku s kotom 300°. Tako nam ostane "mrtvih" 60°, v katerih nismo sposobni dobiti trenutnega položaja.



Slika 52: Delovanje AX-18A motorja

(Vir: http://support.robotis.com/en/techsupport_eng.htm#product/dynamixel/ax_series/dxl_ax_actuator.htm)

Ugodno pa je, motor deli ta "mrtvi kot" na dve enaki polovici. Kadar zaprosimo za položaj, nam AX-18 vrne število 0–1023 (0°–300°). Če pa je trenutni položaj večji od 300°, a manjši ali enak od 30°, nam ravno tako vrne vrednost 1023. Takoj, ko je velikost kota večja od 330°, nam poda vrednost 0.



Slika 53: Sektorji motorja

(Osebni vir)

Tako lahko razdelimo krog na 12 skladnih delov. To nam omogoči, da lahko v vsakem trenutku ugotovimo položaj koles in smo prepričani, da odstopanje ne bo večje kot 30°. Kar pomeni, da je pri kolesih s premerom 75 mm največja napaka manjša od 20 mm.

8.1.2 Izračun premika

Robot med vožnjo po labirintu ne potrebuje konstantnega toka informacij o trenutnem položaju. To potrebuje samo med zapisovanjem lege, kadar slika okolico z LIDAR-jem in med obračanjem. Premer koles na robotu je 75 mm, kar pomeni, da je obseg približno 235 mm. Torej, če želimo ugotoviti, za koliko se je robot premaknil, vzamemo trenutne stopinje in jih pomnožimo z 0.65 mm/°. To število dobimo, če 235 mm delimo z 360°.

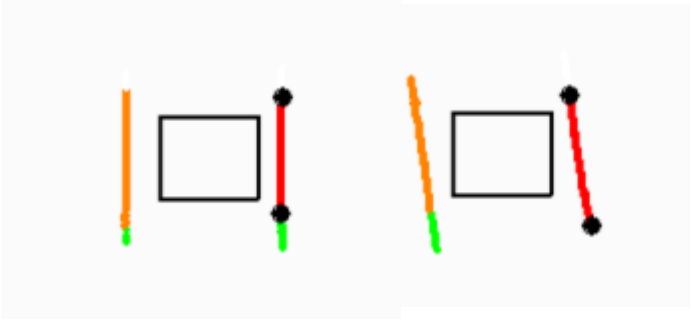
Program za izračun stopinj samo nadzoruje, v katerem sektorju je trenutno motor in v katerem kvadrantu je bil prejšnjikrat, ko je pogledal trenutno pozicijo. Ti dve vrednosti primerja in se odloči, kako bo spremenil trenutne stopinje. Če sta poziciji enaki, ostanejo tudi stopinje enake, če pa sta poziciji različni, program prišteje ali odšteje 30° glede na to, ali se je sektor premaknil v smeri urinega kazalca ali v nasprotni smeri.

Kot smo opisali, ima naš podprogram za zaznavanje trenutne pozicije motorja resolucijo 30°. Zato smo se odločili, da se bo robot obračal samo pod pravim kotom. Takšen premik je precej enostavnejši, hkrati pa tudi zmanjša napako pri računanju pozicije in pri rekonstrukciji arene. Če ima robot samo pravokotne premike, smo lahko dokaj prepričani, da bosta njegov trenutni kot in položaj pravilna. Če bi želeli računati vmesne kote, bi jih morali izračunati preko primerjave rotacije leve in desne strani. Ker pa je največja napaka lahko 20 mm in bi se sčasoma samo še stopnjevala, bi imeli zemljevidi veliko več napak in motenj.

8.2 REGULACIJA PREKO STRANICE

Opazili smo, da s časom motorji nimajo več enake moči. To je posledica padca napetosti baterije. Robot začne vse bolj zavijati v eno smer, odvisno od položaja težišča. Zato smo naredili regulacijo moči motorjev glede na naklonski kot stranic. S tem želimo doseči, da robot vozi čim bolj vzporedno s stenami arene.

Za izračun kota uporabimo stranska kvadranta. Če je v kvadrantu dovolj točk, vzamemo prvo in zadnjo točko ter preko njiju izračunamo koeficient premice, ki ju povezuje. To število nam predstavlja naklonski kot premice.



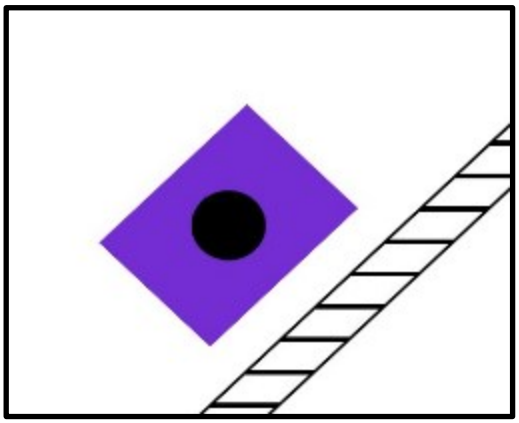
Slika 54: LIDAR slika s poudarjenima pravima in zadnjima točkama

(Osebni vir)

$$k = \frac{\Delta y}{\Delta x}$$

Tako smo “k” vstavili v preprosti P-regulator. S to regulacijo je imela izguba moči veliko manjši vpliv na delovanje robota.

Sistem v areni deluje, kot smo želeli, in pomaga pri vožnji, vendar v realni situaciji ni uporaben. Robot gleda naklonski kot stranice in ker vemo, da so vse stene arene pravokotne druga na drugo, ta sistem lahko uporabimo, ker želimo, da se robot premika samo pod pravim kotom. Če pa bi bile stene pod katerimkoli drugim kotom, bi robot nadaljeval vožnjo vzporedno s stenami in ne "naravnost", kot bi želeli.



Slika 55: Delovanje algoritma ob poševni steni

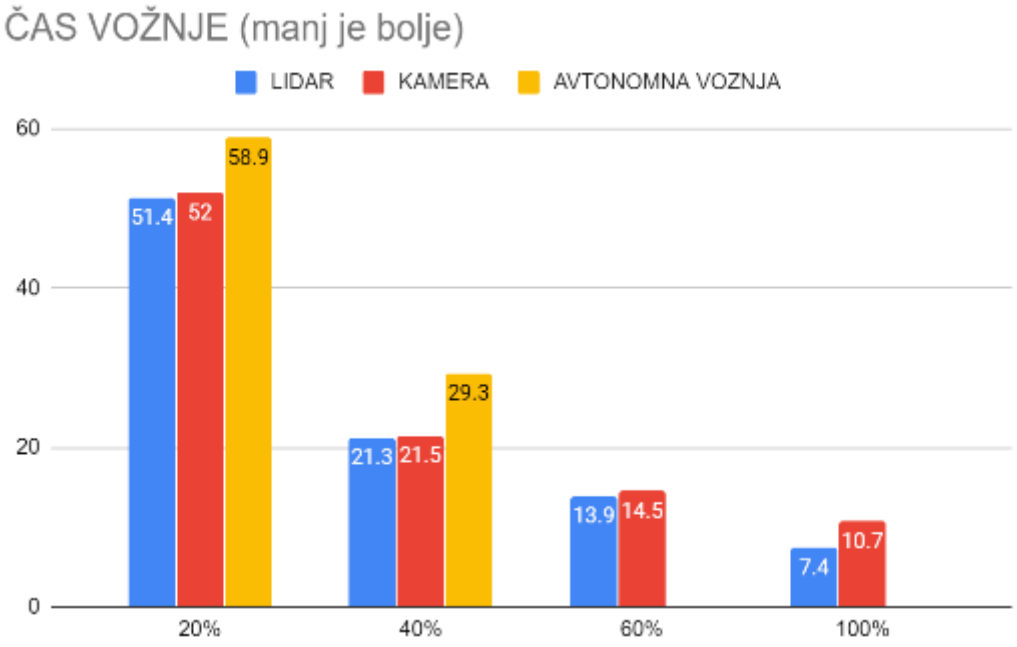
(Osebni vir)

8.3 ZAJEMANJE IN SHRANJEVANJE PODATKOV

Ker ima LIDAR 12 m dometa, nam omogoča, da zajamemo podatke za rekonstrukcijo na dokaj redkih intervalih in s tem zmanjšamo napake oziroma motnje. Tako smo se odločili, da bomo zajemali podatke le na vsakih 118 mm (to je razdalja, ki jo robot prepotuje, ko se kolesa zavrtijo za 180°). Če merimo na takšne intervale, ne izgubimo preveč detajlov, hkrati pa tudi ne zapravljamo čas z beleženjem podatkov. Ko LIDAR zajame podatke, jih shrani v JSON (JavaScript Object Notation) datoteko. To nam omogoči, da podatke hitro prekopiramo na drug računalnik, kjer jih lahko rekonstruiramo. Poleg podatkov LIDAR-ja si robot shrani tudi trenutno razdaljo od začetne pozicije in trenutno orientacijo.

8.4 REZULTATI VOŽNJE

Zanimalo nas je, ali bi lahko trenutni sistem uporabili na tekmovanju, zato smo ga na različnih hitrostih primerjali z vožnjo preko kamere in vožnjo preko LIDAR-ja. Hitrost smo nastavili na 20 %, 40 %, 60 % in 100 % maksimalne hitrosti. Z vsakim načinom vožnje smo naredili 10 meritev in iz njih naredili povprečje.



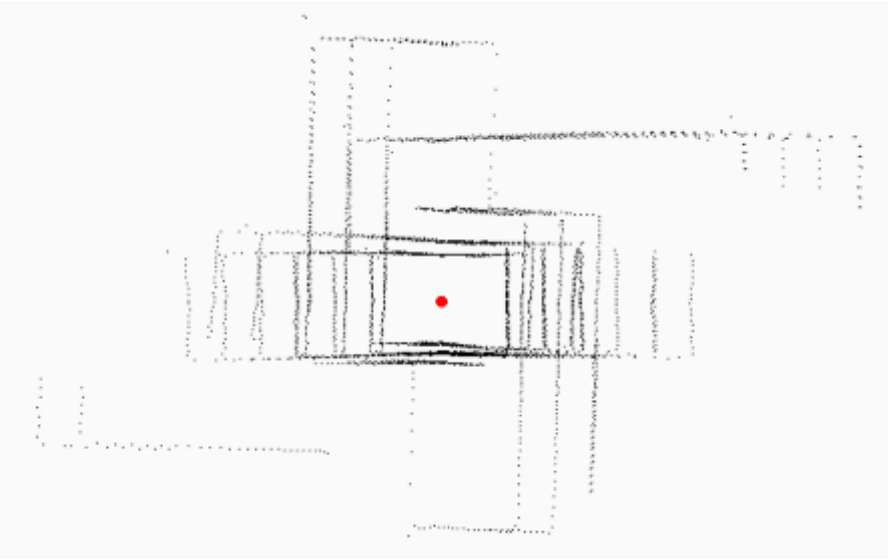
Graf 2: Časi vožnje

Ugotovili smo, da sistem ni uporaben pri večjih hitrostih. Robot nima dovolj časa, da reagira na stene, tako da se jim ni sposoben izogniti. Pri manjših hitrostih pa so bili časi avtonomne vožnje primerljivi. To nam daje upanje, da bi lahko v prihodnosti z boljšimi algoritmi uspeli prevoziti areno hitreje kot človek. Opazili smo tudi, da je vožnja preko LIDAR-ja hitrejša od vožnje preko kamere. Čeprav nam kamera omogoči, da vidimo veliko več detajlov, je časovni zamik prevelik. Vedno, ko zajame sliko, jo mora nato prikazati še na zaslonu in jo poslati na drug računalnik. LIDAR pa mora narisati le preprosto skico okolja, zato je zamik zaradi tega manjši.

9 REKONSTRUKCIJA ZEMLJEVIDA

Program za rekonstrukcijo dobi podatke v JSON-datoteki. Ta format ima podatke urejene tako, da jih najprej imenuje, nato navede (“ime_podatkov”:podatki). Zapis informacij v takšni obliki nam omogoči enostaven prenos in enostavno obdelovanje.

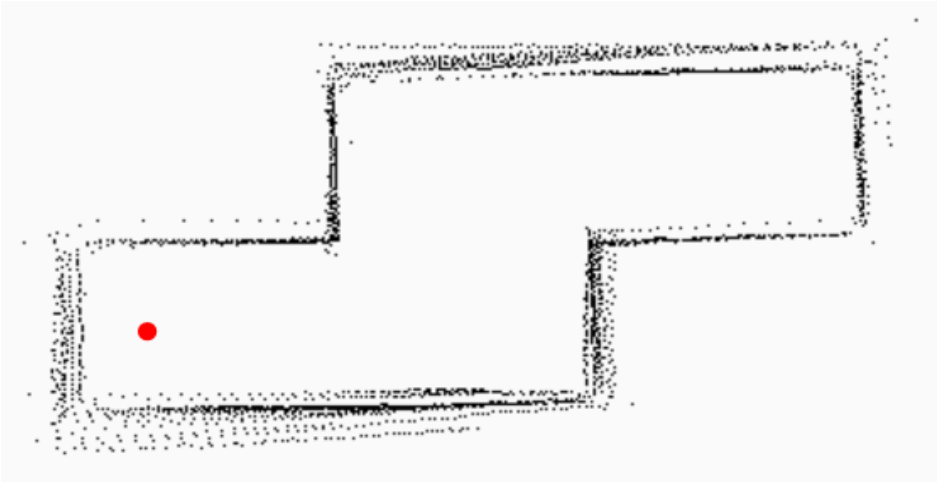
Ko program odpre datoteko, spremeni podatke, ki jih je podal LIDAR v točke. Senzor vedno shrani trenutni kot in razdaljo, ki jo je takrat izmeril. Iz teh dveh spremenljivk lahko preko kotnih funkcij izračunamo koordinate točke. Če to naredimo za vse točke, dobimo spodnjo sliko.



Slika 56: Točke LIDAR-ja brez upoštevanja pomika

(Osebni vir)

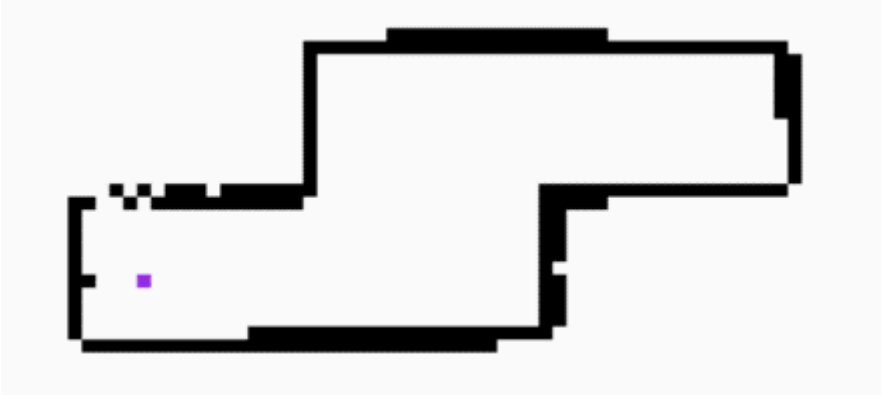
Rdeča točka predstavlja izhodišče. S slike lahko vidimo približno obliko tekmovalne arene. Vendar točk še nismo zamaknili in s tem upoštevali premik robota. Hkrati pa jih je potrebno še zavreti glede na orientacijo robota.



Slika 57: Točke LIDAR-ja z upoštevanjem premika

(Osebni vir)

S spodnje slike je oblika arene že precej očitna, a vseeno ima veliko motenj, zato smo dodali še dva filtra. Prvi izbriše vse točke, ki so bile od robota oddaljene za več kot 500 mm. Drugi pa na sliko postavi mrežo s kvadrati, ki so veliki 6 × 6 pikslov. Nato pogleda, ali ima posamezni kvadrat 3 točke ali več. Če jih ima, obarva kvadrat s črno, drugače pa pusti belega. Tako nastane takšna slika.



Slika 58: Točke LIDAR-ja z upoštevanjem premika in filtrom

(Osebni vir)

Ker smo želeli zmanjšati število motenj, smo hkrati tudi izbrisali manjši del arene, vendar je oblika še vedno jasna. Če pa bi nas to motilo, bi lahko spremenili parametre filtrov (npr. 4 × 4 kvadrati in maksimalna dolžina 600 mm).

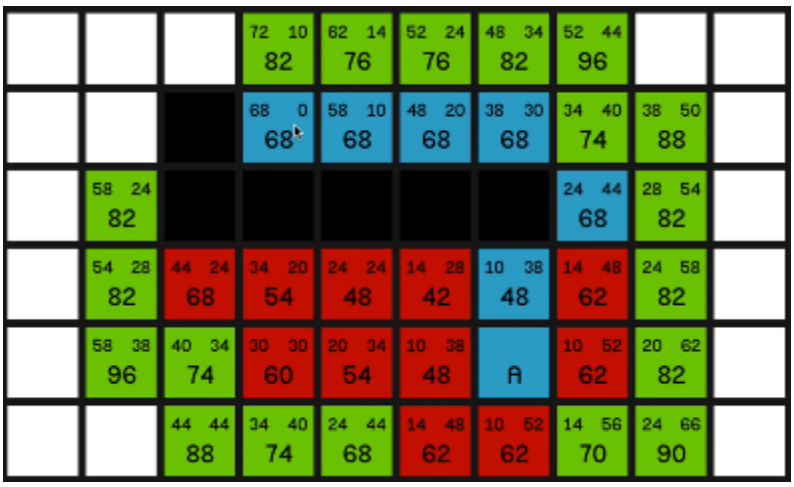
9.1 ODSTOPANJE ZEMLJEVIDA

Za potrditev tretje hipoteze smo z robotom naredili 10 rekonstrukcij in izračunali površino labirinta. Površina tekmovalne arene je 0,96 m^2. Merili smo stranice in preko njih izračunali 10 površin, izračunali njihove relativne napake ter nato iz njih naredili povprečno napako. Tako smo dobili povprečno relativno napako 3.04 %, ki je manjša od naše tolerance. Če pa izračunamo povprečno površino, dobimo 0.9522 m^2.

10 POZICIONIRANJE ROBOTA

Ko je bila konstrukcija zemljevida dokončana, nam je preostalo le še pozicioniranje robota. Želeli smo mu postaviti neko končno točko, do katere bi lahko prišel sam. Težava pa je nastopila pri izdelavi algoritma. Naša prva ideja je bila, da bi uporabili neko poenostavljeno različico A* algoritma.

V osnovi ta algoritem išče najkrajšo razdaljo med dvema točkama. Zemljevid razdeli na posamezne kvadrate in za vsakega izračuna razdaljo med izhodiščem in koncem ter iz teh dveh razdalj dobi vrednost. Manjša kot je vrednost, hitrejša je pot.



Slika 59: A algoritem*

(Vir: https://www.youtube.com/watch?v=-L-WgKMFuhE)

V teoriji bi takšen algoritem lahko hitro našel pot do katerekoli točke, bi želeli. V praksi pa ne upošteva velikosti robota. Če bi želeli uporabiti ta algoritem, bi morali biti kvadrati, za katere program izračuna vrednosti, tako veliki kot robot. Če bi bili manjši in če bi bila idealna pot ob steni, bi se robot vanjo zaletel. V primeru, da bi napisali program na tak način, bi izgubili natančnost pozicioniranja, zato smo morali poiskati drugo rešitev.

Tako smo prišli do ideje, da bi robotu podali tudi vmesne točke. Ključ te rešitve se skriva v tem, da bi se točki razlikovali samo po eni koordinati. Tako bi se robot premikal od točke do točke, kar bi tudi zagotovilo, da ima pravokotne premike, kot smo jih že vajeni. Izračunati razdalje med

točkami je relativno enostavno in hitro jih lahko spremenimo v rotacije motorja. Te rotacije pa potem vnesemo v robota.

Slika 60: Točke za pozicioniranje

(Osebni vir)

Vijolični točki na zgornji sliki predstavljata začetek in konec, rumena pa vmesno. Program najprej izračuna pot med začetno in rumeno točko ter to razdaljo spremeni v rotacije. Poleg tega zapiše tudi orientacijo robota. Začetna orientacija je vedno naravnost. Nato izračuna pot med vmesno in končno točko. Hkrati program preko njunih koordinat ugotovi, da se mora robot v rumeni točki zavrteti za 90° in šele potem nadaljevati pot.

Poleg načrtovanja poti lahko ta program uporabimo za merjenje razdalj. Vedno ko označimo novo točko, nam program poda razdaljo do prejšne.

10.1 PREMIK

Ko program izračuna razdalje med točkami in orientacijo robota, nam izpiše vrstico z ukazi. Vrstica nam poda podatke, ki robotu povedo, koliko stopinj rotacije motorja je oddaljen od naslednje točke in v kateri orientaciji mora biti obrnjen.

Pri sliki 60 nam program poda naslednje podatke:

[[1110.0, 0], [690.0, 1]]

Oglati oklepaj najprej oklepa vse podatke, nato pa še posamezne ukaze do naslednje točke. V prvem notranjem oklepaju nam število 1110 pove stopinje do prve točke, 0 pa orientacijo. Tako program na robotu po vrsti bere oklepaje in se premika od točke do točke.

10.2 ODSTOPANJA PREMIKA

Za potrditev četrte hipoteze smo primerjali dejanske premike robota in izračunane premike v programu za rekonstrukcijo. Najprej smo robota poslali po areni, tako da je lahko ustvaril zemljevid. Nato smo petkrat določili končne točke. Enkrat ni bilo pomožnih točk, dvakrat je bila ena in dvakrat sta bili dve pomožni točki.

Slika 61: Primer točk za eno meritev

(Osebni vir)

Nato smo merili odstopanja po posamezni osi in razdaljo med začetno in končno točko. Izračunali smo relativne napake in ta postopek ponovili desetkrat, tako da smo imeli na koncu 10 različnih rekonstrukcij in 50 končnih točk. Izračunali smo povprečno absolutno napako za vse tri meritve in dobili naslednje rezultate.

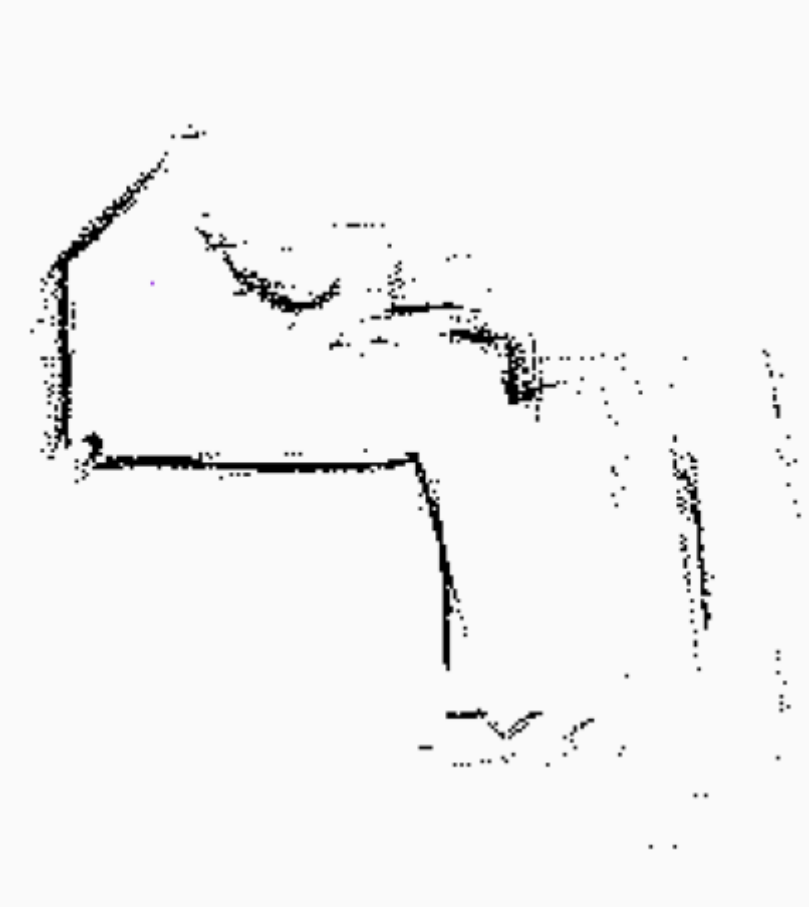
	X-os	Y-os	Razdalja med točkama
Relativna napaka	3.75 %	4.41 %	3.50 %

Tabela 4: Napake pozicioniranja

Iz rezultatov je razvidno, da je naš sistem relativno natančen. Čeprav je vmes odstopanje preseglo tudi 20 % (pri izmerjeni razdalji 10 cm in izračunani 13 cm) so vse tri povprečne relativne napake pod 5 %.

11 VOŽNJA PO SOBI

Na koncu našega raziskovanja smo se odločili, da robot vozi po sobi. Ni nas zanimalo, kako hitro bo prevozil sobo, le kako bo izgledala slika, ki jo bo narisal. Tako smo mu dali nalogo, naj naredi 25 meritev, ki jih bo zabeležil in se nato ustavil. Tako smo dobili spodnjo sliko sobe.



Slika 62: Zemljevid sobe

(Osebni vir)

Na sliki so razvidni določeni predmeti, vendar jih ne moremo natančno določiti. Tako smo dobljeno sliko postavili preko dejanske slike sobe.

Slika 63: Primerjava zemljevida z okoljem

(Osebni vir)

Čeprav so s tega zornega kota določeni objekti in proporci popačeni, je na sliki precej očitno, kaj določene lise predstavljajo. Torej, če bi s to metodo želeli približno oceniti, kako izgleda tloris sobe, bi bili uspešni.

Dodati moramo, da robot ni sposoben zaznati predmetov, ki ležijo nižje kakor LIDAR. To pomeni, da bi lahko robot nasedel ali se zamaknil in slika bi postala popačena.

12 PREDSTAVITEV REZULTATOV

Na osnovi rezultatov naših raziskovanj lahko štiri hipoteze potrdimo in eno zavržemo.

H1 – Zaradi optimizacije topologije se bo potencialna energija pri padcu robota zmanjšala za 10 %.	✗
H2 – Robot bo sam sposoben prevoziti osnovno tekmovalno areno.	✓
H3 – Robot bo sposoben narediti zemljevid arene z maksimalno 5 % odstopanjem.	✓
H4 – Robot bo sposoben priti do določene točke v prej narejenem zemljevidu in odstopanje ne bo večje od 5 %.	✓
H5 – Z uporabo tiskanega vezja bomo dosegli, da bo letošnje vezje manjše od lanskega.	✓

H1 – Zaradi optimizacije topologije se bo potencialna energija pri padcu robota zmanjšala za 10 %.

Masa po optimizaciji je bila manjša le za 56 g, kar je pomenilo, da se potencialna energija ni zmanjšala za 10 %, zato smo hipotezo ovrgli.

H2 – Robot bo sam sposoben prevoziti osnovno tekmovalno areno.

Največja težava pri programiranju vožnje je bila narediti program dovolj hiter. Poleg same vožnje mora robot opravljati še druge funkcije. Uspelo nam je narediti program za vožnjo po tekmovalni areni, zato lahko to hipotezo potrdimo.

H3 – Robot bo sposoben narediti zemljevid arene z maksimalno 5 % odstopanjem.

Pri tej nalogi se niso pojavile večje težave, zato smo lahko naredili natančen program za rekonstrukcjo. Hipotezo smo potrdili.

H4 – Robot bo sposoben priti do določene točke v prej narejenem zemljevidu in odstopanje ne bo večje od 5 %.

Skupaj s programom za rekonstrukcijo smo naredili tudi program za pozicioniranje. Ko se je robot premaknil, je bila njegova pozicija znotraj tolerance, tako da lahko potrdimo tudi to hipotezo.

H5 – Z uporabo tiskanega vezja bomo dosegli, da bo letošnje vezje manjše od lanskega.

Z uporabo tiskanega vezja nam je uspelo narediti vezje manjše, saj tiskanina omogoča, da elektronske komponente pozicioniramo bližje drugo drugi. Poleg tega smo z vezjem zmanjšali tudi verjetnost nastanka kratkega stika in posledično nepričakovanih napak. Hipotezo smo potrdili.

13 MOŽNOST NADALJNJEGA RAZISKOVANJA

Kot smo zapisali na začetku raziskovalne naloge, naš cilj ni bil narediti najhitrejšega ali tudi najnatančnejšega robota. Želeli smo le predstaviti naše ideje za reševanje problemov pri avtonomiji. Tako smo našli še veliko možnosti za nadaljnje raziskovanje. Po našem mnenju bi lahko z boljšim načinom merjenja lege dosegli boljše rezultate. Čeprav naš robot prevozi tekmovalno areno, ima z vožnjo po sobi še vedno težave. Le-te izvirajo iz dejstva, da robotu drastično pade natančnost, ko premiki niso več pravokotni. Tako bi lahko z boljšim načinom določanja lege izboljšali vožnjo, rekonstrukcijo in pozicioniranje. Med delom smo prišli tudi do ideje, da bi robot med vožnjo konstantno obdeloval sliko. Lahko bi iskal mesta požarov, ljudi in to označil na zemljevidu. Vendar bi za to potrebovali grafično močnejši mikroračunalnik, saj Raspberry Pi 4 ni dovolj zmogljiv. V nadaljevanju bi na robotu lahko uporabili tudi strojni vid in zmogljivejši mikroračunalnik.

14 ZAKLJUČEK

Raziskovalna naloga nam je omogočila, da nadgradimo svoje znanje na področjih elektrotehnike, CAD-modeliranja in programiranja. Naučili smo se načrtovanja elektronike, delovanja nekaterih elektronskih komponent in pridobili znanje iz osnov elektrotehnike. Z razvojem tiskanega vezja smo pridobili znanje in izkušnje pri uporabi profesionalne programske opreme za razvoj elektronike, Altium Designer, se naučili razvoja elektronskih shem, računalniških modelov elektronskih komponent, postavitve komponent na tiskano vezje in povezave elektronskih komponent na tiskanem vezju. Pri modeliranju smo spoznali novo funkcijo in postali boljši v izdelovanju CAD-modelov. Nadgradili smo tudi naše znanje programiranja in se naučili uporabe matematike za reševanje realnih problemov pri programiranju.

Vsa novo pridobljena znanja smo združili s problemskim in sodelovalnim učenjem ter ekipnim delom in s tem izboljšali robota, narejenega preteklo leto. Kljub uspešni posodobitvi slednjega pa smo sprejeli tudi določene pomanjkljivosti robota in s tem ustvarili možnost za nadaljnje raziskovanje.

15 VIRI

[1] Our World in Data (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://ourworldindata.org/natural-disasters

[2] ARSO (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://www.arso.gov.si/varstvo%20okolja/poro%C4%8Dila/poro%C4%8Dila%20o%20stanju%20okolja%20v%20Sloveniji/nesrece.pdf

[3] FRIDMAN L. MIT Autonomous Vehicle Technology Study (online). 2017. (citirano 20. 2. 2020). Dostopno na naslovu: https://www.researchgate.net/profile/Bryan_Reimer/publication/321180547_MIT_Advanced_Vehicle_Technology_Study_Large-Scale_Naturalistic_Driving_Study_of_Driver_Behavior_and_Interaction_with_Automation/links/5a16d6e0aca272df0808830e/MIT-Advanced-Vehicle-Technology-Study-Large-Scale-Naturalistic-Driving-Study-of-Driver-Behavior-and-Interaction-with-Automation.pdf.

[4] DARPA Grand Challenge (2007) (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2007)

[5] Nvidia (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://blogs.nvidia.com/blog/2019/04/15/how-does-a-self-driving-car-see/

[6] Lidar (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://en.wikipedia.org/wiki/Lidar

[7] Crowe S. Researchers back Tesla's non-LiDAR approach to self-driving cars (online). 2019. (citirano 20. 2. 2020). Dostopno na naslovu: https://www.therobotreport.com/researchers-back-teslas-non-lidar-approach-to-self-driving-cars/

[8] Simultaneous localization and mapping (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping#Mathematical_description_of_the_problem

[9] Odometry (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://en.wikipedia.org/wiki/Odometry

[10] Printed circuit board (online). (citirano 20. 2. 2020). Dostopno na naslovu: https://en.wikipedia.org/wiki/Printed_circuit_board]