Šolski center Celje

Srednja šola za strojništvo, mehatroniko in medije

STROJNO UČENJE REŠEVALNEGA ROBOTA

Raziskovalna naloga

Področje: Elektrotehnika, elektronika ali robotika

Avtorji:

Blaž Gril, M-4. c

Jernej Romih, M-4. c

Blaž Strajnič, M-4. c

Mentorji:

dr. Matej Veber, univ. dipl. inž.

mag. Andro Glamnik, univ. dipl. inž.

Celje, april 2024

IZJAVA*

Mentorja Andro Glamnik in Matej Veber, v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljama, da je v raziskovalni nalogi z naslovom Strojno učenje reševalnega robota, katere avtorji so Blaž Strajnič, Jernej Romih, Blaž Gril:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 4.4.2024 žig šole Podpis mentorjev
Andro Glamnik
Matej Veber

Podpis odgovorne osebe

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

POVZETEK

Ključne besede: robot, izris, mikroračunalnik, strojno učenje in 3D-tisk.

V okviru raziskovalne naloge smo se s pomočjo strojnega učenja osredotočili na prepoznavanje znakov za nevarnost. V uvodu smo opredelili cilje, hipoteze in metode raziskovanja, zatem pa je sledil izris robota, pri čemer smo uporabili program Creo Parametric za oblikovanje njegovih komponent. V nadaljevanju smo opisali električno vezavo, kamere, motorje, mikroračunalnik in postopek 3D-tiskanja. V poglavju Programi smo se osredotočili na strojno učenje, zaznavo znakov za nevarnost, zaznavanje barv, nadzor robota s kontrolerjem in uporabo QR kod. V zaključku smo analizirali hipoteze in razpravljali o možnostih nadaljnjega razvoja.

SUMMARY

Keywords: robot, design, microcomputer, machine learning and 3D printing.

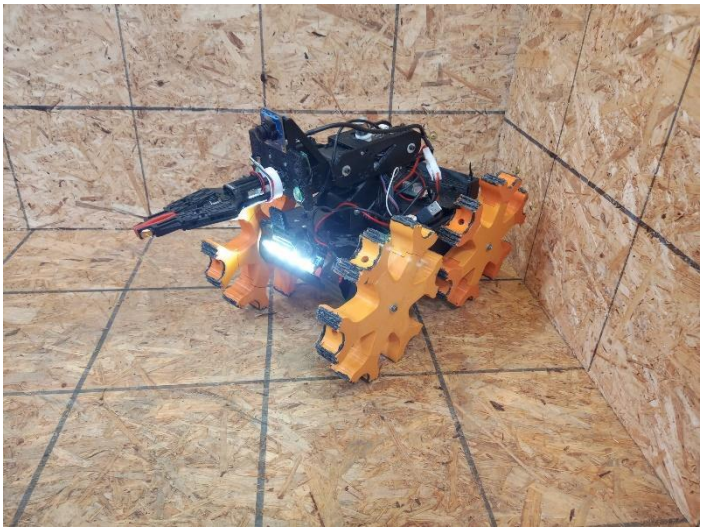
As part of the research ~~project~~ paper, we focused on the recognition of danger signs using machine learning. In the introduction, we defined the objectives, hypotheses, and research methods. What followed was designing the robot and its components using Creo Parametric software. We then described the electrical circuits, cameras, motors, the microcomputer, and the 3D printing process. In the Programs chapter, we focused on machine learning, danger sign detection, color detection, robot control with a controller, and the use of QR codes. In the conclusion, we analyzed the hypotheses and discussed possibilities for further development.

KAZALO VSEBINE:

KAZALO SLIK:

1 UVOD

Reševalni roboti je vrsta robota, ki je namenjen iskanju preživelih, navigaciji v težko dostopnih okoljih in izvajanju reševalnih operacij s ciljem hitrega odziva. Izgled reševalnega robota lahko vidimo na slikah 1 in 2.



Slika 1 : Primer reševalnega robota

(Vir: https://major.robocup.de/ligen/rapidly-manufactured-robot-challenge/?lang=en#&gid=1&pid=8)

Slika 2: Simbolična slika robotske roke pritrjene na reševalnega robota

(Vir: https://www.nbcnews.com)

V današnjem svetu se soočamo z nenehnimi izzivi, ki izhajajo iz naravnih nesreč in katastrof, ki lahko povzročijo uničujoče posledice za življenje. V tem kontekstu je bil razvit naš koncept reševalnega robota, ki bo predstavljal vsestransko orodje za izvajanje različnih nalog v zahtevnih okoljih.

1.1 CILJI

Zadali smo si cilj, da izdelamo delujočega reševalnega robota, ki bo lahko premagoval različne terenske ovire, opremljen bo s senzorji za lažje premikanje po prostoru in imel bo robotsko roko, ki bo zmožna opravljati različne naloge.

1.2 HIPOTEZE

1. S pomočjo strojnega učenja bomo znake za nevarnost prepoznali v majn kot 1 sekundi,
2. na sliki bomo vedno našli pravilno mesto, kjer se nahaja znak za nevarnost,
3. robot bo lahko premagal stopnico, visoko 10 cm,
4. zmožen se bo obrniti na mestu v labirintu,
5. sestavni deli robota bodo hitro zamenljivi,

1.3 METODE RAZISKOVANJA

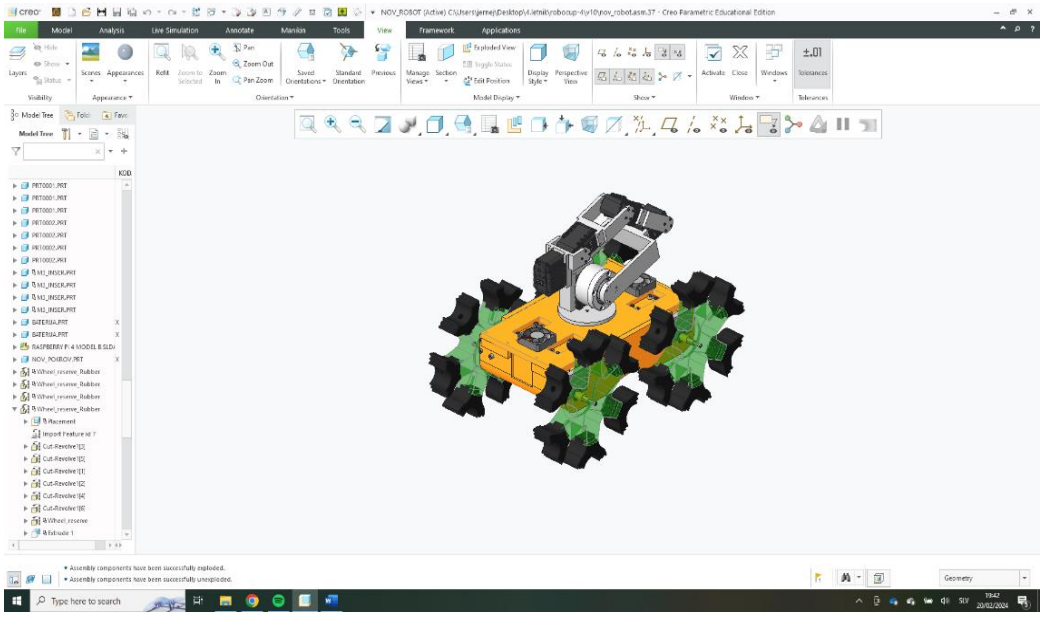
V postopku raziskovanja bomo uporabili sledeče metode:

- Metoda analize temelji na osnovi razčlenitve neke celote na njene osnovne sestavne enote. Na ta način smo si izdelavo robota razdelili na 2 dela: snovanje/konstruiranje in programiranje robota.
- Eksperimentalna metoda bo zajemala načrtovanje in izvedbo preizkusov, ki bodo testirali predpostavljene hipoteze.

2 IZRIS

2.1 CAD: PROGRAM CREO PARAMETRIC

Creo Parametric je 3D računalniški program za načrtovanje (CAD), ki ga je razvilo podjetje PTC (Parametric Technology Corporation). Gre za zmogljivo orodje, ki se uporablja za oblikovanje izdelkov, inženiring in proizvodne procese. Creo Parametric ponuja širok nabor funkcij, vključno s 3D-konstruiranjem kosov, fleksibilnim modeliranjem, konstruiranjem mehanizmov, priprave na 3D-tisk, orodjem za tehnično dokumentacijo, simulacijo in analizo. Uporabnikom omogoča ustvarjanje kompleksnih oblik, simulacijo resničnih pogojev in generiranje podrobnih tehničnih risb. Creo Parametric se široko uporablja v različnih industrijah, kot so avtomobilska, letalska, potrošniška in elektronska industrija, za oblikovanje ter razvoj inovativnih izdelkov. [3][14]

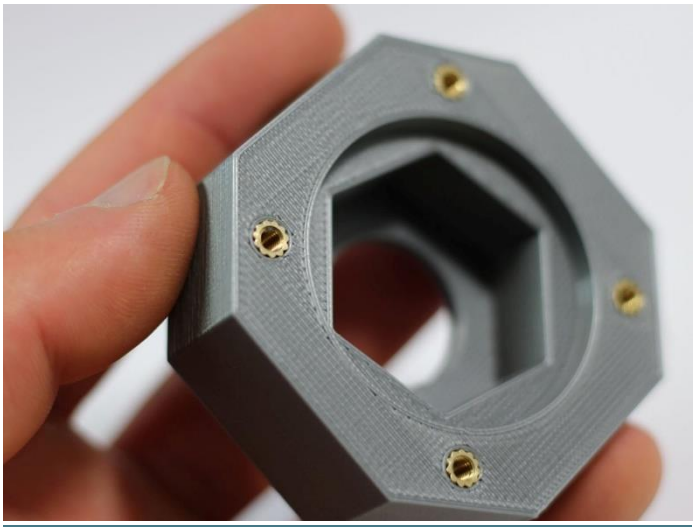


Slika 3: 3D-model reševalnega robota

(Vir: osebni arhiv)

2.2 OHIŠJE

Za izpolnitev zahtev po čim manjši teži robota, kompaktnosti, dvignjenosti od tal, dobri zračnosti in enostavni zamenljivosti smo se odločili, da bomo robot izdelali iz PLA-plastike. PLA-plastika je bila izbrana zaradi svoje enostavnosti tiskanja in izjemnih karakteristik, ki vključujejo dobro trdnost ter vzdržljivost. Pri pritrditvi ostalih delov na ohišje robota smo uporabili posebne vložke za plastiko, ki smo jih s spajkalnikom stopili v natisnjene dele, kot je prikazano na sliki 4. Ta postopek je omogočil stabilno in varno pritrditev komponent na ohišje robota. Vijake ustrezne velikosti smo nato privili v te vložke, kar je zagotovilo trdno pritrditev. Takšen pristop ne zagotavlja le enostavnega vzdrževanja, temveč omogoča tudi hitro in preprosto zamenjavo posameznih delov, če je to potrebno. To je ključnega pomena za učinkovito upravljanje in vzdrževanje robota v različnih situacijah. S tem smo zagotovili, da je naš robot zanesljiv in funkcionalen.[15]



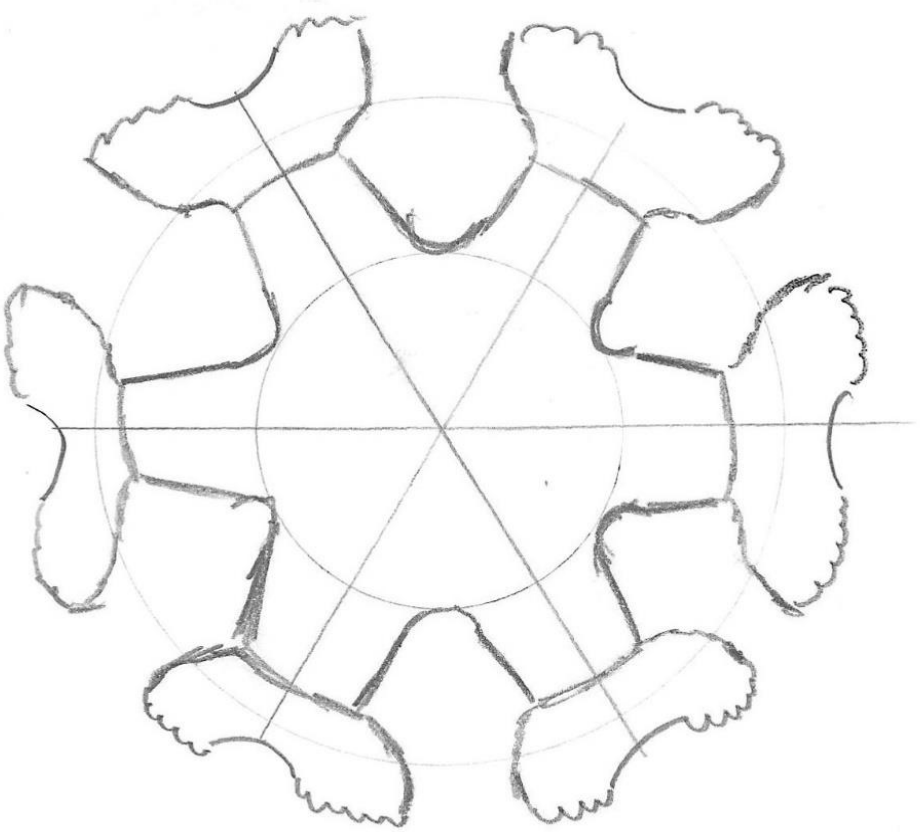
Slika 4: Vložki za plastiko

(Vir: https://www.3dpeople.uk/threaded-inserts-service)

2.3 PRENOS MEHANSKE SILE IN POGON

Imamo veliko možnosti, kot so gosenice, gume, fleksibilni nastavki na pogonskih kolesih … Zaradi lahke izdelave, transporta in velike učinkovitosti smo se odločili za fleksibilne nastavke, katerih skico vidimo na sliki 5. Izdelali smo jih s 3D-tiskalnikom, in sicer iz dveh

materialov. Sredinski del je bil izdelan iz PTEG zaradi trdote in odpornosti na obrabo. Zunanji deli noge pa iz TPU zaradi možnosti upogiba TPU-materiala.

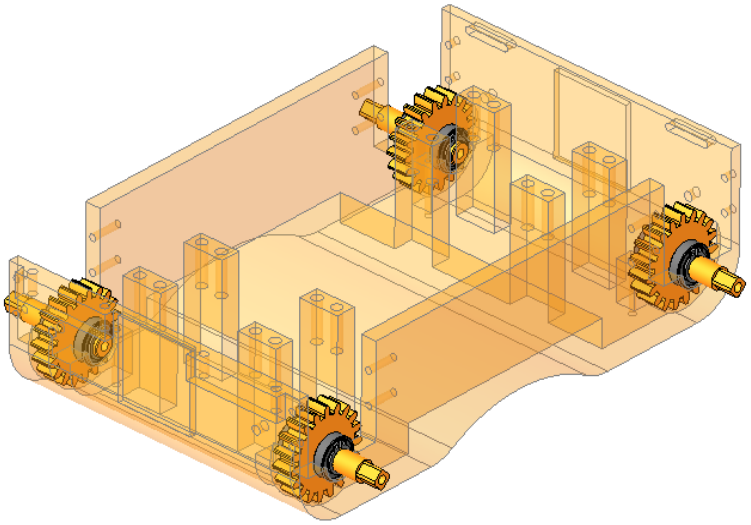


*Slika 5: Fleksibilni nastavk*i

(Vir: osebni arhiv)

2.4 POGONSKA OS

Po vseh raziskavah smo začeli z izdelovanjem prvih skic in risb robota. Pričeli smo pri motorjih. Naša prvotna ideja je bila, da bi pogonsko os priključili neposredno na motor, a zaradi velike sile, ki se pojavi pri padcih robota na motorje, smo se odločili za prenos z zobniki. Sprva smo načrtovali zobniški prenos z razmerjem 1 : 1, vendar smo v kasnejših preizkusih ugotovili, da bi bilo učinkoviteje z zobniškim razmerjem 2 : 3. Vse zobnike smo morali pritrditi na ohišje robota, kar smo naredili s pomočjo ležajev velikosti 19 mm x 10 mm x 5 mm in vijakov velikosti ϕ 10. Takšna pritrditev zagotavlja stabilnost in zanesljivo delovanje zobniškega prenosa ter omogoča učinkovito delovanje motorjev.[1]

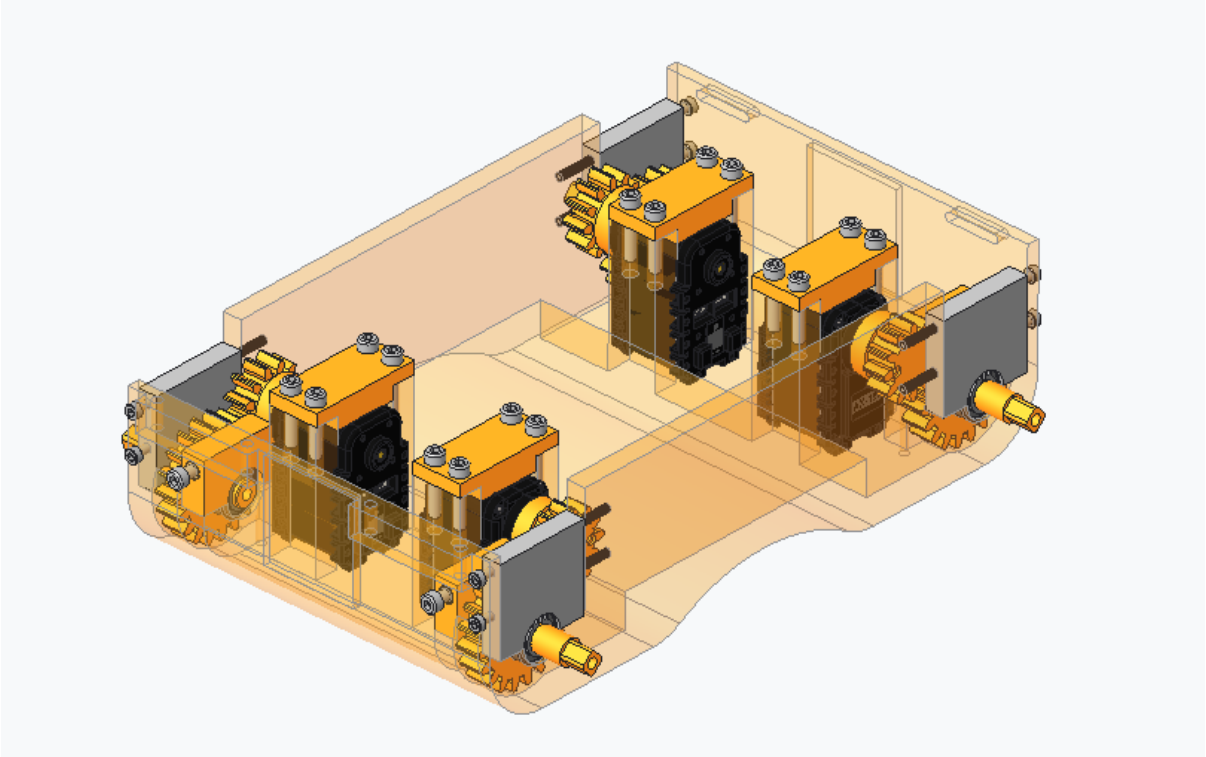


Slika 6: Pogonska os

(Vir: osebni arhiv)

2.5 MOTORJI

Motorje smo vstavili v posebne predele, ki so bili izdelani po njihovih merah, in jih nato pritrdili s pokrovom, ki je bil z vijaki M4 x 20 mm privit v vložke v telesu. Motorji so imeli na svoji pogonski osi s 4 vijaki M2 x 8 mm pritrjen zobnik z 12 zobci in modulom 2. Ti zobniki pa so gnali zobnike, ki so bili predstavljeni že predhodno. Ležaje za pogonsko os smo pričvrstili s posebnimi plastičnimi deli, ki smo jih privili v trup robota.[8]

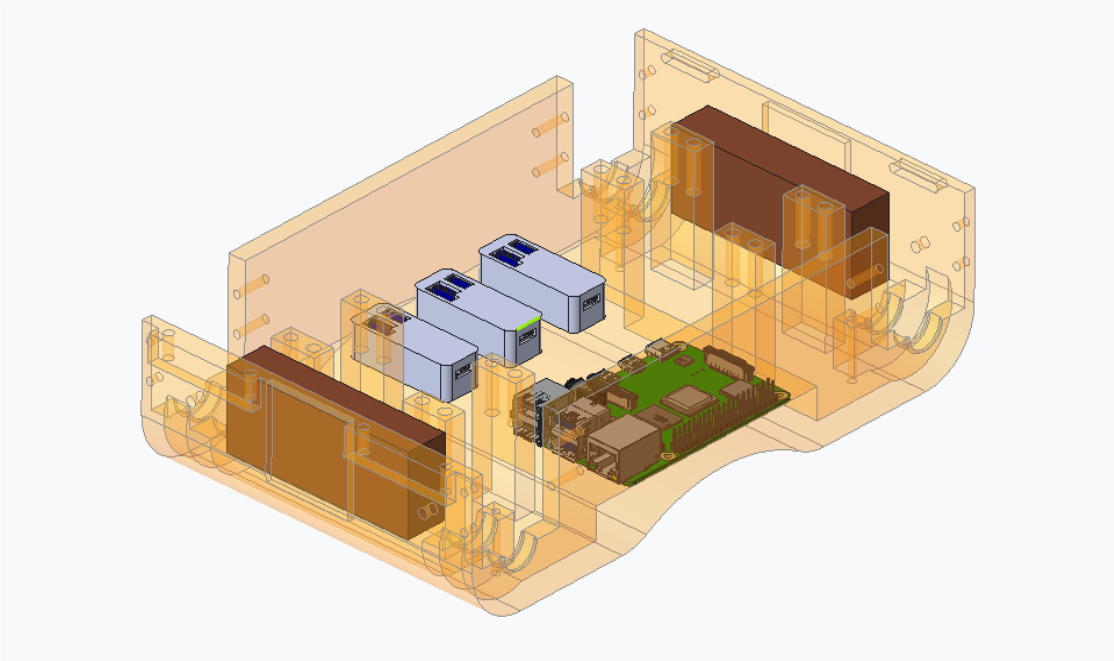


Slika 7: Pogonska os

(Vir: osebni arhiv)

2.6 ELEKTRONIKA

Pri risanju postavitve robota smo morali upoštevati razporeditev Raspberry Pi-ja, gonilnikov za motorje, baterij in ostalih električnih komponent. Pri tem smo se osredotočili na učinkovito postavitev, ki bi zagotovila optimalno delovanje sistema. Baterije smo postavili povsem na dno robota zaradi njihove teže, in sicer na levo in desno stran. Raspberry Pi potrebuje dobro prezračevanje, zato smo ga namestili na sredino robota, kjer je najboljši pretok zraka. Enako smo storili tudi z gonilniki, saj se ti pogosto segrevajo in potrebujejo ustrezno hlajenje.[9]

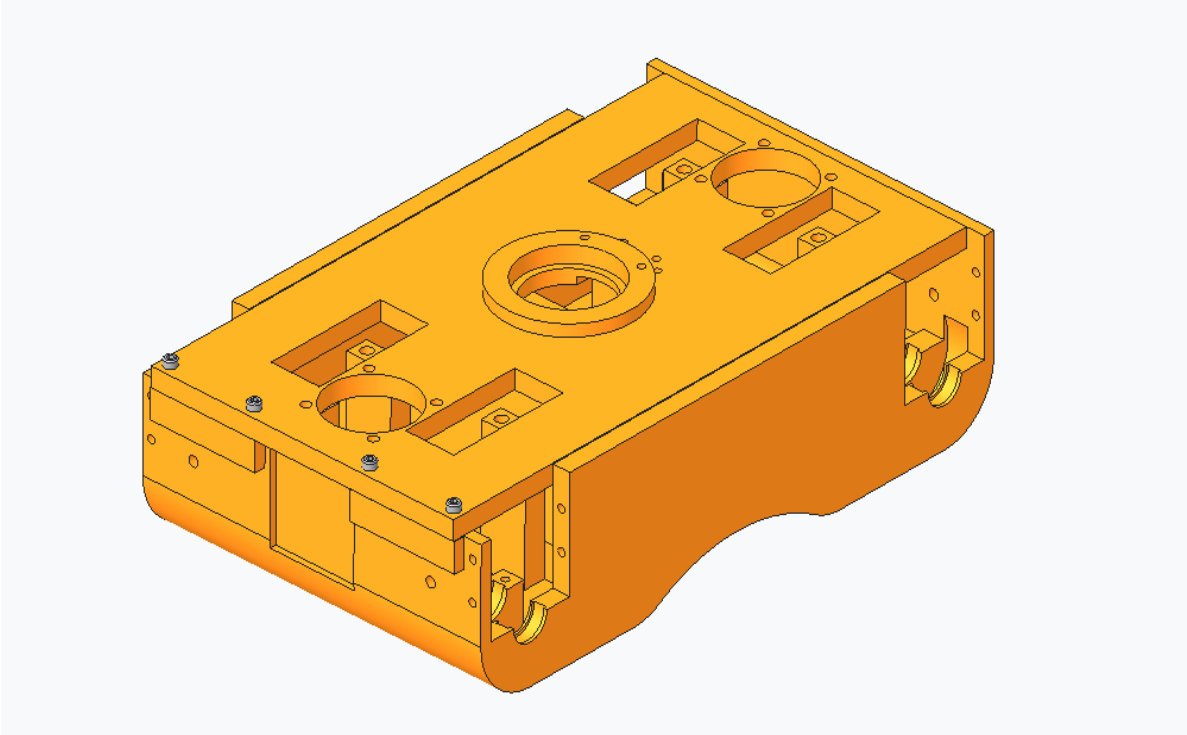


Slika 8: Položaj električnih komponent

(Vir: osebni arhiv)

2.7 POKROV

Vse notranje komponente je bilo potrebno zavarovati pred zunanjimi vplivi, zato smo to naredili s pokrovom, ki se vstavi v posebne luknje na levi strani robota, medtem ko ga na drugi strani pritrdimo v vložke v telesu s pomočjo vijakov M3 x 10 mm. Pokrov ima prav tako namen, da nosi roko in pripomore k trdnosti robota, zato mora biti kakovostno izdelan in močan. Izdelali smo ga iz PLA-materiala, saj je učinkovit.

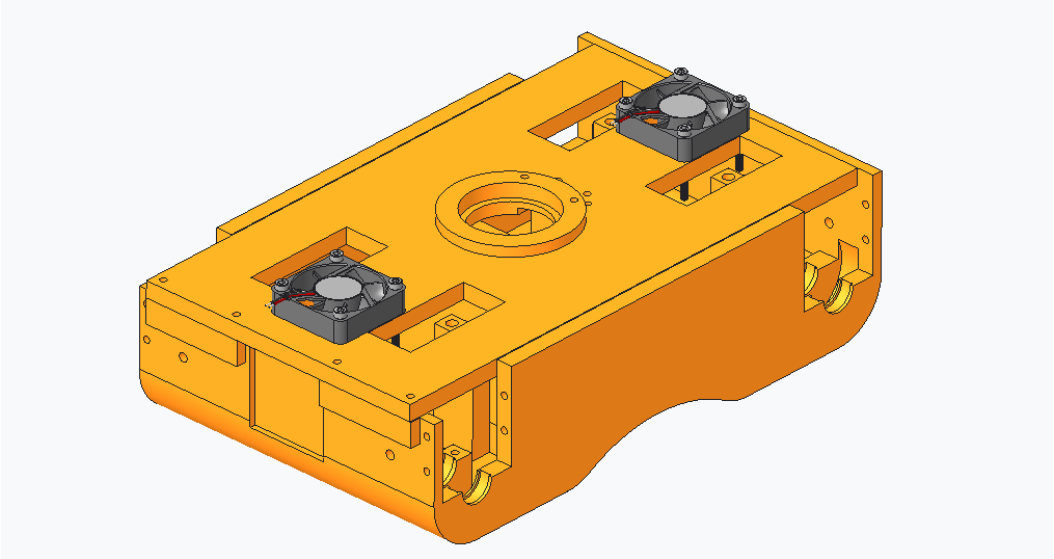


Slika 9: Pokrov robota

(Vir: osebni arhiv)

2.8 VENTILATORJA

Celoten robot mora biti dobro prezračen, zato smo na njegov pokrov dodali dva ventilatorja velikosti 40 mm x 40 mm x 5 mm. Ventilatorja smo namestili tako, da sta obrnjena v nasprotne smeri, kar omogoča pretok zraka skozi celoten robot. Eden od ventilatorjev je usmerjal zrak v notranjost robota, medtem ko je drugi izvlekel zrak iz njega. Ventilatorje smo pritrdili s pomočjo vložkov za plastiko, v katere smo privijačili vijake M3 x 15 mm.[7]



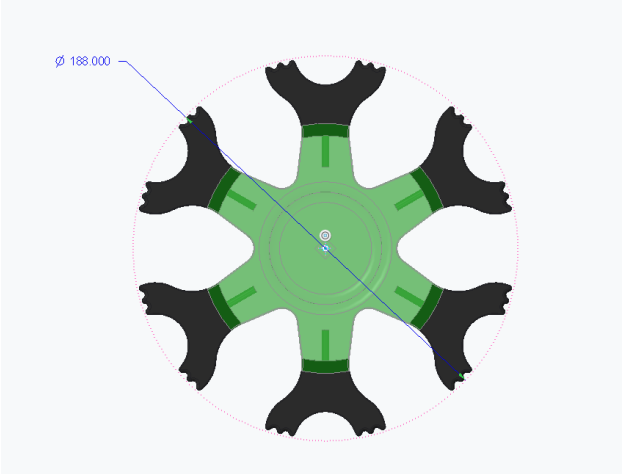
Slika 10: Nameščeni ventilatorji

(Vir: osebni arhiv)

2.9 FLEKSIBILNI NASTAVKI NA POGONSKIH KOLESIH

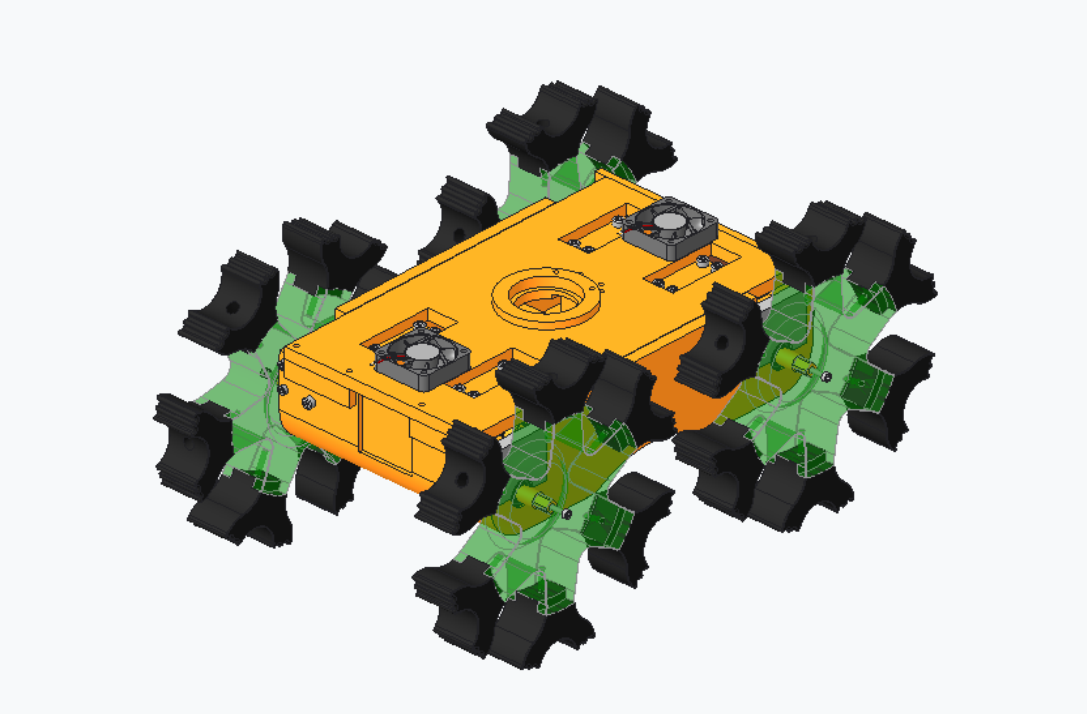
Za zagotavljanje stabilnega premikanja robota po različnih terenih smo se odločili uporabiti fleksibilne nastavke na pogonskih kolesih. Ti so bili izbrani zaradi njihovih sposobnosti, enostavne proizvodnje in zamenjave. Po več preizkusih smo določili pravo dimenzijo koles, pri čemer smo se odločili za največje nastavke, ki smo jih lahko namestili na robota. To smo storili z namenom, da bi zagotovili večjo višino robota, da se ne bi zaskočil na neravnih terenih. Dodatno smo v telesu robota ustvarili zaokrožen sredinski del, da smo povečali razdaljo med robotom in tlemi. Dimenzije fleksibilnih nastavkov smo natančno določili. Širina teh je bila 30 mm, premer njihovega kroga pa 188 mm. Fleksibilne nastavke smo

pritrdili na robota z vijaki M3 x 60 mm, ki smo jih zategnili v pogonsko os, v kateri so bili nameščeni ustrezni vložki. Za učinkovit prenos sil med fleksibilnimi nastavki in pogonsko osjo smo oblikovali kvadratno obliko na pogonski osi, ki se je ujemala z obliko luknje v nastavkih.



Slika 11: Premer kolesa

(Vir: osebni arhiv)

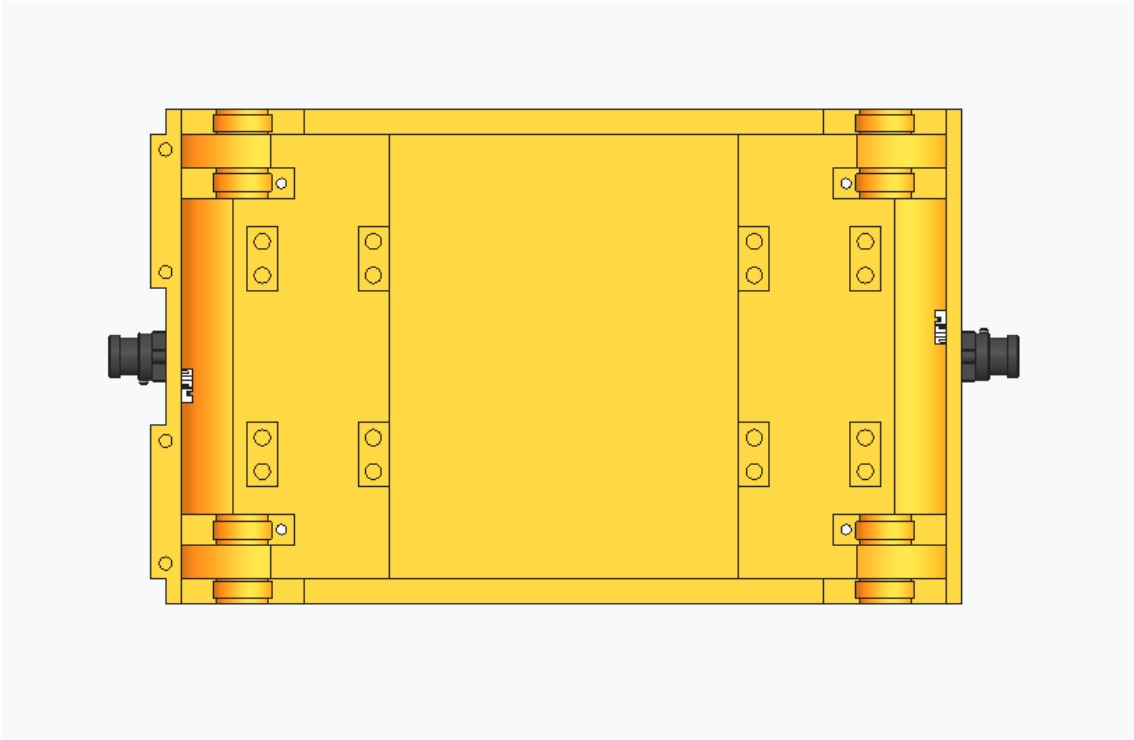


Slika 12: Fleksibilni nastavki na pogonskih kolesih na robotu

(Vir: osebni arhiv)

2.10 KAMERE

Na robotu imamo 2 kameri, in sicer na začetku in na koncu trupa motorja. Obe kameri smo pritrdili s 4 vijaki M2 x 15 mm. Postavitev kamer je vidna na sliki 13.



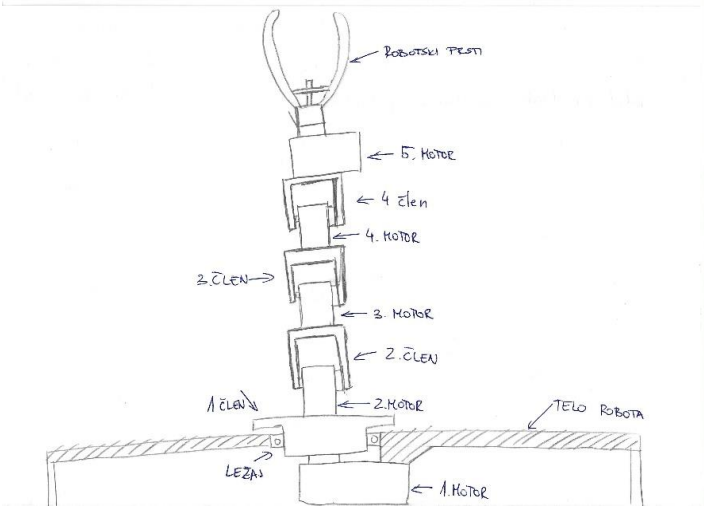
Slika 13: Položaj kamer

(Vir: osebni arhiv)

2.11 ROBOTSKA ROKA

V pravilih tekmovanja je zapisano, da mora robot izvajati naloge natančno in zanesljivo, kar pomeni, da mora imeti ustrezno kontrolo nad gibanjem in pozicioniranjem svojih sklepov.

Začeli smo z osnovnim načrtom roke, ki je bil ključen za nadaljnji razvoj. Naša vizija je bila ustvariti robotsko roko dolžine približno 400 mm, ko je v celoti iztegnjena, kar bi omogočalo raznoliko uporabo v različnih pogojih. Pri izbiri motorjev smo se odločili za AX-12A motorje, ki so se izkazali kot zelo primerni za naše potrebe. Kljub temu pa smo prepoznali njihovo glavno slabost – plastične zobnike v notranjosti. Da bi omilili morebitne težave zaradi tega, smo načrtovali, da bosta 2 od 5 motorjev (konkretno motorja 1 in 5) skrbela za obračanje roke, medtem ko bodo preostali motorji delovali kot človeški komolec in omogočali spreminjanje dolžine roke. Prvi korak v našem načrtu je bil ustvariti skico, ki je služila kot osnova za nadaljnji razvoj. Skica je vsebovala osnovno obliko roke in razporeditev motorjev, kar je omogočilo boljšo predstavo o končnem izdelku. V nadaljnjem razvoju načrta smo se osredotočili na več ključnih vidikov. Prvič, bilo je pomembno izbrati ustrezne materiale za izdelavo, ki bi zagotovili trpežnost in stabilnost roke. Poleg tega smo načrtovali krmiljenje roke. Prav tako smo posvečali posebno pozornost varovanju motorjev, zlasti plastičnih zobnikov, pred preobremenitvijo in poškodbami. S tem v mislih smo oblikovali ustrezne zaščitne mehanizme, ki so podaljšali življenjsko dobo motorjev in povečali zanesljivost celotnega sistema.[12]

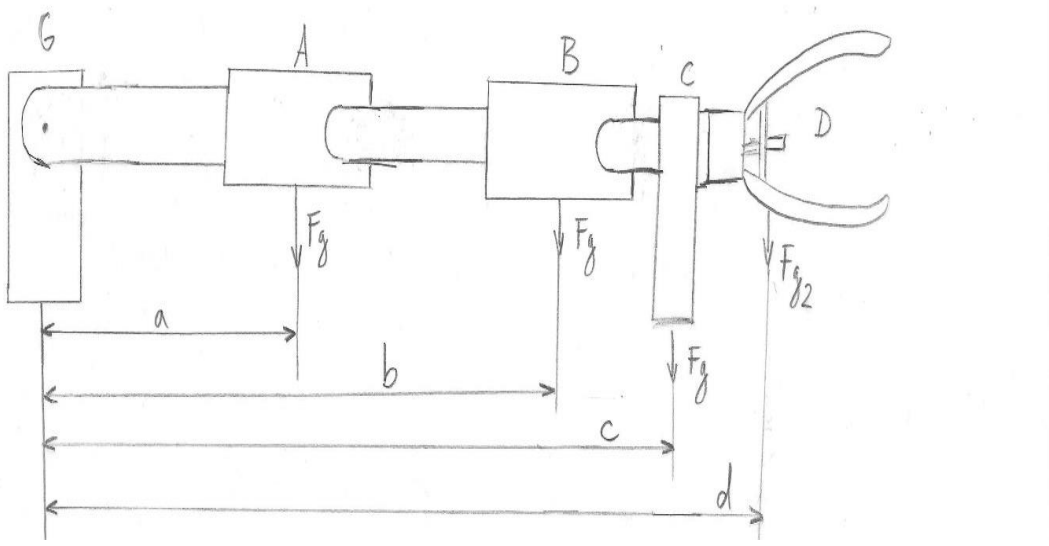


Slika 14: Skica roke

(Vir: osebni arhiv)

2.11.1 Momenti

Po osnovni zasnovi smo začeli z izračuni sil, če se bo vse na robotski roki izšlo. Izračun je temeljil na ravnotežni enačbi momentov, ki govori: *"Da je vsota vseh zunanjih in notranjih sil (momentov) posameznega dela enaka nič (saj tak del dejansko miruje). "(Jerman, 2017: 4).* [10] Z uporabo tega načela smo določili moment okoli prve osi, ki je znašal 0,93915 Nm brez obremenitve in 1,374 Nm med obremenitvijo. Nato smo preučili specifikacije motorjev na uradni spletni strani proizvajalca in ugotovili, da motorji pri statični obremenitvi zmorejo prenesti 1,4 Nm. Na podlagi teh podatkov smo se odločili, da smo na prvi člen roke dodali planetni reduktor, ki je povečal moč na prvem sklepu. Vsi izračuni so vidni spodaj in na sliki 15.



Slika 15: Načrt roke

(Vir: osebni arhiv)

Izračun:

$Fg = 60\ g = 0{,}59\ N$

$Fg_2 = 150\ g = 1{,}47\ N$

$A = 115\ mm = 0{,}115\ m$

$b = 230\ mm = 0{,}23\ m$

$c = 300\ mm = 0{,}3\ m$

$d = 380\ mm = 0{,}38\ m$

$M_G = M_A + M_B + M_C + M_D =$

$= (Fg*a) + (Fg*b) + (Fg*c) + (Fg_2*d) =$

$= 0{,}06785\ Nm + 0{,}1357\ Nm + 0{,}177\ Nm + 0{,}5586\ Nm =$

$= 0_1 93915\ Nm \Rightarrow$ brez obremenitve in plastičnih delov

Z OBREMENITVIJO IN 3D NATISNJENIMI DELI:

$Fg = 70\ g = 0{,}687\ N$ $M_G = M_A + M_B + M_C + M_D =$

$Fg_2 = 250\ g = 2{,}45\ N$ $= 0{,}079\ Nm + 0{,}158\ Nm + 0{,}206\ Nm + 0{,}931\ Nm =$

a = 0,115 m $= 1,\ 374\ Nm \Rightarrow$ Z OBREMENITVIJO IN TEŽO DELOV

$b = 0{,}23\ m$

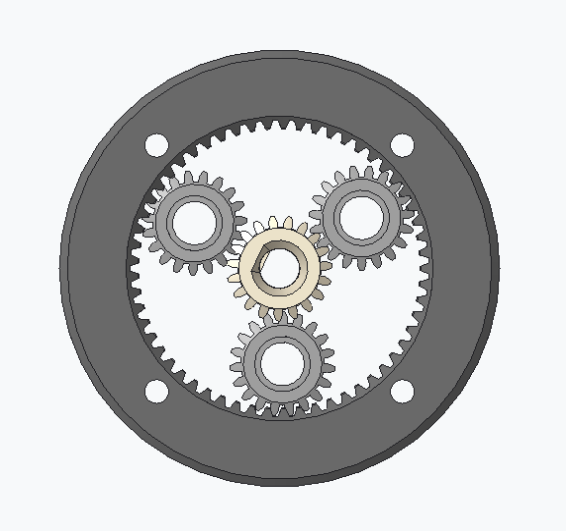
$c = 0{,}3\ m$

$d = 0{,}38\ m$

2.11.2 Reduktor

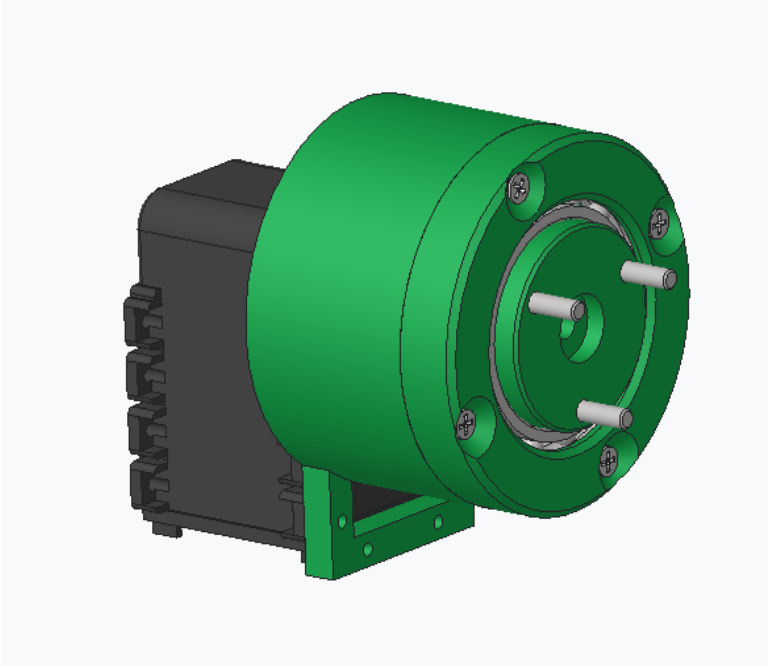
Odločili smo se za uporabo planetarnega reduktorja, saj je občutno manjši od tradicionalnega zobniškega reduktorja, hkrati pa zagotavlja enako zmogljivost. Dodatna prednost planetarnega reduktorja je enakomerna razporeditev sil na 3 planetnih zobnikih, kar zmanjšuje verjetnost zloma delov v reduktorju. Sami smo izdelali reduktor iz PLA-plastike, saj je ta material primeren za naše potrebe in omogoča zadostno trdnost in vzdržljivost.

Začetno načrtovanje reduktorja je potekalo v programu Creo Parametric, pri čemer smo upoštevali več kriterijev, med drugim pritrditev reduktorja na motor, trajnost in kompaktnost. Reduktor je zasnovan z zunanjim ohišjem, ki se pritrdi na motor s pomočjo vijakov M2 x 10 mm. Sestavljen je iz 3 planetnih zobnikov z 19 zobci in 1 sončnega zobnika enake velikosti. V vsak zobnik smo vstavili ležaje velikosti 7 mm x 3 mm x 3 mm, ki zagotavljajo gladko vrtenje zobnikov v reduktorju. Zobnike smo pritrdili na zunanjo os z vijaki M3 x 25 mm, nato pa smo jih dodatno namazali in vstavili v ohišje motorja. Zunanja os je opremljena tudi z ležajem velikosti 42 mm x 30 mm x 5 mm, ki je vstavljen v pokrov reduktorja. Celoten reduktor je prikazan na sliki 17, kjer so vidne vse zunanje komponente, medtem ko notranja struktura in prenos izhajajo iz slike 16.[13]



Slika 16: Planetarno gonilo

(Vir: osebni arhiv)

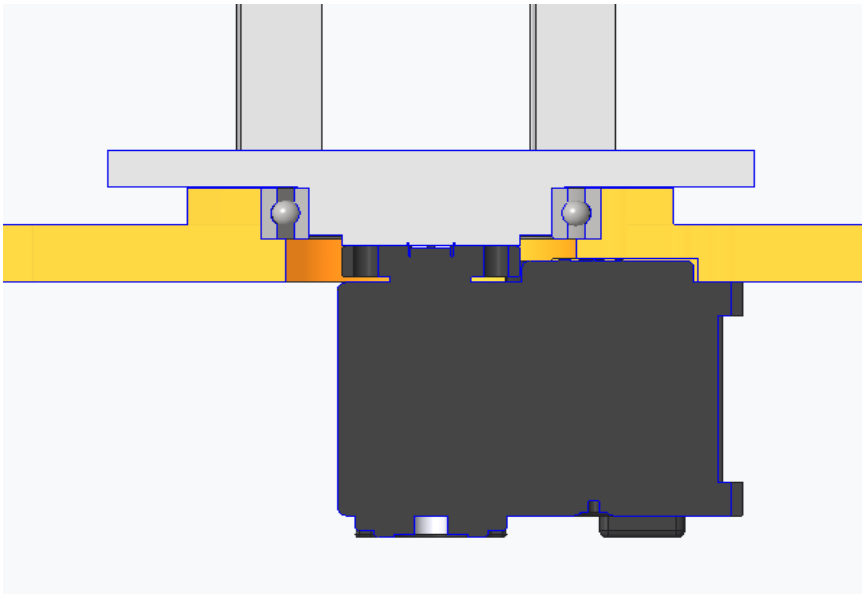


Slika 17: Planetarno gonilo, nameščeno na motor

(Vir: osebni arhiv)

2.11.3 Prva vrtljiva os

Pričeli smo s prvim motorjem, ki smo ga pritrdili na pokrov robota s pomočjo 4 vijakov velikosti M2 x 16 mm. Prvi motor je ključen za omogočanje krožnega obračanja roke za 360° okoli robota. Nadalje smo nanj namestili plastični del, ki je namenjen pritrditvi drugega motorja. Slednji del smo pritrdili s 4 vijaki velikosti M2 x 20 mm. Z namenom izboljšanja prenosa sil in povečanja trdnosti roke smo v pokrov roke vgradili ležaj velikosti 42 mm x 30 mm x 5 mm, v katerega smo namestili prvi del.

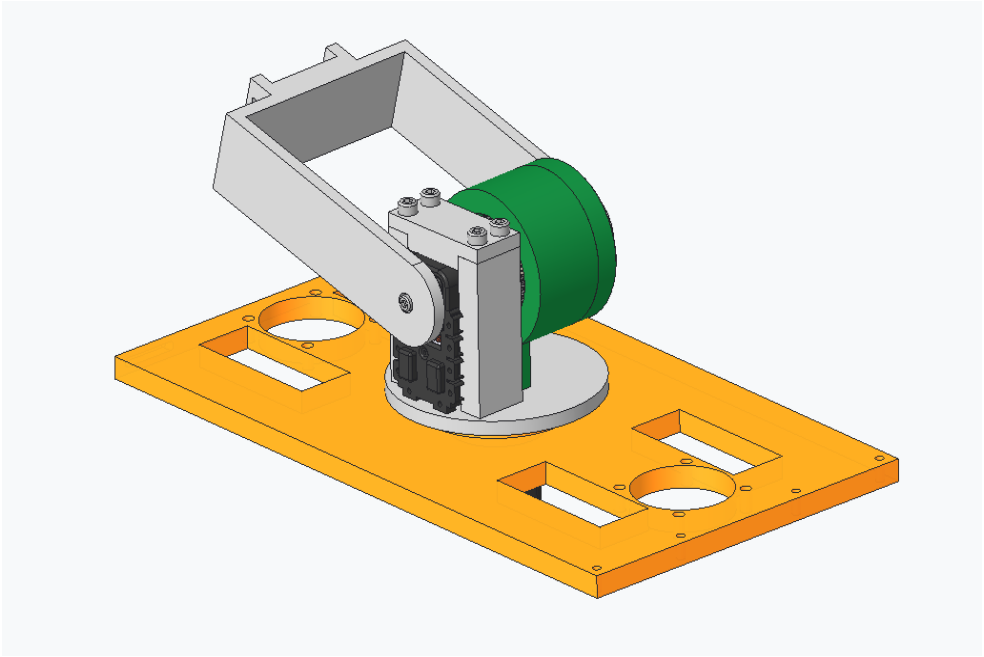


Slika 18: Prva vrtljiva os

(Vir: osebni arhiv)

2.11.4 Druga os

Na prvi del smo najprej pritrdili drugi motor s planetnim reduktorjem, ki je bil že prej predstavljen. Pritrdili smo ga s pomočjo t. i. pokrova motorjev, ki je bil privit v prvi del z vijaki M4 x 20 mm, v katerega so bili predhodno vstavljeni vložki. Na drugi motor smo nato pritrdili drugi del robotske roke, ki služi premiku roke v dolžino in deluje kot človeški komolec. Nato smo ga na motor pritrdili z vijakom M3 x 15 mm, ki je bil privit skozi ležaj velikosti 7 mm x 3 mm x 3 mm.

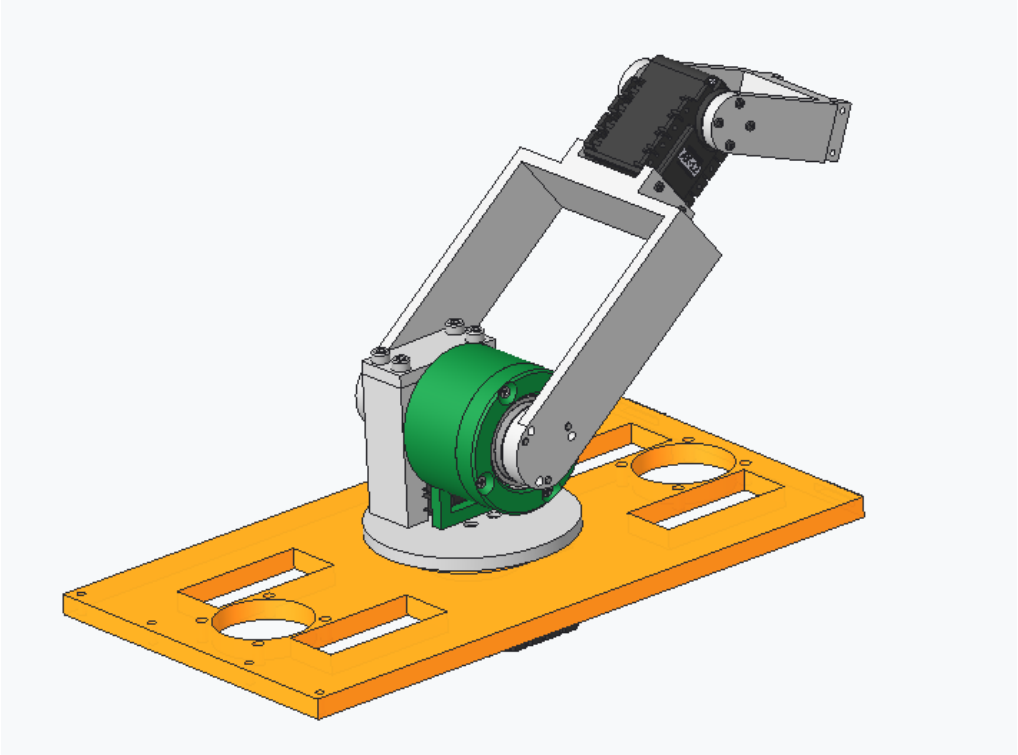


Slika 19: Prvi sklep

(Vir: osebni arhiv)

2.11.5 Tretja os

Drugi sklep je sestavljen iz enega motorja, ki je pritrjen na drugi del s 4 vijaki M2 x 20 mm. Na ta motor smo nato montirali tretji plastični del roke, v katerega smo že predhodno vstavili ležaj velikosti 7 mm x 3 mm x 3 mm, ki izboljšuje gibljivost roke in zmanjšuje obrabo. Tretji del smo pritrdili na motor z vijakom M3 x 15 mm na eni strani in s 4 vijaki M2 x 16 mm na drugi strani.

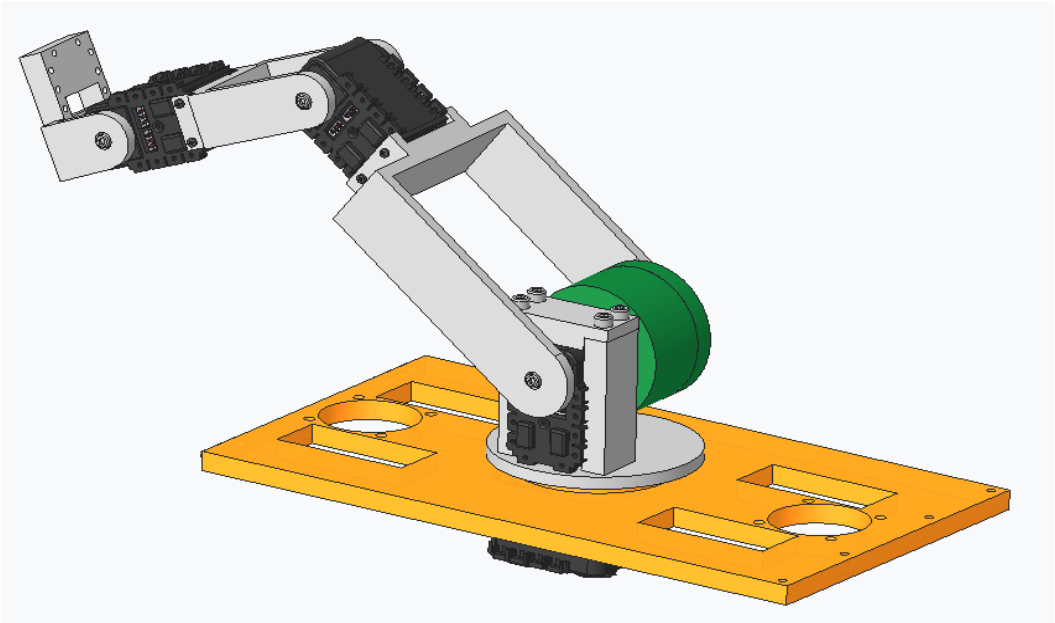


Slika 20: Drugi sklep

(Vir: osebni arhiv)

2.11.6 Četrta os

Tretji sklep je skoraj identičen drugemu, saj uporablja enake komponente. Edina razlika je, da je plastični del oblikovan nekoliko drugače, kar omogoča drugačno pritrditev petega motorja. Četrti motor je pritrjen na tretji del s 4 vijaki M2 x 20 mm. Enako kot prej je v četrti del že vstavljen ležaj velikosti 7 mm x 3 mm x 3 mm, ki izboljšuje gibljivost roke in zmanjšuje obrabo. Nato smo četrti del pritrdili na motor z vijakom M3 x 15 mm na eni strani in s 4 vijaki M2 x 16 mm na drugi strani.

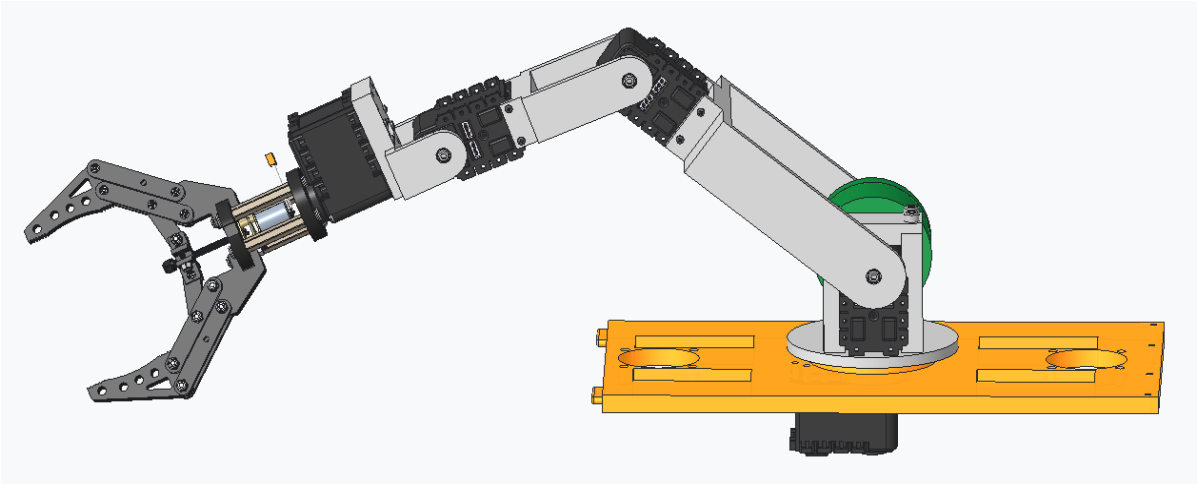


Slika 21: Tretji sklep

(Vir: osebni arhiv)

2.11.7 Peti motor in robotsko prijemalo

Za zaključek roke smo na četrti plastični del pritrdili peti motor, ki služi za rotacijo prstov roke. Motor smo pritrdili na četrti del s 4 vijaki M2 x 20 mm. Nazadnje smo na zadnji motor pritrdili prste, in sicer robotske prste Makeblock Robot Gripper, ki so izjemno zmogljivi glede na svojo težo. Pritrdili smo jih s 4 vijaki M2 x 30 mm.[2]

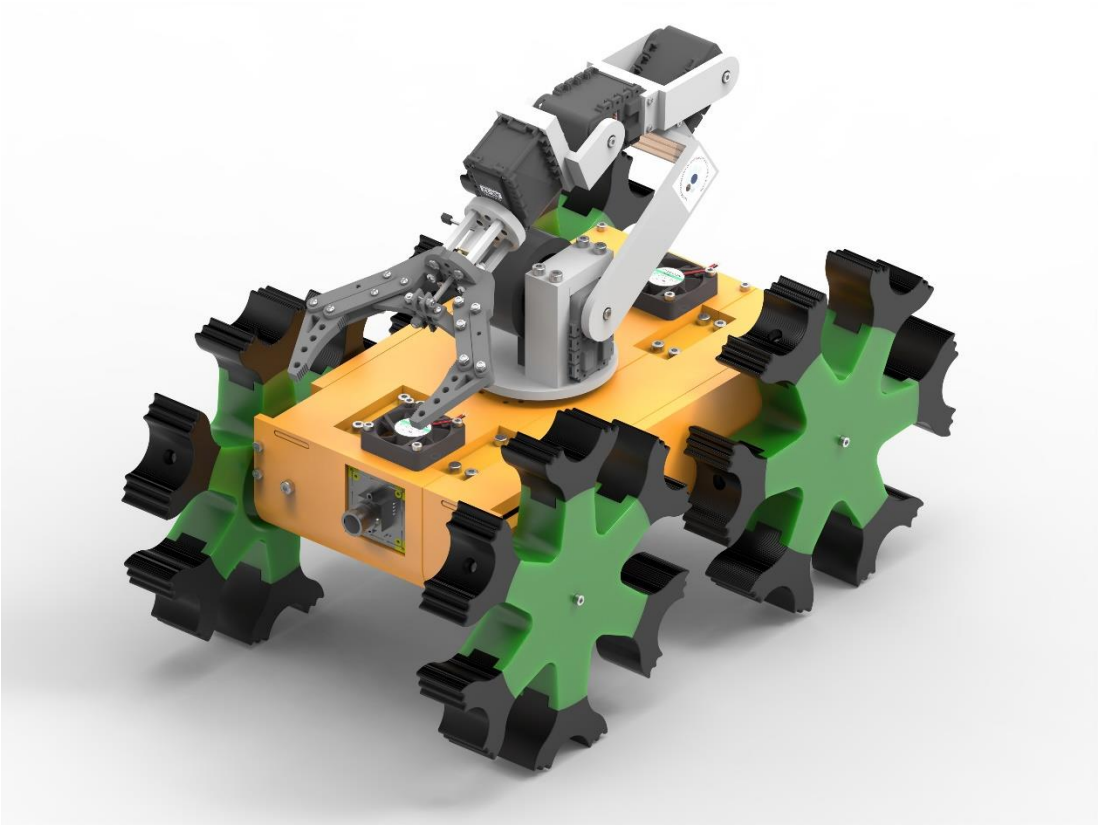


Slika 22: Celotna robotska roka

(Vir: osebni arhiv)

2.12 KONČNI KONCEPT REŠEVALNEGA ROBOTA

Sedaj smo končali tehnični del našega robota – vse kar je potrebno, da se motor premika in deluje nemoteno. Z robotom smo zadovoljni, čeprav obstaja še veliko možnosti za izboljšave. Na sliki 23 lahko vidimo koncept končnega izgleda robota.



Slika 23: Končni koncept reševalnega robota

(Vir: osebni arhiv)

3 ELEKTRIČNA VEZAVA

3.1 KAMERE

Za naš projekt potrebujemo pogled s sprednjega in zadnjega dela robota, kar nam omogoča boljši pregled okolice in lažje izogibanje oviram med vožnjo. Za zaznavo smo se odločili, da dodamo kamero na roko robota.

3.1.1 Sprednja kamera

Za sprednjo kamero smo izbrali Sonyjevo kamero, ki omogoča zajem slike v 180° polju okoli sebe. Ta kamera je ključna za premikanje po prostoru in zaznavanje terena, ki je pred nami.

Slika 24: Sprednja kamera Sony

(Vir: https://www.elpcctv.com)

3.1.2 Zadnja kamera

Za zadnjo kamero smo izbrali lahko pritrdljivo kamero proizvajalca Megapixel. Ta kamera ima visoko ločljivost, in sicer 180° širokokotni objektiv brez popačenja, ki omogoča zajem visokokakovostnih videoposnetkov. Deluje preko povezave USB in ne zahteva dodatnih gonilnikov, kar olajša namestitev in uporabo.[6]

Slika 25: Zadnja kamera MEGAPIXEL

(Vir: https://www.elpcctv.com)

3.1.3 Kamera na roki

Pri tej kameri ne potrebujemo širokokotnega objektiva, temveč oster pogled naravnost za lažje zaznavanje znakov za nevarnost in branje QR-kod.

Slika 26: Kamera na roki Sony

(Vir: https://www.elpcctv.com)

3.2 MOTORJI

Razmišljali smo o različnih vrstah motorjev (servo, koračni, DC …). Po temeljitem raziskovanju smo se odločili za uporabo servomotorjev, saj so primerni za naše potrebe zaradi svoje vodljivosti, kompaktnosti in razpoložljivosti. Končno smo se odločili za motorje Dynamixel AX-18A, saj so opremljeni s številnimi senzorji, ki vključujejo merilnike temperature, kota osi in navora.

Slika 27: Dynamixel AX-18A

(Vir: https://emanual.robotis.com)

3.3 ENERGIJA

Na trgu smo zasledili veliko vrst baterij (litijeve, alkalne, gel baterije, NiMH …). Vsaka od teh baterij ima svoje prednosti in slabosti. Po pregledu smo se odločili za uporabo LiPo baterij, ki so znane tudi kot litij-polimerske baterije. Izbrali smo jih zaradi njihove zanesljivosti ob pravilni uporabi, lahki teži v primerjavi z drugimi baterijami na trgu in njihove modularnosti, ki omogoča različne napetosti (1s, 2s, 3s …). Odločili smo se za 3s, saj potrebujemo 11,1 V za napajanje naših motorjev.

Slika 28: LiPo baterija

(Vir: https://www.lipo.si)

3.4 MIKRORAČUNALNIK (MOŽGANI)

Izbrali smo Raspberry Pi 4, saj je dovolj zmogljiv in izpolnjuje vse zahteve, ki jih potrebujemo. Omogoča komunikacijo z računalnikom in uporablja programski jezik Python, ki ga poznamo že od prej. Med uporabniki je zelo priljubljen, ker omogoča dostop do obsežnih virov in knjižnic. Prav tako je cenovno dostopen, kar ga naredi odličnega za naš projekt z omejenim proračunom. [18]

Slika 29: Raspberry Pi 4

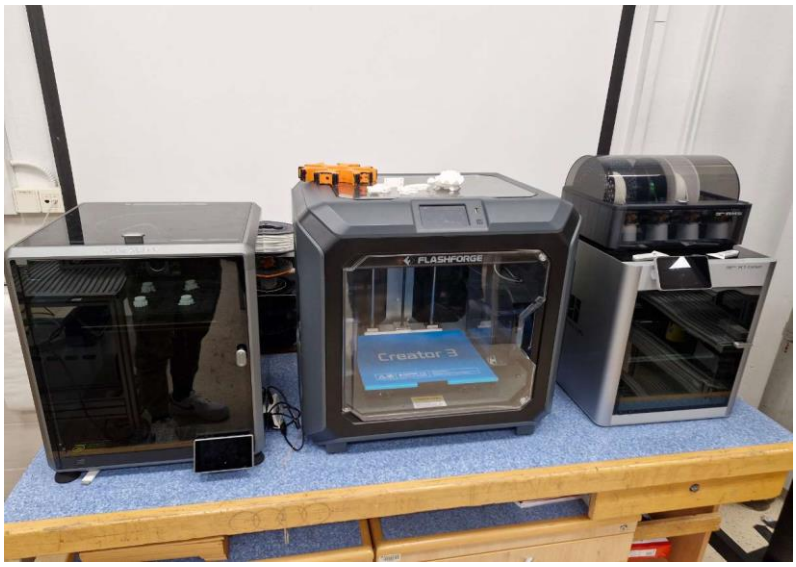
(Vir: https://en.wikipedia.org/wiki/Raspberry_Pi)

3.5 KOMPONENTE (KOLESA, OGRODJE ...)

Vse komponente smo izrisali in jih s pomočjo 3D-tiskanja izdelali. Za to tehniko smo se odločili zato, ker je poceni in lahko razmeroma hitro pretvorimo našo idejo v realnost. 3D-tiskalniki so dostopni na naši šoli, kar nam omogoča enostaven in hiter dostop do tehnologije. S tem se izognemo potrebi po iskanju zunanjih virov za izdelavo komponent, kar pripomore k učinkovitemu napredku projekta.

3.6 3D-TISKANJE

3D-tiskanje je moderna pot izdelave kosa iz digitalnega (CAD) modela, pri kateri 3D-tiskalnik preko šobe po plasteh nanaša raztaljen filament. Oblika kosa je torej le še stvar domišljije. 3D-tiskalniki so vse natančnejši, večji, hitrejši, poleg tega pa pokrijejo širok nabor materialov. Še vedno so najuporabnejši za izdelavo prototipov, vendar pa je 3D-tiskanje uporabno tudi za proizvodnjo v manjših serijah. [4][11]



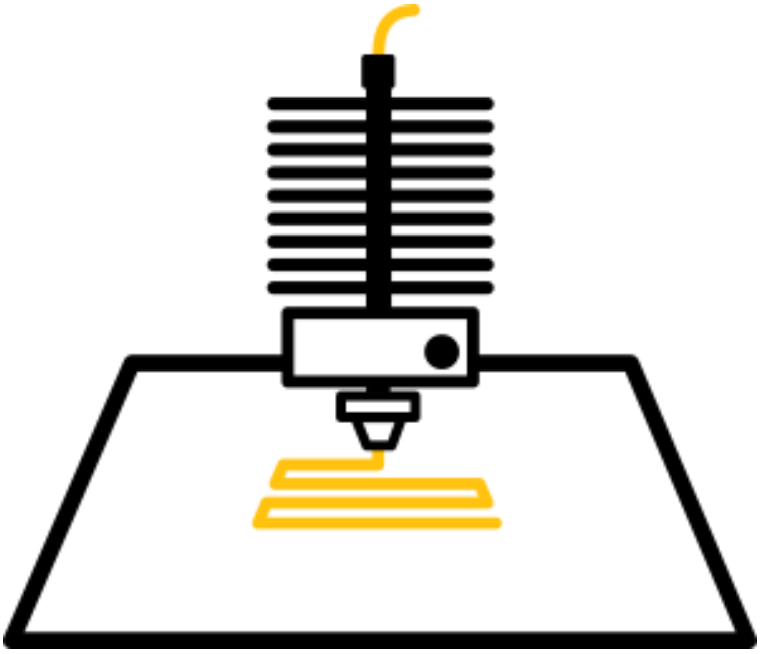
Slika 30 :3D-tiskalnik

(Vir: osebni arhiv)

3D-tiskalniki lahko za tisk uporabljajo veliko različnih materialov (npr. guma, plastika, poliuretanski materiali, kovine ipd.), izbira pa je odvisna od zmogljivosti in tipa tiskalnika. V splošnem se materiali po plasteh nanašajo na različne načine, a najpogostejši v tiskalnikih nižjega cenovnega razreda je nanos topljenega filamenta skozi majhno šobo. V večini se uporablja bioplastika PLA ali vzdržljivejša plastika ABS, ki za uporabo potrebuje ogrevano posteljo za šobo. 3D-tiskalnik filament med tiskanjem vleče v ogrevano glavo, kjer se le ta stopi in se skozi šobo nanese na tiskalno posteljo. [16]

Za delovanje mora biti glava sposobna premikanja po delovnem prostoru vsaj v 3 oseh – torej 3 dimenzijah. Načeloma vsako os poganja svoj motor – zato je kakovost tiska odvisna

od natančnosti in kakovosti pogonskih in transportnih komponent, predvsem pa tudi od vodenja motorjev. [16]



Slika 31: Nanašanje plastike na podlago

(Vir: https://3dtovarna.si)

4 PROGRAMI

Na tekmovanju se ocenjuje zaznavanje različnih znakov. Odločili smo se da jih bomo zaznavali s pomočjo kamere in mikroračunalnika Raspberry Pi.

Znaki, ki jih je potrebno zaznati:

- znaki za nevarnost,
- QR-kode,
- barve,
- gibanje,
- toplota.

Slika 32: Znaki za nevarnost in ostale zaznave

(Vir: https://intelligentrobots.org/files/RoboCup/2019/RMRC/RMRC_2019_Rules_V1.5_-_RKS.pdf)

4.1 PROGRAMSKI JEZIK

Najprej smo preverili, kateri programski jezik bi bil najprimernejši za naš projekt. Razmišljali smo med C/C++, Pythonom in Javo. Po posvetovanju z mentorjema in pregledu razpoložljivih raziskav na spletu smo se odločili za Python. Ta odločitev je bila posledica več dejavnikov. Prvič, Python smo izbrali zaradi obsežne dokumentacije in širokega nabora virov, ki so na voljo za učenje in razumevanje tega jezika. Drugič, izbran je bil zaradi svoje kompatibilnosti z našim nadzornim vezjem, ki je Raspberry Pi. Python ima bogato podporo zanj in je bil zasnovan z mislijo na to platformo, kar olajša integracijo med programom in

strojno opremo. Nazadnje smo ga izbrali zaradi njegove prilagodljivosti in enostavnosti uporabe. Omogoča hitro prototipiranje in razvoj ter olajša odpravljanje morebitnih napak v programu. S to odločitvijo smo zagotovili, da imamo ustrezna orodja in tehnologijo za uspešen razvoj našega projekta.

4.2 STROJNO UČENJE ZAZNAVE ZNAKOV

Strojno učenje v kombinaciji s strojnim vidom je veja umetne inteligence, ki se osredotoča na uporabo velike količine naprej posnetih slik za zaznavanje določenih predmetov. Ti so v našem primeru znaki za nevarnost. Uspešno zaznavo znaka z računalnikom dosežemo s posnemanjem učenja ljudi, tako da na vnaprej posnetih slikah iščemo podobnosti, ki so skupne določenemu predmetu.

Znaki, ki smo jih morali zaznati, sodijo v 9 kategorij, in sicer:

- nevarnost, ko je mokro,
- eksplozivno,
- vnetljiva trdna snov,
- vnetljiv plin,
- nalezljiva snov,
- nevarno za vdihovanje,
- nevnetljiv plin,
- organski peroksid,
- samovnetljivo.

Ugotoviti je bilo potrebno, kje na sliki se znak nahaja in v katero kategorijo sodi.

Slika 33: Znaki za nevarnost

(Vir: https://www.dreamstime.com/illustration/hazardous.html)

4.2.1 TensorFlow

TensorFlow je brezplačna in odprtokodna knjižnica programske opreme za strojno učenje in umetno inteligenco, ki je posebej osredotočena na izobraževanje globokih nevronskih mrež.

Zakaj smo se odločili za uporabo knjižnice TensorFlow?

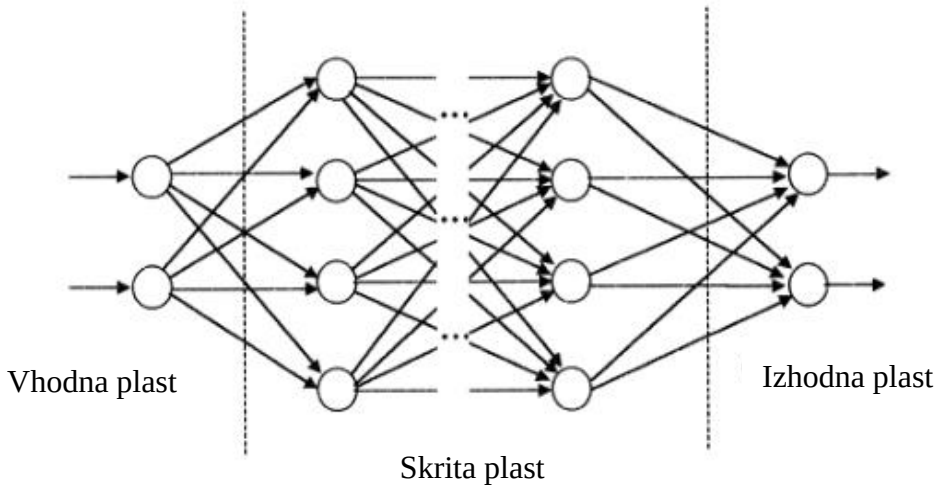
- **Ker je odprtokodna** – vsa koda te aplikacije je javno dostopna in prosta za brezplačno uporabo.
- **Vizualizacija podatkov z grafi** – poenostavi delo s strojnim učenjem in olajša reševanje problemov.
- **Možnost uporabe grafične kartice** – za pospešeno učenje, ki je na procesorju precej zamudno.
- **Kompatibilnost z jezikom Python** – v katerem so napisani vsi naši programi.
- **Pogosto posodabljanje knjižnice** – omogočajo uporabo najnovejše tehnologije s področja robotskega učenja.

4.2.2 Nevronske mreže

Človeški možgani so neverjetno prilagodljivi in zmožni reševati tudi probleme, ki jih ni mogoče predstaviti z algoritmom. Delujejo na podlagi iskanja vzorcev, ravno nasprotno od računalnikov, ki so izvrstni v izvajanju algoritmov. Zaradi višanja potrebe po avtomatizaciji problemov, kot sta prepoznava objektov s slike in govora, so se po zgledu iz narave v računalništvu razvile umetne nevronske mreže, ki poskušajo z računalniškimi algoritmi posnemati učenje človeških možganov. S tovrstnimi algoritmi so bile na nekaj področjih dosežene in presežene zmožnosti ljudi. [5]

4.2.2.1 Zgradba nevronske mreže

- **Vhodna plast** – Predstavlja neobdelano informacijo, ki jo vnesemo v mrežo.
- **Skrite plasti** – V vsaki izmed skritih plasti določimo aktivnost vhodne plasti in uteži med vhodno ter skrito plastjo. Obnašanje izhodne plasti je odvisno od aktivnosti v skritih plasteh in uteži med skritimi ter izhodnimi plastmi. Ker uteži med vhodno in skrito plastjo določajo, kdaj se katera izmed skritih plasti aktivira, lahko te plasti same zgradijo svojo predstavitev vhodnih informacij in zato s spreminjanjem uteži dosežemo, da skrite plasti same odločajo, katero informacijo bodo poslale naprej.
- **Izhodne plasti** – Izpišejo informacije, dobljene iz skritih plasti.



Slika 34: Znaki za nevarnost

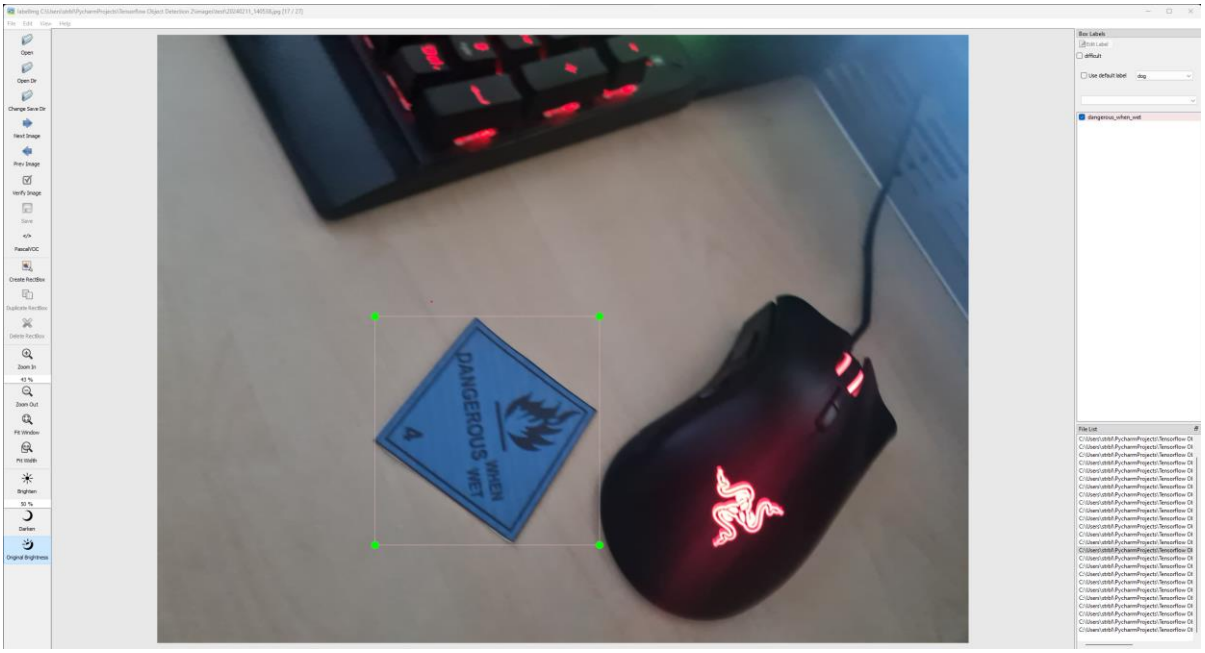
(Vir: https://dijaski.net/gradivo/rif_ref_nevronske_mreze_01)

4.2.3 Ssd mobilenet v2 320x320

SSD MobileNet v2 320x320 je nevronska mreža za zaznavanje objektov, ki je osnova za naš program za strojno učenje. Zanj smo se odločili, ker omogoča hitro in učinkovito zaznavanje več objektov v slikah. Pomembna lastnost, zaradi katere smo ga uporabili, je tudi njegova majhna poraba procesorja, kar pomeni, da zanj ne potrebujemo močnih računalnikov, kljub temu pa omogoča sprotno obdelovanje videoposnetkov in ne samo slik.

4.2.4 Realizacija zaznave znakov za nevarnost

Preden smo lahko začeli trenirati model za zaznavo, smo morali narediti referenčne slike. Te smo naredili s telefonom in jih prenesli na računalnik. Za največjo natančnost modela smo potrebovali veliko število čimbolj raznolikih slik. Nato smo na vsaki sliki posebej označili znake, ki jih bomo zaznavali, in jim določili razrede, v katere spadajo. To smo naredili s pomočjo programa pyqt5.



Slika 35:Program za označevanje slik

(Vir: osebni arhiv)

Ko smo označili vse slike, smo jih razdelili v dve mapi, in sicer mapo za treniranje in mapo za testiranje modela. Prvo mapo smo uporabili med treniranjem modela, zato je bila najobsežnejša. V drugi mapi je bilo nekaj slik, ki jih model še nikoli ni videl. Uporabili smo jo zato, da smo lahko določili njegovo natančnost.

Ko sta bili mapi s slikami pripravljeni, smo naložili vse potrebne knjižnice in začeli trenirati modele. Poizkusili smo z več različnimi količinami slik in različnim številom korakov. Če model treniramo z več koraki, je natančnejši, ampak treniranje traja veliko dlje.

4.2.5 Analize modelov

Željeni rezultat modela je pravokotnik okoli znaka z napisanim pravilnim imenom razreda, v katerega spada ta znak.

Slika 36: Željeni rezultat modela

(Vir: osebni arhiv)

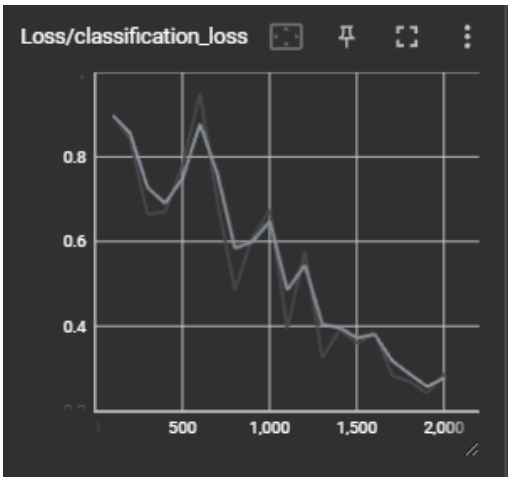
Razlaga neznanih pojmov:

- **Classification loss (izguba pri klasifikaciji)** meri, kako dobro model klasificira ali določa razrede za podane znake.
- **Localization loss (izguba lokalizacije)** meri, kako natančno model določa lokacijo objektov na sliki.
- **Regularization loss (izguba regularizacije)** nam pove, kako se model prilagaja na učne slike. Če se model na te slike preveč prilagodi, ne bo deloval na novih slikah, ki jih še ni videl.
- **Total_loss (skupna izguba)** predstavlja celotno izgubo. Je seštevek vseh ostalih izgub.
- **Steps per second (koraki na sekundo)** nam povedo, koliko korakov treniranja, ki jih sami vnaprej določimo, se opravi vsako sekundo.

Cilj je, da so vse izgube čim nižje.

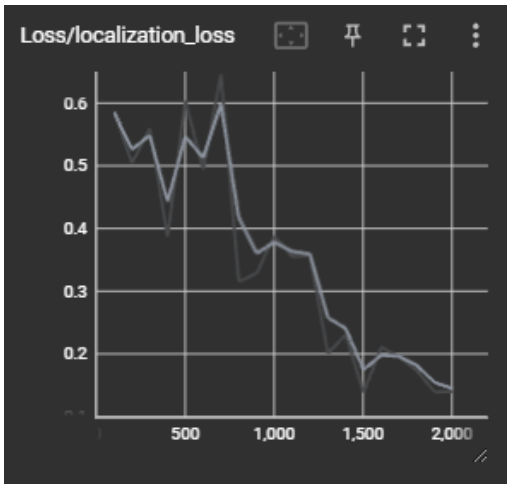
4.2.5.1 Prvi model

Prvi model smo trenirali z 9 referenčnimi slikami za vsakega od 9 razredov znakov za nevarnost, torej smo imeli skupaj 81 slik. Trenirali smo ga 2000 korakov. Iz rezultatov smo ugotovili, da je bil model zelo nenatančen. Treniranje je trajalo 42 minut.



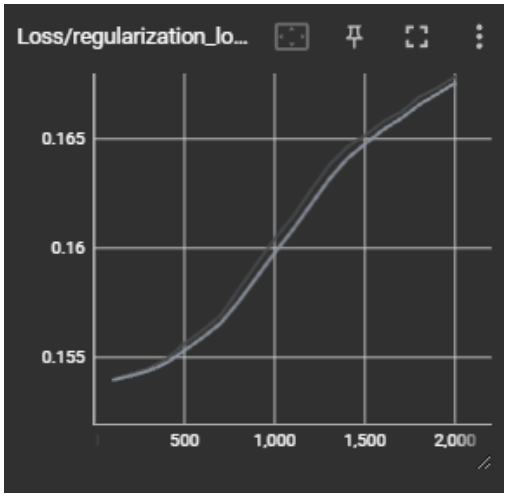
Slika 37: Izguba pri klasifikaciji

(Vir: osebni arhiv)



Slika 38: Izguba lokalizacije

(Vir: osebni arhiv)



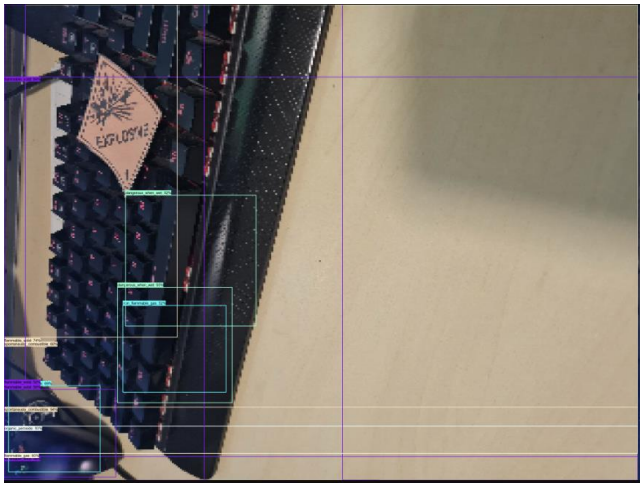
Slika 39: Izguba regularizacije

(Vir: osebni arhiv)

Slika 40: Skupna izguba

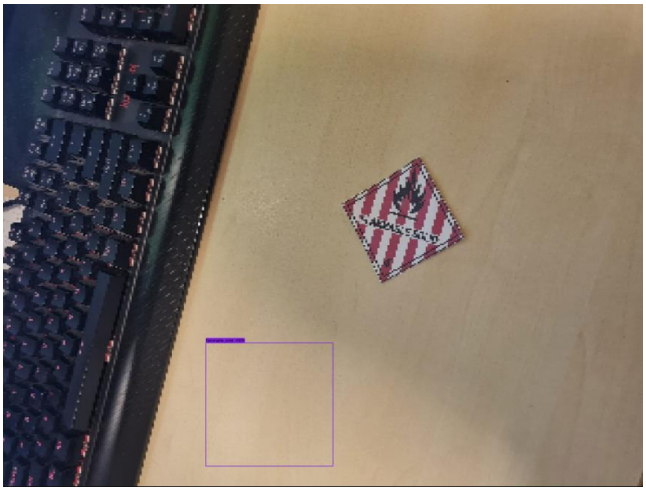
(Vir: osebni arhiv)

Z grafov je razvidno, da se vse izgube nižajo, ko treniramo za več korakov. Edina izjema pri tem je izguba regulacije, ki se viša. To je za model zelo slabo, kar lahko vidimo tudi v rezultatih. Ostale izgube pri koncu treniranja še padajo, kar nam pove, da bi bil model boljši, če bi ga trenirali več korakov.



Slika 41: Slika za preizkus natančnosti modela

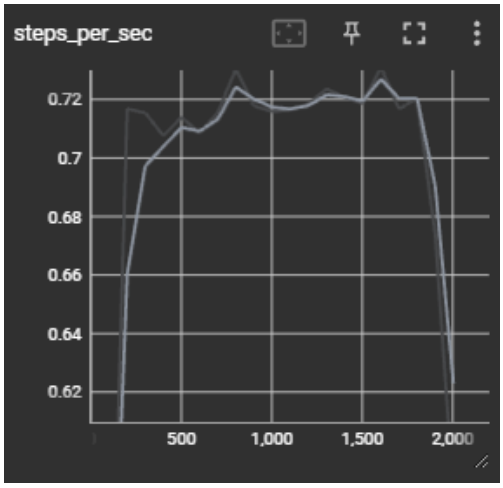
(Vir: osebni arhiv)



Slika 42: Slika za preizkus natančnosti modela

(Vir: osebni arhiv)

Na prvi preizkusni sliki lahko vidimo, da model ni našel znaka, prav tako ni pravilno označil kategorije, v katero spada. Zaznal je tudi veliko napačnih mest, kjer ni bilo znakov. To se je verjetno zgodilo zaradi slabe izgube regularizacije. Ker je to verjetno posledica premajhne količine referenčnih slik, smo za vse naslednje modele dodali več slik. Druga slika nam kaže veliko boljše rezultate, saj je model zaznal pravilen razred znaka, ni pa pravilno določil njegove pozicije. To smo poizkusili rešiti z več koraki.

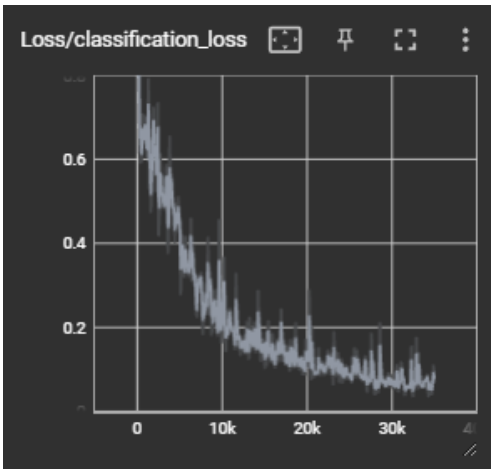


Slika 43: Koraki na sekundo

(Vir: osebni arhiv)

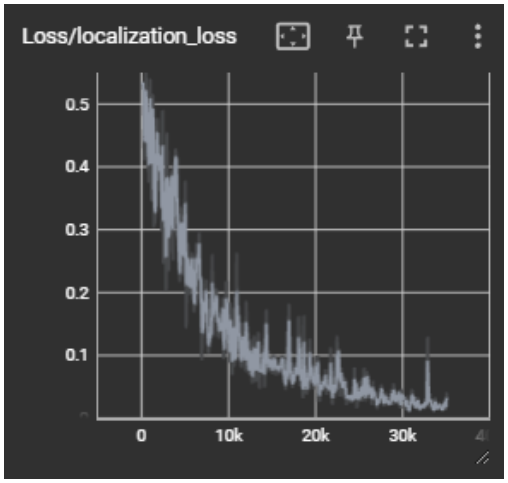
4.2.5.2 Drugi model

Drugi model smo trenirali z 12 referenčnimi slikami za vsakega od 9 razredov znakov za nevarnost, torej smo imeli skupaj 108 slik. Trenirali smo ga 35000 korakov. Rezultati so bili veliko boljši kot pri prvem modelu, ampak lokalizacija še vedno ni bila dovolj natančna. Treniranje je trajalo 13 ur in 25 minut.



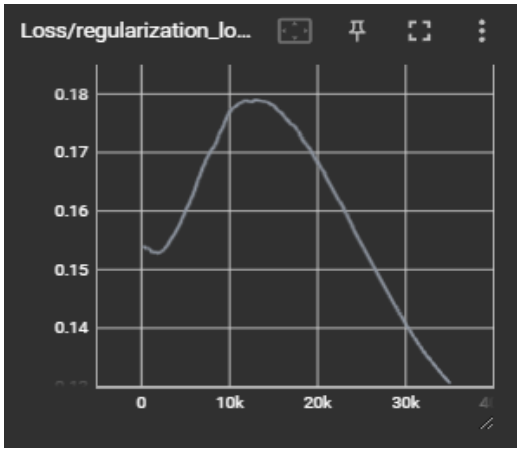
Slika 44: Izguba pri klasifikaciji

(Vir: osebni arhiv)



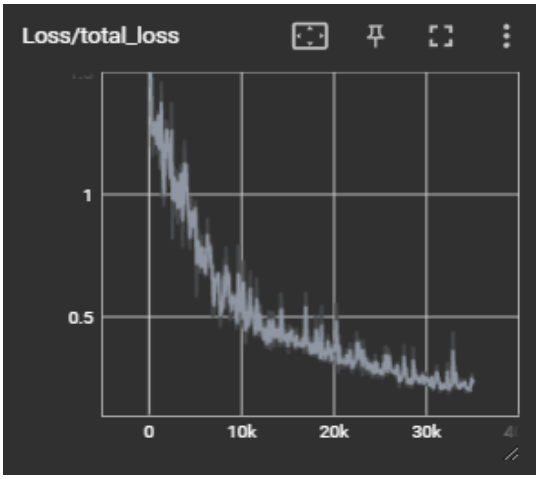
Slika 45: Izguba lokalizacije

(Vir: osebni arhiv)



Slika 46: Izguba regularizacije

(Vir: osebni arhiv)



Slika 47: Skupna izguba

(Vir: osebni arhiv)

Če primerjamo grafe prvega in drugega modela, lahko vidimo, da so dodatni koraki pomagali pri vseh izgubah. Največja razlika je opazna pri izgubi regularizacije, ki se je začela nižati šele pri 13500 korakih. Zato pri prvem modelu z 2000 koraki ta izguba ni padla. Z grafa je razvidno, da bi ta izguba lahko z več koraki še padla, ampak bi bilo daljše treniranje še bolj zamudno.

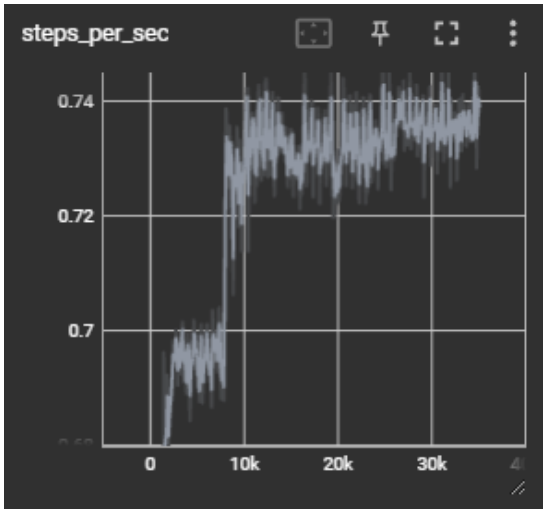
Slika 48: Slika za preizkus natančnosti modela

(Vir: osebni arhiv)

Slika 49: Slika za preizkus natančnosti modela

(Vir: osebni arhiv)

Na preizkusnih slikah vidimo, da je model vedno določil pravilno kategorijo znaka. Tudi napačnih mest, kjer ni znakov, ni več zaznaval. Edini problem tega modela je lokalizacija znaka na sliki. Ta ni bila uspešna niti v enem preizkusu.



Slika 50: Koraki na sekundo

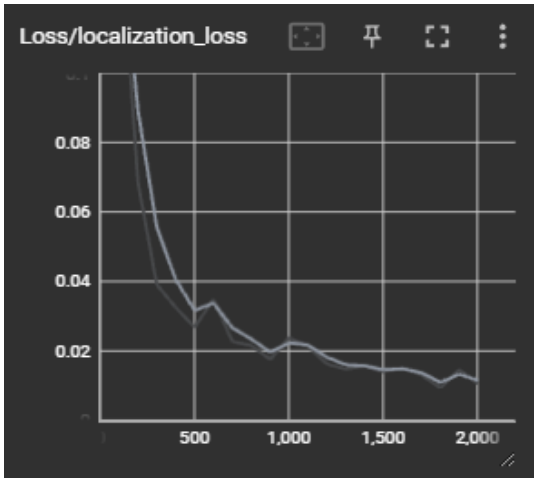
(Vir: osebni arhiv)

4.2.5.3 Tretji model

Tretji model smo trenirali z 28 referenčnimi slikami za vsakega od 9 razredov znakov za nevarnost, torej smo imeli skupaj 252 slik. Trenirali smo ga samo 2000 korakov, ker je treniranje s takšno količino slik počasno. Rezultati so bili boljši kot pri drugem modelu. Treniranje je trajalo 9 ur.

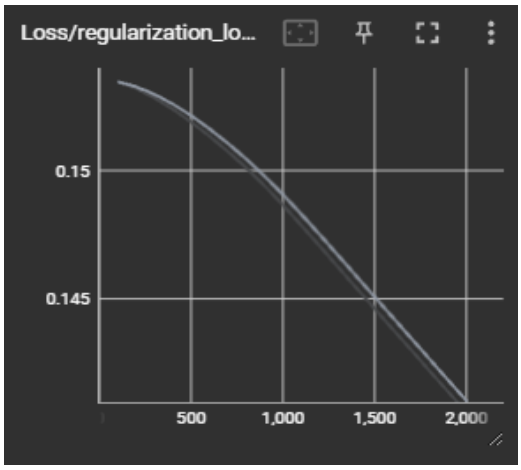
Slika 51: Izguba pri klasifikaciji

(Vir: osebni arhiv)



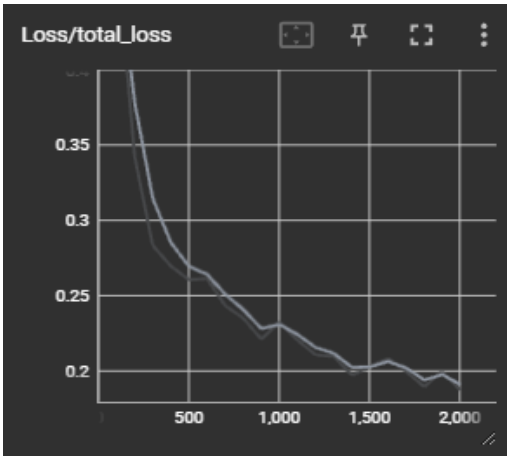
Slika 52: Izguba lokalizacije

(Vir: osebni arhiv)



Slika 53: Izguba regularizacije

(Vir: osebni arhiv)



Slika 54: Skupna izguba

(Vir: osebni arhiv)

Z grafa totalne izgube je razvidno, da je model že pri 2000 korakih natančnejši kot prejšnji model pri 35000 korakih. Torej lahko sklepamo, da je imel prejšnji model največjo težavo s premajhnim številom slik. Prav tako vidimo, da se izguba regularizacije začne takoj nižati.

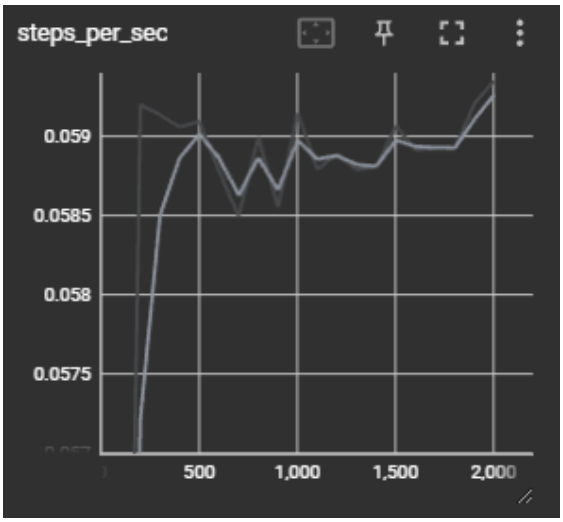
Slika 55: Slika za preizkus natančnosti modela

(Vir: osebni arhiv)

Slika 56: Slika za preizkus natančnosti modela

(Vir: osebni arhiv)

Na prvi preizkusni sliki lahko vidimo željeni rezultat. Na drugi sliki pa lokalizacija še vedno ni natančna. To bil lahko rešili z večjim številom slik in več koraki, ampak bi močno podaljšalo čas treniranja. Treniranje bi v tem primeru na našem računalniku lahko trajalo več dni ali tudi več tednov.



Slika 57: Koraki na sekundo

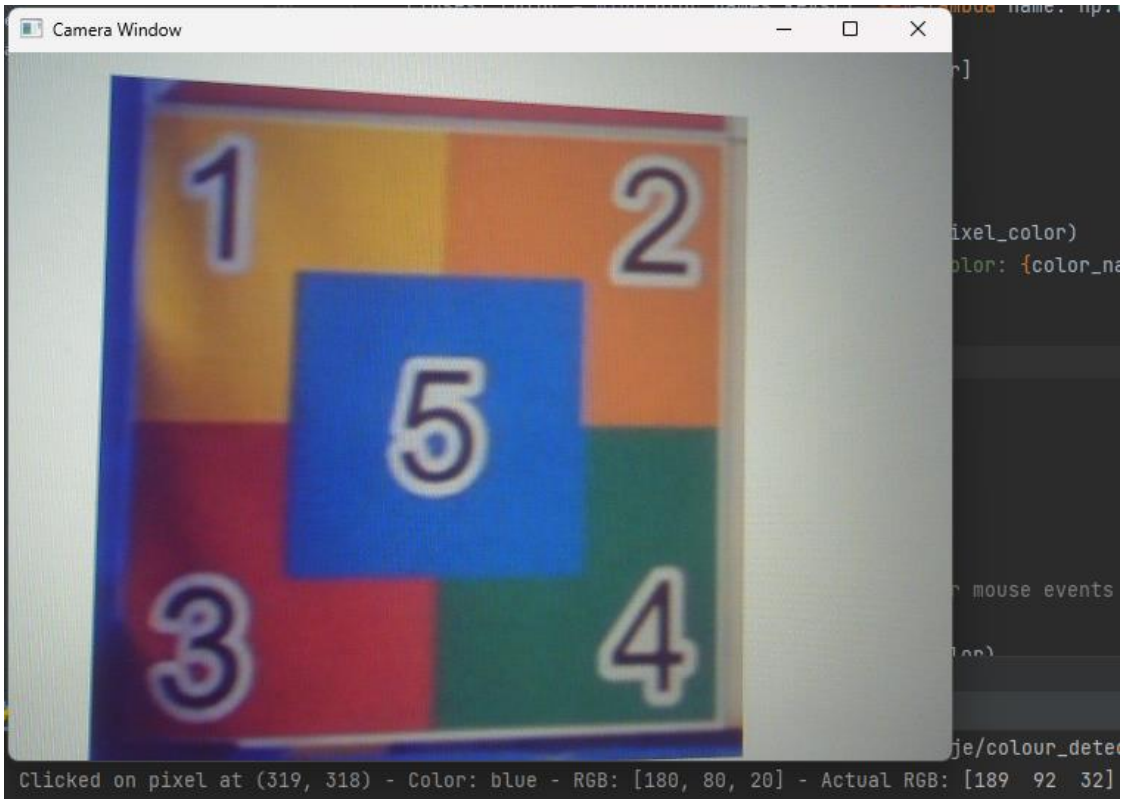
(Vir: osebni arhiv)

4.3 ZAZNAVANJE BARV

Zaznavanja barv smo se lotili s pomočjo knjižnice open-cv. To je knjižnica za Python, ki omogoča uporabo kamer in obdelovanje njihovih podatkov. Ker so v open-cv podatki o barvi posameznih pikslov shranjeni v obliki bgr (3 številke 0–255, ki predstavljajo količino modre, zelene in rdeče barve), smo najprej določili bgr vrednosti vsake barve. Določene so v seznamu, ki nam omogoča, da te vrednosti glede na pogoje, kot je svetloba, spreminjamo za največjo natančnost zaznavanja.

Ko smo določili vrednosti za vsako barvo, je moral program vedeti tudi, na katerem delu slike želimo barvo zaznati. Ta problem smo rešili s pomočjo funkcije v open-cv, ki nam omogoča zaznavo položaja miške na sliki in pritiska leve tipke.

Ko smo vedeli, na katerem pikslu je miška, smo s slike prebrali njegovo bgr vrednost in jo primerjali s seznamom vrednosti za posamezne barve, da smo ugotovili, vrednost katere barve je najbližja vrednosti piksla.



Slika 58: Primer zaznave modre barve s kamero in izpis v konzolnem oknu, ki nam pove barvo.

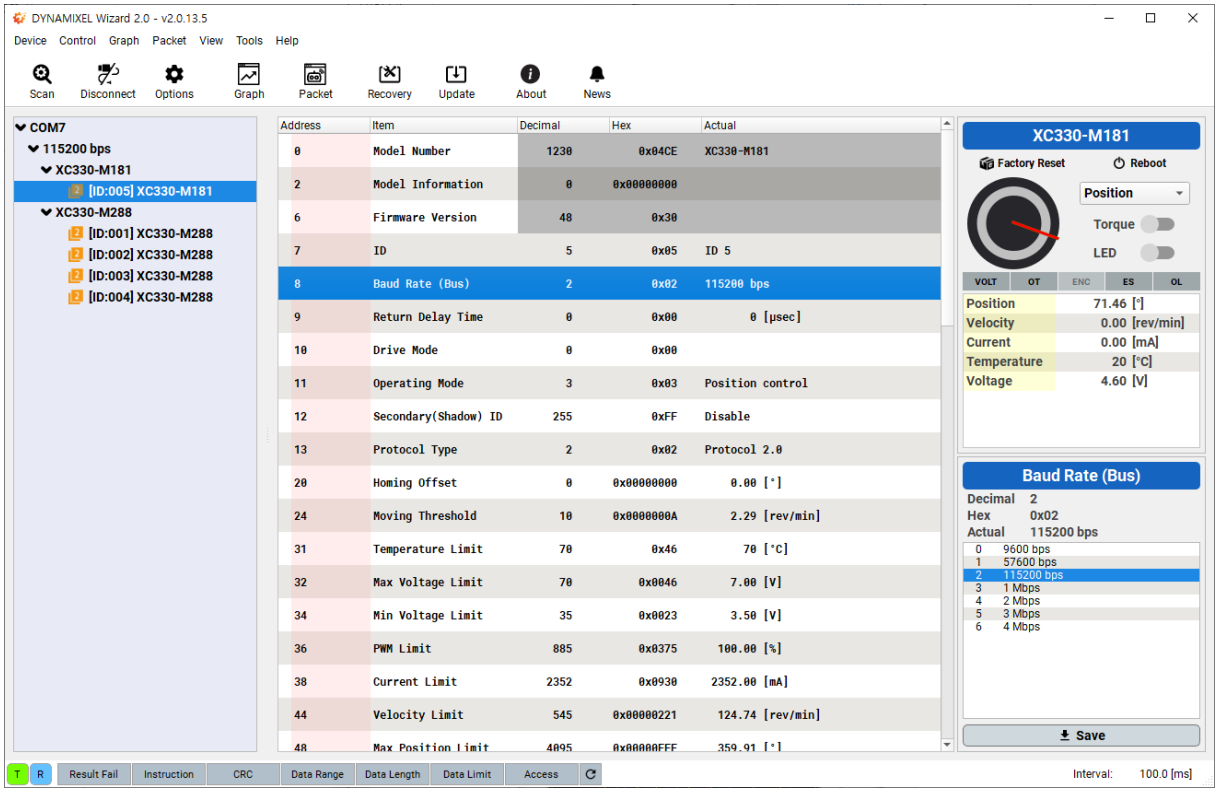
(Vir: osebni arhiv)

4.4 NADZOR ROBOTA S KONTROLERJEM

Za premikanje robota uporabljamo Dynamixel motorje, s katerimi se povežemo s pomočjo knjižnice dynamixel_sdk, za komunikacijo med kontrolerjem in računalnikom pa uporabljamo knjižnico pygame. Zanjo smo se odločili, ker omogoča branje vrednosti vsake igralne palice in gumba na kontrolerju z enostavnimi funkcijami.

Nadziranja robota smo se najprej lotili s pomočjo programa Dynamixel Wizard, s katerim lahko nadziramo vsak motor posebej, da testiramo njegovo brezhibno delovanje. Prav tako nam je ta program omogočil, da smo za vsak motor določili drugačno identifikacijsko številko, s pomočjo katere smo jih lahko pozneje ločili v programu. Poleg tega pa nam

program Dynamixel Wizard daje pomembne povratne informacije o temperaturi, obremenitvi in hitrost motorjev, ki pomagajo pri njihovem nadzoru.



Slika 59: Program Dynamixel Wizard

(Vir: osebni arhiv)

Ko smo končali s kontrolo vsakega motorja posebej, smo morali najti način, da bi nadzirali še vse motorje hkrati. To smo dosegli s programom v dveh delih.

Prvi del je s pomočjo podatkov o poziciji igralne palice na kontrolerju, ki so izražene z dvema številkama 0–1 izračunal, kako hitro se morajo vrteti leva in desna kolesa. Pri tem smo imeli težave, ker je hitrost motorja za vrtenje naprej določena s številkami 0–1023, za vrtenje nazaj pa 1024–2047. Ta razlika v vhodnih in izhodnih vrednostih je zahtevala precej kompleksne kalkulacije. Te težave smo rešili z naslednjimi računi.

```
Lmotor_speed     =     axis0     *     sp1     +     (axis1     *     sp2)
Dmotor_speed = axis0 * sp1 - (axis1 * sp2)
```

Slika 60: Rešitev problema pri kontrolerju

(Vir: osebni arhiv)

V teh izračunih Lmotor_speed in Dmotor_speed predstavljata hitrosti motorjev, axis1 predstavlja pomik igralne palice na kontrolerju gor in dol, axis0 pa pomik levo in desno.

Izračun je deloval dobro, dokler nismo igralne palice pomaknili v skrajno pozicijo, da je bila njena vrednost enaka 1 ali –1. Takrat so vrednosti hitrosti motorjev presegle maksimalne vrednosti in program je javil napako ter se ustavil.

To smo rešili z naslednjimi vrsticami kode.

```
if Lmotor_speed > 1023:
    Lmotor_speed = 1023
    Dmotor_speed = Dmotor_speed - (Lmotor_speed - 1023)

if Dmotor_speed > 1023:
    Dmotor_speed = 1023
    Lmotor_speed = Lmotor_speed - (Dmotor_speed - 1023)

if Lmotor_speed < (-1023):
    Lmotor_speed = (-1023)
    Dmotor_speed = Dmotor_speed - (Lmotor_speed + 1023)

if Dmotor_speed < (-1023):
    Dmotor_speed = (-1023)
    Lmotor_speed = Lmotor_speed - (Dmotor_speed + 1023)

if Dmotor_speed < 0:
    Dmotor_speed = 1023 + abs(Dmotor_speed)

if Lmotor_speed < 0:
    Lmotor_speed = 1023 + abs(Lmotor_speed)
```

Slika 61: Koda za spreminjanje hitrosti motorjev

(Vir: osebni arhiv)

Ta koda preverja, če je katera izmed mejnih vrednosti presežena in jo ponastavi nazaj na maksimalno vrednost, prav tako pa hitrosti na nasprotni strani doda nasprotno vrednost presežene vrednosti, da pospeši obračanje robota.

S tem smo zaključili prvi del programa in začeli drugi del. Cilj drugega dela je, da se poveže z motorji in jim s pomočjo USB-kabla pošlje izračunano hitrost. Ta naloga je precej enostavna, ker ima knjižnica dynamixel_sdk vgrajeno funkcijo, ki naredi prav to. S tem je bil program zaključen in sledilo je testiranje. Med testiranjem smo opazili, da program velikokrat javlja napako, ker povezava prek USB-priključka ni brezhibna.

```
Traceback (most recent call last):
  File "D:\pycharm\usje\motorji.py", line 69, in <module>
    portHandler.setBaudRate(BAUDRATE)
  File "C:\Users\strbl\AppData\Local\Programs\Python\Python310\lib\site-packages\dynamixel_sdk\port_handler.py", line 68, in setBaudRate
    return self.setupPort(baud)
  File "C:\Users\strbl\AppData\Local\Programs\Python\Python310\lib\site-packages\dynamixel_sdk\port_handler.py", line 114, in setupPort
    self.ser = serial.Serial(
  File "C:\Users\strbl\AppData\Local\Programs\Python\Python310\lib\site-packages\serial\serialwin32.py", line 33, in __init__
    super(Serial, self).__init__(*args, **kwargs)
  File "C:\Users\strbl\AppData\Local\Programs\Python\Python310\lib\site-packages\serial\serialutil.py", line 244, in __init__
    self.open()
  File "C:\Users\strbl\AppData\Local\Programs\Python\Python310\lib\site-packages\serial\serialwin32.py", line 64, in open
    raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))
serial.serialutil.SerialException: could not open port 'COM5': FileNotFoundError(2, 'The system cannot find the file specified.', None, 2)
```

Slika 62: Javljena napaka

(Vir: osebni arhiv)

Za rešitev te napake smo poizkusili uporabiti več različnih USB-kablov, ampak se je napaka vedno ponavljala, zato smo jo morali odpraviti programsko. Za to smo uporabili funkcijo 'try, except'. Ta funkcija deluje tako, da sledi kodi pod delom 'try', razen če je javljena napaka. V tem primeru začne slediti kodi v delu 'except', kamor smo napisali program za ponovno vzpostavitev povezave z motorji.

```
except:
    print("error")
    time.sleep(0.5)

    portHandler.closePort()
    print("port                                                    closed")

    time.sleep(1)
    portHandler                    =                PortHandler(DEVICENAME)
    packetHandler              =            PacketHandler(PROTOCOL_VERSION)
    while                      a                ==                      1:
        try:
            portHandler.openPort()
            a=0
        except:
            a=1
    portHandler.setBaudRate(BAUDRATE)
    print("port open")
```

Slika 63: Koda za ponovno vzpostavitev povezave z motorji

(Vir: osebni arhiv)

4.5 QR-KODE

Ena izmed nalog tekmovanja je tudi branje QR-kod. QR-kode so lahko nameščene na različnih lokacijah na polju in lahko vsebujejo različne podatke, ki jih moramo shraniti v določeno mapo. Zato smo preko kamere na roki naredili bralnik QR-kod.

Slika 64: Skeniranje QR-kode

(Vir: https://intuition-software.com/en/)

4.5.1 Kaj so QR-kode?

QR-koda je vrsta dvodimenzionalne kode, ki jo je mogoče prebrati s pomočjo pametnega telefona ali drugih naprav s kamero. Sestavljene so iz črno-belih kvadratov in vsebujejo informacije, kot so besedilo, URL-povezave , WiFi omrežja in še več. [17]

4.5.2 Kako deluje branje QR-kode?

Za branje QR-kode smo najprej omogočili dostop do kamere. Program je moral omogočati zajem slike s kamero in jo nato analizirati, da bi jo prepoznal. To smo dosegli s pomočjo programske knjižnice, in sicer OpenCV. Na koncu se vsi rezultati (vsebine QR-kod) izpišejo na zaslonu in shranijo v mapo.

4.5.3 Koraki za branje QR-kode

1. Zajem slike: Ta korak se zgodi na začetku funkcije extract, kjer smo uporabili knjižnico OpenCV za zajem slike s kamero.
2. Pretvorba slike v črno-belo: Ta korak se prav tako zgodi v funkciji extract, kjer pretvorimo zajeto sliko v črno-belo za lažjo obdelavo.
3. Prepoznavanje QR-kode: Prepoznavanje QR-kode se začne z iskanjem črnih kvadratkov na sliki. To se zgodi v isti funkciji extract, kjer smo uporabili funkcijo findContours, da poišče vse oblike na sliki, nato pa filtrira le tiste, ki bi lahko bile QR-kode.
4. Prepoznava QR-kode: QR-kodo prepoznamo s knjižnico pyzbar.
5. Branje vsebine QR-kode: Ko je QR-koda prepoznana, jo program okviri v rdeč kvadrat, kot lahko vidimo na sliki 65. Naslednji korak je bil prikaz rezultatov.
6. Prikaz rezultatov: Vsebino najdenih QR-kod ali morebitnih razbranih vsebin smo shranili v določeno mapo. Vsebino QR-kod pa smo sproti videli na prikazovalniku.

Slika 65: Bralnik QR-kode

(Vir: osebni arhiv)

5 ANALIZA IN IZBOLJŠAVE

5.1 ANALIZA HIPOTEZ

1. S pomočjo strojnega učenja bomo znake za nevarnost prepoznali v manj kot 1 sekundi: ✓
   Reševalnega robota smo uspešno opremili s strojnim učenjem in algoritmi za obdelavo slik. S tem smo omogočili, da je robot znake za nevarnost prepoznal v manj kot 1 sekundi po vstopu v nevarno območje.
2. Na sliki bomo vedno našli pravilno mesto, kjer se nahaja znak za nevarnost: ✗
   Kljub pravilno ugotovljenemu razredu znakov za nevarnost nismo vedno našli njihove lokacije na sliki.
3. Robot bo lahko premagal stopnico, visoko 10 cm: ✓
   Robot ima fleksibilne nastavke na pogonskih oseh in pogon na vsa štiri kolesa z zobniškim reduktorjem, zato je zmožen premagati stopnico, visoko 10 cm.
4. Zmožen se bo obrniti na mestu v labirintu: ✓
   S pomočjo nadzora robota s kontrolerjem in kompaktnim dizajnom smo dosegli ozek obračalni krog, ki nam omogoča spreminjanje smeri v labirintu.
5. Sestavni deli robota bodo hitro zamenljivi: ✓
   Vsi deli robota se lahko v primeru okvare z uporabo osnovnega orodja v roku 10 minut zamenjajo. To nam omogoča modularna zgradba, sestavljena z vijačno zvezo.

5.2 NADALJNJE IZBOLJŠAVE

Nadgradnja sistema za zaznavanje in prepoznavanje nevarnosti: Kljub uspešnosti sistema za zaznavanje znakov za nevarnost bi lahko sistem nadgradili z uporabo naprednejših algoritmov in tehnik strojnega učenja za še večjo natančnost ter hitrost prepoznavanja. Prav tako bi lahko razširili spekter prepoznavanja znakov na splošno.

Izboljšanje mobilnosti in stabilnosti robota: Čeprav robot uspešno premaguje različne terene in ovire na poti, so še možne izboljšave. Lahko bi povečali moč motorjev in s tem navor na kolesih. Prav tako pa bi lahko dodali poseben mehanizem na robota (podoben repu), ki bi omogočil, da bi splezal čez višje ovire.

Optimizacija prenosa podatkov in uporabe VR-tehnologije: Da bi izboljšali vodenje in nadzor na robotom, bi lahko na robota dodali kamero, ki bi zajela vso okolico (360°). Podatke s kamere bi prenesli na VR-očala, s katerimi bi lahko spremljali okolico robota, in ga vodili preko VR-okolja.

6 ZAKLJUČEK

Razvoj v okviru tekmovanja RoboCupRescueRobot League je predstavljal izjemno pomembno izkušnjo, ki nam je omogočila pridobiti dragocene veščine in znanje na področju robotike, inženiringa ter strojnega učenja. Skozi proces izdelave reševalnega robota smo se soočili z različnimi tehničnimi in organizacijskimi izzivi, ki so zahtevali celovit pristop k reševanju problemov. Cilje, ki smo si jih zadali, smo zadovoljivo opravili in naredili delujočega robota. V sklopu raziskav imamo še veliko prostora za izboljšave, predvsem v segmentu robotske roke in VR-okolja.

7 VIRI IN LITERATURA

[1] Amazon. 10 Bearing 6800-2RS 10x19 Sealed 10x19x5 Ball Bearings VXB Brand (online). 2019. (citirano 25. 1. 2024).

Dostopno na naslovu: https://www.amazon.com/Bearing-6800-2RS-10x19x5-Bearings-VXB/dp/B002BBOCP0

[2] Botland. Robot gripper (online). 2020. (citirano 1. 2. 2024).

Dostopno na naslovu: https://botland.store/withdrawn-products/10554-makeblock-86502-robot-gripper-for-ranger-ultimate-black5904422316396.html

[3] Cad/cam/cae tehnologije. Creo Parametric (online). 2017. (citirano 13. 1. 2024). Dostopno na naslovu: http://cadcam.spts.si/?page_id=5

[4] Creality. 3D printers (online). 2015. (citirano 22. 12. 2023).

Dostopno na naslovu: https://www.creality.com/

[5] Dijaski.net. Nevronske mreže (online). 2013. (citirano 24. 1. 2024). Dostopno na naslovu: https://dijaski.net/gradivo/rif_ref_nevronske_mreze_01

[6] Elp. camera (online). 2020. (citirano 5. 2. 2024).

Dostopno na naslovu: https://www.elpcctv.com/16-megapixel-120-degree-no-distortion-hd-usb-micro-camera-module-wide-angle-p-361.html

[7] Grabcad. DC Fan (online). 2016. (citirano 22. 12. 2023).

Dostopno na naslovu: https://grabcad.com/library/40mm-dc-fan-1

[8] Grabcad. Dynamixel AX-18A (online). 2014. (citirano 18. 12. 2023).

Dostopno na naslovu: https://grabcad.com/library/dynamixel-ax-18a-robotis-1

[9] Grabcad. Robotis U2D2 (online). 2019. (citirano 15. 12. 2023).

Dostopno na naslovu: https://grabcad.com/library/robotis-u2d2-1

[10] Notranje sile. Varnost v strojništvu (online). 2017. (citirano 25. 1. 2024). Dostopno na naslovu: http://lab.fs.uni-lj.si/lasok/index.html/gradivo_jerman_OTV/VvS_06__Statika_D__NotranjeSile.pdf

[11] Računalničke novice. 3D-tiskanje funkcionalnih delov (online). 2023. (citirano 2. 2. 2024). Dostopno na naslovu: https://racunalniske-novice.com/3d-tiskanje-funkcionalnih-delov/

[12] Robots. Dynamixel AX-12 (online). 2010. (citirano 1. 2. 2024). Dostopno na naslovu: https://www.duplichecker.com/domain-age-checker.php

[13] Strojna. Planetna gonila (online). 2017. (citirano 22. 1. 2024). Dostopno na naslovu: https://strojna.si/strojni-park/planetna-gonila-4-generacija/

[14] Traceparts. Knjižnica 3D-datotek (online). 2009. (citirano 12. 12. 2023). Dostopno na naslovu: https://info.traceparts.com/designers/engineers-and-designers/

[15] UltiMaker. How to use heat set inserts to securely fasten 3D printed parts (online). 2010. (citirano 5. 1. 2024). Dostopno na naslovu: https://ultimaker.com/learn/how-to-use-heat-set-inserts-to-securely-fasten-3d-printed-parts/

[16] Wikipedija. 3D-tiskanje (online). 2013. (citirano 12. 2. 2024). Dostopno na naslovu: https://sl.wikipedia.org/wiki/Raspberry_Pi#cite_note-2

[17] Wikipedija. Koda QR (online). 2011. (citirano 5. 2. 2024). Dostopno na naslovu: https://sl.wikipedia.org/wiki/Koda_QR

[18] Wikipedija. Raspberry Pi (online). 2011. (citirano 10. 2. 2024). Dostopno na naslovu: https://sl.wikipedia.org/wiki/Raspberry_Pi#cite_note-2