HITRO PROTOTIPIRAN TERENSKI ROBOT

Raziskovalna naloga

Avtorji:
Nejc LOČIČNIK, M-4. c
Žan SOTOŠEK, M-4. c
David ŽURAJ, M-4. c

Mentorja:
mag. Matej VEBER, univ. dipl. inž.
mag. Andro GLAMNIK, univ. dipl. inž.

Mestna občina Celje, Mladi za Celje

Celje, 2018

IZJAVA*

Mentorja, Matej Veber in Andro Glamnik, v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljava, da je v raziskovalni nalogi z naslovom Hitro prototipiran terenski robot, katere avtorji so Nejc Ločičnik, Žan Sotošek in David Žuraj:

- 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, _____ žig šole Podpis mentorja

Podpis odgovorne osebe

* POJASNILO V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja fotografskega gradiva, katerega ni avtor raziskovalne naloge, pa hrani šola v svojem arhivu.

DOVOLJENJE ZA OBJAVO AVTORSKE FOTOGRAFIJE V RAZISKOVALNI NALOGI

Podpisani Nejc Ločičnik, Žan Sotošek, David Žuraj izjavljamo, da smo avtorji fotografskega gradiva, navedenega v priloženem seznamu in dovoljujemo v skladu z 2. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, da se lahko uporabi pri pripravi raziskovalne naloge pod mentorstvom Mateja Vebra in Andra Glamnika, z naslovom Hitro prototipiran terenski robot, katere avtorji so Nejc Ločičnik, Žan Sotošek, David Žuraj.

Dovoljujemo tudi, da sme Osrednja knjižnica Celje vključeno fotografsko gradivo v raziskovalno nalogo objaviti na knjižničnih portalih z navedbo avtorstva v skladu s standardi bibliografske obdelave.

Celje, _____ Podpis avtorjev:

Priloga:

- seznam fotografskega gradiva

ZAHVALA

Zahvaljujemo se našima mentorjema mag. Mateju Vebru, univ. dipl. inž., in mag. Andru Glamniku, univ. dipl. inž., za koordinacijo celotne raziskovalne naloge in strokovne nasvete. Za ves čas, trud, potrpežljivost in spodbudne besede, ki sta nam jih namenila.

Zahvaljujemo se šoli in ravnateljici Simoni Črep, prof., za uporabo šolskih orodij, prostorov in 3D-tiskalnika, s katerim smo natisnili celotnega robota.

Zahvalili bi se tudi vsem, ki so nam tako ali drugače pomagali pri izpeljavi našega projekta, pa četudi je bila le spodbudna beseda, majhen nasvet ali pomoč pri delu.

Na koncu bi se zahvalili Brigiti Renner, prof., za pregled in lektoriranje naloge ter Simoni Tadeji Ribič, prof., za lektoriranje povzetka v angleškem jeziku.

HITRO PROTOTIPIRAN TERENSKI ROBOT

POVZETEK

Po celem svetu se dogajajo naravne katastrofe, ki prinašajo veliko smrtnih žrtev. Po pustošenju narave pridejo na kraj nesreče reševalne ekipe, ki se izpostavijo nevarnostim in skušajo najti ponesrečence. Ob vzpodbudi tekmovanja RoboCup smo se odločili za izdelavo reševalnega robota, ki bi opravljal razne naloge na nevarnih območjih, s katerimi se srečujejo reševalci. Najprej smo se spoznali s pravilnikom in zahtevami tekmovanja. Z brskanjem po spletu in lastnim znanjem smo zasnovali primerno konstrukcijo. Sledilo je 3D-tiskanje sestavnih delov in njihova montaža z elektronskimi komponentami. Nastali izdelek smo opremili s programsko opremo in nastal je željeni robot. S testiranjem smo preizkusili njegovo uporabnost in trpežnost. Uspelo nam je izdelati reševalnega robota, ki zadošča standardom tekmovanja in je odlična popotnica za izdelavo profesionalnega.

Ključne besede: Reševalni robot, robotska roka, Phyton, brezžična povezava, 3D-tiskanje, RoboCup RRMR

RAPIDLY MANUFACTURED ALL-TERRAIN ROBOT

SUMMARY

Natural catastrophes happen all around the globe and bring along numerous casualties. After the devastation rescuers arrive and put themselves in danger to help the casualties. Thanks to the RoboCup initiative we decided to design and produce a rescue robot, which is capable of performing various tasks in dangerous environments. We got familiar with the competition regulations and requirements and then proceeded to search through the web and use our own knowledge to construct the robot. We equipped it with software, thus creating a finished robot. We thoroughly checked for errors and tested its capabilities and durability. We succeeded in creating a device which satisfies the competition standards and is a great stepping stone to creating a professional one.

Keywords: Rescue robot, robotic arm, Python, wireless connection, 3D-printing, RoboCup RRMR

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

KAZALO PRILOG

UPORABLJENE KRATICE

PVC – Polivinilklorid

CAD – Computer Aided Design

RPi – Raspberry Pi

FDM – Fused Deposition Modelling

RRMR – Rescue Rapidly Manufactured Robot

1 UVOD

Živimo v času, ko se nam zdi, da smo ljudje vsemogočna bitja, ki vladamo in nadzorujemo svet. To bi lahko držalo, ampak še vedno obstaja nekaj, česar nismo zmožni obvladovati. To je tako imenovana »mati narava«, ki nas znova in znova preseneča s svojo močjo in nepredvidljivostjo. Znanstveniki se trudijo, da bi čim bolje preučili obnašanje sveta in ga v nadaljnje uspeli voditi po svojih željah. To nas raziskovalce ne zanima toliko, kakor posledice, ki jih prinese besnenje narave. Ena izmed najhujših naravnih nesreč je zagotovo potres. Še posebej, če se zgodi na območju, kjer so hiše, bloki, stolpnice na enem mestu. Ob premiku tektonskih plošč se sila prenese vse do površja in povzroči silovite tresljaje Zemljinega površja. Zgradbe ne prenesejo tolikšnih obremenitev, zato se porušijo. Iz zgodovine vemo, da takšni dogodki terjajo veliko žrtev. Tudi na Slovenskem poznamo takšne dogodke, na primer potres leta 1895 v Ljubljani.

»V Ljubljani in okolici je umrlo sedem oseb, nekaj pa jih je bilo ranjenih.« (https://sl.wikipedia.org/wiki/Ljubljanski_potres_1895#%C5%BDrtve_potresa, 11. 2. 2018, Potres v Ljubljani)

Ob takšnih nepredvidljivih situacijah so gasilci in reševalci primorani nemudoma odhiteti na kraj nezgode in pričeti z iskanjem in reševanjem ponesrečencev. Tako so neposredno ogroženi tudi sami. Skoraj v vsaki reševalni akciji se zgodi, da reševalec postane žrtev.

Porodila se nam je ideja, da bi izdelali daljinsko vodenega robota, ki bi bil zmožen premagati najrazličnejši teren in prevoziti luknje, v katere se človek ni zmožen splaziti. Naš namen je, da se reševalci ne bi izpostavljali nesrečam, temveč da bi ostali na varnem in preko slike, ki bi jo snemal robot, upravljali z njim.

Slika 1-1: Reševalni robot [12]

1.1 STRUKTURA RAZISKOVALNEGA DELA

Nismo prvi, ki razmišljamo v tej smeri. Po vsem svetu se razvijajo sistemi in mobilni roboti, ki skušajo olajšati reševanje ter zaščititi reševalce pred nesrečami. Zaradi vse večjega interesa po tovrstni robotike se je ustanovila RoboCup Rescue Robot League, ki je mednarodna skupnost ekip iz celega sveta. Izvršitelj tekmovanja je US Department of Commerce National Institute of Standards and Technology, v sodelovanju z mednarodnim odborom organizatorjev. Ta skupnost uporablja tekmovanja in učne tabore za napredek na področju robotike. Po natančnem pregledu njihove spletne strani: [*http://oarkit.intelligentrobots.org/home/home/about-us/] (11. 2. 2018),* smo ugotovili, da je to prava pot do našega uspeha. Imajo točno določene kriterije in naloge, ki jih mora rešiti robot.

»A wide variety of challenges face response robots operating in confined spaces. This competition encourages teams to develop solutions to problems that include driving over rough terrain, sensing the environment, picking up and delivering objects, intuitive operator control and autonomous behaviours. Solutions to these challenges will be highly interdisciplinary, covering mechanics, electronics, computing and art. The competition is structured so as to reward teams that do well overall, as well as teams that demonstrate innovative advances in specific challenges.« (http://oarkit.intelligentrobots.org/home/the-arena/, 11. 2. 2018, The Challenge)

Prevod: Pestra izbira izzivov za funkcionalnost robota v omejenih prostorih. To tekmovanje spodbuja ekipe k razvijanju rešitev za probleme, ki vsebujejo vožnjo čez grob teren, čutenje okolja, pobiranje in dostavljanje predmetov, inovativne kontrole za opravljanje in avtonomno vedenje. Rešitve teh problemov bodo močno interdisciplinarne, pokrivale bodo mehaniko, elektriko, računalništvo in umetnost. Tekmovanje je zgrajeno tako, da nagrajuje ekipe, ki se nasploh dobro odrežejo, prav tako kot ekipe, ki demonstrirajo inovativne napredke pri specifičnih izzivih.

1.2 HIPOTEZE

Cilj naše raziskovalne naloge je preučiti tekmovalno polje in na podlagi tega zasnovati ter izdelati mobilnega robota, ki bo zadoščal vsem kriterijem zgoraj omenjenega tekmovanja. Pri tem smo si zadali nekaj najpomembnejših hipotez:

- H1 – dolžina in širina robota ne bosta presegli 230 mm x 230 mm;
- H2 – imeti mora robotsko roko, opremljeno s kamero in primežem;
- H3 – konstrukcija mora biti 3D natisnjena;
- H4 – robot mora biti zmožen premagovati ovire do višine 150 mm;
- H5 – komunikacija med robotom in upravljavcem mora biti brezžična;
- H6 – motorji morajo biti Dynamixel AX-12A znamke Robotis;
- H7 – mikroračunalnik mora biti Raspberry Pi;
- H8 – robot mora biti odporen na trke in prah;
- H9 – sestavni deli morajo imeti možnost hitre menjave;
- H10 – upravljanje robota mora biti enostavno.

1.3 METODE RAZISKOVANJA

Tekmovanje naj bi bilo »open source«. To pomeni, da bi vsaka ekipa, ki bi tekmovala na tem tekmovanju, objavljala svoje ideje, sestavne dele, program in ugotovitve na skupnem forumu. Tako bi nove ekipe lažje začele z delom. Mi tega foruma nismo našli ali pa celo ne obstaja, tako da smo imeli zelo malo virov za nadaljnje delo. Da bi zastavljene cilje čim bolje realizirali, smo si pomagali z dvema metodama:

- metodo analize, ki temelji na razčlenitvi celote na njene osnovne sestavne dele. Tako smo si našo kompleksno nalogo razdelili na več delov, kot so: snovanje, konstruiranje, električno vezje, programiranje itd. S tem je bilo delo bolj sistematično in nazorno;
- primerjalno metodo, ki temelji na primerjavi več ugotovitev. To metodo smo največkrat uporabili pri snovanju. Svoje ideje o konstrukciji smo primerjali z ostalimi raziskovalnimi in mobilnimi roboti.

2 TEKMOVANJE ROBOCUP RESCUE

Gre za zelo priznano robotsko tekmovanje s poudarkom na mobilni robotiki in humanoidih. Deli se na JUNIOR in MAJOR. Že iz imena ugotovimo, da je prva kategorija namenjena mlajšim tekmovalcem od 12 do 19 let. Prav tako so tekmovanja na nižji zahtevnostni stopnji. Druga kategorija je bistveno zahtevnejša. V njej sodelujejo tudi univerze in tekmovalci z nazivom doktor. Kategorija MAJOR se deli na 4 kategorije. Ena izmed njih je tudi naša, RESCUE. Preostale tri so: @HOME, INDUSTRIAL in SOCCER. Tudi kategorija RESCUE se deli na dve tekmovanji. Eno je namenjeno dijakom do 19. leta starosti, druga pa starejšim. [19]

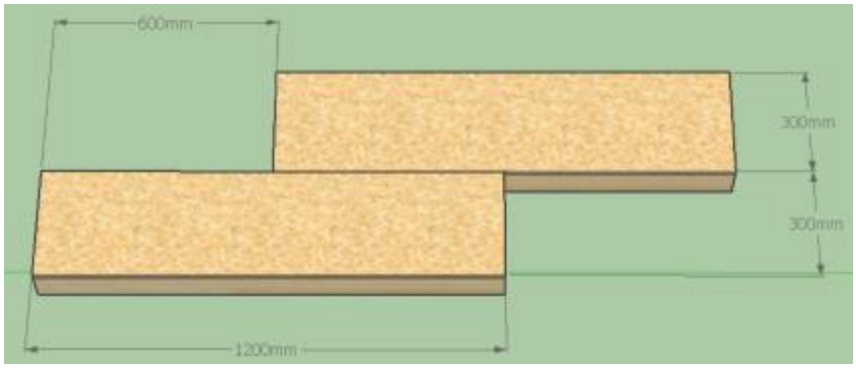
Cilj tekmovanja je, da ekipa izdela mobilnega robota, ki je lahko avtonomen ali pa ga upravlja en član ekipe. Pilot ne sme biti v času tekmovanja v bližini tekmovalnega poligona. Ne sme imeti neposrednega stika z robotom in tekmovalno površino. To pomeni, da mora ekipa primerno opremiti svojega robota z vso potrebno senzoriko, tako da lahko pilot nemoteno upravlja z njim. Zanimivo je, da tekmovanje ni časovno omejeno, temveč se točkuje odzivnost robota na terenu, natančnost krmiljenja, kakovost senzorjev, ponovljivost voženj posameznih sektorjev poligona itn.

Slika 2-1: Tekmovalci RoboCup RRMR 2017 [23]

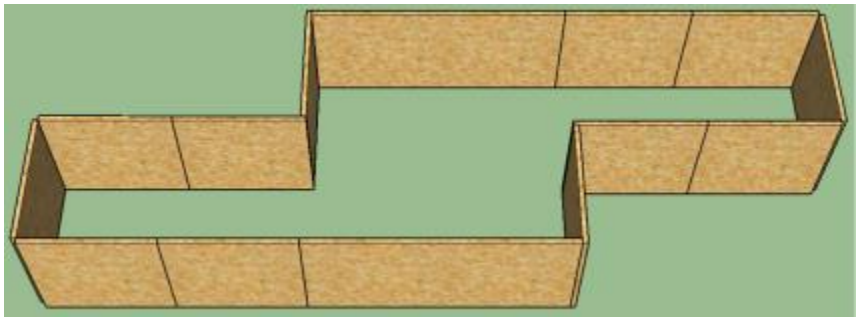
2.1 TEKMOVALNA POVRŠINA

Tekmovanje se odvija v dvorani, mi pa smo razvijali robota, ki naj bi se gibal v naravi, po ruševinah, predorih in ostalem raznovrstnem reliefu. Načeloma dvorana in narava nimata podobne površine, vendar so organizatorji tekmovanja poskrbeli za podobnost. Z opazovanjem narave in posledic, ki jih prinesejo naravne katastrofe, so organizatorji skušali narediti poligon, ki naj bi zajemal čim več preprek, s katerimi se lahko robot sreča v realnosti.

Določili so velikost tekmovalne površine. Stranske stene, ki omejujejo gibanje robota, so visoke 300 mm. [2]



Slika 2-2: Dno tekmovalne površine [2]

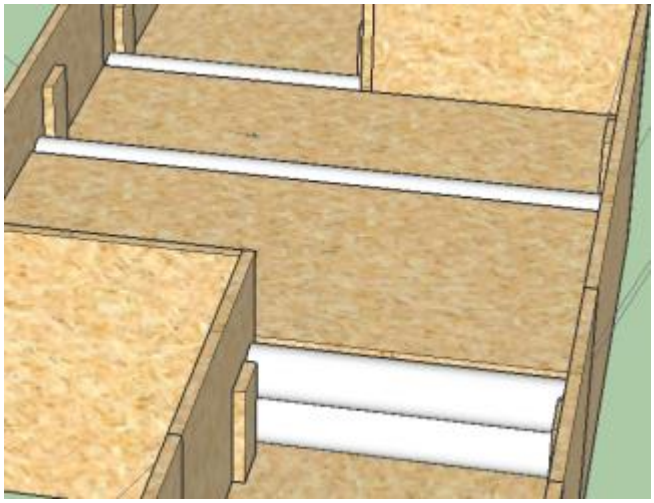


Slika 2-3: Stene tekmovalne površine [2]

Na tekmovanju je 24 tekmovalnih površin. Vsaka ima drugačno notranjo zgradbo oziroma relief. V vsakem polju se robot sreča z različnimi izzivi, zato mora ekipa dobro premisliti in skonstruirati robota, ki je zmožen opraviti z vsemi izzivi. [2]

2.1.1 Okrogla prepreka

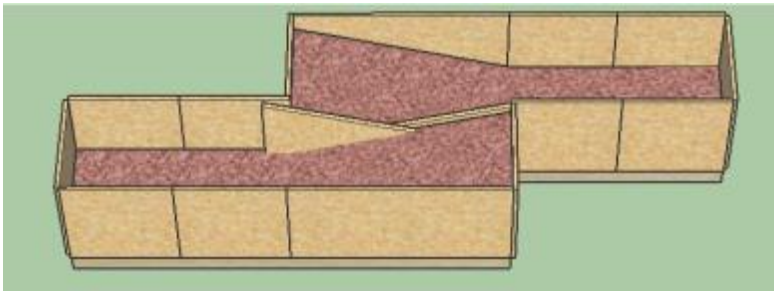
Te vrste preprek preverjajo, ali je robot zmožen premagati stopnico v višini 100 mm. Stopnica je sestavljena iz dveh 50 mm debelih, okroglih PVC-cevi. V drugo smer sta narejeni dve stopnici in vsaka vsebuje po eno cev. Te niso fiksirane, zato se ob stiku vrtečega predmeta vrtijo. To pomeni, da ima pogonski sklop robota zelo majhen oprijem. [14]



Slika 2-4: Okrogla prepreka [14]

2.1.2 Peščena podlaga

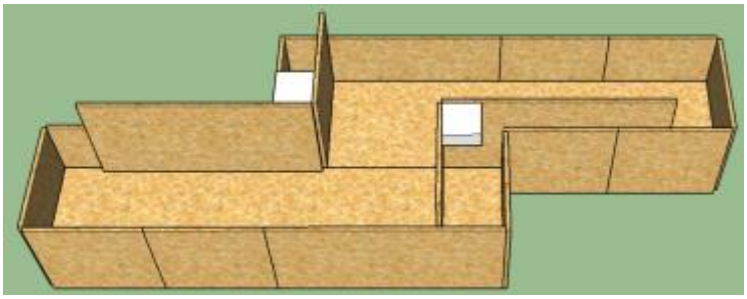
Z vsake strani je narejen 30˚klanec. Torej se 2 klanca na sredini križata. Vse polje je posuto z mivko, s peskom ali z žagovino. Posut material ni prilepljen ali na kakršenkoli način pritrjen. Tako robot nima veliko oprijema na površini, še posebej ne v klancu. [14]



Slika 2-5: Peščena podlaga [14]

2.1.3 Slalom

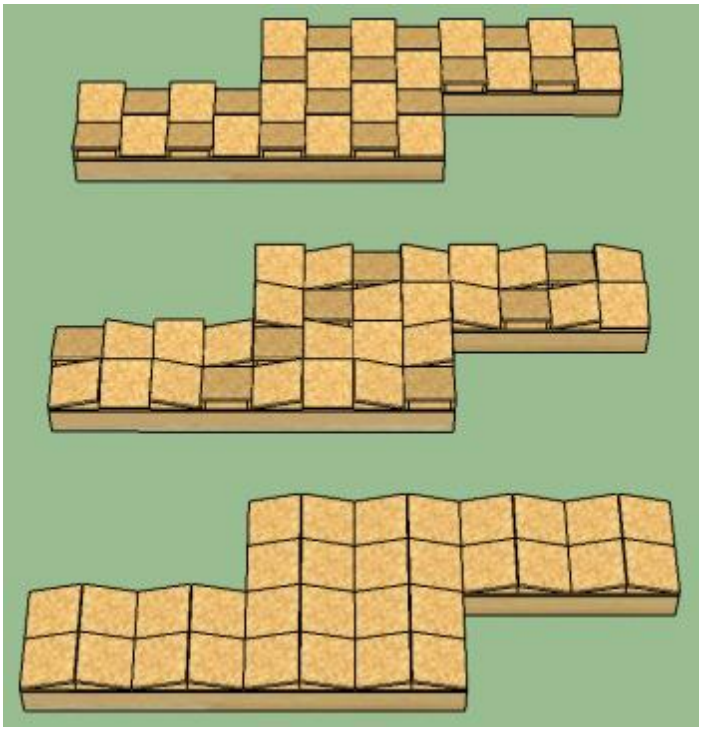
Postavljeni sta dve steni v obliki črke L, ki se lahko prestavljata na željeno razdaljo. Robota se testira, če je zmožen priti skozi ožino, ki je malo širša od njegove širine. [14]



Slika 2-6: Slalom [14]

2.1.4 Klančina

Klančine so pod kotom 15˚. Prehodi med njimi so nizki, zato za robota niso zahtevni. S slik je razvidno, na koliko načinov je možna postavitev klančin. [14]



Slika 2-7: Klančina [14]

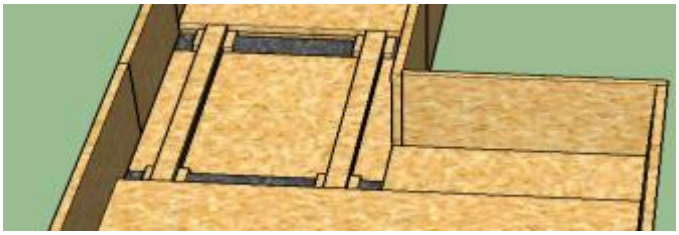
2.1.5 Klančine s stopnico

Ta klančina ima enak kot nagiba kot klančina, predstavljena v prejšnjem podpoglavju. Razlika je le, da je ta dvignjena za 100 mm ali 150 mm. Kot je razvidno s slike, lahko naredimo poligon, ki vsebuje vse tri klančine. Relief je zelo razgiban, kar je velik izziv za robota. [14]

Slika 2-8: Klančine s prehodom [14]

2.1.6 Tirnica

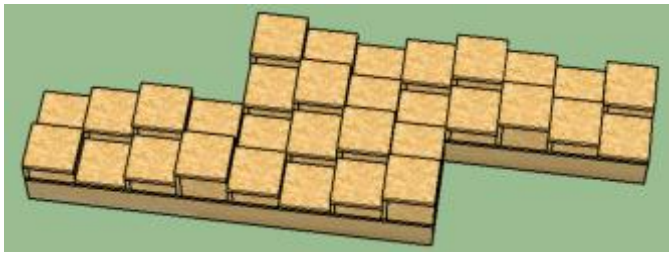
Tirnici se lahko premikata po črni podlagi. Naloga robota je, da se pelje natanko po tirnicah. [14]



Slika 2-9: Tirnica [14]

2.1.7 Stopnice

Gre za podobno situacijo kot pri klančinah, le da so v tem primeru zgornje ploskve vzporedne s površino in niso pod nagibom. Stopnice so lahko visoke 100 mm ali 150 mm. Tudi tukaj nastane zahteven relief, saj so prehodi veliki do 150 mm, kar je za robota manjših dimenzij zelo veliko. [14]



Slika 2-10: Stopnice [14]

2.2 VELIKI POLIGON

Najprej se ekipe srečajo s prej predstavljenimi tekmovalnimi površinami. Ekipe, ki so s svojimi roboti uspele premagati osnovne poligone, se preizkusijo še na velikem. Ta je sestavljen iz več osnovnih poligonov, ki so med seboj povezani v celoto. Da pa ne bi bilo preveč enostavno, so tudi osnovni poligoni lahko pod kotom 30˚. Tako dobimo progo z več nivoji. Kot je razvidno s slike, so različni sektorji označeni z različnimi barvami. Modra barva označuje najlažje dele, rumena zahtevnejše, rdeča pa najzahtevnejše. Prevoženi sektorji se temu primerno tudi točkujejo. [16]

Slika 2-11: Veliki poligon [16]

3 POTEK DELA

Preučili smo tekmovalno površino in se lotili snovanja robota. Vse sestavne dele smo narisali v SolidWorksu, s pomočjo MakerBot programa smo jih pretvorili v obliko, ki jo je razumel 3D-tiskalnik. Večina delov smo natisnili. Ko smo imeli izdelane vse komponente, smo jih skupaj z elektronskimi moduli in vezji sestavili v celoto. Sledilo je pisanje programa v programskem jeziku Python. Robota smo nato testirali in optimizirali njegovo vodenje.

3.1 SNOVANJE

Pri snovanju izdelka ali stroja je najpomembneje, da vemo, kakšen je njegov namen. S tem, ko smo spoznali tekmovalni poligon, smo si začeli predstavljati obliko robota. Poligon je 300 mm širok, kar pomeni, da moramo narediti robota, ki bo čim ožji. Zadali smo si, da ne sme presegati širine 230 mm. Druga najpomembnejša predpostavka je, da je robot zmožen prevoziti 150 mm visoko stopnico. Mora biti trpežen na zunanje sile ter odporen na pesek in prah. Na podvozju oziroma jedru robota mora biti pritrjena robotska roka s kamero. Prav tako mora imeti dovolj velik vir energije, da lahko zdrži vsaj nekaj ur na terenu.

3.1.1 EMU-robot

Organizatorji tekmovanja so zasnovali zelo enostavnega robota, ki naj bi zadoščal kriterijem tekmovanja. Njihov namen je, da bi ekipe na svojem robotu uporabile enake komponente kot so na EMU-robotu, mislili so na motorje (Dynamixel AX-12A) in mikroračunalnik (Raspberry Pi). Tudi konstrukcija naj bi bila podobna njihovi ali celo enaka z nekaj nadgradnjami. S tem hočejo doseči kompatibilnost med ekipami. Nam se zdi ta ideja dobra, zato smo se je tudi skušali čim bolj držati.

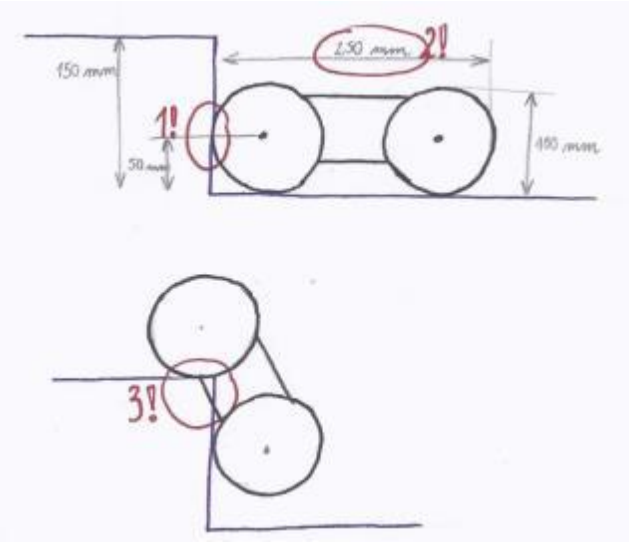
Slika 3-1: EMU-robot [4]

3.1.2 Prvi koncept robota

Najprej smo se posvetili podvozju, saj menimo, da je najpomembnejša vozna dinamika. To pomeni, da je robot zmožen prevoziti vse ovire, ki jih ponuja tekmovanje. Najprej smo si ogledali, kako je sestavljen EMU-robot. Na jedrni del ima pritrjene 4 motorje, na vsakem je pritrjeno kolo, ki ima premer približno 80 mm.

Opazili smo tri velike težave:

1. Če robota postavimo pred 150 mm visoko oviro, opazimo, da so njegova kolesa veliko manjša od nje. Če so kolesa narejena tako, da imajo zasekana platišča in na zunanjih delih gumijasto oblogo, obstaja možnost, da se robotu uspe vzpeti. To je verjetno pri idealni tekmovalni površini. Ta je lahko posuta s peskom, ki povzroči manjši oprijem in posledično oteži robotovo delovanje.
2. S povečevanjem koles bi se približali višini prepreke. Imeli bi večjo možnost, da bi jo premagali, saj bi bilo potrebno opraviti manjšo pot do točke, ko kolo preide na vrh ovire. Težava je, da se z večanjem koles povečuje tudi dolžina. Že pri kolesu s 120 mm premera bi robot meril v dolžino 250 mm, če upoštevamo, da je med kolesoma 10 mm razmika.
3. V primeru, da robotu uspe priti s prvim kolesom na vrh, se težava pojavi v območju med kolesoma. Še posebej problematične so prepreke, ki imajo oglate robove. Če pride rob stopnice med kolesa, se lahko robot zagozdi.



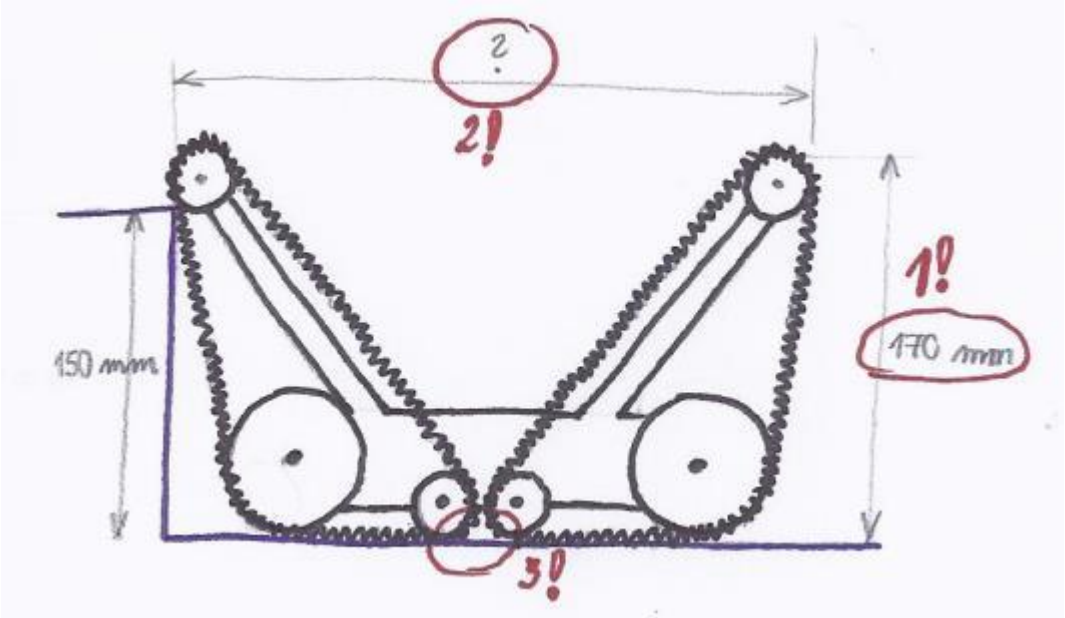
Slika 3-2: Prvi koncept robota

3.1.3 Drugi koncept robota

Ključni pomanjkljivosti koles sta, da je z njimi težko premagovati višinske ovire in da nimajo veliko oprijema. Pri tem so gosenice veliko boljše, predvsem pri oprijemu na podlago. Pri kolesih ima zelo majhna površina stik s tlemi, gosenica pa ima toliko stika, kolikor ji ga sami določimo.

Na spodnji sliki je koncept goseničarja. Na vsaki strani ima po 2 gosenici, kar pomeni, da ima 4 pogonske motorje in veliko pomožnih koles, kar je zelo neracionalno. Poleg tega smo opazili 3 napake, do katerih lahko pride.

1. Goseničar je tako visok, da so gosenice višje kot 150 mm visoka ovira. Dobra lastnost je, da prepreko prevozi z lahkoto, slabša pa je, da je zelo velik. Gosenice so statične, kar pomeni, da so vedno v istem položaju. V nizkih predorih ali podobnih situacijah si s takim robotom ne bi mogli pomagati.
2. Vprašljiva je tudi njegova dolžina, saj je bila že pri robotu s kolesi dolžina velik problem. Res je, da so premeri koles, ki poganjajo gosenice, manjši, vendar so tu še pomožni koleščki, ki prav tako potrebujejo dovolj prostora.
3. Tudi pri goseničarju se pojavi nevarno območje med gosenicama. Tako kot pri prvem konceptu se lahko tudi v tem primeru robot zatakne ali nasede.



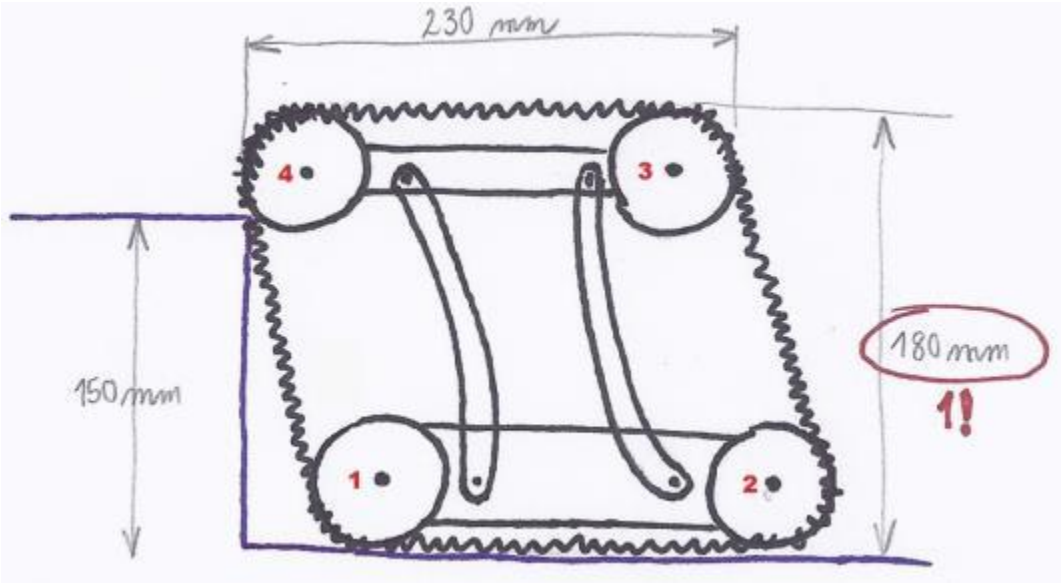
Slika 3-3: Drugi koncept robota

3.1.4 Tretji koncept robota

Zamislili smo si drugo varianto goseničarja. V prejšnjem primeru nam je ustrezalo dejstvo, da gosenice sežejo nad oviro. V drugem oziru pa bi radi imeli čim nižjega robota, ki bi bil kos nizkim predelom. Navdih smo iskali v kmetijski mehanizaciji. Variabilne balirke za okrogle bale z gumijastimi pasovi imajo narejen sistem, ki enakomerno stiska balo od 500 mm do 1850 mm premera, pri tem pa so pasovi vedno enako nategnjeni. Želeli smo narediti robota, ki bi spreminjal položaj gosenic, pri tem pa bi ostale vseskozi enako nategnjene.

Na naslednjih 6 slikah prikazujemo, kako smo si zamislili gibanje gosenic in kako bi robot s takšnim sistemov prečkal najvišjo možno oviro. Sistem je sestavljen iz pogonskih koles (1 in 2), pomožnih koles (3 in 4), gosenice, ki je napeljana okoli koles, in ostalih mehanskih delov, do katerih pridemo v nadaljevanju. Na drugi strani je identičen sistem, ki se premika sinhrono. Z rdečo smo označili njegovo višino, ki je v osnovnem položaju gosenic[1] velika, vendar se v skrajnih legah višina zniža pod 100 mm, se pa s tem podaljša njegova dolžina.

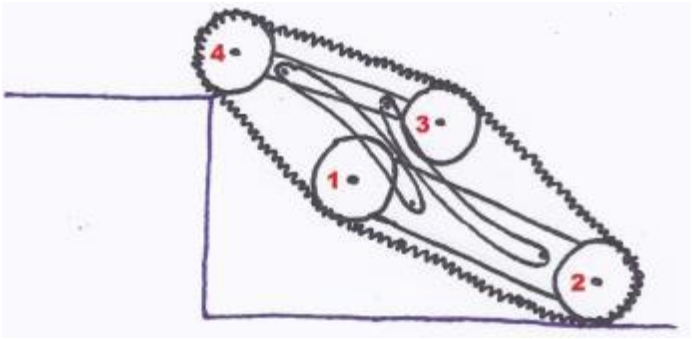
Sistem pomaknemo malo v levo. Z robotom se pripeljemo do ovire tako, da se pomožno kolo (4) nasloni na rob ovire.



Slika 3-4: Tretji koncept robota 1

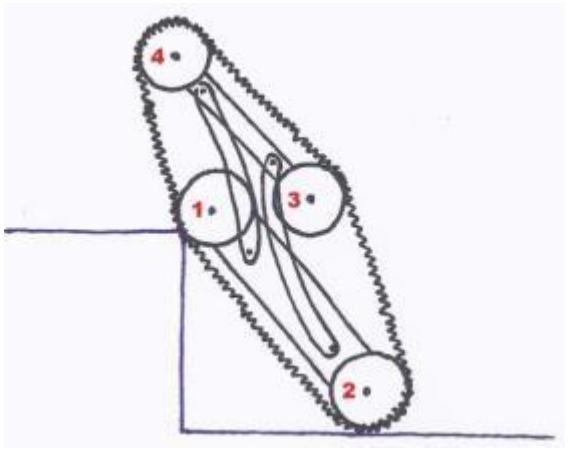
[1] *Osnovni položaj gosenic je položaj, ko središča pogonskih in pomožnih koles tvorijo oglišča pravokotnika.*

Sistem premaknemo v skrajni levi položaj, tako da je pogonsko kolo (1) v zraku.



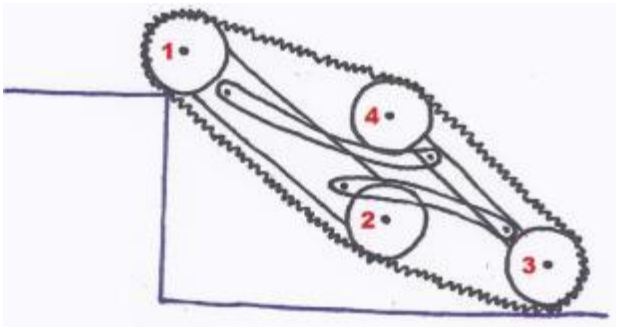
Slika 3-5: Tretji koncept robota 2

Z robotom se peljemo naprej tako dolgo, dokler pogonsko kolo (1) ne zapelje preko roba.



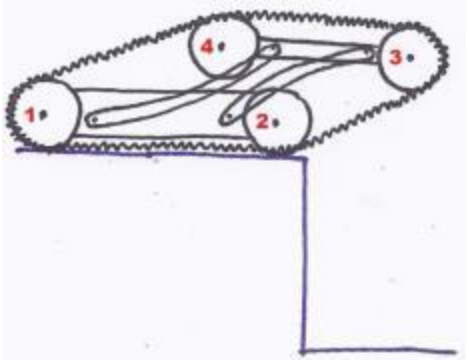
Slika 3-6: Tretji koncept robota 3

Sitem pomaknemo v skrajni desni položaj, tako da je pogonsko kolo (2) v zraku.



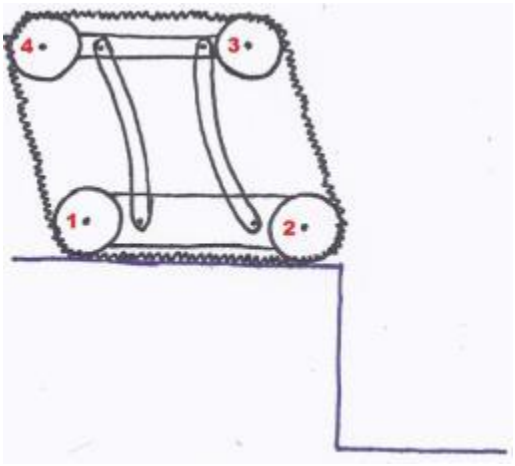
Slika 3-7: Tretji koncept robota 4

Z robotom se peljemo naprej, dokler nista obe pogonski kolesi na vrhu ovire.



Slika 3-8: Tretji koncept robota 5

Na koncu premaknemo gosenice v poljubni položaj za nadaljnjo pot.



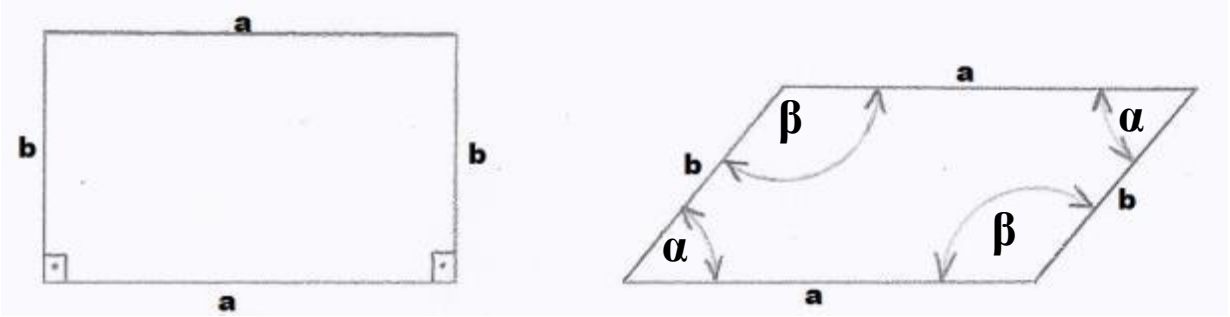
Slika 3-9: Tretji koncept robota 6

Teoretično bi zadostovalo samo eno pogonsko kolo oziroma samo en motor bi lahko poganjal gosenico. Predpostavimo, da bi bilo kolo (1) pogonsko. Na sliki 3-8 vidimo, da se obseg kolesa 40 % prilega na gosenico, kar pomeni, da bi bilo dovolj oprijema za pogon. Na sliki 3-6 opazimo, da se le 10 % obsega kolesa prilega na gosenico, kar je zelo malo in vprašljivo je, če bi bilo dovolj za poganjanje gosenice. Na podlagi tega smo se odločili, da bomo uporabili dve pogonski kolesi. S tem smo dosegli nemoteno delovanje gosenice, posledično pa povečali porabo energije.

3.1.5 Preračun gosenic

Ob analiziranju skic tretjega koncepta smo se vprašali, ali pri spreminjanju lege gosenice dolžina le-te ostane nespremenjena. Na podlagi skic smo si zadevo težko predstavljali, zato smo skice pretvorili v lik in ga z matematičnimi zakoni razrešili. Ko je gosenica v osnovnem položaju, ima obliko pravokotnika. Ob premikanju pa se koti α in β spreminjajo, vendar to ne vpliva na dolžino stranic in prav tako ne na obseg.

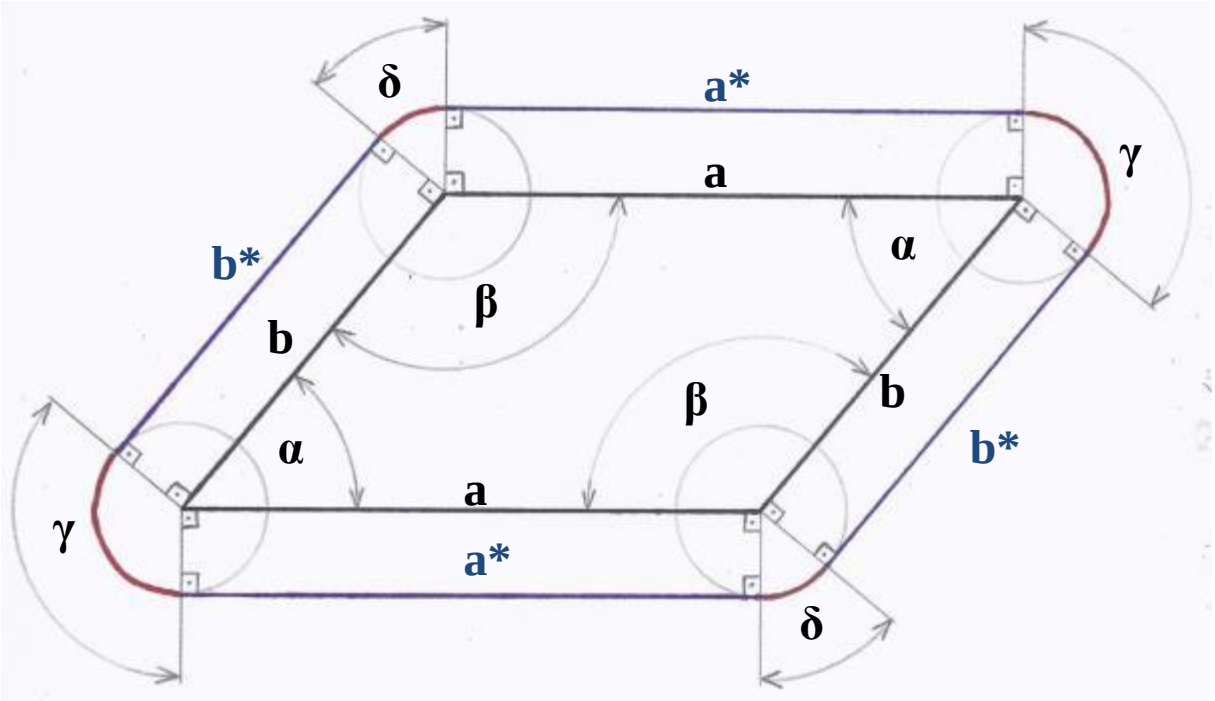
$$o = 2 \times (a + b)$$



Slika 3-10: Obseg paralelograma

V vsako oglišče smo narisali krog. Ti krogi predstavljajo kolesa. Daljici a in b smo vzporedno preslikali, tako da so oglišča tangentna na krožnico. Obarvali smo jih modro ter jih označili a* in b*. Mali kvadratki s pikami predstavljajo kot 90˚. Rdeče črte prikazujejo dele gosenice, ki se prilegajo na kolesa.

Ob premikanju gosenice daljice a* in b* ostanejo v katerem koli položaju enako dolge. Zanimalo nas je, kaj se dogaja z dolžinami krožnih lokov. Najprej smo določili notranje kote, nato še zunanje. Ugotovili smo, da je kot α enak kotu δ, kot β pa kotu γ. To pomeni, da je seštevek zunanjih kotov v katerem koli položaju gosenice enak polnemu kotu 360˚. S tem smo potrdili teorijo o ohranjanju dolžine gosenice.



Slika 3-11: Obseg gosenice

Vsota notranjih kotov paralelograma:

$$\sum_{i=1}^{n} = (n-2) \times 180° \quad => \quad \sum_{i=1}^{4} = (4-2) \times 180° = 360°$$

Velikost kotov:

$$2\alpha + 2\beta = 360° \quad => \quad \alpha + \beta = 180°$$

$\alpha = 180° - \beta$

$\alpha + \gamma + 90° + 90° = 360°; polni\ kot$

$\alpha = 360° - 180° - \gamma$

$\alpha = 180° - \gamma$

$\beta = \gamma$

$\beta = 180° - \alpha$

$\beta + \delta + 90° + 90° = 360°; polni\ kot$

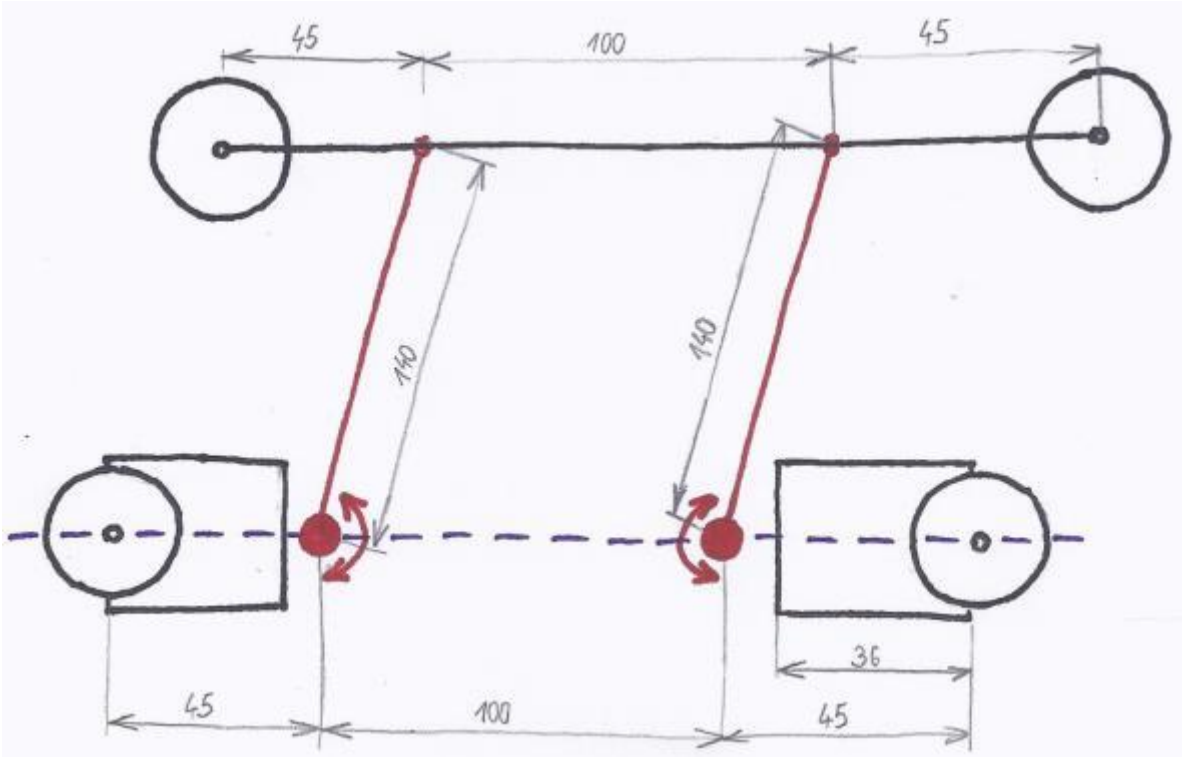
$\beta = 360° - 180° - \delta$

$\beta = 180° - \delta$

$\alpha = \delta$

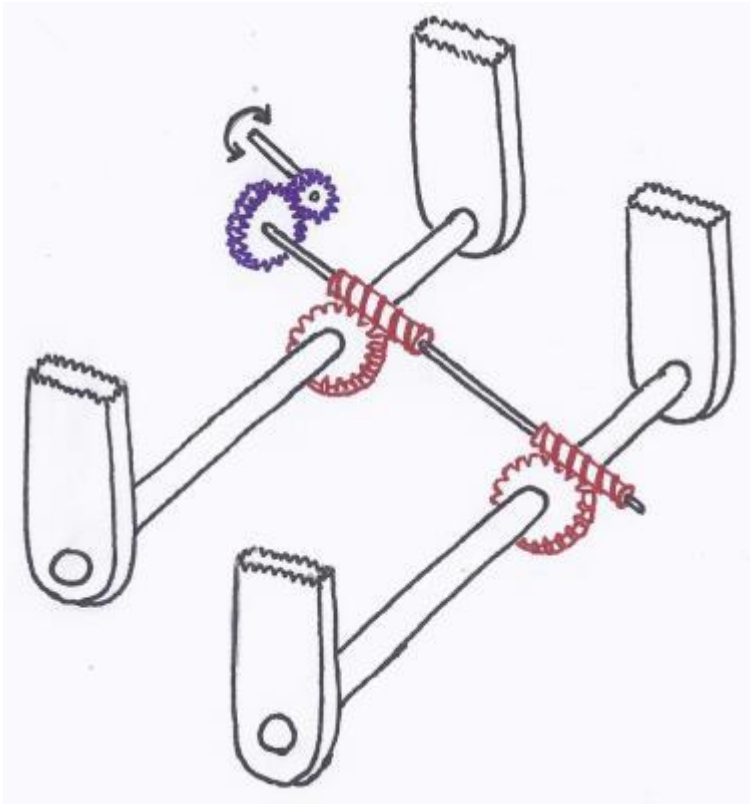
3.1.6 Sistem za premikanje gosenice

Teorija o ohranitvi dolžine gosenice pri premikanju drži v primeru pravilno narejenega mehanizma. Daljica, ki jo omejujeta središči pomožnih koles, mora biti v katerem koli položaju vzporedna s premico, ki poteka skozi središči pogonskih koles. Rdeči daljici predstavljata glavni vodili za premikanje daljice pomožnih koles. Ti premici morata biti med seboj vzporedni in istih dolžin, le tako lahko dosežemo, da je daljica pomožnih koles v poljubnem položaju vzporedna na premico. Na sliki so kotirane mere, ki smo jih upoštevali v nadaljnji izdelavi. Dolžina motorja je od središča pogonske osi do konca ohišja 36 mm. Velika rdeče pobarvana kroga predstavljata glavni osi za premikanje gosenice. Sistem smo skušali čim bolj minimalizirati.



Slika 3-12: Sistem za premikanje gosenice

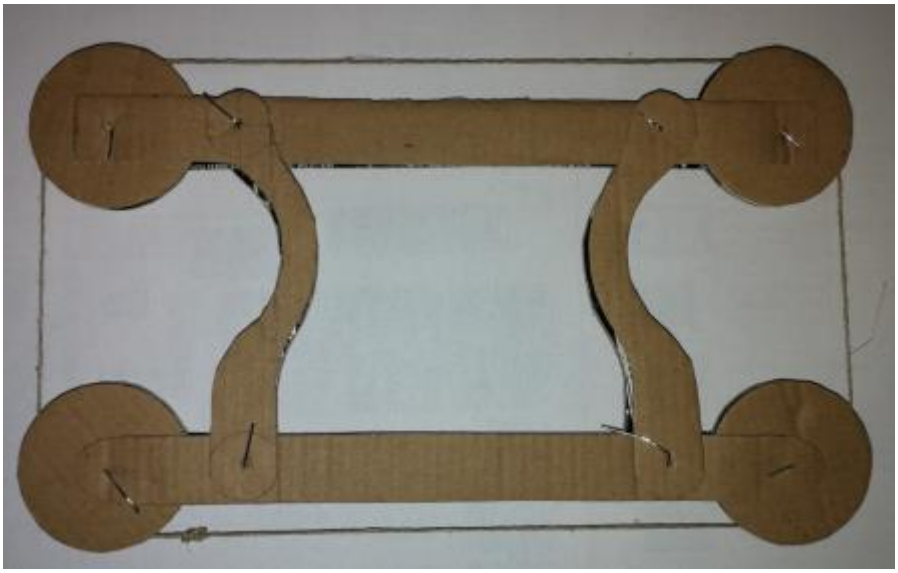
Na vsaki strani robota je ena gosenica. Lahko bi naredili, da bi se gosenici na levi in na desni premikali vsaka posebej. Za to bi potrebovali dva mehanizma, dva motorja in veliko prostora. Zadevo smo racionalizirali in povezali levo in desno stran s pogonskima osema. V tem primeru sta obe gosenici na istem položaju in se premikata hkrati z enako hitrostjo. Mehanizem poganja motor Dynamixel AX-12A. Navor se poveča v reduktorju. Sestavljata ga dva zobnika, prenos je iz manjšega na večjega. S tem zmanjšamo hitrost vrtenja, ampak se bistveno poveča navor. Posledično motor porabi manj energije in je zmožen premagovati večje sile. Iz reduktorja poteka prenos na polžasto gonilo. Gonilo je v razmerji 1 : 50, kar pomeni, ko se polž zavrti 50-krat, se zobnik 1-krat. Sorazmerna je hitrost, obratno sorazmeren pa je navor, saj je kar 50-krat večji. S tem dosežemo, da sistem za premikanje gosenic lahko dvigne celega robota. Zaradi polžastega gonila se poveča natančnost premikanja. Najpomembneje je, da zobnik ne more premakniti polžastega vretena, zato posledično ne moremo z zunanjimi silami premakniti položaja gosenice, temveč jo lahko samo z motorjem.



Slika 3-13: Mehanizem za premikanje gosenice

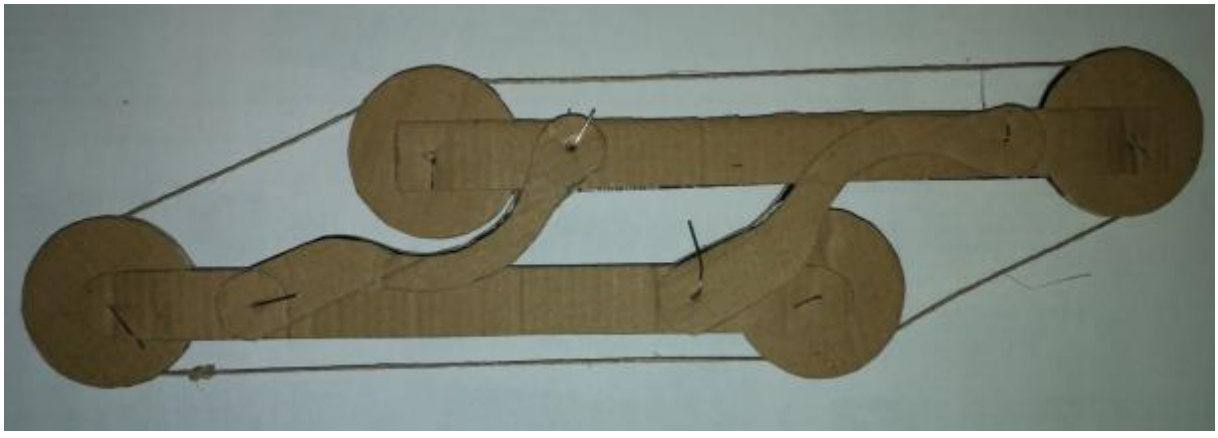
3.1.7 Maketa gosenice

Pred postopkom modeliranja in izdelave smo naredili maketo gosenice, da se prepričamo, ali naša teorija drži. Maketa je naravne velikosti. Narejena je iz kartona, osi pa so iz raztegnjenih aluminijastih sponk. Maketo smo najprej postavili v osnovni položaj, nato pa okoli koles na tesno napeljali vrvico.



Slika 3-14: Maketa gosenice v osnovnem položaju

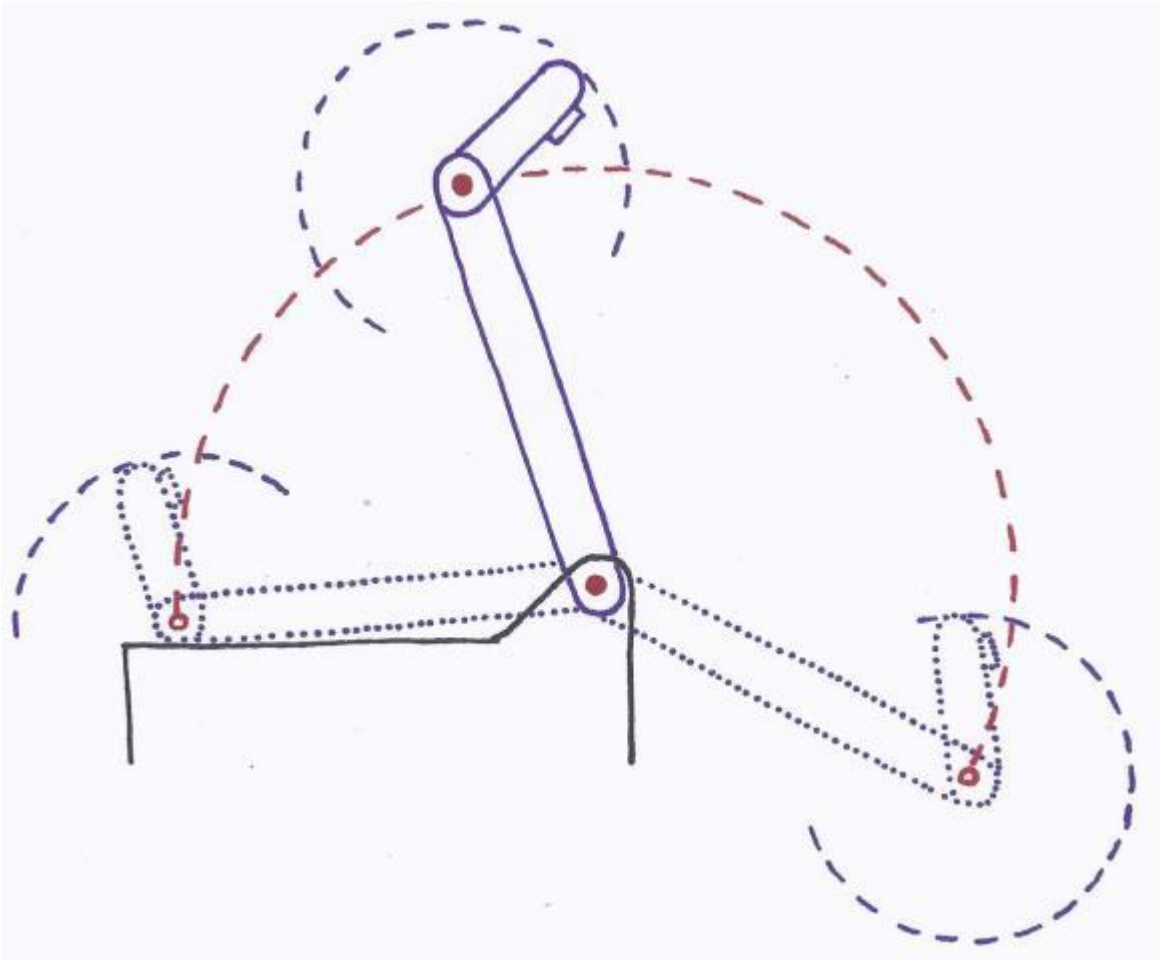
Sistem smo nato premikali sem in tja ter opazili, da je vrvica v katerem koli položaju enako nategnjena. S tem smo potrdili našo teorijo in dali projektu zeleno luč za nadaljnji postopek.



Slika 3-15: Maketa gosenice v poljubni legi

3.1.8 EMU robotska roka

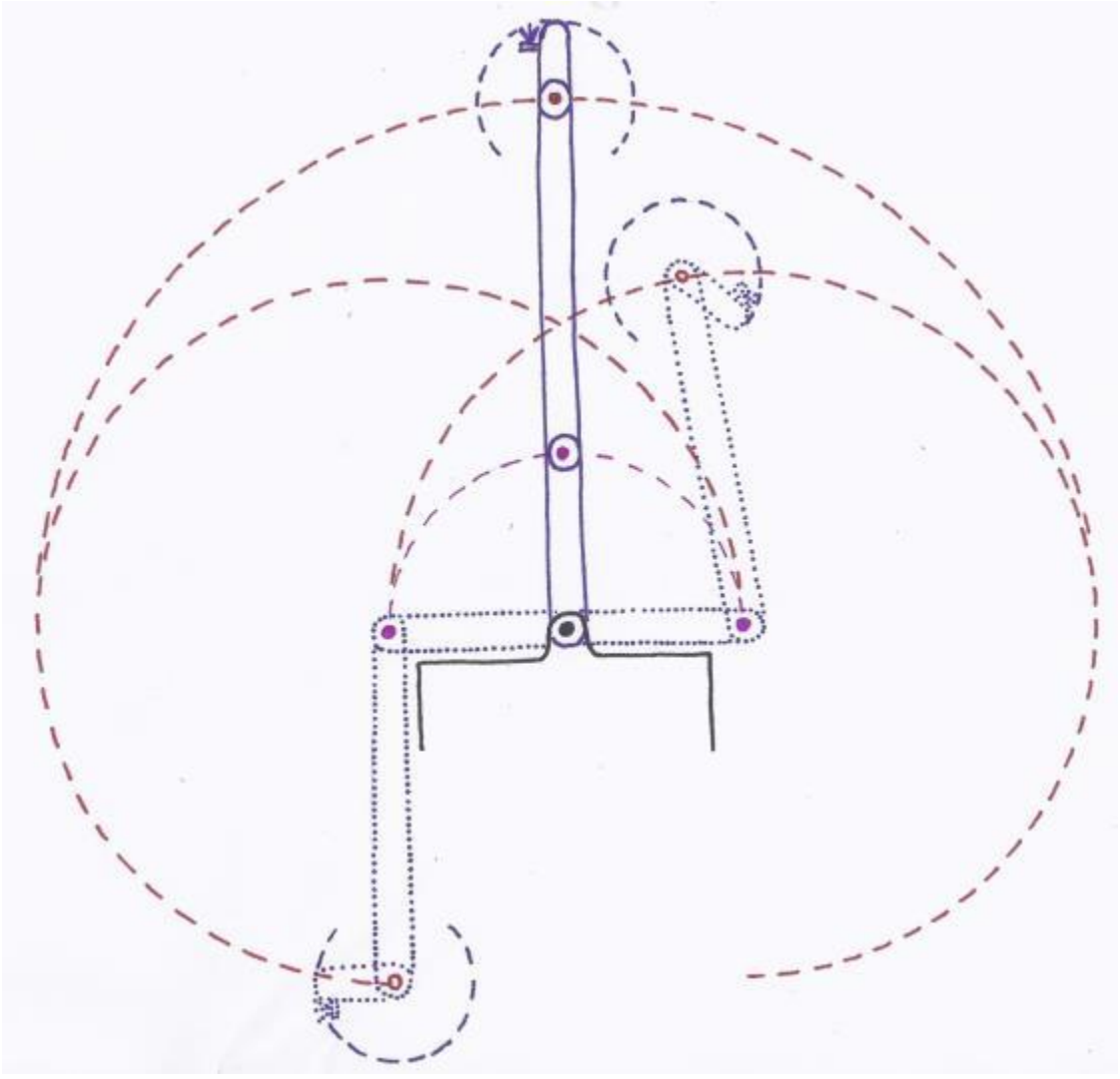
Primarni namen robotske roke je, da ima na sebi pritrjeno kamero, preko katere operater spremlja robota na ekranu. Roka je pritrjena na jedrni del robota. Kot vidimo s slike, je roka pritrjena na skrajni desni del, kar je nepraktično. Robot mora biti zmožen voziti v obe smeri (naprej in nazaj). V tem primeru ima operater več razpoložljivih mest za postavitev kamere, če se pelje v desno stran. V obratni smeri bi imel veliko več težav z iskanjem najprimernejšega zornega kota. Roka je sestavljena iz dveh delov, kar ji ne daje velike fleksibilnosti. Rdeča črtkana črta prikazuje območje, ki ga lahko zajame daljši del roke. Modra črta pa prikazuje območje, ki ga lahko zajame krajši del roke, na kateri je kamera.



Slika 3-16: EMU robotska roka

3.1.9 Idealna robotska roka

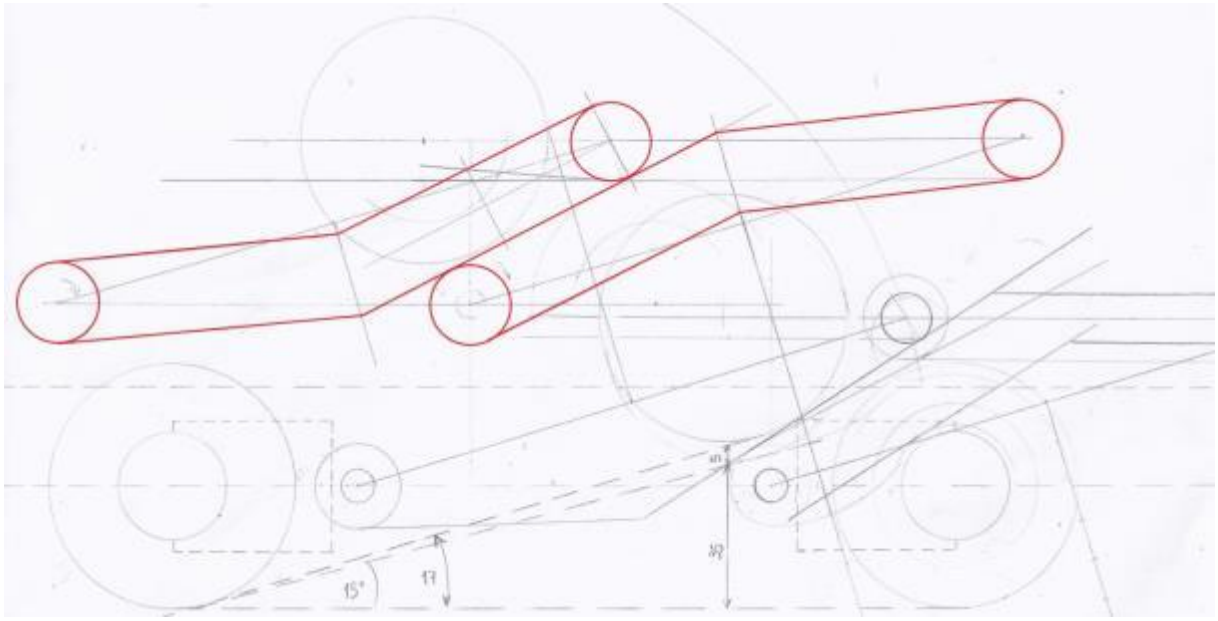
Zasnovali smo roko, ki je sestavljena iz treh delov. Posledično potrebujemo en motor več in več energije. Težje je tudi vodenje, saj je potrebno obvladovati vse tri dele. Vendar nam daje ta roka veliko dobrih lastnosti. Najprej opazimo, da je roka pritrjena na sredini. To nam omogoča enakovredno vidljivost v obe smeri. Zunanji rdeči krog označuje najdaljši raztezek srednjega dela roke. Treba pa je vedeti, da se lahko kamera pomika kjerkoli v notranjosti kroga, zahvaljujoč tridelni roki. Iz slike je razvidno, da je območje raztezka globoko pod robotom. Vsekakor se zavedamo, da je roka precej velika, vendar jo je mogoče zložiti v zelo malo območje.



Slika 3-17: Idealna robotska roka

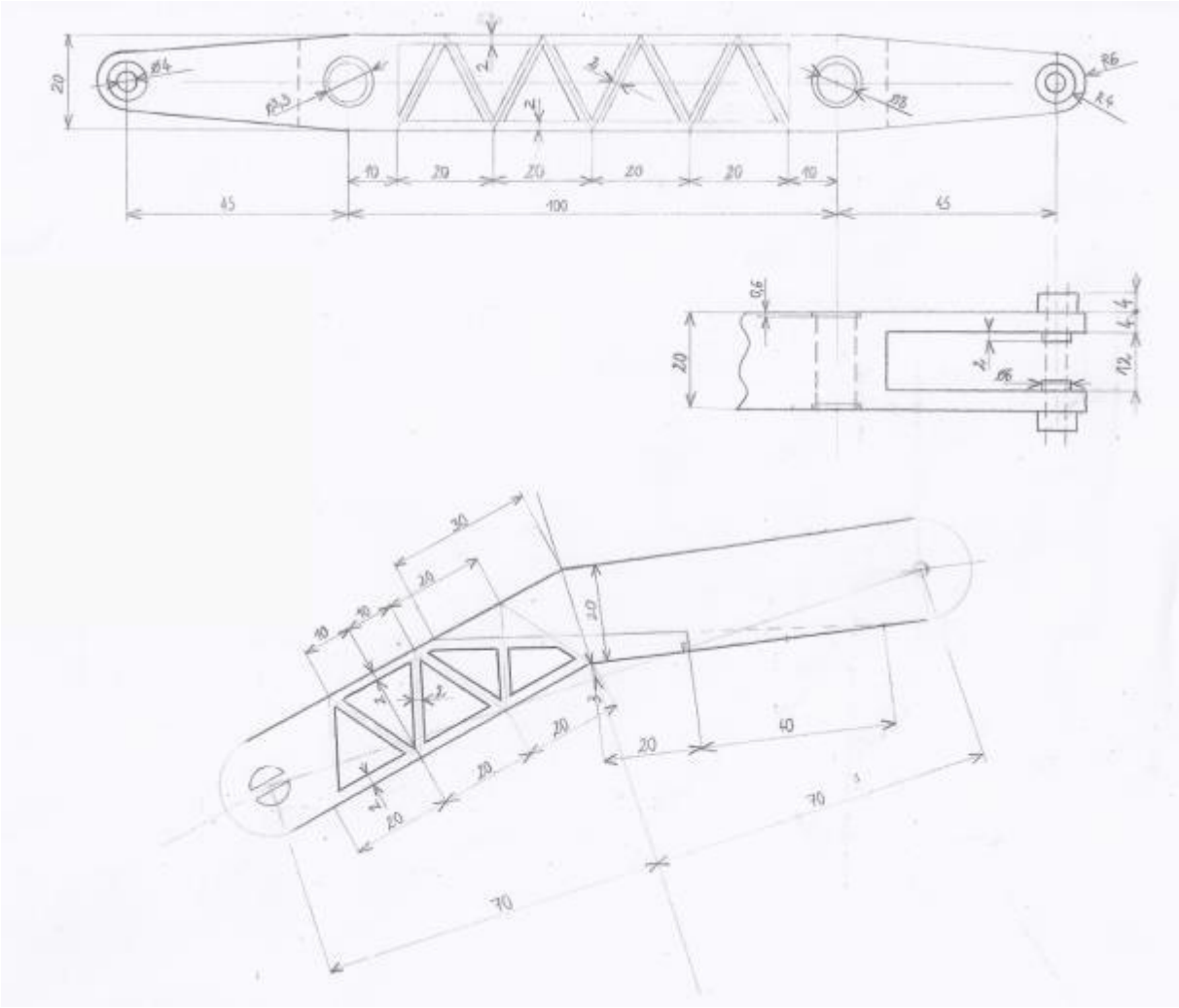
3.2 KONSTRUIRANJE

Svoje ideje oziroma koncepte smo morali realizirati. Pri snovanju smo nekatere dele narisali s črtami. Določili smo obliko, material, barvo in lastnosti posameznih sestavnih delov. Največ težav nam je povzročal sistem za premikanje gosenic. Predvsem problematični sta bili vodilni roki, ki prestavljata nosilec pomožnih koles. Na spodnji sliki vidimo veliko črt in krogov. Na prvi pogled vse skupaj deluje nepregledno. Vendar se z dobrim opazovanjem vidi, da veliki krogi predstavljajo kolesa, mali krogi osi, črtkane črte ohišje motorja in črte povezave. Pomožna kolesa smo postavili v skrajni desni položaj pri premikanju gosenic. Zavedali smo se, da bodo deli 3D-natisnjeni. Takšni plastični elementi imajo veliko manjšo trdnost kot vliti plastični elementi ali železni. Posledično smo morali paziti, da so dovolj veliki za prenašanje sil. S poizkušanjem smo prišli do idealne rešitve, ki je označena z rdečo barvo.



Slika 3-18: Konstruiranje vodilnih rok

Ročno smo narisali delavniško risbo nosilca pomožnih koles in vodilne roke. Najpomembnejša je funkcionalnost delov, pri čemer smo pazili tudi na estetiko. Racionalno smo gledali na material. Osrednji del nosilca za pomožni kolesi bi lahko bil poln. S trikotniki smo dosegli isto togost nosilca, zmanjšali porabo plastike za 40 %, zmanjšali čas izdelave in ga nazadnje vizualno izboljšali.



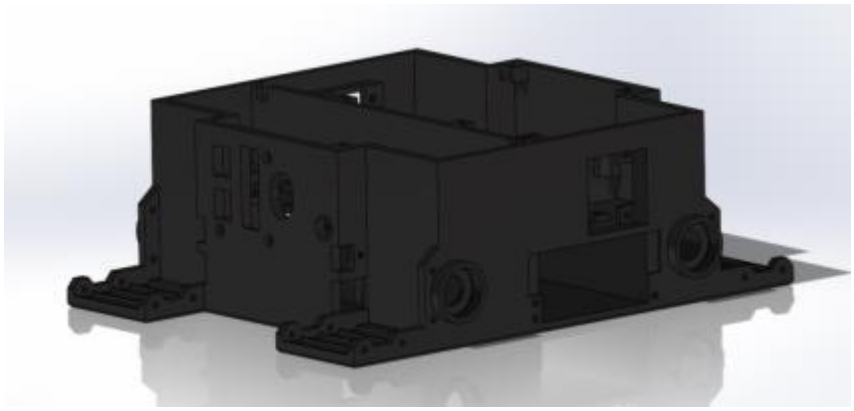
Slika 3-19: Ročno narisana delavniška risba

Opazili smo, da je ročno risanje precej zamudno, tehnologija pa je toliko napredovala, da v industriji na ta način ne moremo biti konkurenčni, zato smo začeli uporabljati CAD-program.

3.3 MODELIRANJE

Na trgu obstaja veliko CAD-programov za modeliranje. V šoli smo se v 3. letniku spoznali s programom CREO 2.0. Učili smo se ga uporabljati skozi celo leto. Prve sestavne dele robota smo naredili z njim. Pri nam znanih podjetjih smo povprašali, kateri CAD-program uporabljajo. Skoraj vsa podjetja so nam predlagala program SolidWorks, ki je sicer zahtevnejši za uporabo. Osnovne funkcije med programoma so si podobne, vendar ima slednji veliko več možnosti. Z njim se je enostavneje orientirati, predvsem pa je lažje kotiranje. Tudi sestavnico je veliko lažje narediti. V celoti je program preglednejši in bolj profesionalen. V njem lahko opravljamo tudi simulacije.

Velika prednost modelirnih programov je uvažanje komponent iz spletnih knjižnic. Mi smo uporabljali spletno stran GrabCAD [https://grabcad.com/]. Prihranila nam je veliko časa, saj smo z nje prenesli Dynamixel AX-12A, ventilator, tranzistor, upore in razne spojke. Najprej smo začeli s podvozjem. Izhajali smo iz motorja. Najprej smo naredili vpetje motorja, naslednji korak je bilo jedro z vpetji za vse 4 pogonske motorje, odprtina za baterijo, vezje itd.



Slika 3-20: Jedrni del

Po končani izdelavi vseh sestavnih delov robota smo se lotili sestave robota v funkciji "Assembly". Tukaj dobi končni produkt svojo realno podobo. Eden izmed naših pomembnih ciljev je bil, da iz vseh sestavnih delov dobimo sestavnico, ki bo premikajoča. Pri premiku naklona gosenic smo morali povezati zobnik iz motorja na zobnik, ki je nameščen na os. Na tej osi pa sta polžasta pogonska sklopa, ki prav tako skrbita, da se premika os, na katero so pritrjene vodilne roke za premik gosenic.

Slika 3-21: Sestavnica mehanskega dela gosenic

V programu SolidWorks imamo na voljo nekaj različnih načinov spajanja sestavnih delov. Osnove sestavljanja smo spoznali že v šoli pri predmetu konstruiranje, kjer so nam predstavili osnovne načine sestavljanja dveh sestavnih delov.

Mi pa smo raziskali nekoliko podrobneje in uporabili še ostale načine, kot je spajanje zobnikov »Gear mate«, ki smo ga uporabljali za sestavne dele, ki poganjajo bodisi pogonska kolesa ali pogonski sklop za premikanje naklona gosenic. Druga za nas nova možnost sestavljanja pa je bila »Advanced mates«, pri kateri lahko nastavimo kot med dvema sestavnima deloma. Tako smo lahko omejili premikanje robotske roke, vodilnih rok za premik naklona gosenic itd.

S sestavljanjem na takšen način smo prišli do veliko natančnejšega vpogleda, kako bo izgledal končni izdelek. Hkrati pa smo lahko simulirali različne okoliščine in lažje predvideli težave pri izdelavi.

Slika 3-22: Sestavnica robota

3.4 3D-TISKANJE

3D-tiskanje je postopek, pri katerem izdelujemo tridimenzionalne predmete iz različnih materialov: papir, guma, kovine ter najbolj pogosto različne vrste plastike. S tem postopkom lahko naredimo izdelek skoraj vsake oblike. 3D-tiskanje se razlikuje od ostalih pogostejših postopkov izdelave predmeta v tem, da se tukaj material dodaja, medtem ko pri ostalih postopkih material po navadi odvzemamo, na primer vrtanje, frezanje struženje itd. Postopku, ki dodaja material, pravimo aditivni postopek. To pomeni, da se postopoma doda nova plast materiala in se tako oblikuje končna oblika.

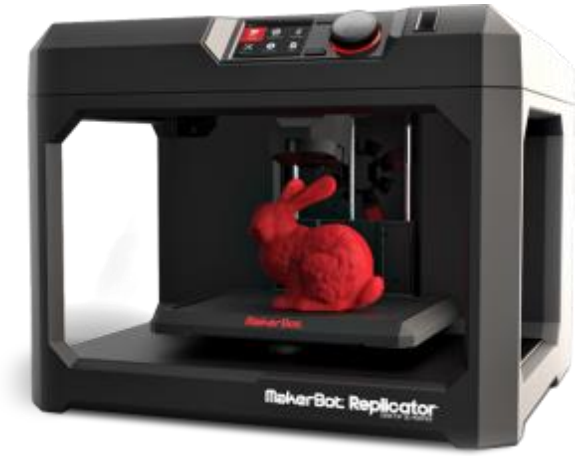
Področje uporabe 3D-tiskalnika je zelo široko. Najpogosteje se uporablja pri tiskanju prototipov na področju oblikovanja nakita, industrijskega oblikovanja, arhitekture, inženirstva, konstrukcije, letalske in avtomobilske industrije. Čedalje pogosteje se uporablja tudi v medicini, saj lahko natisnejo nov organ, del kosti, proteze, zobe itd. Zadnje pa se je razvilo 3D-tiskanje stavb. Ogromni tiskalniki uporabljajo namesto plastike beton, ki ga nanašajo po slojih, dokler ne nastane stavba. [26] [27]

Slika 3-23: 3D natisnjen del človeške lobanje [26]

3.4.1 MakerBot Replicator (Fift Generation)

Že od samega začetka je bil naš glavni cilj 3D-tiskanje konstrukcije robota. Poleg tega je najprimernejši proces izdelovanja prototipnih izdelkov. S svojim unikatnim procesom tiskanja je tiskalnik zmožen natisniti zapletene oblike po ugodni ceni. 3D-tiskanje je bilo za nas izziv, saj se do sedaj še nikoli nismo srečali s to tehnologijo. Tako smo pridobili dodatno znanje iz sodobnih tehnologij izdelovanja izdelkov.

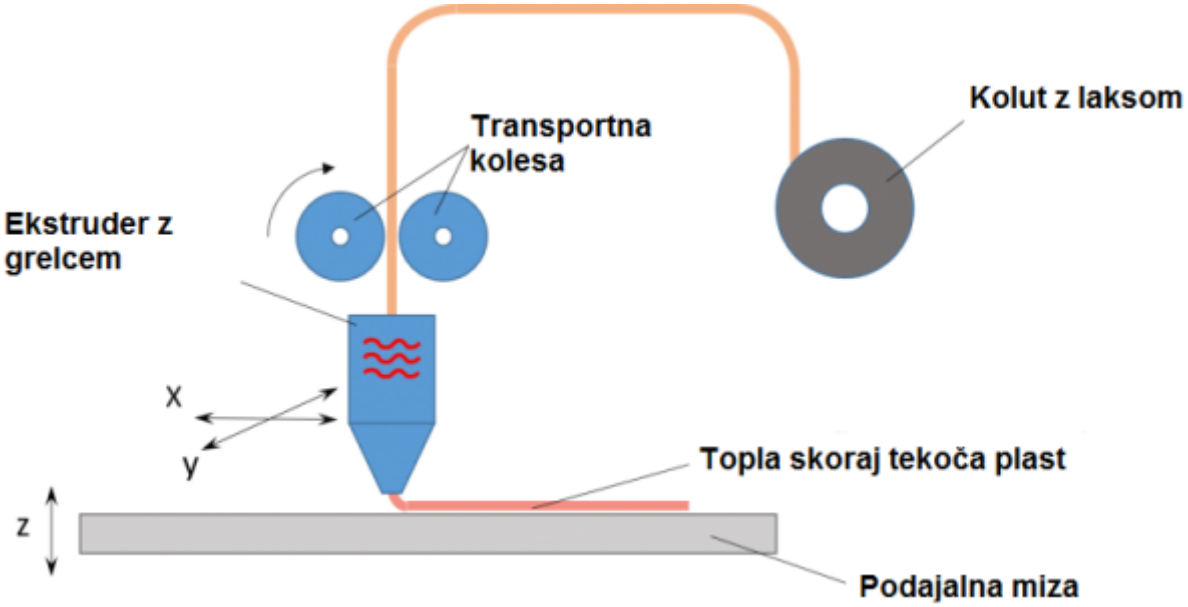
Uporabljali smo šolski 3D-tiskalnik MakerBot Replicator (Fift Generation), ki je najboljši namizni MakerBotov 3D-tiskalnik. Po novem ima »Smart extruder«, ki je izredno produktiven, zanesljiv in natančen. MakerBot Replicator (Fift Generation) je hitrejši in tišji kakor njegov predhodnik. Je zelo zanesljiv in natančen, saj uporablja 1,75 mm debelo polilaktično kislino za polnilo. Prav tako je uporabniku zelo prijazen in namenjen vsem, ki želijo hitro pridobiti znanje na tem področju.



Slika 3-24: MakerBot Replicator (5th generation) [10]

Izdelki so lahko narejeni v ločljivosti vse od 0,05 mm do 0,4 mm. Pri tem je temperatura tiskalne konice lahko 215 ℃–260 ℃. Izdelki so lahko maksimalne velikosti 252 mm x 199 mm x 150 mm in so natisnjeni z minimalno debelino enega sloja 100 mikronov. [10]

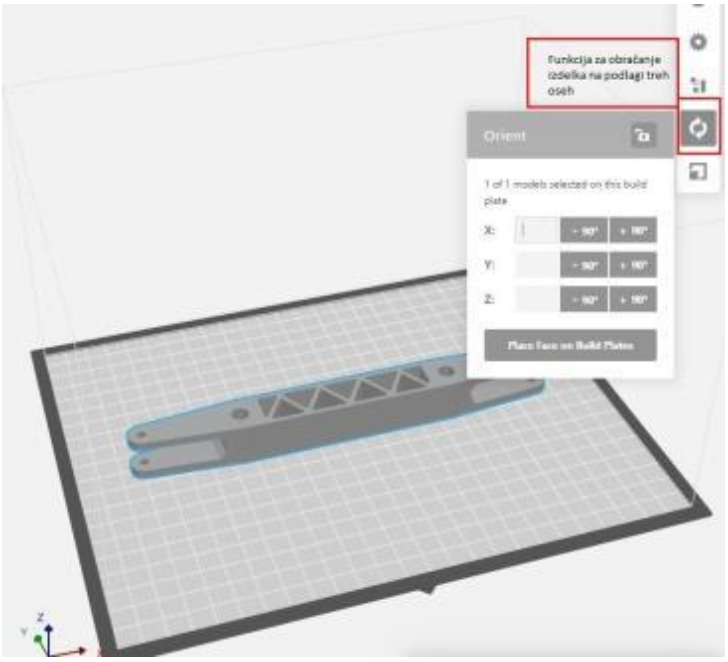
Deluje po principu FDM, kar pomeni modeliranje s taljenjem polnila. Iz koluta, ki je pritrjen na ohišju, ekstruder sam odvija plastično žico. Ta potuje skozi glavo ekstruderja in konico, ki segreje polnilo do skoraj tekočega stanja. Glava se premika pod računalniškim nadzorom, da definira tiskano obliko. Premika se v dveh dimenzijah, in sicer po X-osi ter Y-osi, z vertikalnim pomikom mize pa dobimo še tretjo Z-os. S pomočjo aditivnega postopka ekstruder naredi vsak sloj posebej. Tako z velikim številom slojev dobimo obliko izdelka. [6]



Slika 3-25: FDM-tehnologija 3D-tiskanja [Prirejeno po 6]

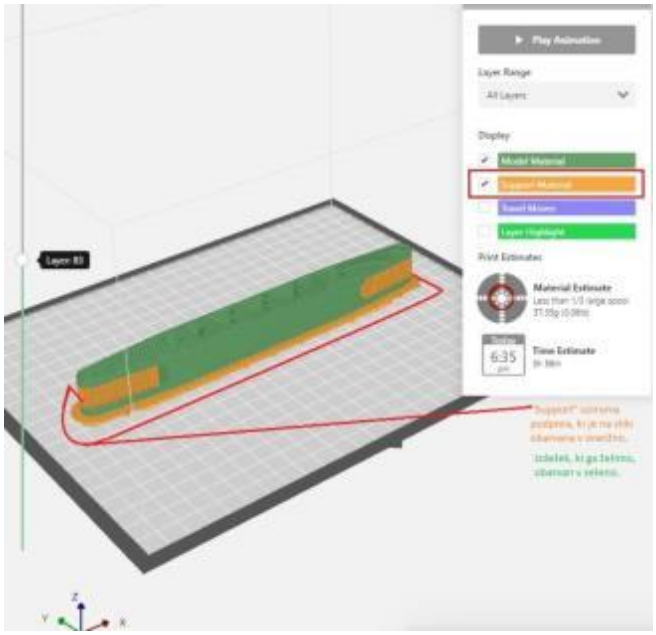
3.4.2 Tiskanje robota

Predmet oziroma izdelek, ki ga želimo natisniti, moramo najprej narisati v modelirnem programu. V našem primeru so bili vsi sestavni deli narisani v SolidWorksu. Shranjene datoteke, ki jih razume SolidWorks, smo morali pretvoriti v tiskalniku razumljiv jezik. MakerBot ima svoj program, ki nam omogoča uvažanje CAD-modelov za tiskanje. S tem prihranimo veliko časa, saj je vse, kar od nas zahteva program, da uvozimo željeni model v STL-obliki. S pomočjo programskih funkcij lahko zavrtimo uvožen model v kateri koli smeri. S spreminjanjem lege spreminjamo trdnost, čas izdelave in hrapavost našega izdelka. Stremimo k temu, da izdelek zavrtimo v položaj, ki bo po končani izdelavi dosegel čim večjo odpornost proti zlomu in bo za tiskanje uporabili čim manj polnila.



Slika 3-26: Postavitev modela v Makerbot programu

Ko izdelek postavimo v željeno lego, mu po potrebi dodamo funkcijo »Support«. Ta delom izdelka, ki niso samostoječi, naredi podporo. Na primer, ko smo natisnili nosilec pomožnih koles, smo morali uporabiti to funkcijo, saj bi bil izdelek drugače povsem neuporaben.



Slika 3-27: Funkcija »Support« v Makerbot programu

Pri modeliranju moramo upoštevati tolerance, da dobimo izdelek s popolnoma natančnimi merami. Najprej smo imeli s tem nekoliko težav, saj nismo imeli podatka, v kakšnih mejah morajo biti tolerance. Ko pa smo natisnili nekaj prvih prototipov, smo prišli do ugotovitve, da moramo pri luknjah za ležaje, baterijo in ventilatorja upoštevati določeno toleranco: željena mera +0.4 mm dodatka. Na primer, ko smo naredili luknjo za ležaj, ki je nameščen na os za premik naklona gosenic, smo morali ležaju, katerega zunanji obseg meri 8 mm, dodati 0.4 mm za tolerance tiskalnika. Program pa nam ponudi tudi vpogled, kako bo izdelek praktično natisnjen. Prikaže nam tudi veliko ostalih informacij, kot so: število potrebnih slojev, čas tiskanja, poraba polnila itd. Tako lahko lažje vidimo morebitne napake.

Slika 3-28: Jedro robota pred 3D-tiskanjem

3.5 KOMPONENTE

Vse v nadaljevanju omenjene komponente so kupljene. Pred nakupom smo dobro premislili in na spletu poiskali najboljšega ponudnika. Večkrat je zaradi čakanja na dostavo določene komponente prišlo do zastoja dela.

3.5.1 Raspberry Pi

Raspberry Pi Model 3 B je mikroračunalnik velikosti kreditne kartice, ki ga uporabljamo za krmiljenje našega robota. Izbrali smo ga zaradi števila možnosti, ki nam jih ponuja v primerjavi z Arduinom, prav tako je bil priporočen za kompatibilnost na tekmovanju. Za našo uporabo so predvsem pomembni GPIO (vhodi in izhodi za splošno uporabo), CSI (priključek za kamero), USB-priključki in WIFI-modul za brezžično povezavo. [18]

Tabela 1: Splošne specifikacije Raspberryja Pi 3 Model B

Raspberry Pi 3 Model B	
Sistem na čipu	Broadcom BCM2837
Centralna procesna enota	4x ARM Cortex-A53, 1.2 GHz
Grafična procesna enota	Broadcom VideoCore IV
RAM	1GB LPDDR2 (900 MHz)
Povezava	10/100 Ethernet, 2.4 GHz 802.11n wireless
Bluetooth	4.1 Classic, Low Energy
Shramba	microSD
GPIO	40-pin header, populated
Priključki	HDMI, jack, 4 x USB 2.0, Ethernet, CSI, DSI
Operacijski sistem	Raspbian (različica Debiana, Linux)

Slika 3-29: Raspberry Pi 3 Model B [18]

3.5.2 Aktivni del

Dynamixel AX-12A

AX-12A so pametni servomotorji, ki so sposobni slediti hitrosti, poziciji, temperaturi in obremenitvi. Prav tako kot Raspberry Pi so bili priporočeni za kompatibilnost na tekmovanju. Na robotu uporabljamo štiri motorje za gibanje, enega za spreminjanje oblike in tri za premik roke. [3]

Tabela 2: Splošne specifikacije motorja Dynamixel AX-12A

AX-12A	
Teža	54,6 g
Dimenzije	32 mm x 50 mm x 40 mm
Resolucija	0,29 °
Obremenitev, pri kateri obtiči	1,5 Nm (pri 12 V in 1,5 A)
Hitrost brez obremenitve	59 rpm (pri 12 V)
Meje obračanja	• 0°–300° • Neomejeno
Temperatura obratovanja	–5 °C– +7 °C
Napetost	9 V–12 V (priporočeno 11,1 V)
Signal ukazov	Digitalni paket
Tip protokola	Poldvojna asinhronska serijska komunikacija
ID	254 ID (0–253)
Hitrost komunikacije	7343 bps–1 Mbps
Material	Inženirska plastika

Slika 3-30: Dynamixel AX-12A [3]

Makeblock Mini gripper

Prijemalo Makeblock se nahaja na roki robota. Izbrali smo ga zaradi preprostega mehanizma in lahkosti. Deluje preko 9 g servomotorja, ki ga krmilimo s pomočjo PWM-regulacije. Prijemalo ima vgrajeno mehansko varovalo pred preobremenitvijo, ki preprečuje obrabo motorja in zagotavlja dolgotrajno uporabo. Je zelo primeren za dvigovanje majhnih predmetov dimenzij med 22 mm in 60 mm ter teže do 60 g. [11]

Slika 3-31: Makeblock Mini gripper [11]

3.5.1 Komunikacija

Predstavili bomo različne vrste komunikacije med motorji in RPijem, ki smo jih preizkusili, ter izbiro daljinca za vodenje robota.

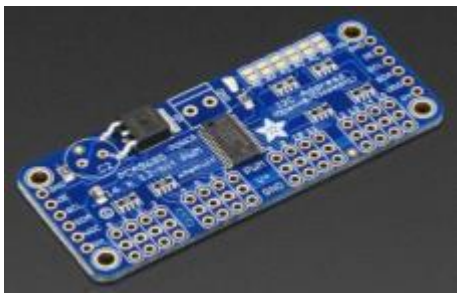
OpenCM9.04 tipa C

OpenCM je vmesni mikrokrmilnik, saj predstavlja vmesni člen med RPijem in motorji. Za uporabo je potrebno najprej vzpostaviti serijsko povezavo med RPijem in OpenCMom in nato dalje na motorje. Ker je OpenCM neke vrste mikrokrmilnik, moramo nanj naložiti drug program, ki informacije in ukaze z RPija pretvarja v protokol, ki ga razumejo aktuatorji v motorjih. Za ta način se nismo odločili, ker je bil prezahteven in na spletu nismo našli dovolj uporabnega gradiva. [13]

Slika 3-32: OpenCM9.04 tipa C [13]

Adafruit 16-Channel 12-bit PWM/servo driver

Zamislili smo si, da bi preko GPIO-pinov in 16-kanalnega vmesnika krmilili motorje s pomočjo PWM-regulacije, vendar se je to na koncu izkazalo povsem neuporabno, saj aktuatorji motorjev Dynamixel uporabljajo popolnoma drug komunikacijski protokol. Če bi jih želeli krmiliti preko PWM-regulacije, bi potrebovali neke vrste adapter, ki pa trenutno ne obstaja. [1]



Slika 3-33: 16-Channel PWM/Servo driver [1]

USB2AX v3.2a

Na koncu smo le našli to, kar smo iskali. To je USB2AX, ki je majhen in ponuja preprosto komunikacijo z motorji. Deluje tako, da sam vzpostavi serijsko povezavo preko USB-priključka na RPiju. Na internetu smo našli še zadovoljivo število gradiv, med katerimi je najpomembnejši Python modul z osnovnimi funkcijami za krmiljenje motorjev. [25]

Slika 3-34: USB2AX v3.2a [25]

Logitech Gamepad F710

Za daljinsko vodenje uporabljamo Logitechov Gamepad F710. Zanj smo se odločili zaradi lastnih izkušenj o kakovosti izdelka in številu programirljivih tipk. Deluje preko sprejemnika v obliki USB-ključka, s katerim vzpostavimo 2.4 GHz brezžično povezavo. Povezava je dovolj dobra, da ne vidimo vidnih zaostankov. Daljinci te vrste so uporabniku najprijaznejši in preprosti za uporabo. [9]

Tabela 3: Pomembnejše specifikacije za Logitech Gamepad F710

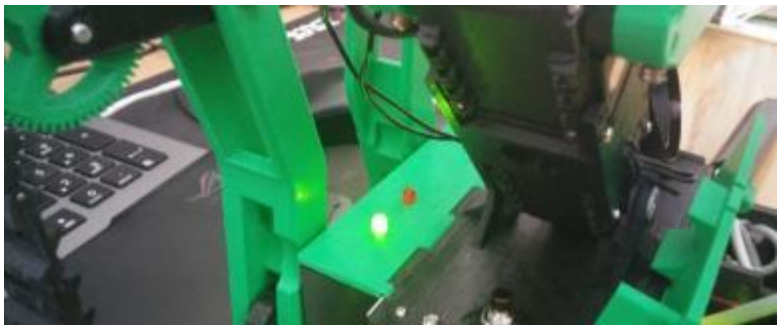
Logitech Gamepad F710	
Teža	285 g
Tip povezave	Brezžični (nano USB)
Frekvenca	2.4 GHz
Domet povezave	10 m

Slika 3-35: Logitech Gamepad F710 [9]

3.5.2 Signalizacija

Za signalizacijo smo uporabili 3 LED-diode. Prva zelena LED-dioda signalizira vklop glavnega stikala in je pozicionirana poleg stikala v notranjosti robota. Drugi dve LED-diodi, rdeča in zelena, pa signalizirata potek programa. Na primer, ko je program v glavni zanki, sveti zelena, če niso povezani vsi motorji, sveti rdeča in v primeru prazne baterije izmenično utripata obe.

Slika 3-36: LED-dioda glavnega stikala



Slika 3-37: Signalizacijski LED-diodi

3.5.3 Hlajenje

V polnem obratovanju robota se elektronske komponente čez čas začnejo segrevati. Še posebej obremenjeni so RPi, Polulu in tranzistor. Povišana temperatura lahko negativno vpliva na delovanje posameznih komponent in posledično na robota. Z ventilatorjema znamke Gostime dovajamo hladen zrak v prostor z elektroniko in s tem zmanjšamo temperaturo. Nahajata se ob bokih robota in obratujeta skozi ves čas njegovega delovanja. Vklapljamo jih z GPIO-izhodi preko tranzistorja.

Slika 3-38: Gostime brushless DC fan

3.5.4 Senzorika

Končna stikala so zelo pomembna pri gibljivih delih, saj lahko z njimi omejujemo gibanje in tako preprečujemo kakršnekoli zlome in poškodbe motorjev.

SW152-ND

Mikrostikali SW152-ND uporabljamo kot končni stikali pri premikanju gosenic. Stanje stikal beremo preko GPIO-vhodov. Zanje smo se odločili, ker so ravno prave velikosti in ker so bili na zalogi v lokalni trgovini z elektromehanskimi komponentami. [21]

Slika 3-39: SW152-ND končno stikalo [21]

Push Button Momentary Switch

Ti tipkali uporabljamo kot končni stikali za prvi sklop rok. Zanju smo se odločili, ker se v primerjavi z mikrostikali SW152-ND, zaradi vgrajenega navoja, veliko lažje pritrdita.

Raspberry Camera Board – Night Vision & Adjustable-Focus Lense

Kamero uporabljamo za deljenje slike skozi brezžično povezavo, tako da lahko robota krmilimo, ne da ga vidimo. Za lažje upravljanje in pogled z več zornih kotov je kamera pritrjena na robotski roki. Ta model smo si izbrali, ker je opremljen z nočnim vidom, v primeru reševanja v temi. [18]

Tabela 4: Splošne specifikacije kamere

5 Megapixel OV5647 Camera	
Resolucija slike	5 MP
Resolucija videa	1080 p
FPS	30
Dimenzije	25 mm x 24 mm

Slika 3-40: Kamera z nočnim vidom in nastavljivo lečo [18]

3.5.5 Napajanje

Napajanje za RPija in ostale porabnike je ločeno, ker napetost preko Poluluja ni bila dovolj stabilna za zagon in napajanje RPija.

Turnigy 3200 mAh 3S 30C LiPoly Pack

Baterijo Turnigy uporabljamo za napajanje motorjev in ostalih porabnikov (LED, ventilatorja, stikala itd.), razen RPija. Morali smo paziti na prekomerno polnjenje in praznjenje, saj jo lahko s tem uničimo. Napetost mora biti med 12.5 V in 10.5 V. [24]

Tabela 5: Splošne specifikacije baterije Turnigy

Turnigy LiPoly Pack	
Kapaciteta	3200 mAh
Konfiguracija	3S1P / 11.1 V / 3 Cell
Teža	265 g
Dimenzije	139 mm x 45 mm x 22 mm

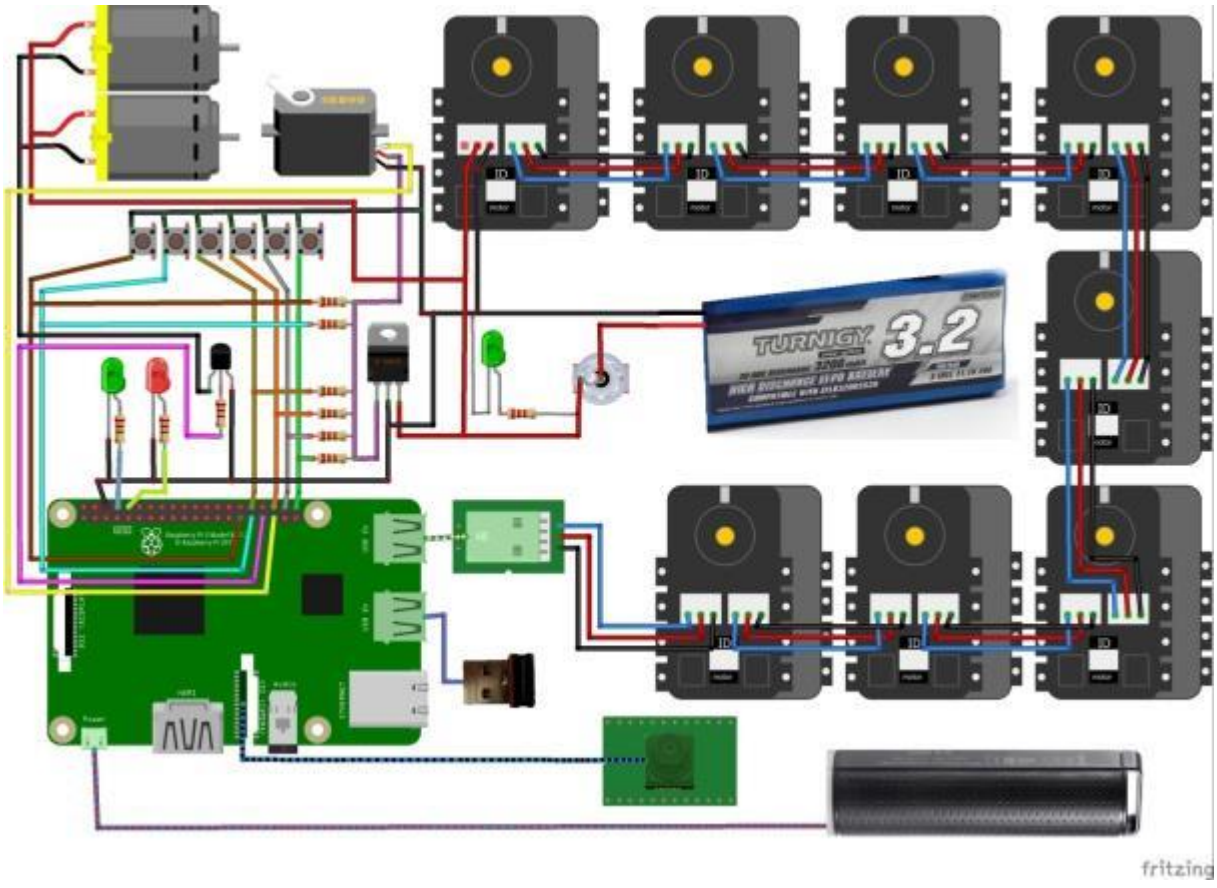
Slika 3-41: Turnigy baterija [24]

Prenosna baterija »Oval«

Prenosno baterijo Oval uporabljamo za napajanje RPija. Ima kapaciteto 2600 mAh in izhodno napetost 5 V z nazivnim tokom 1 A, kar je ravno prav za večurno uporabo. Za to prenosno baterijo smo se odločili zaradi njenih majhnih dimenzij.

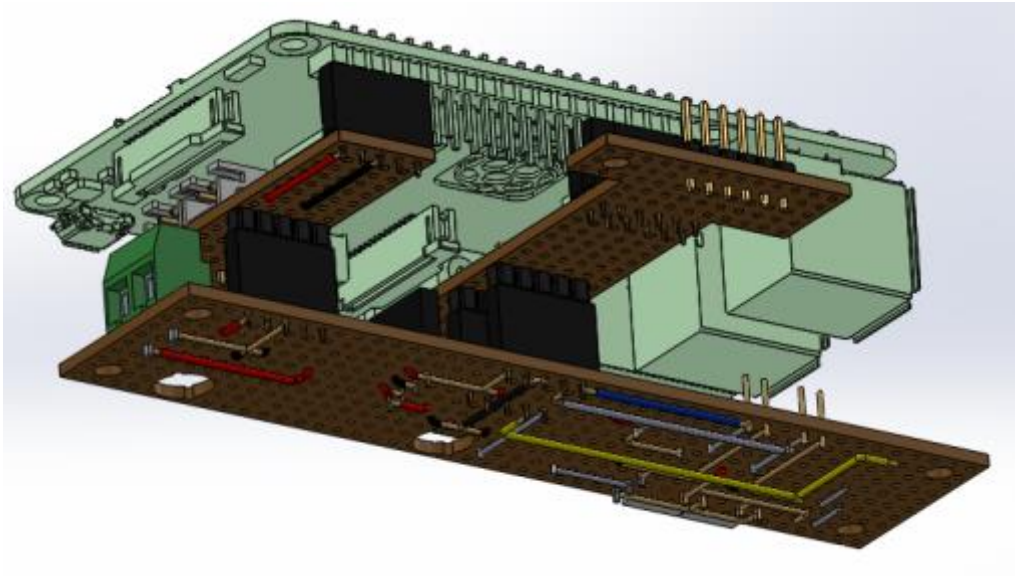
3.6 ELEKTRIČNA INŠTALACIJA

Za napajanje smo uporabili Li-po baterijo Turnigy 3.2. Pozitivni del gre najprej skozi glavno stikalo. Ko ga sklenemo, začne svetiti zelena LED-dioda. S tem ko sveti, nas opozarja, da je tokokrog sklenjen. Tako dobijo neposredno napetost Dynamixel AX-12 A motorji. Na pretvorniku napetosti se z 12 V reducira na 5 V. Slednja napetost se uporablja pri šestih pull down uporih oziroma stikalih in kot napajalna napetost Mini gripperja. Z njo smo hoteli preko pina napajati Rpi, vendar zaradi nestabilnosti toka ta varianta ni delovala. Napajanje Rpija smo rešili s prenosno baterijo. Programsko smo definirali 4 izhodne pine. Z dvema krmilimo signalizacijski diodi. Vsaka dioda dobi iz pina napetost 3,3 V. Diodi varujeta 65 Ω upora. Z enim pinom krmilimo Mini gripper, z drugim pa odpiramo tranzistorsko stikalo. Tranzistor spusti 12 V napetost iz ventilatorjev na negativni pol. Za boljšo preglednost je v prilogi 5 A3-list el. vezja z legendo.



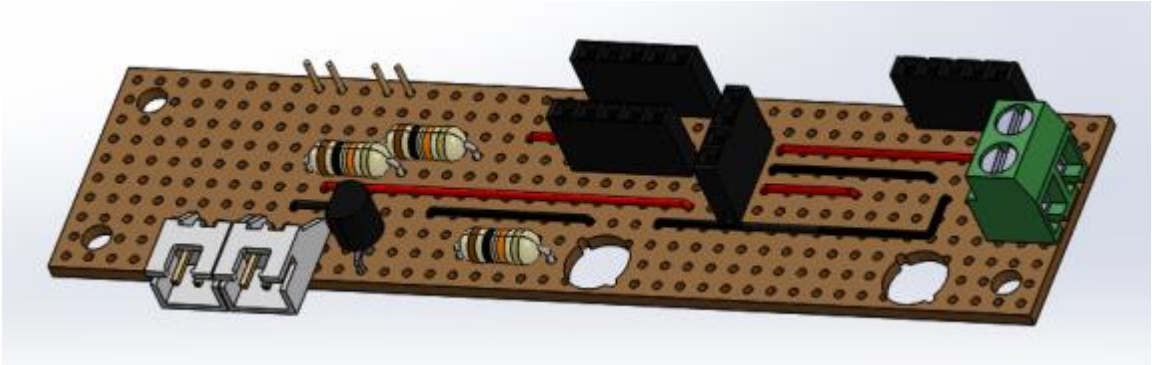
Slika 3-42: Blokovna shema el. vezja

V jedru robota je zelo malo prostora. Pri konstruiranju smo se trudili zasnovati zaprt prostor, namenjen samo elektroniki. S tem smo zaščitili občutljive komponente pred mehanskimi deli ter zunanjimi vplivi. Celotno vezje je sestavljeno iz treh bakrenih ploščic s komponentami in Rpijem. Vezje je razdeljeno na 3 nadstropja. V prvem je glavna bakrena ploščica, na katero pride napajanje iz baterije. Na njej je tudi Polulu. V drugem nadstropju sta dve bakreni ploščici. Manjša skrbi za priključitev signalnih diod in negativnega dela na Rpi. Na drugi ploščici pa so narejene vezave za končna stikala in elemente na robotski roki. Zadnje nadstropje zaseda Rpi. Vsa nadstropja so med sabo povezana s pini, ki jih lahko hitro in enostavno združujemo.

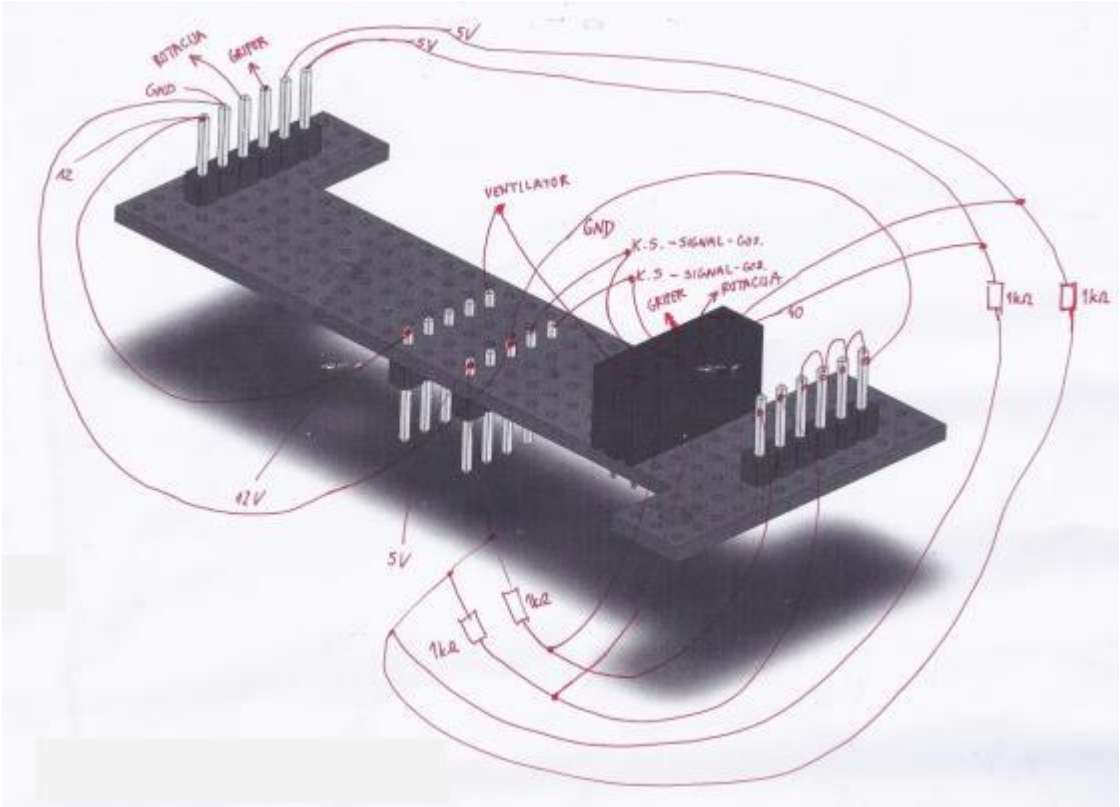


Slika 3-43: Model električnega vezja

V SolidWorksu smo narisali vsako bakreno ploščico posebej. Nanjo smo postavili vse komponente na točno določeno mesto. Tako s spajkanjem ploščice nismo imeli težav. V glavnem vezju smo narisali tudi povezave. Ob spajkanju lažje vidimo, kako bolje leži kabel, zato smo na osrednji bakreni ploščici narisali povezave s kemičnim svinčnikom. Oseba, ki je spajkala, si je sama določila napeljavo kablov.



Slika 3-44: Glavna bakrena ploščica



Slika 3-45: Sredinska bakrena ploščica

3.7 RPI-PROGRAM

Programirali smo v Pythonu, verzije 3.6, ker ga priporočajo za uporabo na RPiju in je eden najboljših jezikov za začetnike. Na spletu ponuja ogromno učnega materiala in že delujočih projektov.

3.7.1 Python

Python je močan in po izgledu privlačen jezik. Prednost daje lahki berljivosti in pisanju, kar omogoča zelo hitro učenje v primerjavi z ostalimi jeziki. Temu tudi služi čista sintaksa in prisiljena uporaba zamikov namesto zavitih oklepajev, tako da lažje vidimo, kaj vse spada pod določene sklope oziroma funkcije.

3.7.2 Moduli in knjižnice

V našem programu smo uporabili naslednje knjižnice:

```
from evdev import InputDevice
from pyax12.connection import Connection
import RPi.GPIO as GPIO
from time import sleep
from functions import *
```

Slika 3-46: Uporabljeni moduli oz. knjižnice

- evdev je modul za vhodne dogodke za Linux operacijske sisteme. Potrebovali smo ga za komuniciranje z daljincem Logitech F710. [5]
- pyax12.connection je modul, posebej narejen za komunikacijo z motorji Dynamixel, preko USB2AX-ključka. [17]
- RPi.GPIO je modul za kontroliranje vhodov in izhodov na RPiju. [7]
- time je modul s časovno povezanimi funkcijami. [22]
- functions je naša osebna knjižnica funkcij.

3.7.3 Funkcije

Ker je funkcij za gibanje robota veliko, smo se odločili, da zanje naredimo majhno knjižnico. Ta vsebuje funkcije za premikanje robota, preoblikovanje gosenic, premikanje roke, funkcije za specifične položaje roke in funkcije za ustavitev motorjev.

```
# UVOZ MODULOV/FUNKCIJ
from pyax12.connection import Connection
from time import sleep

# POSTAVITEV SERIJSKE POVEZAVE Z MOTORJI
sc = Connection(port="/dev/ttyACM0", baudrate=1000000)
```

Slika 3-47: Začetek knjižnice

Knjižnico začnemo z uvozom modulov. Tukaj potrebujemo modul pyax12 za komunikacijo z motorji in modul za časovno upravljanje. Nato postavimo serijsko povezavo z motorji, preko katere pošiljamo oziroma prejemamo podatke o motorjih.

```
def naprej(hitrost):
    sc.set_ccw_angle_limit(1, 0, degrees=False)
    sc.set_cw_angle_limit(1, 0, degrees=False)
    sc.set_ccw_angle_limit(2, 0, degrees=False)
    sc.set_cw_angle_limit(2, 0, degrees=False)
    sc.set_ccw_angle_limit(3, 0, degrees=False)
    sc.set_cw_angle_limit(3, 0, degrees=False)
    sc.set_ccw_angle_limit(4, 0, degrees=False)
    sc.set_cw_angle_limit(4, 0, degrees=False)
    sc.set_speed(1, speed=hitrost)
    sc.set_speed(2, speed=hitrost)
    sc.set_speed(3, speed=hitrost+1024)
    sc.set_speed(4, speed=hitrost+1024)
```

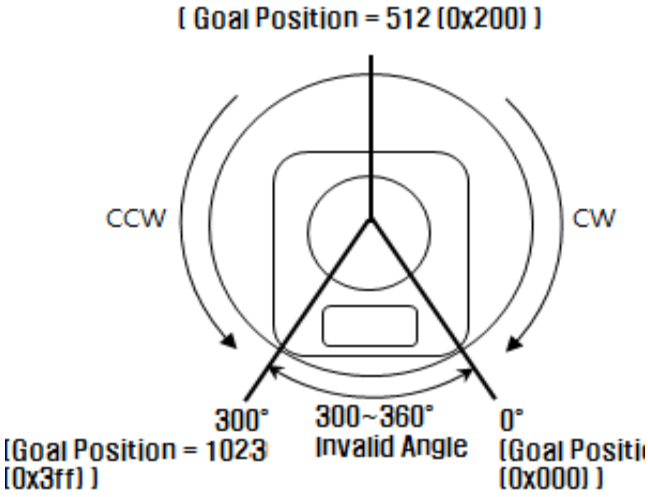
Slika 3-48: Primer funkcije za pogon

Najprej so podane osnovne funkcije za gibanje robota, kjer krmilimo motorje 1–4. Na sliki vidimo funkcijo za premik robota naprej. Najprej nastavimo motorje na način neskončnega vrtenja. To storimo tako, da nastavimo limit v smeri urinega kazalca in v nasprotni smeri na 0 (cw, ccw). Sledi nastavitev hitrosti, namesto katere imamo spremenljivko, saj imamo v programu možnost različnih hitrosti. Previdni moramo biti, kako so na robotu obrnjeni motorji. Če želimo, da se robot premika naprej, se morajo vrteti v različne smeri. Hitrost motorjev se giblje med vrednostmi 0 in 1023 v eno smer in 1024 in 2047 v drugo, zato pri motorju 3 in 4 prištejemo vrednost 1024. Ostale funkcije za gibanje (nazaj, levo, desno) so enake, z edino razliko, kje prištevamo vrednost 1024. Prav tako je podobna funkcija za motor za preoblikovanje gosenic (motor 5). Razlikuje se le v tem, da kontroliramo en motor, ne štirih.

```
def rokam1(poz):
    sc.set_ccw_angle_limit(6, 1023, degrees=False)
    sc.set_cw_angle_limit(6, 0, degrees=False)
    sc.goto(6, poz, speed=128, degrees=False)
```

Slika 3-49: Primer funkcije za premik roke

Sledijo funkcije za krmiljenje motorjev roke (motorji 6–8). Zgoraj vidimo funkcijo za prvi motor naše robotske roke. Od motorjev za pogon in preoblikovanje gosenic se razlikujejo tako, da moramo tokrat motorje nastaviti na način omejenega gibanja in da je spremenljivka namesto hitrosti željena pozicija motorja (v glavnem programu trenutni poziciji prištevamo neko vrednost). Način omejenega gibanja je posebej narejen za robotske roke. Motorji se zaklenejo, ko se ne premikajo, in tako ne padajo ob večjih težah same roke. Motor v eni smeri omejimo na maksimalno vrednost 1023 oziroma 300°. Kot vidimo na spodnji sliki, imajo motorji v tem načinu 60° nedosegljivih. V primeru, da želimo motor spraviti v neveljavno pozicijo, nam program javi napako. Če pa se motor že znajde v neveljavni poziciji, se sam premakne v najbližjo veljavno pozicijo (300° ali 0°). Hitrost premikanja pri teh funkcijah določimo že tukaj. Manjše so, ker ne želimo poškodovati robota ali motorjev.



Slika 3-50: Omejitve motorja [3]

Tretji motor roke, ki skrbi za premikanje kamere in gripperja, nosi najmanjšo težo, zato nas ne skrbi, da bi »padal«. Torej je funkcija za večjo funkcionalnost ista kot funkcija za pogon.

```
def zac_poz():
    sc.set_ccw_angle_limit(6, 1023, degrees=False)
    sc.set_cw_angle_limit(6, 0, degrees=False)
    sc.set_ccw_angle_limit(7, 1023, degrees=False)
    sc.set_cw_angle_limit(7, 0, degrees=False)
    sc.set_ccw_angle_limit(8, 1023, degrees=False)
    sc.set_cw_angle_limit(8, 0, degrees=False)
    sc.goto(6, 205, speed=128, degrees=False)
    sc.goto(7, 1020, speed=128, degrees=False)
    sleep(3)
    sc.goto(8, 205, speed=128, degrees=False)
    sleep(1)
    sc.set_ccw_angle_limit(7, 0, degrees=False)
    sc.set_cw_angle_limit(7, 0, degrees=False)
    sc.set_speed(7, 64)
    sleep(1)
    sc.set_speed(7, 0)
    sc.goto(8, 512, speed=128, degrees=False)
```

Slika 3-51: Funkcija za zložitev roke

Sledijo funkcije za specifične pozicije roke. Od teh je najkompleksnejša funkcija za zložitev robotske roke. Kot vidimo na zgornji sliki, je funkcija povsem brez zunanjih spremenljivk, vse pozicije in hitrosti so določene v njej. Časovni zamiki se uporabljajo, da se izognemo poškodbam. Ker se roka zaradi omejitev ne more povsem zložiti, moramo drugemu motorju roke spremeniti način iz omejenega gibanja na neomejeno in ga spustiti proti robotu za 1 sekundo. Za točno to pozicijo smo se odločili, ker najmanj škoduje kablu za kamero.

```
def gobica():
    pos1 = sc.get_present_position(6)
    pos2 = sc.get_present_position(7)
    pos3 = sc.get_present_position(8)
    sc.set_ccw_angle_limit(1, 0, degrees=False)
    sc.set_cw_angle_limit(1, 0, degrees=False)
    sc.set_ccw_angle_limit(2, 0, degrees=False)
    sc.set_cw_angle_limit(2, 0, degrees=False)
    sc.set_ccw_angle_limit(3, 0, degrees=False)
    sc.set_cw_angle_limit(3, 0, degrees=False)
    sc.set_ccw_angle_limit(4, 0, degrees=False)
    sc.set_cw_angle_limit(4, 0, degrees=False)
    sc.set_ccw_angle_limit(5, 0, degrees=False)
    sc.set_cw_angle_limit(5, 0, degrees=False)
    sc.set_ccw_angle_limit(6, 1023, degrees=False)
    sc.set_cw_angle_limit(6, 0, degrees=False)
    sc.set_ccw_angle_limit(7, 1023, degrees=False)
    sc.set_cw_angle_limit(7, 0, degrees=False)
    sc.set_ccw_angle_limit(8, 1023, degrees=False)
    sc.set_cw_angle_limit(8, 0, degrees=False)
    sc.set_speed(1, speed=0)
    sc.set_speed(2, speed=0)
    sc.set_speed(3, speed=0)
    sc.set_speed(4, speed=0)
    sc.set_speed(5, speed=0)
    sc.goto(6, pos1, speed=64, degrees=False)
    sc.goto(7, pos2, speed=64, degrees=False)
    sc.goto(8, pos3, speed=64, degrees=False)
```

Slika 3-52: Funkcija za izklop v sili

Na koncu nam je ostala še funkcija za izklop v sili. Motorje z neomejenim gibanjem preprosto ustavimo. Iz motorjev za roko preberemo trenutno pozicijo, nato jim ukažemo, naj se te pozicije držijo. S tem zagotovimo, da roka ne pade, če slučajno kaj nosi.

3.7.4 Glavni program

Tipke na daljincu smo poskusili optimizirati tako, da je vodenje robota čim bolj enostavno. To smo dosegli tako, da tipke bližje vrhu daljinca skrbijo za premik (robota ali roke) naprej, medtem ko tipke bližje spodnjemu delu daljinca skrbijo za premikanje nazaj. Sprva smo želeli imeti celotno premikanje robota na eni analogni palčki, vendar smo kmalu ugotovili, da pride s tem do velikega časovnega zamika med ukazi.

Slika 3-53: Načrtovanje vhodov na daljincu [Prilagojeno po 9]

1/2 – Premik robota naprej/nazaj (D-PAD gor/dol)

3/4 – Premik robota levo/desno (D-PAD levo/desno)

5 – Ponastavitev/izklop programa (BACK)

6 – Zložitev roke v začetno pozicijo (START)

7/8 – Premik prvega motorja roke naprej/nazaj (A/Y)

9/10 – Premik drugega motorja roke naprej/nazaj (B/X)

11/12 – Premik tretjega motorja roke naprej/nazaj (leva analogna palčka po Y-osi)

13 – Stisk/spust prijemala (leva analogna palčka kot tipka)

14/15 – Razteg roke naprej/nazaj (desna analogna palčka po Y-osi)

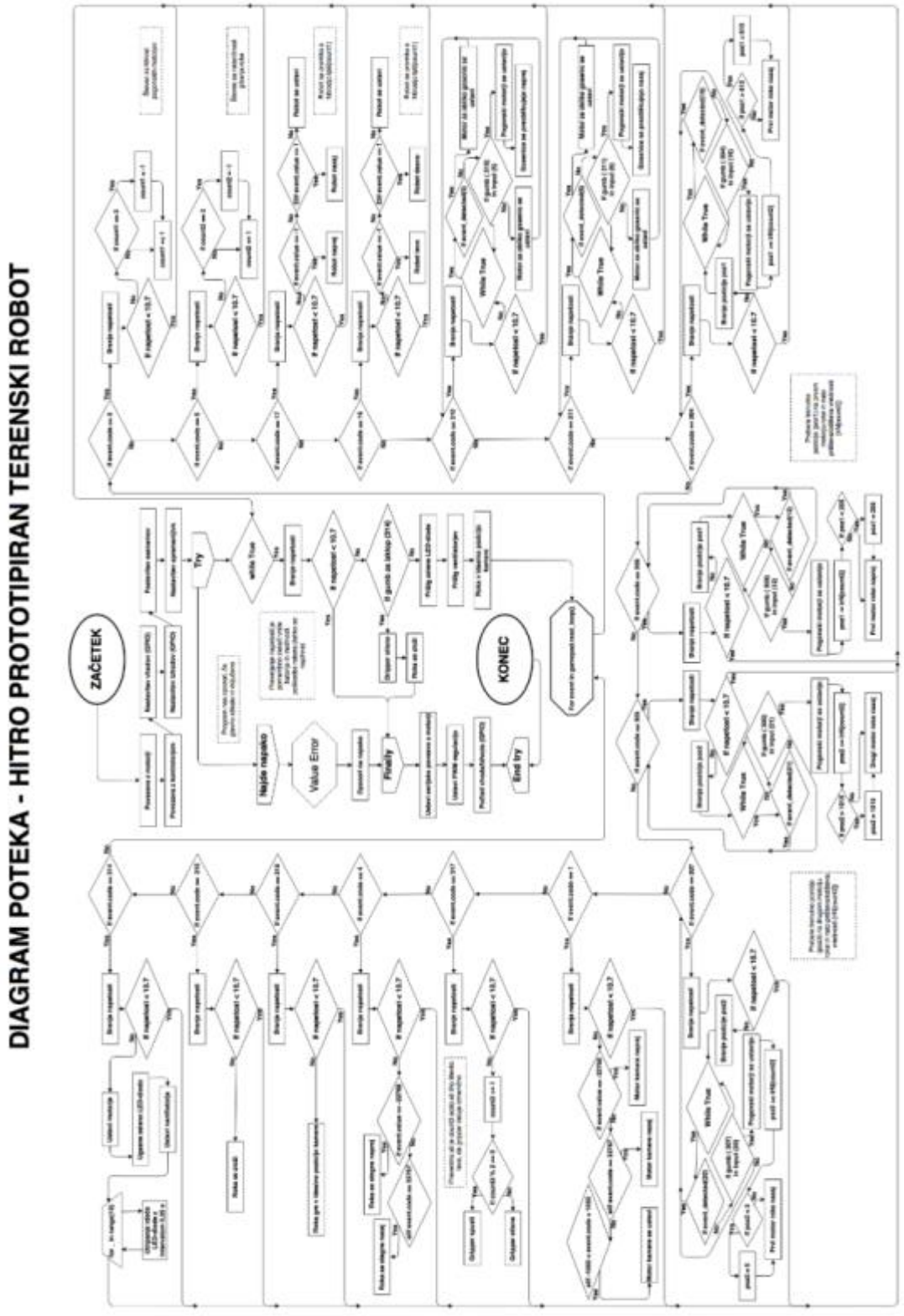
16 – Premik roke v idealno pozicijo za kamero (desna analogna palčka kot tipka)

17/18 – Preoblikovanje gosenic naprej/nazaj (LB/RB)

19 – Menjavanje natančnosti gibanja roke (RT)

20 – Menjava hitrosti premika (LT)

Na sliki vidimo celoten diagram poteka glavnega programa našega robota. Ker je zelo obsežen lahko v prilogi 6 pogledamo diagram na A3-formatu.



Slika 3-54: Diagram poteka

Ker je program preobsežen, da bi ga v celoti opisali, si lahko pogledamo popolnoma komentirano verzijo v prilogi 7. V nalogi bomo predstavili temeljno zanko našega programa in sklope za različne tipke daljinca, ki se v njej nahajajo.

```
for event in gamepad.read_loop():
```

Slika 3-55: Temelj programa

Okoli zanke, ki izhaja iz evdev knjižnice, je zgrajen cel program, saj v njej obravnavamo vsak dogodek, ki nam ga sporoči kontroler. Vsak dogodek ima svojo kodo in vrednost, npr. tipka A ima kodo 304 in vrednosti 0 ali 1. V primeru, da je tipka stisnjena, je vrednost 1. Poleg tipk ima kontroler tudi »D-PAD« (»directional pad«, dobesedno prevedeno smerna blazinica), ki ga uporabljamo za premik robota in nam ponuja vrednosti –1, 0 in 1 v x- in y-oseh. Imamo tudi dva različna potenciometra, prvi (uporabljen pri LT in RT) ima niz vrednosti med 0 in 255. Drugi, bolj natančnejši (dva sta uporabljena na vsaki analogni palčki) pa ima niz vrednosti od –32678 do 32767 v x- in y-oseh.

Skozi celoten program opazimo uporabo print funkcije, ki služi poročanju o tem, kaj robot trenutno dela oziroma kaj je naredil. S tem omogočimo vodenje robota, ne da ga vidimo. Poleg tega se pri vsakem sklopu pojavi pregled napetosti, saj lahko baterijo uničimo, če jo preveč spraznimo.

```
if event.code == 2 and event.value == 255:
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    if count1 == 2:
        count1 = -1
    count1 += 1
    print("Sprememba hitrosti! Trenutna hitrost je:", lp8[count1])
```

Slika 3-56: Primer števca

Ta sklop predstavlja števec za spremembo hitrosti premika. S pomočjo navadnega prištevanja in omejevanja števila smo naredili zanko števil 0, 1 in 2, ki gre kasneje skozi seznam (predmeti na seznamu se začnejo z 0, ne z 1) in nam poda določeno hitrost premikanja (hitrosti so četrtinska, polovična in polna). Na primer, če je count1 enak 2, se bo robot premikal s polno hitrostjo. Enak proces uporabimo pri menjavi natančnosti gibanja roke.

```
if event.code == 17:
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    if event.value == -1:
        print("Robot se premika naprej!")
        naprej(lpS[count1])
    elif event.value == 1:
        print("Robot se premika nazaj!")
        nazaj(lpS[count1])
    else:
        stop_pogon()
```

Slika 3-57: Primer sklopa za premik robota

Kot smo omenili že prej, uporabljamo za premik D-PAD-tipke. Če pritisnemo zgornjo tipko, se aktivira funkcija za premik naprej in hitrost glede na prejšnji sklop. Če pritisnemo spodnjo tipko, se aktivira funkcija za premik nazaj. Če je zaznana vrednost, ki ne pripada tema dvema tipkama, se pogonski motorji ustavijo. Za smeri levo in desno je postopek popolnoma enak, le da so tipke postavljene v X-osi.

```
if event.code == 310:
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    while True:
        if GPIO.event_detected(5):
            print("STOP - Končno stikalo (naprej)")
            gosenice(0)
            break
        if gamepad.active_keys(310) and GPIO.input(5):
            stop_pogon()
            print("Gosenice se preoblikujejo naprej!")
            gosenice(2047)
        else:
            gosenice(0)
            break
```

Slika 3-58: Primer sklopa za preoblikovanje gosenic

Sklop za preoblikovanje gosenic je prvi sklop, ki potrebuje končna stikala, saj ne želimo zlomiti robota ali obrabiti motorja. Ko program vidi, da je željena tipka aktivna, postavi neskončno zanko, v kateri konstantno pregledujemo stanje končnega stikala in tipke. V primeru, da program zazna padec vhoda končnega stikala z 1 na 0, se bo neskončna zanka prekinila. Gosenice pa se bodo preoblikovale le, ko bo aktivna željena tipka in končno stikalo ne bo podrto. V primeru, da eden izmed teh dveh primerov ni izpolnjen, se motor ustavi in neskončna zanka se prekine. Enak postopek je pri preoblikovanju v drugo smer, le da tam uporabljamo drugo tipko in končno stikalo.

```
if event.code == 304:
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    poz1 = sc.get_present_position(6)
    while True:
        if GPIO.event_detected(16):
            print("STOP - Končno stikalo (M1 - nazaj)")
            break
        if gamepad.active_keys(304) and GPIO.input(16):
            stop_pogon()
            print("Prvi motor roke nazaj!")
            poz1 += lrN[count2]
            if poz1 > 815:
                poz1 = 815
            rokam1(poz1)
        else:
            break
```

Slika 3-59: Primer sklopa za premik motorja roke

Sklop za premik motorjev roke je enak prejšnjemu, in sicer v zvezi s končnimi stikali. Razlika je v načinu premikanja motorjev, saj tokrat trenutni poziciji motorja prištevamo oziroma odštevamo neko vrednost, ki je odvisna od števca za natančnost gibanja roke. Dobljeno pozicijo tudi omejimo, kar poleg končnih stikal deluje kot dvojno varovalo (še posebej pri prvem motorju, saj se lahko giblje samo 180°). Pri drugih motorjih oziroma smereh se razlikujejo le željena tipka, končno stikalo, ali vrednost prištevamo ali odštevamo (odvisno, ali je gibanje roke naprej ali nazaj) in omejitev maksimalne ter minimalne pozicije.

```
if event.code == 1:
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    if event.value == -32768:
        print("Motor kamere in griperja naprej!")
        rokam3(256)
    elif event.value == 32767:
        print("Motor kamere in griperja nazaj!")
        rokam3(256 + 1024)
    elif -1000 < event.value < 1000:
        rokam3(0)
```

Slika 3-60: Primer sklopa za uporabo y-osi analogne palčke

Na sliki vidimo sklop za premikanje tretjega motorja roke, na katerega sta pritrjena kamera in prijemalo. Kot smo že omenili, imajo analogne palčke niz vrednosti med –32768 in 32767. Da se izognemo neželjenim časovnim zamikom, nas zanima le, ko je analogna palčka v minimalni oziroma maksimalni poziciji. Edino, kar je v tem sklopu zapleteno, je poiskati pravilen razpon vrednosti, kdaj se motor ustavi. To smo kot mnogo drugih problemov rešili s poskušanjem, dokler se nam motor ni zdel najbolj odziven.

```
if event.code == 317 and event.value == 1:
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    count3 += 1
    if count3 % 2 == 0:
        print("Prijemalo spusti!")
        p1.ChangeDutyCycle(3.5)
        sleep(0.1)
    else:
        print("Prijemalo stisne!")
        p1.ChangeDutyCycle(11.5)
        sleep(0.1)
```

Slika 3-61: Sklop za izmenično delovanje prijemala

Pri sklopu za stisk oziroma spust prijemala je posebnost, kako smo s pomočjo opazovanja sodosti spremenljivke count3 naredili izmenično delovanje. Vedno, ko pritisnemo željeno tipko, spremenljivki prištejemo 1. Nato preverimo sodost s pomočjo deljenja spremenljivke z 2 in prav tako preverimo, ali je ostanek enak 0. V primeru, da to drži, prijemalo stisne, če ne drži, pa prijemalo spusti.

```
if event.code == 314 and event.value == 1:
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    print("STOP - Motorji se ustavijo!")
    gobica()
    GPIO.output(13, 0)
    GPIO.output(14, 0)
    for _ in range(10):
        GPIO.output(15, 1)
        sleep(0.25)
        GPIO.output(15, 0)
        sleep(0.25)
    break
```

Slika 3-62: Sklop za ponastavitev programa

Na koncu nam je ostal še sklop za ponastavitev programa. Vse motorje preprosto ustavimo, roka obstane na mestu, izklopimo zeleno LED-diodo in ventilatorje, nato pa z utripanjem rdeče LED-diode signaliziramo, da se bo program resetiral.

3.8 RPI-CAM-WEB-INTERFACE

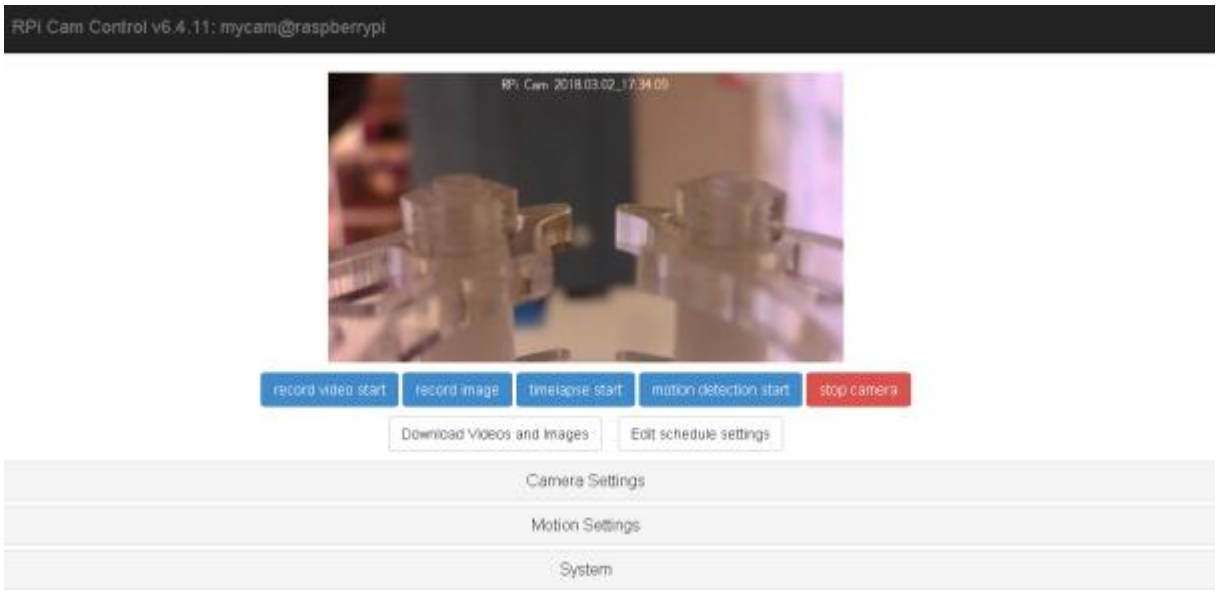
Za prenos videoslike smo uporabili RPi-cam-web-interface. To je spletni vmesnik za RPi-kamero, ki ponuja veliko funkcij. Uporabljamo predvsem naslednje:

- ogled, ustavitev in ponovni zagona predogleda v živo
- kontrola nastavitev kamere (svetlost, kontrast …)
- zajemanje full-HD-videov in slik

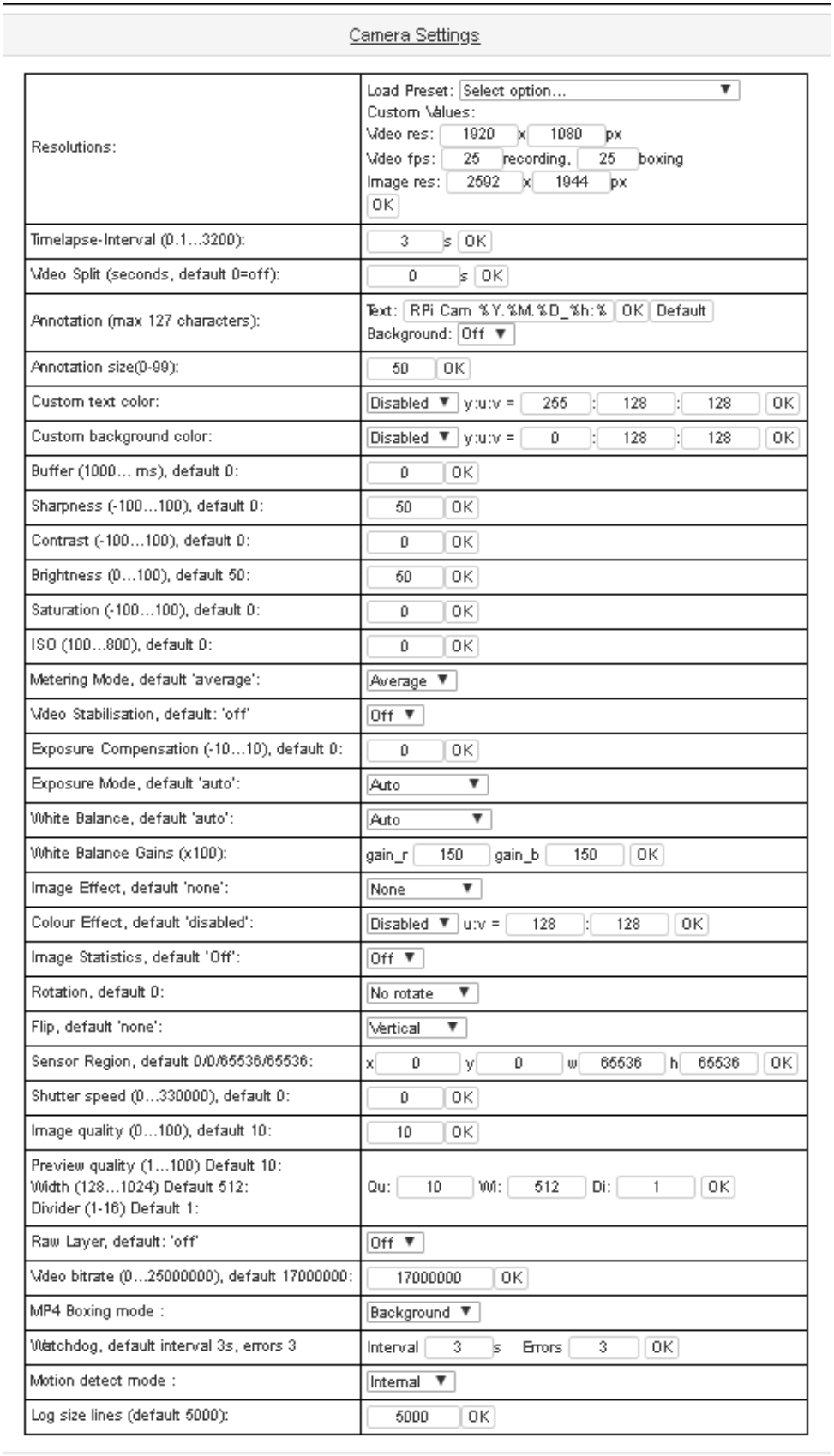
RPi-cam-web-interface se preprosto inštalira na RPi. Potrebno je nastaviti, da se kamera vedno vklopi, ko zaženemo RPi. Na katero koli napravo na istem omrežju lahko v spletni brskalnik vpišemo »ip-naslov-RPi-ja/html/«, kot vidimo na spodnji sliki. [20]

Slika 3-63: Primer url naslova

Odpre se nam RPi-cam-web-interface, kjer lahko vidimo posnetek kamere. Spodaj lahko spreminjamo različne nastavitve.



Slika 3-64: RPi-cam-web-interface



Camera Settings

Resolutions:	Load Preset: Select option... ▼ Custom values: Video res: 1920 x 1080 px Video fps: 25 recording, 25 boxing Image res: 2592 x 1944 px OK
Timelapse-Interval (0.1...3200):	3 s OK
Video Split (seconds, default 0=off):	0 s OK
Annotation (max 127 characters):	Text: RPi Cam %Y.%M.%D_%h:% OK Default Background: Off ▼
Annotation size(0-99):	50 OK
Custom text color:	Disabled ▼ y:u:v = 255 : 128 : 128 OK
Custom background color:	Disabled ▼ y:u:v = 0 : 128 : 128 OK
Buffer (1000... ms), default 0:	0 OK
Sharpness (-100...100), default 0:	50 OK
Contrast (-100...100), default 0:	0 OK
Brightness (0...100), default 50:	50 OK
Saturation (-100...100), default 0:	0 OK
ISO (100...800), default 0:	0 OK
Metering Mode, default 'average':	Average ▼
Video Stabilisation, default: 'off'	Off ▼
Exposure Compensation (-10...10), default 0:	0 OK
Exposure Mode, default 'auto':	Auto ▼
White Balance, default 'auto':	Auto ▼
White Balance Gains (x100):	gain_r 150 gain_b 150 OK
Image Effect, default 'none':	None ▼
Colour Effect, default 'disabled':	Disabled ▼ u:v = 128 : 128 OK
Image Statistics, default 'Off':	Off ▼
Rotation, default 0:	No rotate ▼
Flip, default 'none':	Vertical ▼
Sensor Region, default 0/0/65536/65536:	x 0 y 0 w 65536 h 65536 OK
Shutter speed (0...330000), default 0:	0 OK
Image quality (0...100), default 10:	10 OK
Preview quality (1...100) Default 10: Width (128...1024) Default 512: Divider (1-16) Default 1:	Qu: 10 Wi: 512 Di: 1 OK
Raw Layer, default: 'off'	Off ▼
Video bitrate (0...25000000), default 17000000:	17000000 OK
MP4 Boxing mode :	Background ▼
Watchdog, default interval 3s, errors 3	Interval 3 s Errors 3 OK
Motion detect mode :	Internal ▼
Log size lines (default 5000):	5000 OK

Slika 3-65: RPi-cam-web-interface nastavitve kamere [20]

3.9 NAČRTI ZA PRIHODNOST

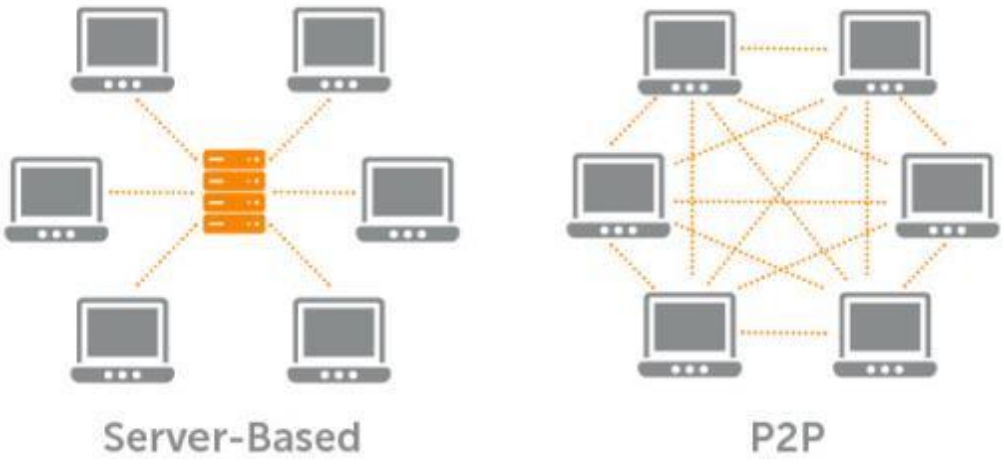
Marsičesa nismo naredili, saj v delovanje nekaterih reči nismo povsem prepričani.

3.9.1 BREZŽIČNA POVEZAVA

Naš robot je bil zasnovan kot terenski robot, zato mora imeti temu primerno brezžično povezavo. Trenutno uporabljamo deljenje podatkov s telefona (4G). Vendar vemo, da se lahko naravne nesreče zgodijo v območjih brez signala, zato mora povezava delovati brez internetne povezave. Našli smo dve verjetni rešitvi, a ju še nismo preizkusili zaradi našega pomanjkanja znanja o omrežjih. Skušali bomo realizirati in usposobiti vsaj eno izmed njih pred tekmovanjem.

Peer to peer

Prva rešitev, za katero menimo, da bo najverjetneje delovala, je »peer to peer« povezava. Kot vidimo na sliki, je posebnost te povezave, da povežemo dva računalnika direktno med seboj oziroma postavimo neke vrste privatno omrežje, brez kakršne koli internetne povezave ali vmesnega serverja.



Slika 3-66: Primerjava s skupnim strežnikom ali brez njega (P2P) [8]

RPi kot točka dosegljivosti

Druga rešitev je, da RPi nastavimo kot neke vrste točko dosegljivosti. Deluje po podobnem principu kot navaden WIFI-router (glej desno stran spodnje slike). RPi bo imel odprto povezavo, na katero se lahko priključijo ostale naprave, kot na primer telefon ali prenosni računalnik.

Slika 3-67: Primer delovanja WIFI-routerja [8]

4 PREDSTAVITEV REZULTATOV

Po sestavi vseh delov in komponent je nastal končni izdelek – robot, kakršnega smo si zamislili. Združili smo mehanski in programski del ter opravili prvi zagon. Po nekaj programskih popravkih je robot v popolnosti deloval. Predenj smo postavili razne prepreke in jih skušali premagati. Poleg tega smo testirali tudi delovanje roke in prenos slike. Po nekajurnem testiranju smo lahko potrdili oziroma zavrgli hipoteze ter zapisali ugotovitve.

Slika 4-1: Končni izdelek

Slika 4-2: Testiranje robota

Tabela 6: Potrditev hipotez

H1 – dolžina in širina robota ne bosta presegli 230 mm x 230 mm;	✅
H2 – mora imeti robotsko roko, opremljeno s kamero in primežem;	✅
H3 – konstrukcija mora biti 3D-natisnjena;	✅
H4 – robot mora biti zmožen premagovati ovire do višine 150 mm;	✅
H5 – komunikacija med robotom in upravljavcem mora biti brezžična;	✅
H6 – motorji morajo biti Dynamixel AX-12A znamke Robotis;	✅
H7 – mikroračunalnik mora biti Raspberry Pi;	✅
H8 – robot mora biti odporen na trke in prah;	✅
H9 – sestavni deli morajo imeti možnost hitre menjave;	⚠️
H10 – upravljanje robota mora biti enostavno;	✅

H1 – Prva hipoteza se nanaša na velikost robota. Zastavili smo si cilj, da robot ne sme presegati več kot 230 mm v širino in dolžino. Med konstruiranjem smo vseskozi pazili, da bi naredili čim manjšega in kompaktnega. Na koncu nam je uspelo narediti robota, ki meri v širino 200 mm. Z rezultatom smo zelo zadovoljni, saj imamo še več prostora, kot smo načrtovali. V osnovnem položaju gosenic je robot po dolžini ravno v meji 230 mm. S premikanjem gosenic se dolžina povečuje, kar je bil tudi naš namen. S tem povečujemo oprijem in težišče robota, kar nam pomaga pri premagovanju ovir. Hipoteza je potrjena, tekmovanje pa bo pokazalo, ali so dimenzije primerne.

H2 – Robotsko roko smo morali narediti. Brez nje ne bi mogli dosegati željenih slik s kamero. Naša roka je nekaj posebnega, saj ima enakovredne funkcije za vožnjo v eno ali drugo smer. Zaradi njene velike gibljivosti imamo veliko prednost pred ostalimi ekipami. Na koncu ima tudi gripper oziroma prijemalo, s katerim lahko prenašamo male objekte. Na začetku smo imeli težave z njeno težo. V določenih položajih motorji niso bili zmožni obdržati pozicije roke. Najprej smo težavo reševali z vzmetmi, ki so jo podpirale. Niso se obnesle, ker so v različnih položajih roke, različno vplivale nanjo. Vzmeti je nadomestil program, ki regulira položaj roke.

H3 – Tekmovanje zahteva, da je konstrukcija 3D-natisnjena. Vse sestavne dele, ki smo jih zasnovali, smo natisnili, razen nekaj izjem. Osi, ki prenašajo torzijske sile, so iz kovine. Nemogoče bi bilo, da bi natisnjene osi zdržale takšne sile, zato se nam zdi upravičeno, da smo izbrali tovrsten material. Po našem mnenju lahko potrdimo hipotezo, saj so vsi ostali elementi natisnjeni.

H4 – Skozi celotno snovanje smo omenjali prepreko v višini 150 mm. To je najvišja možna ovira na tekmovalni površini, zato smo si zadali hipotezo, da jo mora robot premagati. Robot je nastal po tretjem konceptu (stran 16). S preizkušanjem končnega izdelka na 150 mm visoki oviri smo ugotovili, da jo robot prevozi točno tako, kot smo si zamislili. S tem lahko potrdimo hipotezo.

H5 – Komunikacija med robotom in pilotom mora biti brezžična. Z daljincem nismo imeli težav, saj smo USB-ključek le vstavili v RPi in brezžična povezava je bila vzpostavljena. Od robota smo lahko oddaljeni do 15 m, pa povezava nemoteno poteka. Težje je bilo vzpostaviti povezavo s kamero. V poglavju **3.8 RPI-CAM-WEB-INTERFACE** je zadeva natančno opisana. Tudi to hipotezo potrjujemo.

H6 in H7 – Na tekmovanju naj bi ekipe pri sestavljanju svojega robota uporabljale enake komponente. Izpostavili so motor Dynamixel AX-12A in mikroračunalnik Raspberry Pi. Predloga smo upoštevali in uporabili obe komponenti. S tem so organizatorji dosegli kompatibilnost ekip. V času tekmovanja si lahko ekipe med seboj pomagajo. Hipoteza je potrjena.

H8 – Glavni namen robota je zmožnost vožnje po grobem in prašnem terenu. Ves čas smo se zavzemali, da bi izdelali čim robustnejšega in vzdržljivega robota. Jedro robota je pravokotne oblike s 4 mm in 3 mm debelo steno, ki je zelo trpežna. Vsi elektronski deli so v notranjosti robota in s tem zaščiteni pred zunanjimi silami in prahom. Zunanji deli, ki so neposredno izpostavljeni trkom, so močno izdelani. Testiranje je pokazalo, da 3D tiskani elementi niso tako šibki, kot smo si predstavljali. Hipoteza je potrjena.

H9 – Zadali smo si izdelati robota, ki ga lahko na hiter in enostaven način razstavimo na osnovne dele. S tem bi dosegli hiter servis in zamenjavo delov, ki bi utrpeli kakršnokoli poškodbo ali okvaro. Konstrukcijo robota smo naredili profesionalno, kar pomeni, da smo vse elektronske dele skrili v jedru. Konstrukcija je zapletena in natančno narejena. V jedrnem delu se križajo 3 osi, ki so med seboj povezane z zobniškimi sklopi. Vse zunanje dele robota je enostavno zamenjati. Tudi v jedru smo se trudili postaviti komponente čim bolj sistematično. Jedro je zaščiteno pred zunanjimi vplivi, vendar če pride do okvare elektronike, je proces zamenjave dolg, zato ni hipoteza v popolnosti potrjena.

H10 – Želeli smo, da bi robota upravljali na enostaven način. To pomeni, da bi bile ukazne tipke smiselno postavljene in uporabljene. Naš kontroler Logitech Gamepad F710 opravlja zanesljivo vlogo. Je ravno prave velikosti in vsebuje dovolj tipk, ki so na dosegu prstov. To hipotezo potrjujemo.

Ob preizkušanju robota smo prišli do dodatnih ugotovitev, ki so neposredno ali pa sploh niso povezane s hipotezami.

1. Motorji niso tako zmogljivi, kot smo si predstavljali. Iz njihove karakteristike izvemo, da so zelo močni, saj imajo navor 1,5 Nm. V praksi številke ne pridejo do izraza. Pri pogonu ni težav, saj štirje motorji poganjajo gosenici. Tudi pri sistemu za premikanje gosenic ni vidnih težav glede primanjkovanja moči, zahvaljujoč reduktorju in polžastemu gonilu. Težava nastopi pri robotski roki. Z njo smo imeli namen dvigovati prvi konec robota, če bi se ta kam zataknil. To ni mogoče, saj motorji komaj držijo prazno roko na željenem položaju.
2. Ob testiranju voznih lastnostih smo bili zelo zadovoljni. Presenetilo nas je odlično delovanje sistema za premikanje gosenic in njegova funkcionalnost. Težava se pokaže le v območju jedra, ki je med gosenicama. To je od dna dvignjeno samo 10 mm. Ob zelo grobem terenu lahko jedro nasede na podlago, saj gosenici nimata oprijema in posledično lahko robot obtiči. Težavo bi lahko rešili s povečanjem koles, vendar nam konstrukcija tega ne dopušča. Edina rešitev, ki jo trenutno vidimo, je, da na gosenici pritrdimo zatiče in s tem dvignemo robota od tal.
3. Robot se je nepričakovano prevrnil nazaj in pristal na drugi strani gosenic (na pomožnih kolesih). V takem položaju smo lahko nemoteno vozili naprej, le da je morala biti roka zložena. S tem smo dokazali, da je robot pripravljen na težke razmere, iz katerih se lahko vedno izvleče.

5 ZAKLJUČEK

Raziskovalna naloga nam je prinesla nov pogled in znanja s tehničnega področja. S tovrstno nalogo smo se srečali prvič, saj smo projekt izpeljali od začetka do konca. Na začetku se nam zastavljeni cilji niso zdeli težavni. A ko smo se lotili projekta s profesionalnim pristopom, smo ugotovili, da je potrebno veliko narediti za dosego željenega cilja. Pri nalogi smo se srečevali s številnimi težavami, ki smo jih spretno reševali s pomočjo mentorjev in literature. Veseli smo, da nam je uspelo izdelati reševalnega robota, ki ima prav poseben sistem za premikanje gosenic, ki ga na trgu še nismo opazili. S končnim izdelkom potrjujemo vse kriterije tekmovanja, zato upamo, da nas bo komisija sprejela na svetovno prvenstvo RoboCup RMRR, ki bo to leto potekalo 17. 6. v Montrealu v Kanadi. V prihodnje upamo, da bomo s projektom nadaljevali in ga razvijali še naprej. Iz trpežnejših materialov in boljšega komunikacijskega sistema nam lahko uspe narediti robota, ki bo nekega dne zares reševal življenja.

6 VIRI IN LITERATURA

[1] *Adafruit (*svetovni splet). (citirano 27. 2. 2018). Dostopno na naslovu: https://www.adafruit.com/product/815

[2] *Base pallets* (svetovni splet). (citirano 12. 2. 2018). Dostopno na naslovu: http://oarkit.intelligentrobots.org/home/the-arena/v0-7-arena/v0-7-arena-pallets/base-pallets/

[3] *Dynamixel AX-12A* (svetovni splet). (citirano 15. 2. 2018). Dostopno na naslovu: http://www.trossenrobotics.com/images/productdownloads/AX-12(English).pdf

[4] *EMU-robot* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: http://oarkit.intelligentrobots.org/home/the-build/tools-needed-for-emu-build/

[5] *Evdev* (svetovni splet). (citirano 29. 2. 2018). Dostopno na naslovu: http://python-evdev.readthedocs.io/en/latest/

[6] *FDM* (svetovni splet). (citirano 30. 2. 2018). Dostopno na naslovu: http://3d-druckcenter.at/fdm/

[7] *GPIO* (svetovni splet). (citirano 29. 2. 2018). Dostopno na naslovu: https://pythonhosted.org/RPIO/

[8] *Internet connection* (svetovni splet). (citirano 2. 3. 2018). Dostopno na naslovu: https://superuser.com/questions/857612/how-to-configure-switch-router-and-two-different-computer-with-one-single-intern

[9] *Logitech gamepad F710* (svetovni splet). (citirano 15. 2. 2018). Dostopno na naslovu: https://www.logitechg.com/en-gb/product/f710-wireless-gamepad

[10] *Makerbot replicator 5th gen* (svetovni splet). (citirano 25. 2. 2018). Dostopno na naslovu: https://www.digitaltrends.com/3d-printer-reviews/makerbot-replicator-5th-gen-review/

[11] *Mini gripper* (svetovni splet). (citirano 20. 2. 2018). Dostopno na naslovu: https://makeblockshop.eu/products/makeblock-mini-gripper-kit

[12] *More robots to the rescue* (svetovni splet). (11. 2. 2018). Dostopno na naslovu: https://spectrum.ieee.org/automaton/robotics/industrial-robots/japan-earthquake-more-robots-to-the-rescue

[13] *OpenCM-9.04-C-Microcontroller* (svetovni splet). (citirano 27. 2. 2018). Dostopno na naslovu: https://www.bananarobotics.com/shop/OpenCM-9.04-C-Microcontroller

[14] *Overall RoboCup parts list* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: http://oarkit.intelligentrobots.org/home/the-arena/v0-7-arena/overall-robocup-parts-list/

[15] *Potres v Ljubljani* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: https://sl.wikipedia.org/wiki/Ljubljanski_potres_1895#%C5%BDrtve_potresa

[16] *Putting it together* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: http://oarkit.intelligentrobots.org/home/the-arena/v1-0-arena/putting-it-together/

[17] *Pyax12* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: http://pyax-12.readthedocs.io/en/latest/

[18] *Raspberry Pi* (svetovni splet). (citirano 15. 2. 2018). Dostopno na naslovu: https://www.raspberrypi.org/

[19] *RoboCup* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: http://www.robocup.org/

[20] *RPi-Cam-Web-Interface* (svetovni splet). (citirano 2. 3. 2018). Dostopno na naslovu: https://elinux.org/RPi-Cam-Web-Interface

[21] *Switch* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: https://www.digikey.com/product-detail/en/omron-electronics-inc-emc-div/D2F-L/SW152-ND/83251

[22] *Time* (sleep) (svetovni splet). (citirano 29. 2. 2018). Dostopno na naslovu: https://docs.python.org/3/library/time.html

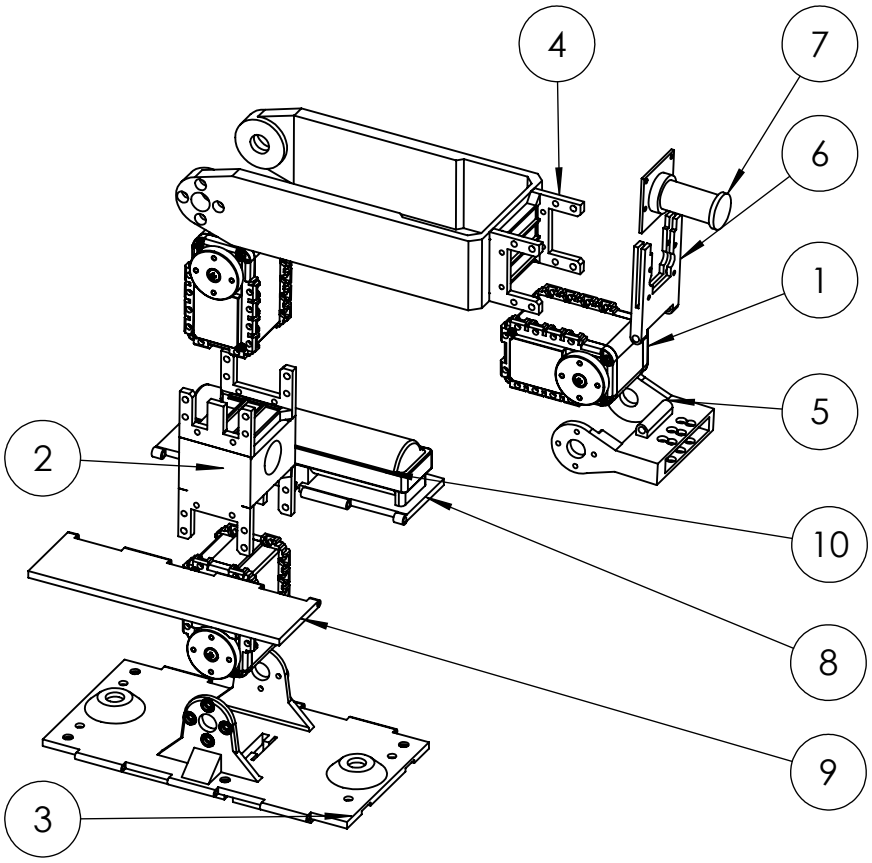
[23] *The open academic robot kit* (svetovni splet). (citirano 11. 2. 2018). Dostopno na naslovu: http://oarkit.intelligentrobots.org/home/

[24] *Turnigy* (svetovni splet). (citirano 15. 2. 2018). Dostopno na naslovu: https://hobbyking.com/en_us/turnigy-3200mah-3s-30c-lipoly-pack-w-ec3-e-flite-compatible-eflb32003s30.html?___store=en_us

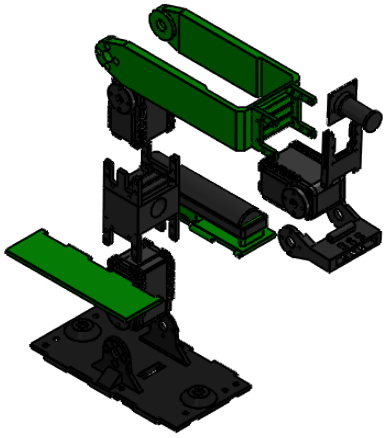
[25] *Usb2ax* (svetovni splet). (citirano 27. 2. 2018). Dostopno na naslovu: http://www.trossenrobotics.com/usb2ax

[26] *3D printed skull* (svetovni splet). (citirano 25. 2. 2018). Dostopno na naslovu: http://www.afr.com/technology/3d-printing-set-to-aid-surgeons-as-asxlisted-3d-medical-signs-global-partnership-20150824-gj68zc

[27] *3D printet house* (svetovni splet). (citirano 25. 2. 2018). Dostopno na naslovu: https://www.engadget.com/2017/03/07/apis-cor-3d-printed-house



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	DYNAMIXEL-AX-12A		3
2	Majhna_roka		1
3	Pokrov		1
4	Srednja_roka		1
5	Nastavek_kamere		1
6	Drzalo_kamere		1
7	Pi_kamera		1
8	Stranska_ploščica_prenosne_baterije		1
9	Stranska_ploščica_signalizacijskih_diod		1
10	Baterija		1



	IME IN PRIIMEK	PODPIS	DATUM		
NARISAL	Nejc Ločičnik		8.3.2018		
NARISAL	Žan Sotošek		8.3.2018		
NARISAL	David Žuraj		8.3.2018		

NASLOV

Sestavnica_robotske_roke

MATERIJAL:

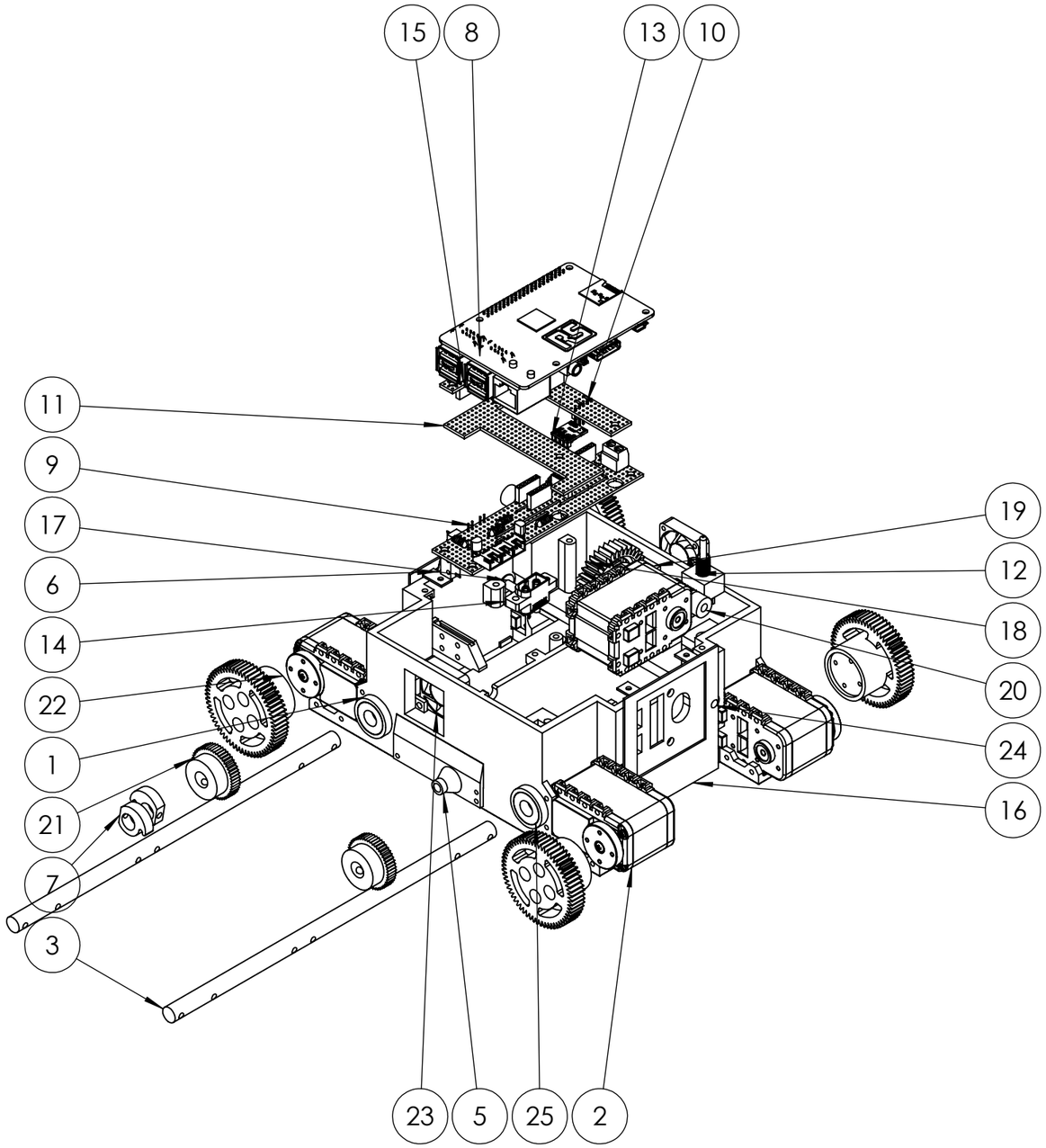
ŠT. RISBE

1

A4

MERILO: 1:3

LIST 1 OD 1



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Jedrni_del		1
2	DYNAMIXEL-AX-12A		5
3	Os_8mm		2
4	Stranski_nosilec_desni		1
5	Stranski_nosilec_levi		1
6	Končno_stikalo		2
7	Prožilec_stikala		2
8	Raspberry		1
9	Glavno_vezje		1
10	Napajanje_jagode		1
11	Bakrena_ploščica_5x3		1
12	Glavno_stikalo		1
13	Polulu		1
14	XT60E-M_konektor		1
15	Vezje_zgornje_plošče		1
16	Dotična_stikala		2
17	Lezaj_4x8mm		3
18	Zobnik_veliki_36		1
19	Zobnik_majhen		1
20	Polž		2
21	Zobnik_polža		2
22	Pogonsko_kolo		4
23	25x25x10-ventilator		2
24	Os_4mm		1
25	Lezaj_8x16mm		4

	IME IN PRIIMEK	PODPIS	DATUM
NARISAL	Nejc Ločičnik		8.3.2018
NARISAL	Žan Sotošek		8.3.2018
NARISAL	David Žuraj		8.3.2018

NASLOV: Sestavnica_jedrnega_dela

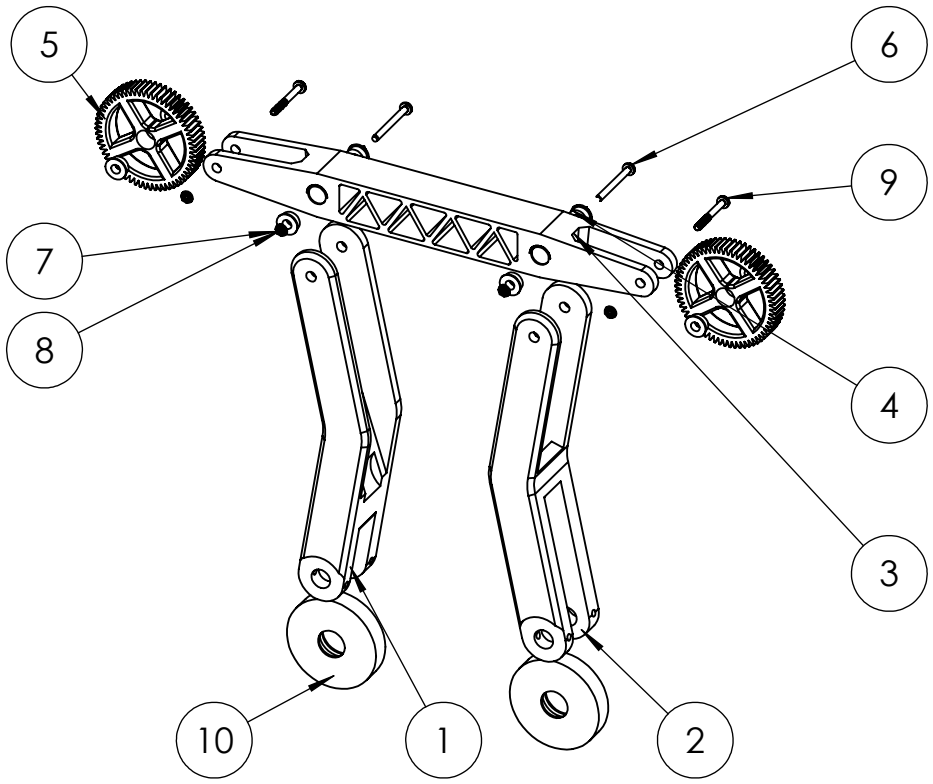
MATERIJAL:

ŠT. RISBE: 2

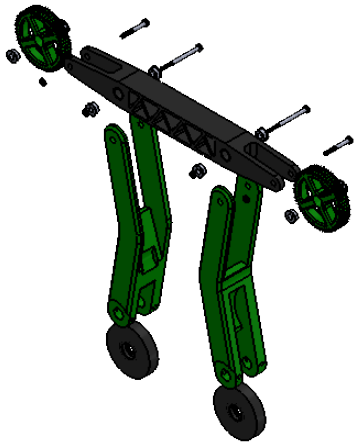
A3

MERILO: 1 : 2.5

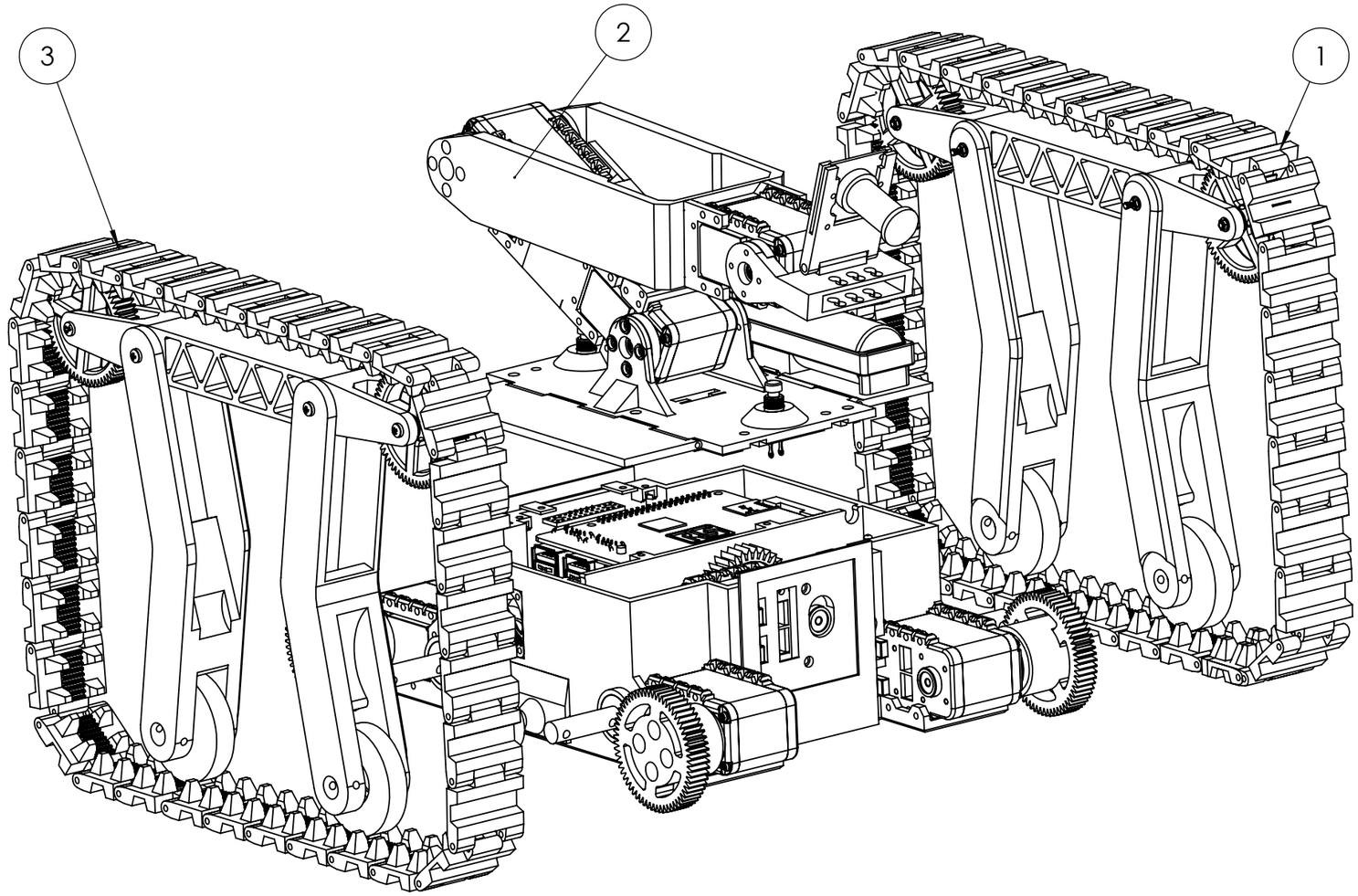
LIST 1 OD 1

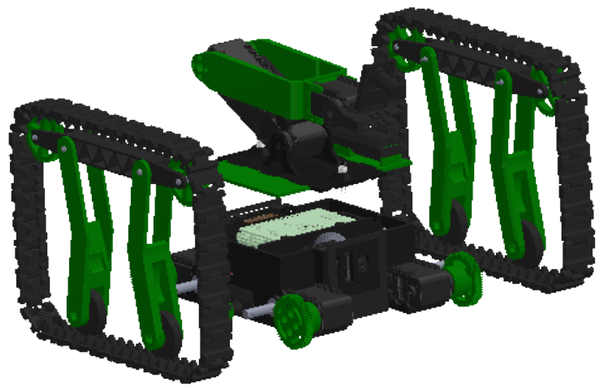


ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Desna_vodlina_roka		1
2	Leva_vodilna_roka		1
3	Nosilec_pomožnih_koles		1
4	Lezaj_4x8mm		8
5	Zobnik_45_gosenica		2
6	Vijak_m4x35		2
7	Matica		4
8	Podložka		4
9	Vijak_m3x15		2
10	Pomožno_kolo		2



	IIME IN PRIIMEK	PODPIS	DATUM				NASLOV
NARISAL	Nejc Ločičnik		8.3.2018				Sestavnica_pogonskega_sklopa
NARISAL	Žan Sotošek		8.3.2018				
NARISAL	David Žuraj		8.3.2018				
				MATERIJAL:			ŠT. RISBE 3
							A4
							MERILO: 1:3
							LIST 1 OD 1





ITEM NO.	ASSEMBLY NUMBER	DESCRIPTION	QTY
1	Sestavnica_pogonskega_sklopa_desni_del		1
2	Sestavnica_jedrnega_dela		1
3	Sestavnica_pogonskega_sklopa_levi_del		1

	IME IN PRIIMEK	PODPIS	DATUM			
NARISAL	Nejc Ločičnik		8.3.2018			
NARISAL	Žan Sotošek		8.3.2018			
NARISAL	David Žuraj		8.3.2018			

NASLOV: Sestavnica_robota

MATERIJAL:

ŠT. RISBE: 4

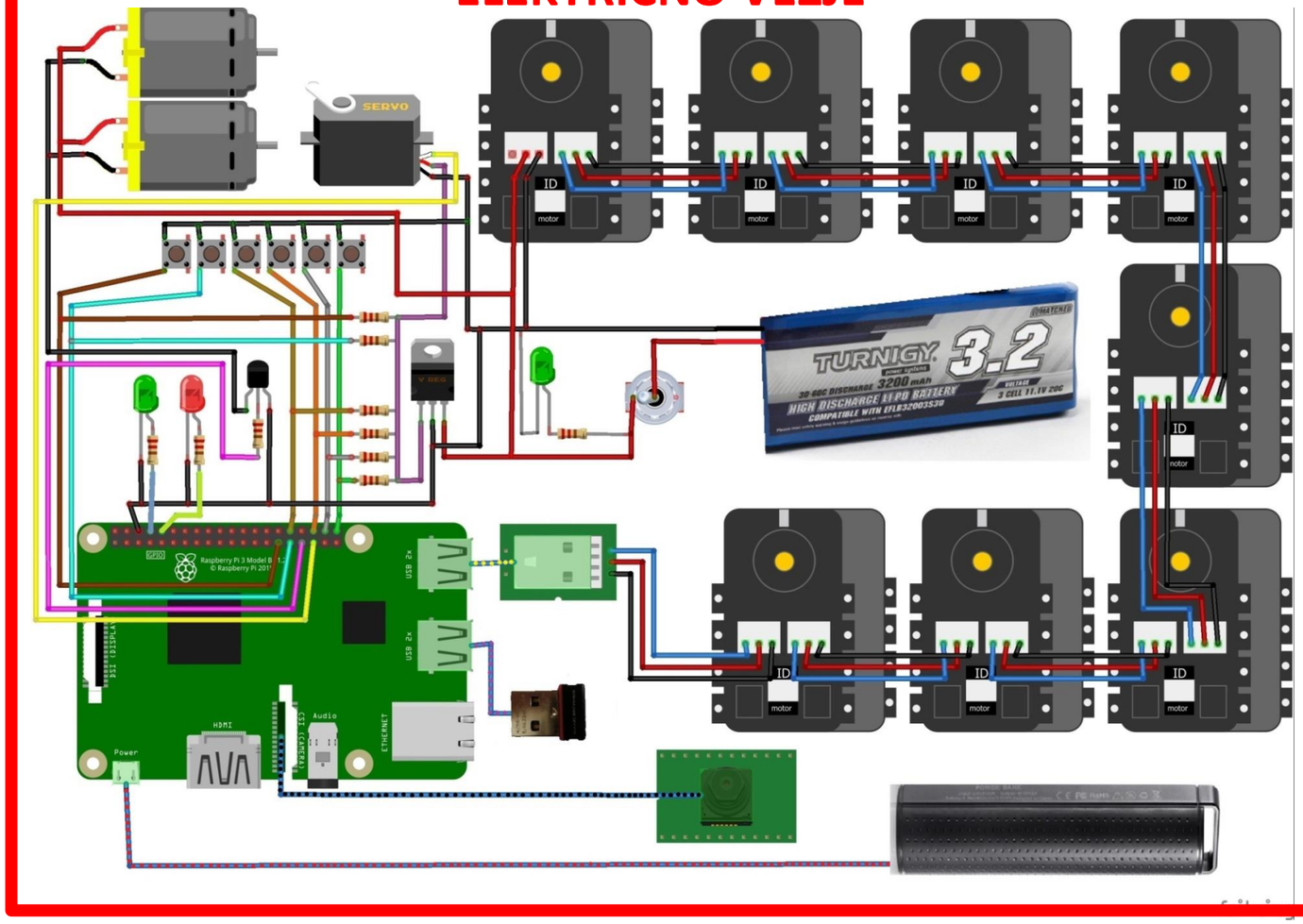
A3

MERILO: 1:2

LIST 1 OD 1

Priloga 5: Blokovna shema el. vezja z legendo

ELEKTRIČNO VEZJE

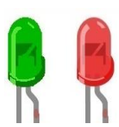


- 12 V
- 5 V
- GND
- DATA
- GPIO19 (0 V – 3,3 V)
- GPIO21 (0 V – 5 V)
- GPIO20 (0 V – 5 V)
- GPIO16 (0 V – 5 V)
- GPIO12 (0 V – 5 V)
- GPIO6 (0 V – 5 V)
- GPIO5 (0 V – 5 V)
- GPIO13 (0 V – 3,3 V)
- GPIO15 (0 V – 3,3 V)
- GPIO14 (0 V – 3,3 V)
- POVEZAVA (MICRO USB KABEL)
- FLEX CABLE
- POVEZAVA (USB)
- POVEZAVA (USB)

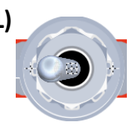
TURNIGY LIPOLY PACK
12 V => RDEČI KABEL
GND => ČRNI KABEL

PRENOSNA BATERIJA »OVAL«
Z RASPBERRY PI3 SE POVEŽE Z MIKRO USB KABLOM

KONČNO STIKALO
PREDUPOR (1 kΩ – 10 kΩ)



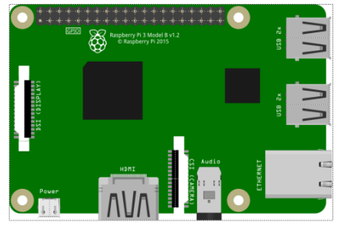
ZELENA/RDEČA LED-DIODA
PREDUPOR:
- 3,3 V => 65Ω
- 12 V => 500Ω



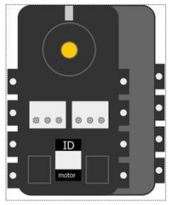
GLAVNO STIKALO

UPOR
220 Ω – 10 kΩ

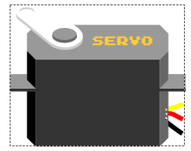
LEGENDA:



RASPBERRY PI 3
1X USB => LOGITECH
1X USB => USB2AX v3.2a
1X CSI => KAMERA
6X PIN => VHODI
4X PIN => IZHODI
1X PIN => GND

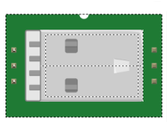


DYNAMIXEL AX-12a
12 V => SREDINJSKI PIN
GND => DESNI PIN
DATA => LEVI PIN
MOTORJI SE MED SABO POVEZUJEJO S POSEBNIMI KABLI, KI SO PRILOŽENI ZRAVEN MOTORJEV



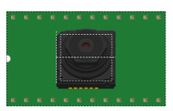
MAKEBLOCK MINI GRIPPER
5 V => RDEČI KABEL
GND => ČRNI KABEL
PMW (SIGNAL) => BELI KABEL

GOSTIME BRUSHLESS DC FAN
12 V => RDEČI KABEL
GND => ČRNI KABEL



USB2AX v3.2a
VSTAVIMO V ENEGA IZMED RASPBEERY PI 3 USB IZHODOV
POVEŽE SE Z MOTORJEM S DYNAMIXEL KABLOM

USB – LOGITECH GAMEPAD F710
VSTAVIMO V ENEGA IZMED RASPBEERY PI 3 USB IZHODOV



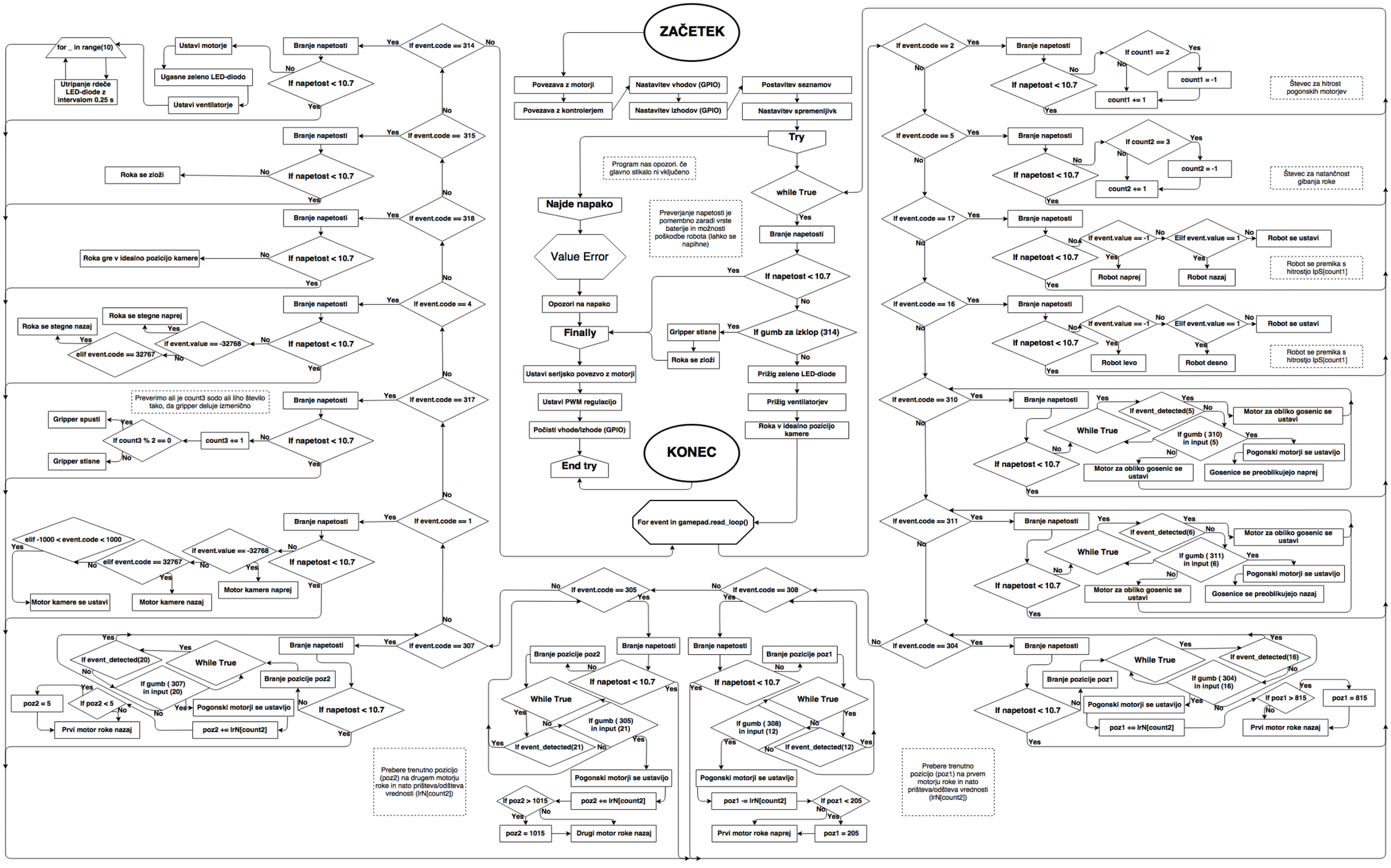
RASPBERRY CAMERA BOARD
POVEŽE SE S »FLEX CABLE«, KATERI JE PRILOŽEN ZRAVEN KAMERE

POLULU 12 V TO 5 V
12 V => VCC
GND => GND
5 V => OUT

TRANZISTOR
BAZA => GPIO13 Z PREDUPOROM 1 kΩ
EMITOR => GND
KOLEKTOR => GND IZ VENTILATORJA

Priloga 6: Diagram poteka programa

DIAGRAM POTEKA - HITRO PROTOTIPIRAN TERENSKI ROBOT



Priloga 7: Glavni program

```
# Uvoz modulov oziroma knjižnic.
from evdev import InputDevice
from pyax12.connection import Connection
import RPi.GPIO as GPIO
from time import sleep
from functions import *

# Postavimo serijsko povezavo z motorji Dynamixel preko USB2AX ključka.
sc = Connection(port="/dev/ttyACM0", baudrate=1000000)

# Nastavimo vhod s katerega beremo podatke daljinca.
gamepad = InputDevice('/dev/input/event0')

# Nastavimo tip številčenja vhodnih in izhodnih pinov.
GPIO.setmode(GPIO.BCM)

# Postavimo vhode.
GPIO.setup(5, GPIO.IN)                      # Končno stikalo za preoblikovanje gosenic 1.
GPIO.add_event_detect(5, GPIO.FALLING)      # Nastavimo "edge" na padec oziroma spremembo iz 1 na 0 na vhodu
                                            # in to ponovimo pri vseh stikalih.
GPIO.setup(6, GPIO.IN)                      # Končno stikalo za preoblikovanje gosenic 2.
GPIO.add_event_detect(6, GPIO.FALLING)
GPIO.setup(12, GPIO.IN)                     # Končno stikalo za prvi motor roke 1.
GPIO.add_event_detect(12, GPIO.FALLING)
GPIO.setup(16, GPIO.IN)                     # Končno stikalo za prvi motor roke 2.
GPIO.add_event_detect(16, GPIO.FALLING)
GPIO.setup(20, GPIO.IN)                     # Končno stikalo za drugi motor roke 1.
GPIO.add_event_detect(20, GPIO.FALLING)
GPIO.setup(21, GPIO.IN)                     # Končno stikalo za drugi motor roke 2.
GPIO.add_event_detect(21, GPIO.FALLING)

# Postavimo izhode.
GPIO.setup(15, GPIO.OUT, initial=0)         # Rdeča LED-dioda.
GPIO.setup(14, GPIO.OUT, initial=0)         # Zelena LED-dioda.
GPIO.setup(13, GPIO.OUT, initial=0)         # Ventilatorja.
GPIO.setup(19, GPIO.OUT)                    # Prijemalo.
p1 = GPIO.PWM(19, 50)                       # Na pinu za prijemalo omogočimo PWM regulacijo in določimo frekvenco 50.
p1.start(11.5)                              # RPi začne spuščati pulze, določimo začetno pozicijo servo prijemala.

# Postavimo sezname.
lpS = [256, 512, 1023]                      # Seznam hitrosti za pogonsko gibanje robota.
lrN = [11, 22, 33, 44]                      # Seznam natančnosti za gibanje roke.

# Postavimo spremenljivke.
count1 = 2                                  # Spremenljivka za števec za spreminjanje pogonske hitrosti.
count2 = 2                                  # Spremenljivka za števec za spreminjanje natančnosti roke.
count3 = 1                                  # Spremenljivka za izmenično delovanje prijemala.

# Program najprej poskusi glavno zanko:
try:
    while True:                             # Preberemo napetost na osmem motorju, ker je prvi motor v verigi
        volt = sc.get_present_voltage(8)    # in se tako izognemo še večjemu padcu napetosti.
        if volt < 10.7:                     # Preverimo, ali je napetost nad dovoljeno vrednostjo.
            print("POZOR - nizka napetost! Trenutna napetost baterije je", volt, "Program se bo ugasnil!")
            break                           # V primeru, da je, sprinta trenutno napetost in prekine program.
                                            # "print" funkcija se vseskozi program uporablja, tako da vemo, kaj robot
                                            # počne brez, da ga sami vidimo.
        if gamepad.active_keys(314):        # Preveri ali je aktivna tipka za izklop,
            print("Izklop programa!")       # v primeru, da je, se prijemalo stisne, roka se zloži v začetno
            p1.ChangeDutyCycle(11.5)        # pozicijo, program pa se prekine.
            sleep(0.1)
            zac_poz()
            break

        print("Vsi motorji so povezani.") # Program nam pove, kdaj je glavna zanka aktivna, napetost baterije,
        print("Trenutna napetost baterije je:", volt) # hitrost gibanja, natančnost gibanja roke in da se bo
        print("Robot se premika s hitrostjo", lpS[count1]) #  roka pomaknila v idealen položaj za kamero.
        print("Roka se premika z natančnostjo", lrN[count2])
        print("Roka v idealno pozicijo kamere!")

        GPIO.output(13, 1)                  # Prižgemo ventilatorja.
        GPIO.output(14, 1)                  # Prižgemo zeleno LED-diodo.
        kamera_poz()                        # Funkcija za idealno pozicijo kamere.

        # for zanka za vse vhodne dogodke, ki jih dobimo iz daljinca:
        for event in gamepad.read_loop():
            # Menjava hitrosti pogona. Count1 gre čez seznam za hitrosti, ki vsebuje 3 predmete (256, 512, 1023)
            # če je tipka LT in vrednost 255 (ker LT ni dejanska tipka, ampak potenciometer, nas zanima le,
            if event.code == 2 and event.value == 255: # ko je ta v maksimalni vrednosti).
                volt = sc.get_present_voltage(8) # Preberemo napetost na osmem motorju. Ta del se ponavlja pri vsakem
                if volt < 10.7:                  # dogodku. Če je napetost nižja od dovoljene, prekinemo for zanko.
                    break
                if count1 == 2:             # Preverimo, če je spremenljivka count1 enaka 2, če je, jo spremenimo na -1.
                    count1 = -1             # Na koncu spremenljivki count1 prištejemo 1, tako smo naredili zanko med
                count1 += 1                 # števili 0, 1 in 2.  (npr. če je count1 enak 2 je hitrost gibanja 1023).
                print("Sprememba hitrosti! Trenutna hitrost je:", lpS[count1]) # Napovemo spremembo hitrosti.

            # Menjava natančnosti gibanja roke. Count2 gre čez seznam za hitrosti, ki vsebuje 4 predmete (11, 22, 33, 44).
            if event.code == 5 and event.value == 255: # Ta proces je enak prejšnemu, razen da je tokrat tipka RT in
                volt = sc.get_present_voltage(8)       # maksimalno število count2 enako 3.
                if volt < 10.7:
                    break
                if count2 == 3:
                    count2 = -1
                count2 += 1
                print("Sprememba natančnosti! Trenutna natančnost je:", lrN[count2]) # Napovemo spremembo natančnosti.
```

```
# Premik robota naprej/nazaj.
if event.code == 17:          # Če je D-PAD gor/dol, vrednosti -1, 0, 1.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    if event.value == -1:     # Če je vrednost -1(gor), se robot premika naprej s hitrostjo odvisno od count1.
        print("Robot se premika naprej!")
        naprej(lpS[count1])
    elif event.value == 1:    # Če je vrednost 1(dol), se robot premika nazaj s hitrostjo odvisno od count1.
        print("Robot se premika nazaj!")
        nazaj(lpS[count1])
    else:                     # Če vrednost ni -1 ali 1, se robot ustavi.
        stop_pogon()

# Premik robota levo/desno.
if event.code == 16:          # Če je D-PAD levo/desno, vrednosti -1, 0, 1.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    if event.value == -1:     # Če je vrednost -1(levo), se robot premika levo s hitrostjo odvisno count1.
        print("Robot se premika v levo (na mestu)")
        levo(lpS[count1])
    elif event.value == 1:    # Če je vrednost 1(desno), se robot premika desno s hitrostjo odvisno od count1.
        print("Robot se premika v desno (na mestu)")
        desno(lpS[count1])
    else:                     # Če vrednost ni -1 ali 1, se robot ustavi.
        stop_pogon()

# Preblikovanje gosenic naprej.
if event.code == 310:         # Če je zaznana tipka LB.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    while True:               # Postavimo neskončno zanko, kjer konstantno gledamo stanje končnega stikala.
        if GPIO.event_detected(5): # Če je zaznana sprememba končnega stikala, se motor za preoblikovanje
            print("STOP - Končno stikalo (naprej)") # gosenic ustavi in neskončna zanka se prekine.
            gosenice(0)
            break
        if gamepad.active_keys(310) and GPIO.input(5): # Če je tipka LB aktivna (vrednost 1) in končno stikalo
            stop_pogon()          # (vezano na pin 5) ni stisnjeno, najprej ustavimo pogonske motorje, nato
            print("Gosenice se preoblikujejo naprej!") # pa se gosenice preoblikujejo naprej.
            gosenice(2047)
        else:                 # V primeru, da tipka LB ni več aktivna ali pa je končno stikalo stisnjeno, se
            gosenice(0)       # motor za preoblikovanje gosenic ustavi in neskončna zanka se prekine.
            break
# Preblikovanje gosenic nazaj.
if event.code == 311:         # Če je zaznana tipka RB.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    while True:               # Proces je enak kot pri preoblikovanju naprej, le da gledamo aktivnost druge
        if GPIO.event_detected(6): # tipke (RB), drugega končnega stikala (vezanega na pin 6) in da se
            print("STOP - Končno stikalo (nazaj)") # motor za preoblikovanje tokrat vrti v drugo smer.
            gosenice(0)
            break
        if gamepad.active_keys(311) and GPIO.input(6):
            stop_pogon()
            print("Gosenice se preoblikujejo nazaj!")
            gosenice(1023)
        else:
            gosenice(0)
            break

# Prvi motor roke nazaj.
if event.code == 304:         # Če je zaznana tipka A.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    poz1 = sc.get_present_position(6) # Preberemo trenutno pozicijo prvega motorja roke
    while True:               # Postavimo neskončno zanko, kjer konstantno gledamo stanje končnega stikala.
        if GPIO.event_detected(16): # Če je zaznana sprememba končnega stikala, se prvi motor roke ustavi
            print("STOP - Končno stikalo (M1 - nazaj)") # in neskončna zanka se prekine.
            break
        if gamepad.active_keys(304) and GPIO.input(16): # Če je tipka A aktivna (vrednost 1) in končno stikalo
            stop_pogon()      # (vezano na pin 16) ni stisnjeno, najprej ustavimo pogonske motorje, nato pa
            print("Prvi motor roke nazaj!") # trenutni poziciji prištejemo vrednost glede na count2.
            poz1 += lrN[count2]
            if poz1 > 815:    # Novo pozicijo omejimo na 815, ker se roka nima več kam premakniti, kar deluje
                poz1 = 815    # kot dvojno varovalo (poleg končnega stikala).
            rokam1(poz1)      # Prvemu motorju za tem navedemo pozicijo, v katero naj se premakne (roka gre nazaj).
        else:                 # V primeru, da tipka A ni več aktivna ali pa je končno stikalo stisnjeno, prvemu
            break             # motorju roke nehamo prištevati vrednosti in neskončna zanka se prekine.
# Prvi motor roke naprej.
if event.code == 308:         # Če je zaznana tipka Y.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    poz1 = sc.get_present_position(6)
    while True:               # Proces je enak kot pri delu za premik roke nazaj, le da gledamo aktivnost druge
        if GPIO.event_detected(12): # tipke (Y), drugega končnega stikala (vezanega na pin 12), da tokrat vrednost
            print("STOP - Končno stikalo (M1 - naprej)") # glede na count2 odštevamo (tako, da gre roka naprej).
            break             # Novo pozicijo tokrat omejimo na 205 oziroma drugo stran robota.
        if gamepad.active_keys(308) and GPIO.input(12):
            stop_pogon()
            print("Prvi motor roke naprej!")
            poz1 -= lrN[count2]
            if poz1 < 205:
                poz1 = 205
            rokam1(poz1)
        else:
            break
```

```
# Drugi motor roke nazaj.
if event.code == 305:          # Če je zaznana tipka B.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    poz2 = sc.get_present_position(7) # Paziti moramo, da pri drugem motorju trenuten položaj shranimo pod drugo
    while True:                # spremenljivko. Proces je sicer enak kot pri prvem motorju roke, le da gledamo
        if GPIO.event_detected(21): # aktivnost druge tipke (B), drugega končnega stikala (vezanega na pin 21),
            print("STOP - Končno stikalo (M2 - nazaj)") # da tokrat vrednost glede na count2 spet prištevamo
            break              # (tako, da gre roka nazaj). Tokrat novo pozicijo omejimo na čim večjo vrednost
        if gamepad.active_keys(305) and GPIO.input(21): # (1015), saj drug motor roke ne potrebuje omejitve.
            stop_pogon()       # V primeru, da motorju podamo pozicijo izven dovoljene vrednosti (0-300 stopinj
            print("Drugi motor roke nazaj!") # oziroma 0-1023) nam ta javi napako.
            poz2 += lrN[count2]
            if poz2 > 1015:
                poz2 = 1015
            rokam2(poz2)
        else:
            break
# Drugi motor roke naprej.
if event.code == 307:          # Če je zaznana tipka X.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    poz2 = sc.get_present_position(7)
    while True:                # Proces je enak kot pri delu za premik roke nazaj, le da gledamo aktivnost druge
        if GPIO.event_detected(20): # tipke (X), drugega končnega stikala (vezanega na pin 20), da tokrat vrednost
            print("STOP - Končno stikalo (M2 - naprej)") # glede na count2 odštevamo (tako, da gre roka naprej)
            break              # in zato tudi novo pozicijo omejimo na čim manjšo vrednost(5).
        if gamepad.active_keys(307) and GPIO.input(20):
            stop_pogon()
            print("Drugi motor roke naprej!")
            poz2 -= lrN[count2]
            if poz2 < 5:
                poz2 = 5
            rokam2(poz2)
        else:
            break

# Tretji motor roke naprej/nazaj.
if event.code == 1:            # Če je leva analogna palčka po y osi.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    if event.value == -32768: # Če je a. palčka potisnjena do konca naprej (ima potenciometer, s skrajnima
        print("Motor kamere in prijemala naprej!") # vrednostima med -32768 in 32767), torej nas zanima, ko je
        rokam3(256)            # vrednost -32768, da gre tretji motor roke (skrbi za obračanje kamere in prijemala)
    elif event.value == 32767: # naprej. Če je a. palčka potisnjena do konca nazaj, torej nas zanima, ko je
        print("Motor kamere in prijemala nazaj!") # vrednost 32767, gre tretji motor roke nazaj.
        rokam3(256 + 1024)
    elif -1000 < event.value < 1000: # Za ustavitev motorja moramo podati nek razpon vrednosti, saj včasih
        rokam3(0)              # določenih vmesnih vrednostih ne zazna. Izbrali smo vse vrednosti med -1000 in 1000,
                               # tako da je ustvitev motorja dovolj odzivna.
# Izmenično stiskanje/spuščanje prijemala.
if event.code == 317 and event.value == 1: # Če je leva analogna palčka (kot tipka) in vrednost 1.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    count3 += 1                # Spremenljivki count3 prištejemo 1, nato pa gledamo, če je count3 liho ali sodo. To
    if count3 % 2 == 0:        # storimo, tako, da spremenljivko delimo z dva in pogledamo ostanek. Če je ostanek 0,
        print("Prijemalo spusti!") # je število sodo  in prijemalo spusti, če pa je ostanek 1, je število liho in
        p1.ChangeDutyCycle(3.5) # prijemalo stisne (3.5 je pozicija servo motorja za spust, 11.5 pa za stisk).
        sleep(0.1)
    else:
        print("Prijemalo stisne!")
        p1.ChangeDutyCycle(11.5)
        sleep(0.1)

# Steg roke naprej/nazaj.
if event.code == 4:            # Če je zaznana desna analogna palčka.
    volt = sc.get_present_voltage(8)
    if volt < 10.7:
        break
    if event.value == -32768: # Podobno kot pri levi analogni palčki nas zanima, le ko je palčka v skrajnih
        print("Roka se stegne naprej!") # poziijah, vendar tokrat imamo za pozicijo palčke naprej funkcijo, ki
        poz_roke_naprej()      # stegne roko naprej in obratno za nazaj. Ta položaj roke je koristen, ko želimo
    elif event.value == 32767: # prevržti težišče na prvo ali zadnjo stran robota ali pa želimo pred robotom
        print("Roka se stegne nazaj!") # kaj pobrati.
        poz_roke_nazaj()

# Pomik roke v idealno pozicijo kamere.
if event.code == 318 and event.value == 1: # Če je zaznana desna analogna palčka (kot tipka) in vrednost 1 se
    volt = sc.get_present_voltage(8) # roka postavi v idealno pozicijo kamere.
    if volt < 10.7:
        break
    print("Roka v idealno pozicijo kamere!")
    kamera_poz()

# Zloženje roke v začetno pozicijo.
if event.code == 315 and event.value == 1: # Če je zaznana tipka START in vrednost 1, se roka zloži v začetno
    volt = sc.get_present_voltage(8) # pozicijo.
    if volt < 10.7:
        break
    print("Roka se zloži v začetno pozicijo!")
    zac_poz()
```

```
            # STOP v sili
            if event.code == 314 and event.value == 1: # Če je tipka BACK in vrednost 1, se požene funkcija za izklop v sili,
                volt = sc.get_present_voltage(8) # kar pomeni, da se ustavijo vsi motorji, roka obstane, kjer je bila,
                if volt < 10.7:          # ventilatorji se ustavijo in rdeča LED-dioda utripa za pet sekund in s tem signalizira
                    break                # reset programa (ko se for zanka prekine, gre prejšna while zanka od začetka).
                print("STOP - Motorji se ustavijo!") # Če držimo BACK tipko po resetu, se bo program ustavil.
                gobica()
                GPIO.output(13, 0)
                GPIO.output(14, 0)
                for _ in range(10):      # For zanka za utripane rdeče LED-diode. Programu povemo, da desetkrat ponovi ugašanje
                    GPIO.output(15, 1)   # in prižiganje diode z intervalom 0.25 sekund.
                    sleep(0.25)
                    GPIO.output(15, 0)
                    sleep(0.25)
                break

# V primeru, da zazna napako ValueError (ki se pojavi v primeru, da komunikacija z motorji ne deluje oziroma
# glavno stikalo ni vključeno), nam program pove, zakaj je do te napake najverjetneje prišlo.
except ValueError:
    print("Glavno stikalo ni vključeno ali motorji nimajo stika!")

# "Finally" se vedno izvede, če pride do napake v "try" zanki ali pa zanko sami zaključimo. Takrat se ustavi serijska
# povezava z motorji in PWM regulacija (RPi neha pošiljati pulze). Čisto na koncu pa še očistimo GPIO pine (resetiramo
# vhode in izhode).
finally:
    sc.close()
    p1.stop()
    GPIO.cleanup()
```