TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

LAOROBOTI PROTOTÜÜP ROS BAASIL

Protype of a warehouse robot based on ROS

LÕPUTÖÖ

Üliõpilane: Vladislav Brusnitsin
Üliõpilaskood: 212577EDTR
Juhendaja: Sergei Pavlov

Kohtla-Järve, 2024

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS

Mina Vladislav Brusnitsin (sünnikuupäev: 13.04.2024)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Laoroboti prototüüp ROS baasil, mille juhendaja on Sergei Pavlov,

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

SISUKORD

EESSÕNA

Lõputöö teema põhineb Dmitri Kozlovi poolt Baltic Bolti ettevõttele välja pakutud tööstusliku laoroboti arendamise projektil. Sergei Pavlov aitas konsultatsiooniga. Samuti aitasid töös masinaehitus- ja energiatehnoloogia protsesside juhtimine teaduskonda 3. kursuse õpilased.

SISSEJUHATUS

Autor töötab oma lõputöö raames välja laoroboti prototüübi. Üliõpilane osaleb täismahus laoroboti väljatöötamise projektis. Mehhanismide ja programmide testimiseks luuakse põhifunktsionaalsusega prototüüp.

Automatiseerimise ja optimeerimise ajastul leidub roboteid üha enam erinevates ettevõtetes erinevate ülesannete jaoks: värvimine, kokkupanek, teisaldamine jne. Arendatav robot on mõeldud konkreetse lao ülesannete täitmiseks, et liigutada erinevatelt riiulitelt kuni 50 kg kaaluvaid kaste ning vabastada töötajad rutiinsest tööst.

Praegu on saadaval erinevaid tööstuslikke laoroboteid, kuid kõik need on mõeldud kindlale laotüübile ehk selle kasutamiseks tuleb laod täielikult ümber kujundada. Meie robot, vastupidi, on väljatöötamisel konkreetse lao jaoks, kuid soovi korral saab seda kasutada ka teistes ladudes. Teine suur probleem selliste robotite rakendamisel on nende hind. Kõik sõltub konkreetsest robotist ja ülesandest, kuid igal juhul ei saa kõik ettevõtted seda endale lubada. Meie eesmärk on ka teha toode mõistliku hinnaga ka väikeladudele.

Autor tegeleb üksi roboti prototüübi väljatöötamise ja silumisega. Prototüüp on vineerist raam mõõtmetega umbes 50 cm x 30 cm See raam on varustatud 2 mootoriga liikumiseks, 2 mootoriga platvormi tõstmiseks ja mehhanismiga kastide riiulitelt tõstmiseks. Seda kõike juhitakse juhtseadmena Raspberry pi 4 ja kontrollerina Arduino Mega abil.

Roboti juhtimistarkvara on kokku pandud ROS (Robot Operating System) baasil ja kasutab C++ kood.

Töö eesmärk on kokku panna täisfunktsionaalne roboti prototüüp, mis suudab käske vastu võtta ja neid täita. Näiteks tuua teatud riiulilt kast. Robot peab saama käsu, sõitma kindlasse kohta, võtma kasti, asetama selle oma platvormile ja jõudma alguspunkti.

1 LAOROBOTI EELISED

Laorobotit on vaja monotoonse töö tegemiseks kaupade, kastide jms tõstmisel, ülekandmisel ja ladustamisel. Selline robot aitab parandada lao efektiivsust, samuti varustab inimest ohtlikust ja tervist mõjutavast tööst. Roboti tüüp ja funktsionaalsus sõltuvad lao tüübist, kaupade tüübist ja tööülesannetest. Näiteks Amazon on suurepärane näide lao optimeerimisest. Nad kasutavad roboteid, mis sõidavad kaubaga riiuli alla, tõstavad seda veidi üles ja transpordivad vastavasse kohta. Seega nende ladudes kuid kasutatakse inimtööd.

Meie projektis kasutame teistsugust lähenemist. Meil on ladu, mida me ei saa ümber ehitada. Selles laos on kuni 3,5 meetri suurused riiulid, millel on kuni 50 kg kaaluvad plastmahutid. Robot peab pärast serverilt käsu saamist jõudma soovitud riiulile, haarava mehhanismi abil võtma konteineri, panema selle oma platvormile ja viima õigesse kohta. Seega on robot ratastel platvorm ja sellel platvormil on tõstemehhanism vähemalt 2,5 meetrit kõrge. Kuna töö on suur ja vastutustundlik, tuleb kõik punktid eelnevalt läbi mõelda, just selleks luuakse prototüüp, mis kujutab endast roboti väiksemat versiooni.

1.1 Kirjanduse ülevaade

Viimastel aastatel on aktiivselt tõusnud robotisüsteemide kasutuselevõtu trend ladudes ja tööstusharudes. Üha enam ettevõtteid mõistab antud optimeerimise olulisust ja kasulikkust. Allpool uurib autor erinevate uuringute ja artiklite põhjal automatiseerimise erinevaid aspekte. [5]

Tootlikkus

Järjepidevuse ja hea korratavuse tõttu suurendavad tööd oluliselt tootlikkust. Inimesed vajavad väsimuse tõttu pause, tähelepanematuse tõttu on suur ka vigade oht, samas kui robotitel puuduvad need miinused. Näiteks töö (AMR) võib kaupa ladu ümber liigutada, vabastades töötajad monotoonsest tööst.[1]

Kiva robotite abil suutis Amazon vähendada kauba kogumise aega 40% võrra, mis võimaldas säilitada ladustamiskulud. [4]

Tegevuskulude vähendamine

Robot saab pidevalt mingit tööd teha, see võimaldab säästa töötajate palkadelt. See minimeerib ka vea või kauba kahjustamise võimaluse, millel on positiivne mõju ka kokkuhoiule. Samuti aitavad tööd kaasa ladustamisruumide optimeerimisele. See väljendub nii marsruutide optimeerimises, see tähendab liikumises laos, kui ka ruumi suurendamise vajaduse puudumisel on võimalik ladu ehitada vertikaalselt, kasutades seeläbi laoruumi maksimaalselt ära. [2]

Paindlikkus ja kohanemisvõime

Kaasaegsete robotisüsteemide suur eelis on suur kohanemisvõime. Kui nõudlus muutub või lao ümberehitamine, saab roboteid hõlpsasti ümber programmeerida ilma struktuuri muutmata. [1]

Ökoloogiline jätkusuutlikkus

Optimaalse marsruudi valimine ja läbisõidu minimeerimine avaldab positiivset mõju süsiniku jalajälje vähendamisele. Samuti kuna kõik toimingud on digitaalsed, aitab see oluliselt vähendada paberi tarbimist. [1]

Probleemid ja piirangud

Vaatamata ilmsetele eelistele ja mugavustele on neid mitmeid tegurid mis takistavad üldlevinud robotiseerimist [3]:

- Suured esialgsed kulud: robotisüsteemide ostmise ja paigaldamise kulud võivad olla märkimisväärsed, muutes selle sammu väikestele ja keskmise suurusega ettevõtetele kättesaamatuks [3].
- Hooldusvajadus: tööd vajavad regulaarset hooldust ja moderniseerimist, mis toob kaasa lisakulusid [3].
- Personali koolitus: töötajaid tuleb koolitada uue tehnikaga suhtlemiseks, mis nõuab ka aega ja rahalisi ressursse [3].
- Küberturvalisus: automatiseeritud süsteemide kasutamine suurendab küberrünnakute riski, mis nõuab täiendavaid investeeringuid andmekaitsesse [3].

Arenguväljavaated

Tänu tehisintellektile ja IoT-le on robotiseerimine ainult hoogustumas ja näitab aktiivset kasvu erinevates valdkondades. Research and Markets-i uuringu kohaselt võib ülemaailmne laorobootika turg 2030.aastaks ulatuda 15 miljardi dollarini, keskmine aastane kasvumäär on umbes 15%. [6]

1.2 *ROS* platform

Robot Operating System (*ROS*) on avatud tarkvaraplatvorm robotsüsteemide loomiseks, mis on saanud standardiks tööstuses ja teadustöös. See pakub tööriistu, teeke ja arhitektuuri robotitarkvara arendamiseks.

ROS paistab silma populaarse lahendusena naasturobotite väljatöötamiseks oma võimalustele ja mugavustele.

Näiteks, võimaldab *ROS* luua keerulisi süsteeme, jagades need mooduliteks, mida saab sõltumatult välja töötada ja testida. See muudab platvormi skaleeritavaks ja mugavaks mistahes raskusastmega projektidega tegelemiseks.

Niisamuti pakub *ROS* ulatuslikku tööriistaraamatukogu robotitega tegelemiseks: alates liiklusjuhtimisest kuni piltide töötlemise ja trajektooride planeerimiseni. Mõned neist: ***RViz***: robotite andmete visualiseerimise tööriist.

Gazebo: simulaator robotite testimiseks virtuaalses keskkonnas.

MoveIt!: manipuleerijate liikumise planeerimise platvorm

Veel üks suur eelis on *Pythoni* ja *C++* tugi. See võimaldab erineva kodeerimiskogemusega inimestel antud platvormi kasutada.

Kogukond ja avatud lähtekood: *ROS* arendajate suur kogukond jagab aktiivselt koodi, dokumentatsiooni ja näiteid, mis kiirendab uute lahenduste väljatöötamist ja juurutamist.

Nende eeliste tõttu on *ROS* laialdaselt levinud erinevates valdkondades.

ROS-i kasutatakse droonide ja mobiilsete sõidukite arendamiseks. Põllumajanduses ehitatakse selle baasil automatiseeritud kombainid. Meditsiinis sobib see suurepäraselt täpsete kirurgiliste jaamade jaoks, mis võimaldab teil teha erineva keerukusega operatsioone. Ka *ROS*-i haridusest pole mööda läinud, tänu juba valmis tööriistade komplektile ja suurele funktsionaalsusele saavad õpilased keskenduda algoritmidele ja õppeprotsessile. [7]

1.2.1 *ROS* arhitektuur

Kõik ülaltoodud on võimalik tänu sellele, kuidas *ROS* andmed töötavad ja milliseid põhimõtteid ta kasutab, analüüsib autor seda allpool üksikasjalikumalt.

Kõigepealt peate mõistma *ROS*-failisüsteemi. Süsteem koosneb:

- **Paketid**: paketid on *ROS*-i tarkvara korraldamise peamine üksus. Pakett võib sisaldada *ROS*-i käitusprotsesse (sõlme), *ROS*-ist sõltuvat teeki, andmekogumeid, konfiguratsioonifaile või midagi muud, mis on koos kasulikult korraldatud. Paketid on kõige aatomi ehitada kirje ja vabastada kirje *ROS*. See

tähendab, et kõige teralisem asi, mida saate ehitada ja vabastada, on pakett. [8]

- **Metapaketid**: Metapaketid on spetsiaalsed paketid, mis esindavad ainult seotud muude pakettide rühma. [8]
- **Pakendi manifestid**: manifestid (pakett.xml) esitage paketi metaandmed, sealhulgas selle nimi, versioon, kirjeldus, litsentsiteave, sõltuvused ja muu metateave, näiteks eksporditud paketid. [8]
- **Hoidlad**: pakettide kogum, millel on ühine *VCS* süsteem. Paketid, mis jagavad *VCS*-i, jagavad sama versiooni ja neid saab koos välja anda catkini väljalaske automatiseerimise tööriista bloom abil. Sageli kaardistavad need hoidlad teisendatud rosbuildi Korstnad. Hoidlad võivad sisaldada ka ainult ühte paketti. [8]
- **Sõnumi (msg) tüübid**: sõnumi kirjeldused, salvestatud *my_package/msg/MyMessageType.msg*, määrake ROS-is saadetud sõnumite andmestruktuurid. [8]
- **Service (srv) tüübid**: service kirjeldused, salvestatud *my_package/srv/MyServiceType.srv*, määrake *ROS*-i teenuste päringu ja vastuse andmestruktuurid. [8]

Meie jaoks on peamised paketid ja sõnumid. Just nendega peame kõige rohkem töötama.

ROS-i sees suhtlemiseks kasutatakse Computation Graph-süsteemi.

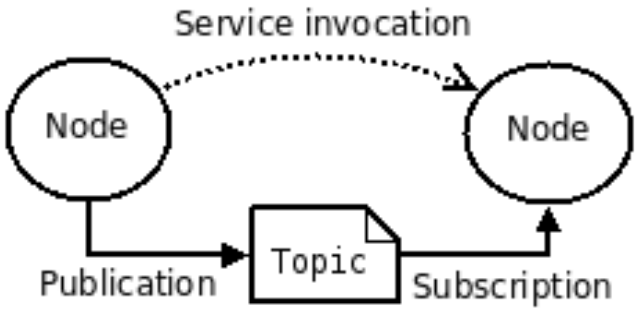
Computation Graph on peer-to-peer võrk *ROS* protsessid, mis töötlevad andmeid koos. *ROS* node, Kapteni, Parameetriserveri, sõnumite, teenuste, teemade ja kottide põhilised Arvutusgraafiku kontseptsioonid, mis kõik annavad graafikule andmeid erineval viisil.

- **Nodes**: Nodesa on protsessid, mis teostavad arvutamist. *ROS* on kavandatud olema modulaarne peeneteralises mõõtkavas; roboti juhtimissüsteem koosneb tavaliselt paljudest sõlmedest. Näiteks kontrollib üks node laseri vahemiku leidjat, üks node kontrollib rattamootoreid, üks node teostab lokaliseerimist, üks node teostab tee planeerimist, üks node pakub süsteemi graafilist vaadet jne. *ROS*-sõlm on kirjutatud *ROS*-i klienditeegi abil, näiteks roscpp või rospy. [8]
- **Master**: *ROS* Master annab nime registreerimise ja lookup ülejäänud arvutus graafik. Ilma kaptenita ei saaks sõlmed üksteist leida, sõnumeid vahetada ega teenuseid kasutada. [8]
- **Sõnumid**: Nodes suhtlevad omavahel sõnumeid edastades. Sõnum on lihtsalt andmestruktuur, mis koosneb sisestatud väljadest. Standardsed primitiivsed

tüübid (täisarv, ujukoma, tõeväärtus jne.) on toetatud, nagu ka primitiivsete tüüpide massiivid.[8]

- **Topics**: sõnumid suunatakse transpordisüsteemi kaudu, millel on avaldamise / tellimise semantika. Node saadab sõnumi, avaldades selle antud teemale. Teema on nimi, mida kasutatakse sõnumi sisu tuvastamiseks. Sõlm, mis on huvitatud teatud tüüpi andmetest, tellib sobiva teema. [8]

Joonisel 1.1 on näidatud peamine on mõista, et on olemas koodiplokke (Nodes), milles toimub mingisugune arvutamine või andmete kogumine ja tänu teemadele saavad sõlmed teavet vahetada. [8]



Joonis 1.1 ROS basic concept [8]

2 PROTOTÜÜBI EHITAMINE

Prototüübi ehitamine on arenduse oluline osa. Selles etapis muutub toode teooriast füüsiliseks vormiks. Siin töötatakse välja peamised ideed ja põhimõtted, millele lõpptoode ehitatakse. On väga oluline selgelt seada ülesanded ja funktsioonid, mida prototüüp peab täitma, ja lõpuks analüüsida, kui palju konkreetne lahendus neid täidab.

Selle töö jaoks on autor määranud järgmised roboti tööpõhimõtted:

- Robot peab olema võimeline manööverdamiseks rattaid eraldi juhtima.
- Robot peab olema võimeline juhtima samm-mootoreid platvormi tõstmiseks.
- Roboti juhtimine peab toimuma kaugjuhtimisega.
- Robot peab takistuse ilmnemisel peatuma.
- Roboti disain peab olema piisavalt tugev ja lootusrikas erinevate testide rakendamiseks.
- Töö peaks olema võimeline käsu peale riiulile sõitma, kasti võtma ja sellega tagasi tulema.

Seatud põhimõtete täitmiseks kasutas autor teatud komponente ja programme, mida kirjeldatakse allpool.

2.1 Tehnilised andmed

Prototüüp koosneb järgmistest osadest:

1. Liitium -aku 24V (vt Joonis 2.1)
2. DC mootor rattaga 2tk
3. Samm-mootor Nema17 2tk (vt Joonis 2.2)
4. TB6600 samm-mootor driver 2tk (vt Joonis 2.3)
5. HW-039 DC mootor driver 2tk (vt Joonis 2.5)
6. Rassbery Pi 4 (vt Joonis 2.6)
7. Arduino Uno (vt Joonis 2.7)
8. Toitemoodul DC/DC step-down 5-40V/1.2-35V 10A (vt Joonis 2.8)
9. Alumiiniumprofiilist
10. Laadimismoodul Li-Po/Li-Ion 5VDC 1A USB B micro TP4056
11. Ultraheli andur

Kõik süsteemid algavad toiteelemendist, see on kõige olulisem aspekt ja tagab töö stabiilsuse. 24V aku valiti võimsuse ja hinna/kvaliteedi kombinatsiooni jaoks.

Joonis 2.1 Lithium-ion battery [15]

Mootorid on tavalised, ilma enkodeerijateta. Praegu piisab sellest, kuid tulevikus paigaldatakse tagasisidega mootorid.

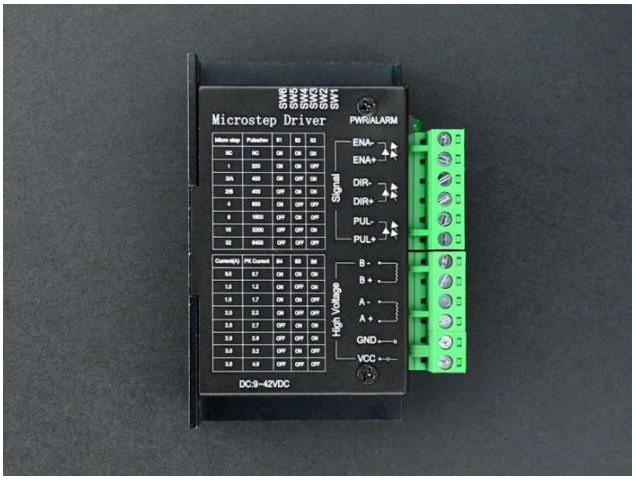
Tõsteplatvormi toomiseks kasutatakse T8 keermega tihvtiga samm-mootoreid. *Nema17* samm-mootoreid kasutatakse sageli 3d-printerite jaoks ja need on väga levinud, mis tagab nende taskukohasuse ja hea hinna. Prototüübi jaoks piisab neist, kuna põhimõtteliselt võimsamaid mootoreid juhitakse samamoodi.

Joonis 2.2 Nema 17 [18]

Kõigi mootorite juhtimiseks on vaja draivereid. Draiver (inglise k. driver – juht, antud kontekstis juhtlülitus, saksa k. Treiber) on elektroonikalülitus mingi seadme elektriahela tüürimiseks.

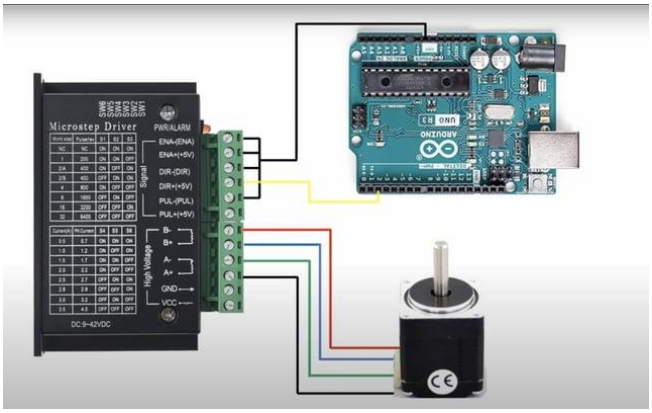
Tavaliste samm-mootorite juhtimiseks kasutatakse erinevaid draivereid, kaalume üksikasjalikumalt.

TB6600 - mootori draiver on hõlpsasti kasutatav professionaalne samm-mootori draiver, mis suudab juhtida kahefaasilist samm-mootorit. See ühildub Arduino ja teiste mikrokontrolleritega, mis suudavad väljastada 5 V digitaalse impulsi signaali. TB6600 samm-mootori draiveril on lai sisendvõimsus vahemikus 9 kuni 42 VDC. See on võimeline väljastama 4a tippvoolu, mis on enamiku samm-mootorite jaoks piisav.



Joonis 2.3 TB6600 [16]

Draiveril (vt Joonis 2.4) on ühendamiseks 12 terminali. VCC ja GND on peamised toiteühendused. A+ A -, B+ B-samm-mootori enda mähiste ühendamine. PUL+ PUL- signaalijuhtmete ühendamiseks, mis kontrollivad mähistele tarnitavate impulsside arvu. DIR+ DIR on ka signaalkontaktid, mis kontrollivad liikumissuunda. ENA+ ENA- saab draiveri lubada või keelata . Seal on palju erinevaid draiveri ühendusi. Sel juhul kasutas autor järgmist:

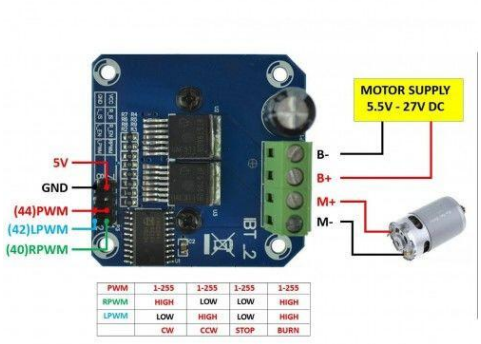


Joonis 2.4 TB6600 ühendus [17]

Selle ühenduse korral on *draiver* pidevalt sisse lülitatud ja seda juhitakse, muutes suunda ja impulsside arvu. Samuti võimaldab see draiver reguleerida mikrosammude arvu ja voolu. Autor pani 1600 impulssi 360-kraadise pöörde ja 2,0 ampri jaoks. Need

seadistused on prototüübi jaoks optimaalsed.

HW-039 (BTS7960) - on täielikult integreeritud suure vooluga H-silla moodul mootoriajami rakenduste jaoks. Liidestaminemikrokontrolleri teeb lihtsaks integreeritud draiveri IC, millel on loogikataseme sisendid, diagnoosimine praegune meel, pöördekiiruse reguleerimine, surnud aja genereerimine ja kaitse ületemperatuuri eest, ülepinge, alapinge, ülevool ja lühis. BTS7960 pakub kuludele optimeeritud lahendust kaitstud suure vooluga PWM-mootoriajamite jaoks, millel on väga väike plaadiruumi tarbimine. [10]



Joonis 2.5 HW-039 [10]

Siin kasutatakse klemme peamise toite B+ B ühendamiseks-ja mootori ühendamiseks.

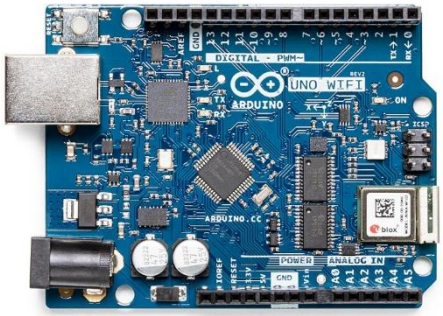
Samuti on 8 signaalkontakti. +5V, GND, RPWM, LPWM, ENR, ENL, ISR, ISL. Draiveri juhtimiseks peate ühendama toite ja RPWM, LPWM. Ülejäänud kontaktid saab sulgeda plussjuhtmega.

Raspberry Pi 4 on väga odav arvuti, mis töötab *Linuxiga*, kuid pakub ka GPIO (üldotstarbeline sisend/väljund) komplekti, mis võimaldab teil juhtida füüsilise andmetöötluse elektroonilisi komponente ja uurida asjade Internetti (IoT). [11]

Raspberry täidab roboti juhtimisblokifunktsiooni, see tähendab, et kogu vajalik suhtlus ja arvutused toimuvad sellel. See on Raspberry, kes käivitab koodi ja annab Arduinos käske.

Joonis 2.6 Rasberry Pi 4 [11]

Arduino Uno - kujundab, toodab ja toetab elektroonilisi seadmeid ja tarkvara, võimaldades inimestel kogu maailmas hõlpsasti juurde pääseda arenenud tehnoloogiatele, mis suhtlevad füüsilise maailmaga. Arduino on lihtsad ja võimsad, valmis rahuldama kasutajate vajadusi õpilastest tegijateni ja professionaalsete arendajateni. [12]



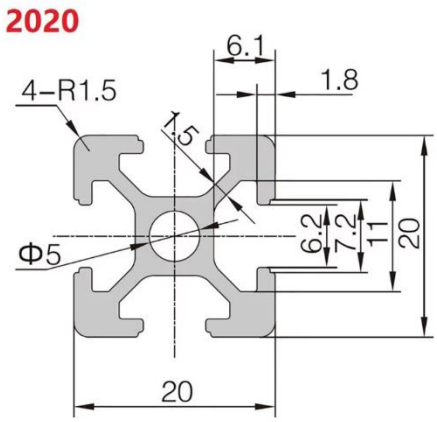
Joonis 2.7 Arduino Uno [12]

Arduino toimib kontrollerina, just sellega on ühendatud kõik mootorid.

Toitemoodul on vajalik pinge vähendamiseks. Kuna nii Arduino kui ka ülejäänud komponendid töötavad madalama pingega, umbes 5-6V. Vastasel juhul põleb kõik lihtsalt läbi.

Joonis 2.8 Toitemoodul [19]

Alumiiniumprofiilist (vt Joonis 2.9) on vajalikud rullide raami ehitamiseks. Valiti profiilid 20x20 mm.

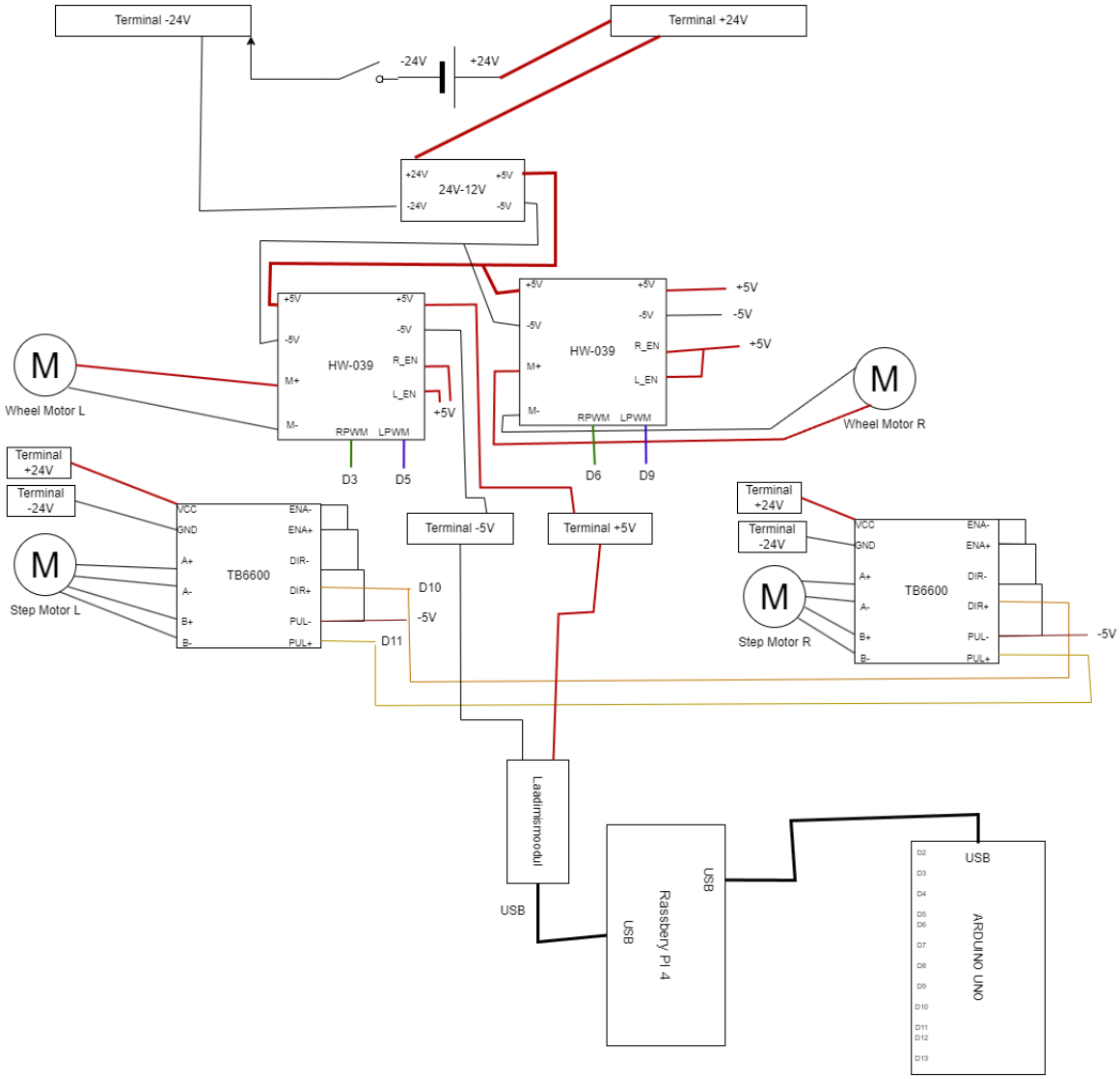


Joonis 2.9 Alumiiniumprofiil [20]

2.2 Ühendamine skeemi koostamine

Järgmine samm pärast komponentide valimist on vooluahela kokkupanek. On vaja ühendada vajalikud tihvtid, jootma juhtmed, jaotada toide õigesti ja proovida mitte midagi põletada.

Joonisel 2.10 on näidatud kõik ühendused. Vooluahel oli paigutatud 2 ahelasse: 24V ja 5V. Samuti kasutatakse Raspberry toitmiseks USB-d, kuna see toiteviis on kõige ohutum.



Joonis 2.10 Ühenduse skeem

Diagrammil näeme 24 V akut. sellest lähevad juhtmed muundurisse ja muundatakse 5 V-ks. siis tulevad need 5 V terminalidesse ja erinevad juba komponentide poolest. Esiteks läheb toide Mootorite draiverile. See pinge on nende mootorite jaoks piisav. Samuti toidab 5 V draiveri loogilist elementi. Seejärel tulevad draiverilt RPM ja RPM signaalijuhtmed, nende kaudu edastatakse Arduino PWM-signaal. Ülejäänud ühendused on suletud +5V-ga, kuna neid pole kontrollerist juhtimiseks vaja. 5 V kasutatakse ka Raspberry ja Arudio toitmiseks.

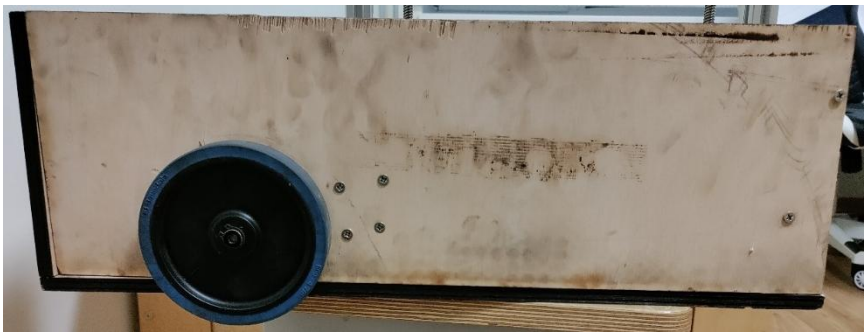
Ülejäänud kontaktid on juhid. ENA -, ENA+ lülitab draiveri sisse ja välja, need on omavahel suletud, see tähendab, et draiver on pidevalt sisse lülitatud. DIR -, DIR+ vastutavad liikumissuuna eest. DIR-on suletud-5V, DIR+ on ühendatud Arduino digitaalse kontaktiga. PUL -, PUL+ määrab impulsside arvu. PUL-on ka-5V juures suletud ja PUL+ läheb arudinosse. Andes sellele kontaktile signaali teatud sagedusega, saate saavutada erineva liikumiskiiruse.

2.3 Roboti kokkupanek

Kui kõik komponendid ja elektriskeem olid valmis, autor alustas robotit kokku panema. Kokkupanek võttis üsna palju aega, kuna autoril pole selles küsimuses kogemusi, kokkupanek toimus mitmes etapis. Selle käigus tuvastati mõned puudused algses plaanis või teatud tehniliste lahenduste läbivaatamine maksimaalse tulemuse saavutamiseks.

2.3.1 Kere

Nagu eespool mainitud, on korpuse materjal vineer. Vineerist oleks pidanud kuubi kokku panema ilma ülemise servata mugavaks paigaldamiseks. Samuti oli vaja vineeri sisse teha augud rataste paigaldamiseks (vt Joonis 2.11, Joonis 2.12).



Joonis 3.11 Taga ratta (autori foto)

Joonis 2.12 Eesimene ratta (autori foto)

Kasutatakse 3 ratast, 2 tagaratast koos mootoritega, eesmist tugiratast. Selline skeem võimaldab saavutada piisava pööratavuse ja kokkupaneku lihtsuse.

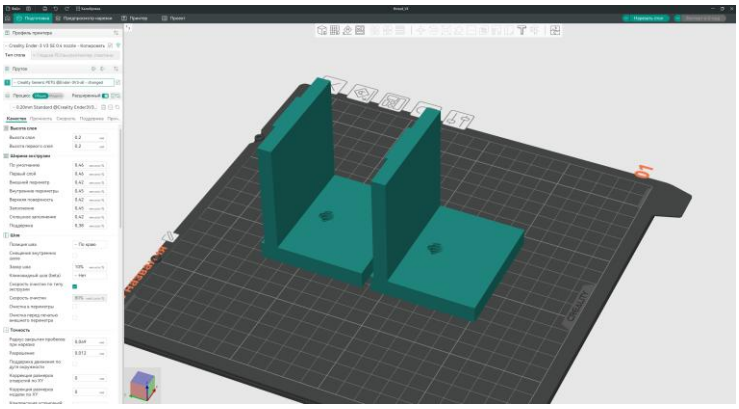
2.3.2 Alumiiniumprofiili kokkupanek

Eraldi punktina soovib autor esile tõsta rullide alumiiniumraami kokkupanekut. Kuigi esmapilgul ei tundu see kõige keerulisem, on igal pool oma nüansid.

Alumiiniumraam on disain, kus on kaks profiili, millel rullid sõidavad, ja 2 sama profiili ülal ja all hoiavad seda kõike kindlalt. Sellises konstruktsioonis on vuukide ja nurkade tasasus väga oluline, kui raamil on märkimisväärne painutus, kus see mõjutab konstruktsiooni tugevust ja täpselt samm-mootoreid. Selle konstruktsiooni joondamise keerukus on profiilide kinnitustes üksteise vahel. Esiteks kasutas autor metallklambreid, mis sisestatakse profiili sisse ja kinnitatakse kruviga. Seda tüüpi kinnitus ei suutnud tagada piisavat tasasust, kuna sulg võis profiili sees veidi nihkuda ja seda hetke oli võimatu reguleerida. Pärast mitmeid katseid leidis autor teise kinnitusdetaili. Nüüd on need metallist nurgad, mis paigaldatakse profiilile ja kinnitatakse kruviga. Kuigi need pole ideaalsed, võimaldasid nad siiski tagada konstruktsiooni väärika tasasuse ja tugevuse. Joonisel 2.13 on näidatud ka plastist nurki. Need nurgad modelleeris ja printis autor 3D printerile.

Joonis 2.13 Alumiiniumraama (autori foto)

Modelleerimiseks kasutati programmi Autodesk Fusion 360 ja mudeli seadistamiseks printimiseks programmi *OrcaSlicer* (vt Joonis 2.14).

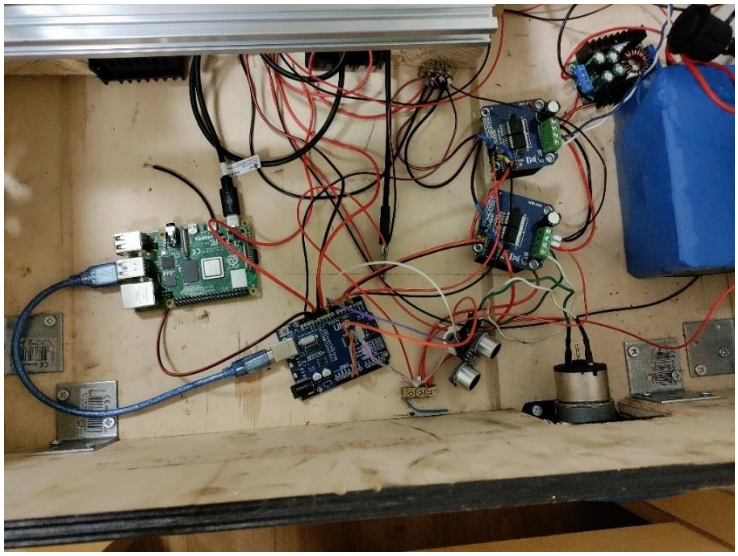


Joonis 2.14 Plastist nurki 3D model (autori foto)

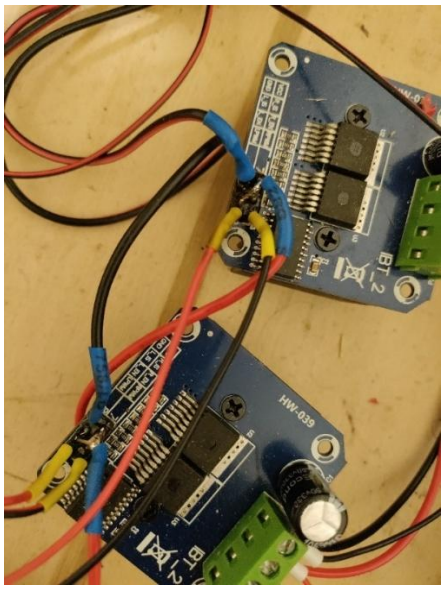
2.3.3 Komponentide jootmine ja paigaldamine

Jootmine tehti vastavalt skeemile. Autor järgis jootmise ja Ohutustehnika põhiprintsiipe. Hea kontakti tagamiseks kasutati ferruleid, Juhtmete õige juhtmestiku tagamiseks kasutati terminale ja enne toite andmist nimetati kõiki ühendusi multimeetriks.

Joonisel 2.15 ja Joonisel 2.16 on näidatud üldine kokkupanek. Kuna projekt pole veel lõpule jõudnud ning komponentide paigutuses ja nende ühendamises tehakse muudatusi, näevad juhtmed lohakad välja, kuid see parandatakse projekti lõpuks.



Joonis 2.15 Elektrooniline kokkupanek

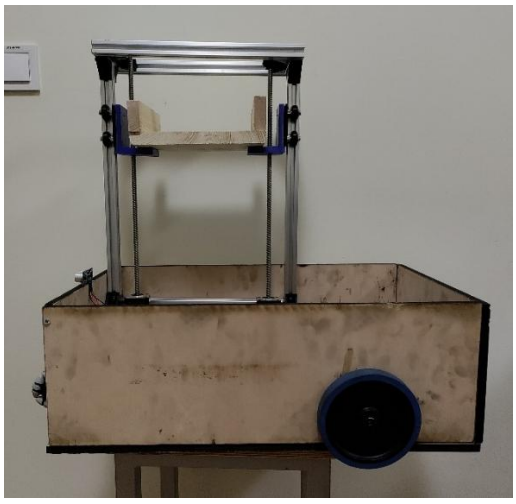


Joonis 2.16 Elektrooniline kokkupanek

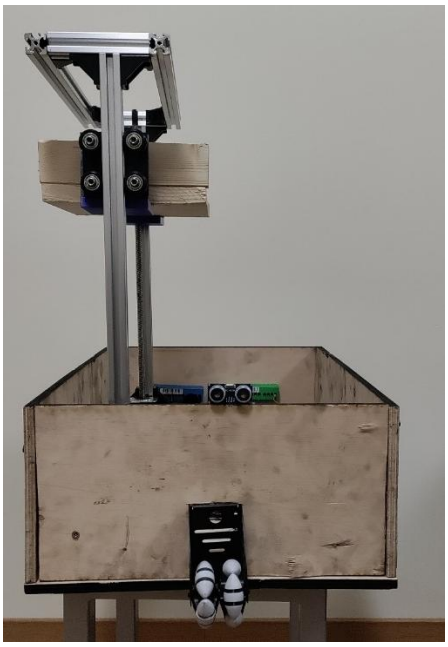
See osa oli kõige töömahukam, kuna autoril pole jootmisega palju kogemusi. Samuti on elektriskeemi mitu korda muudetud või täiendatud.

2.3.4 Valmis kokkupanek

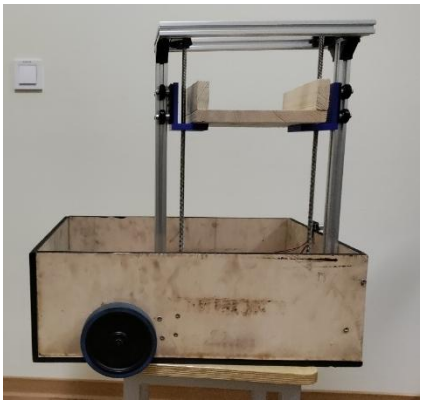
Joonised 2.17, 2.18, 2.19 näitavad prototüüpi kolmest küljest. Töö koosneb panele korpusest, 3 ratast, alumiiniumraamist, platvormist, mis on valmistatud kastide haaramiseks. See välimus ei ole lõplik, kuna mõned tellitud komponendid pole veel saabunud ja kõiki roboti funktsioone pole rakendatud.



Joonis 2.17 Vaade vasakule



Joonis 2.18 Vaade eest



Joonis 2.19 Vaade paremale

Selles etapis saab robot füüsiliselt liikuda tasapinnal, tõsta või langetada platvormi ja lugeda teavet ultraheliandurilt.

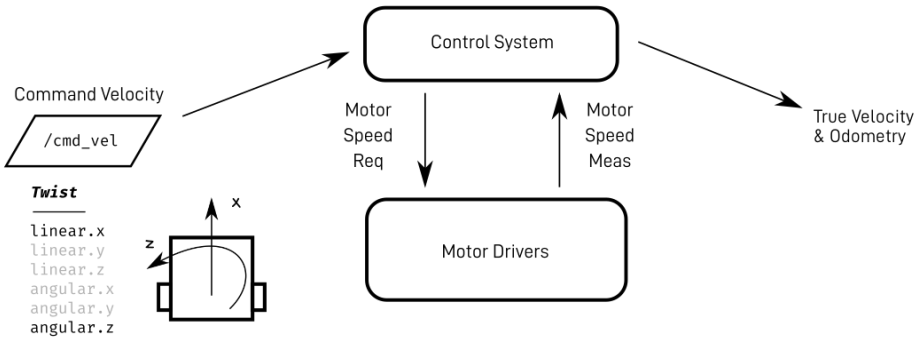
3 PROGRAMMEERIMINE

Roboti enda paralleelse kokkupanekuga hakkas autor ROS-i baasil struktuuri lahti rullima. *ROS* installimiseks vajame *Ubuntu 20.04* arvutit. Järgmisena, järgides ametlikul veebisaidil olevaid juhiseid, installime ROS. Paralleelselt tuleb sama teha ka Raspberry. Raspberry mugavaks kasutamiseks kasutas autor ssh ühendus. See protokoll võimaldab teil seadmega kaugühendust luua ja terminali kaudu kõiki vajalikke toiminguid teha.

3.1 Projekti loomine

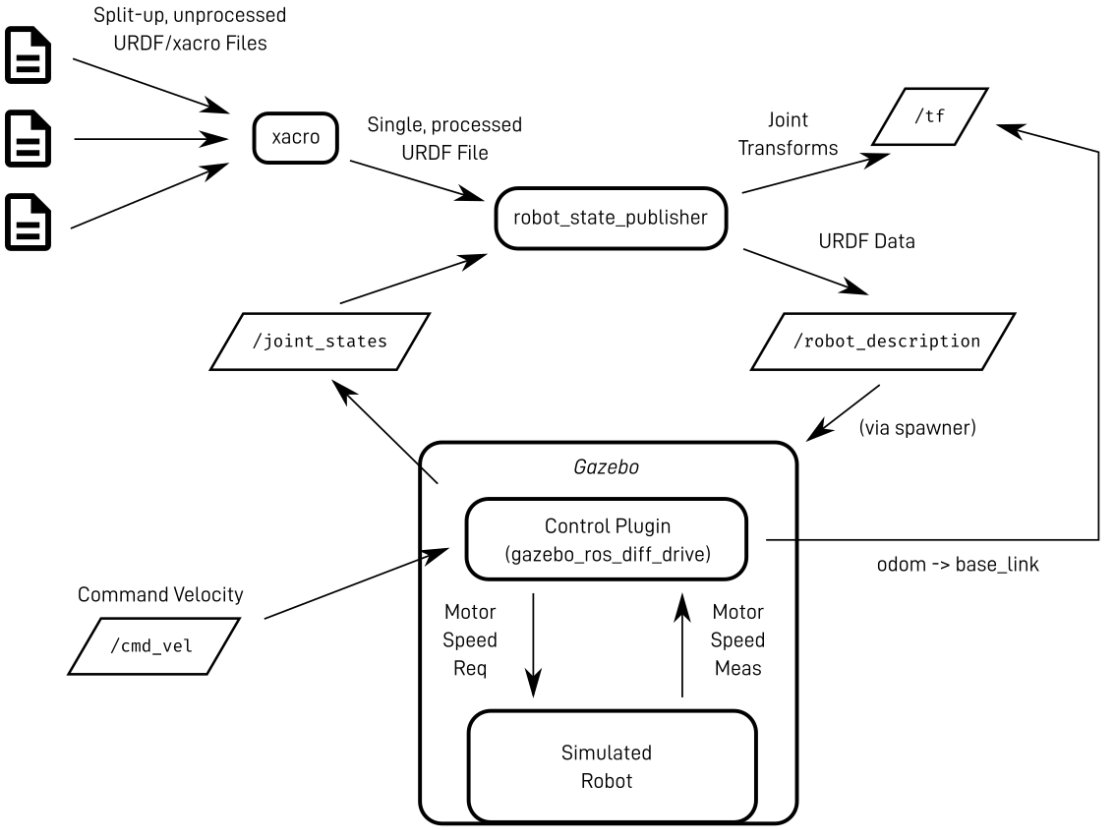
Nüüd on vaja luua kaustad, kuhu kogunemised kogutakse. Põhiarvutis luuakse kaust *dev_ws, Raspberry robot_ws*. Seejärel looge build tool "*colcon*" abil tööruum. Ilmus kolm kausta, meie jaoks on kõige olulisem src, just selles luuakse kõik meie paketid.

Pärast kõigi vajalike pakettide installimist saate *Gazebo* keskkonnas alustada roboti 3D-mudeli loomist. See on vajalik tulevikus mugavamaks ja visuaalsemaks programmeerimiseks. *ROS* kasutab *urdf*-faile, nende struktuur sarnaneb html-koodiga ja kasutatakse ka silte. See fail kirjeldab kõiki roboti parameetreid, nagu rataste suurus, liigesed, mass ja inerts. Pärast urdf-vormingus töö kirjeldust saab simulatsiooni käivitada. Selleks peate kirjutama skripti, mille käivitamisel käivitatakse kõik paketid ja nodes, selle faili nimetab autor *launch_sim.launch.py* tellides *ROS 2* meeskonna launch *my_bot launch_sim.launch.py Gazebo* simulaator käivitub meie robotiga. Nüüd saate alustada liikumise simuleerimist. Kõigepealt peate kirjutama sparco faili, kuhu kirjutame meile vajalikud pistikprogrammid, et gazebo saaks roboti liikumisega töötada. Liikumise juhtimiseks kasutab autor *ros2_control* (vt Joonis 3.1), see on pakett, mis võimaldab teil kontrollida liikumisi telgedel.



Joonis 3.1 ros2_control [13]

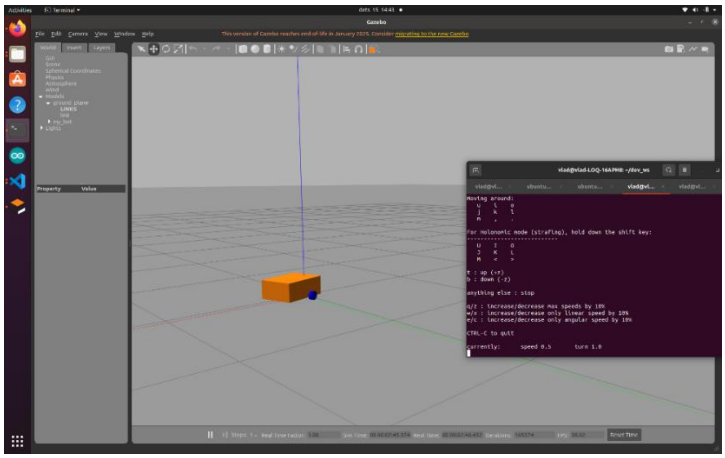
Joonisel 3.2 on näidetud edastamine *urdf*-failide väärtuse saidile *robor_state_publisher*, sellest sõlmest edastab väärtused teemale ja see ros2_controlile, tehes kõik arvutused, avaldatakse väärtused jaotises *joint_states* ja sõlme kaudu tehakse ühine teisendus.



Joonis 3.2 ROS kommunikatsioon [13]

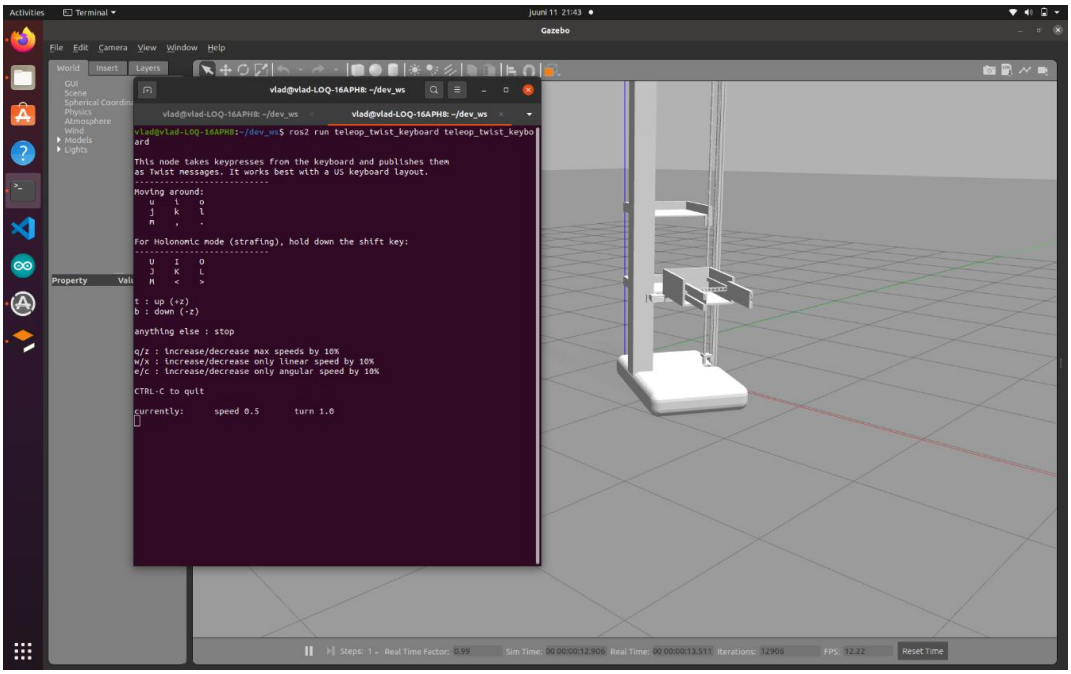
3.2 Simulatsiooni käivitamine

Nüüd tööriista *teleop_twist_keyboard* abil saame kiiruse simulatsiooni edastada. Joonisel 3.3 on näidatud *Gazebo* simulaator ise, roboti lihtsustatud 3D-mudel ja *teleop_twisti*ga töötav Terminal. *Teleop_twist* võimaldab klaviatuuri abil robotit juhtida.



Joonis 3.3 Gazebo

Selles simulaatoris (vt Joonis 3.4) oli ka täielik 3D-roboti mudel, mille tegid mehaanikud.



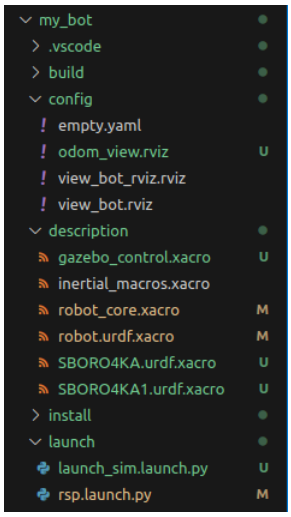
Joonis 3.4 3D mudel

Mudel loodi *Solidworks*is ja teisendati spetsiaalse pistikprogrammi abil *UDRF* faili. Hiljem ei kasutanud autor mudelit, kuna see on programmi jaoks üsna keeruline töödeld.

3.3 Projekti struktuur ja koodid

Selleks, et kõik korralikult töötaks, vajate palju faile, nii et failisüsteem kasvab kiiresti. Selles etapis näeb ta välja selline:

Joonis 3.5 näitab paketi (kausta) sisu *my_robot*. Paketis *my_bot* on palju kaustu, config, description, launch on peamised. Kaustas launch on failid süsteemi käivitamiseks, käivitades need failid terminali kaudu *ROS* installib kõik registreeritud paketid, loob sõlmed ja nendevahelised ühendused. Kaustas description on *PDF*-failid. Need kirjeldavad *3D*-mudeleid ja tööks vajalikke pluginaid. Näiteks, failis *gazebo_control.xarco* ühendab *diff_drive* pistikprogramm, mis edastab kiiruse väärtuse edasi. *Config* kausta on vaja pluginate seadistamiseks.



Joonis 3.5 Faili struktuur

Joonisel 3.6 on esitatud fail *launch_sim.launch.py*. Siin kirjeldatakse, milliseid faile peate käivitama ja nende seadeid, seda on vaja kõigi pakettide korraga käivitamiseks ühe käsuga.

```
20      package_name='my_bot' #<--- CHANGE ME
21
22      rsp = IncludeLaunchDescription(
23                  PythonLaunchDescriptionSource([os.path.join(
24                      get_package_share_directory(package_name),'launch','rsp.launch.py'
25                  )]), launch_arguments={'use_sim_time': 'true'}.items()
26      )
27
28      # Include the Gazebo launch file, provided by the gazebo_ros package
29      gazebo = IncludeLaunchDescription(
30                  PythonLaunchDescriptionSource([os.path.join(
31                      get_package_share_directory('gazebo_ros'), 'launch', 'gazebo.launch.py')]),
32                  )
33
34      # Run the spawner node from the gazebo_ros package. The entity name doesn't really matter i
35      spawn_entity = Node(package='gazebo_ros', executable='spawn_entity.py',
36                          arguments=['-topic', 'robot_description',
37                                     '-entity', 'my_bot'],
38                          output='screen')
39
40
41
42      # Launch them all!
43      return LaunchDescription([
44          rsp,
45          gazebo,
46          spawn_entity,
47      ])
```

Joonis 3.6 launch_sim fail

Joonisel 3.7 on näidatud *rsp.launch.py* kirjeldatakse pakette, sätteid ja failiteid simulatsiooni genereerimiseks, siin käivitatakse ka peamine sõlm nimega *robot_state_publisher*, kus asuvad kõik roboti parameetrid, nagu näiteks tööasend, tagasiside mootoritelt ja liikumine ruumis.

```python
    use_sim_time = LaunchConfiguration('use_sim_time')

    # Process the URDF file
    pkg_path = os.path.join(get_package_share_directory('my_bot'))
    xacro_file = os.path.join(pkg_path,'description','robot.urdf.xacro')
    robot_description_config = xacro.process_file(xacro_file)

    # Create a robot_state_publisher node
    params = {'robot_description': robot_description_config.toxml(), 'use_sim
    node_robot_state_publisher = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        output='screen',
        parameters=[params]
    )


    # Launch!
    return LaunchDescription([
        DeclareLaunchArgument(
            'use_sim_time',
            default_value='false',
            description='Use sim time if true'),

        node_robot_state_publisher
    ])
```

Joonis 3.7 rsp.launch.py fail

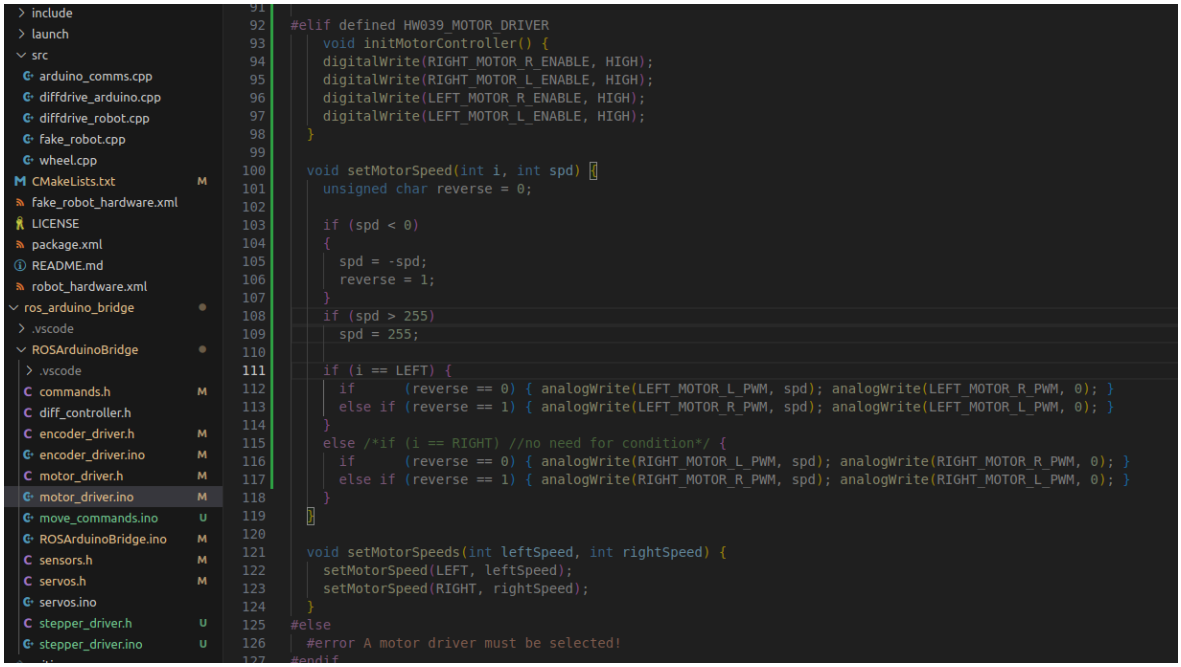
Joonisel 3.8 on näide *URDF*-failist. Siin kirjeldatakse *HTML*-vormingus roboti välimust ja parameetreid nagu materjal ja suurus.

```xml
    <xacro:property name="wheel_mass" value="0.05"/>
    <xacro:property name="wheel_offset_x" value="0.226"/>
    <xacro:property name="wheel_offset_y" value="0.1485"/>
    <xacro:property name="wheel_offset_z" value="0.01"/>
    <xacro:property name="caster_wheel_radius" value="0.01"/>
    <xacro:property name="caster_wheel_mass" value="0.01"/>
    <xacro:property name="caster_wheel_offset_x" value="0.075"/>
    <xacro:property name="caster_wheel_offset_z" value="${wheel_offset_z - wheel_

    <material name="white">
        <color rgba="1 1 1 1" />
    </material>

    <material name="orange">
        <color rgba="1 0.3 0.1 1"/>
    </material>

    <material name="blue">
        <color rgba="0.2 0.2 1 1"/>
    </material>

    <material name="black">
        <color rgba="0 0 0 1"/>
    </material>

    <material name="red">
        <color rgba="1 0 0 1"/>
    </material>
```

Joonis 3.8 robot_core fail

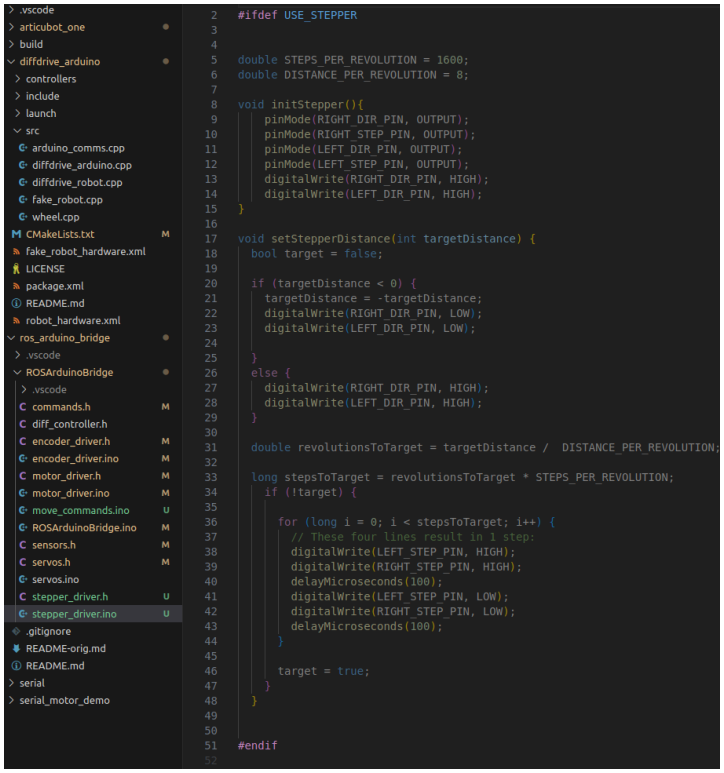
Nüüd saate alustada Simulatsioonist tõelise robotini. Kuna Arduino valiti kontrolleriks, on kõigepealt vaja juhtkoodi. See kood on üldkasutatav ja seda nimetatakse *ros_aruino_bridge*. [9] See kood annab mitu faili Mootorite, servoajamite, andurite, endkoodrite ja PID regulaatori juhtimiseks. Need failid kirjeldavad funktsioone, mida kasutatakse *loop ().* Põhifail on *Ros_arduino_bridge*, siin toimub Arudino suhtlus *Ros*-iga seriali kaudu ja kutsutakse kõiki funktsioone.

Joonis 3.9 näitab draiverit mootorite juhtimiseks. Juht võtab kiiruse väärtuse vahemikus 0 kuni 255 ja väljastab PWM signaal draiveritele.



Joonis 3.9 Motor_driver

Samm-mootorite juhtimiseks vajate ka oma draiverit (vt Joonis 3.10), autor lõi selle juba nullist.

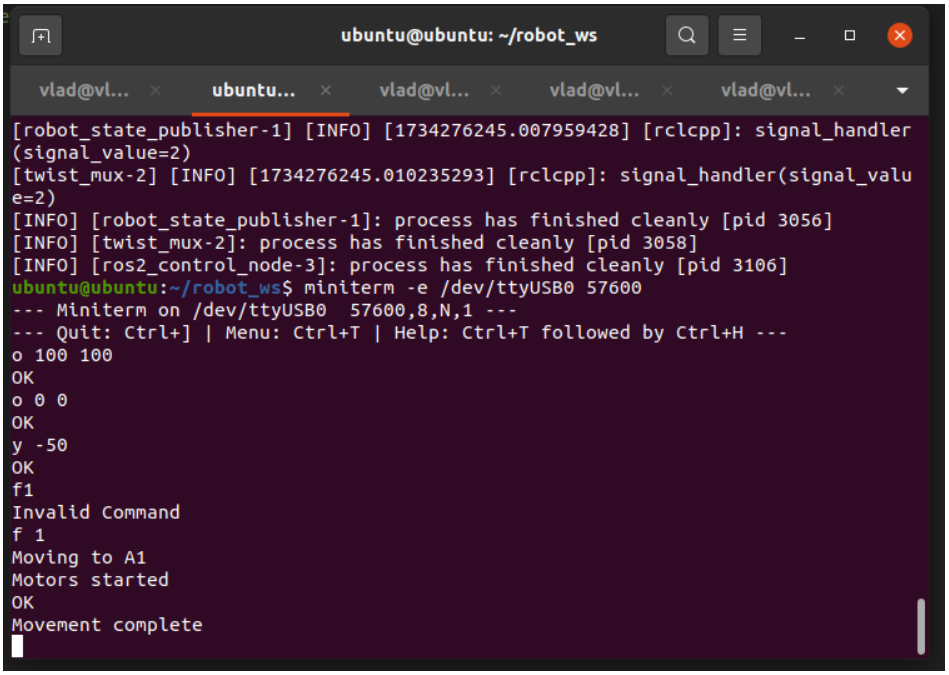


Joonis 3.10 Step_driver

Juht saab väärtuse millimeetrites, seejärel arvutatakse füüsilise draiveri sätete ja

samm-mootori parameetrite põhjal vajalik impulsside arv.

Kõigi funktsioonide testimiseks kasutatakse terminali, minitermi teeki abil saate porti ühendada. Porti ühendades saate kirjutada käske, mis täidavad erinevaid funktsioone. Kommandide loetelu joonisel 3.11.

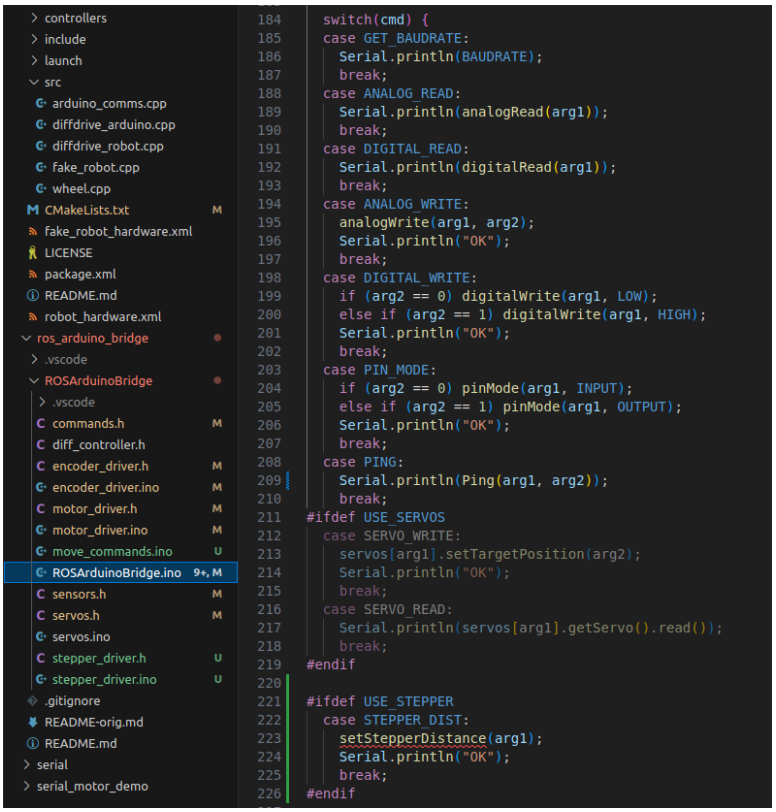


Joonis 3.11 Miniterm

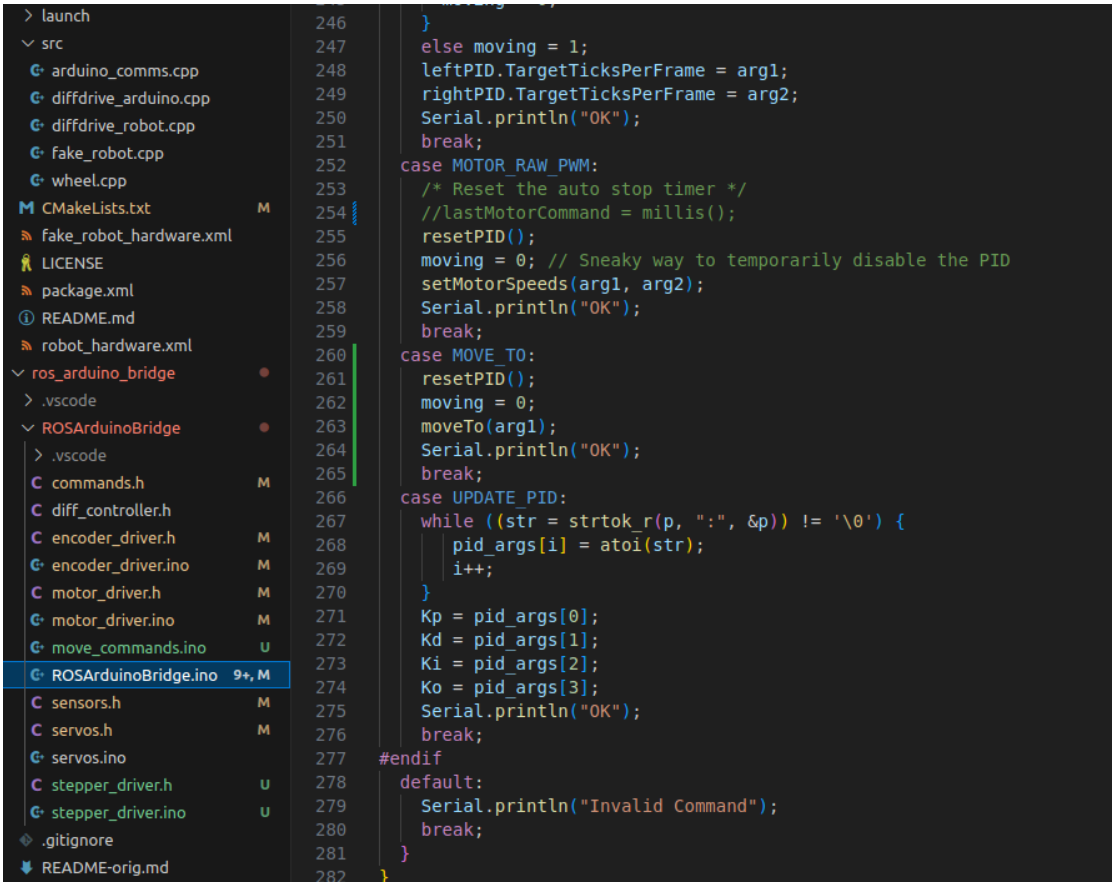
Näiteks kirjutades terminali "o" ja signaali PWM väärtuse, hakkavad mootorid töötama. Kommand "y" käivitage samm-mootorid kuni nad läbivad määratud vahemaa.

Selleks, et kõik toimiks, oli vaja ka redigeerida ros_arduino_bridge faili. Sellele lisati kõne funktsioon *setStepperDistance()* ja teatud koodiplokid edastati tööülesannete jaoks.

Joonised 3.12 ja 3.13 näitavad osa failist *ros_arduino_bridge*. Neil on nähtavad funktsioonide väljakutsed.



Joonis 3.12 ros_arduino_bridge setStepper



Joonis 3.134 ros_arduino_bridge kommand loogika

Komponentide töövõime demonstreerimiseks täiendati faili ka funktsiooniga *moveTo()* (vt Joonis 3.14), mis toimib algoritmina. See näeb ette funktsioonide täitmise jada, et simuleerida roboti liikumist riiulile ja tõsta platvormi teatud kaugusele.

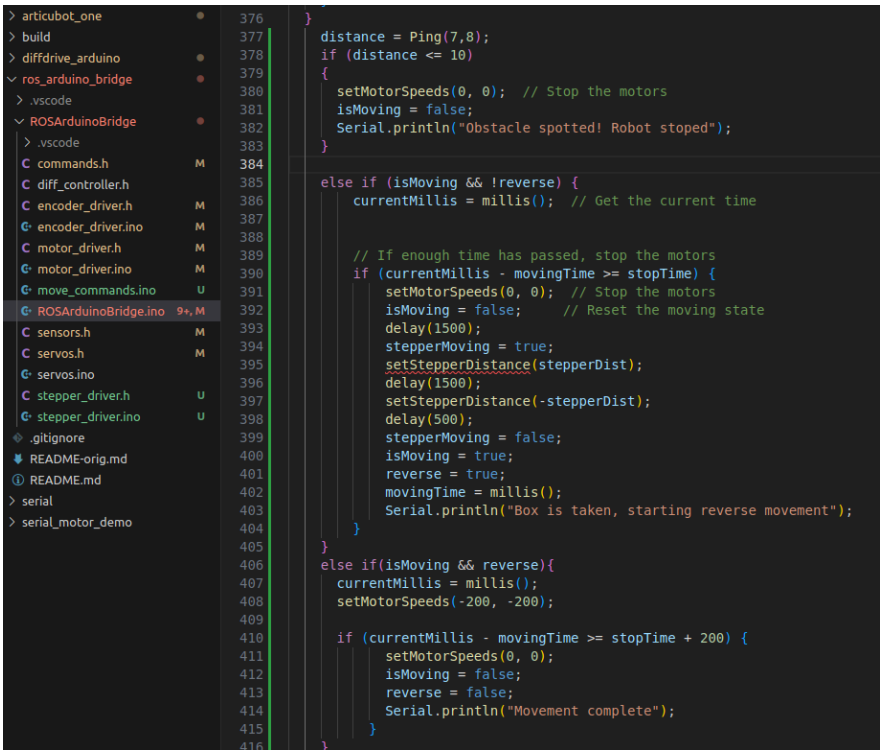
Antud funktsioon võtab argumendi 1 või 2, Esimene või teine rügement. Seejärel kasutades konstruktsiooni switch case määrab funktsioonidele erinevad parameetrid.

Seejärel lülitatakse mootorid sõltuvalt valitud riiulist kõigepealt sisse, et liikuda edasi määramata ajaks. Kui robot hakkas, lülituvad samm-mootorid sisse ja tõstavad platvormi etteantud väärtusele.

Samuti on rakendatud *Ping()* funktsioon (vt Joonis 3.15), mis kontrollib kaugust, mida ultraheliandur mõõdab, ja kui takistus on lähemal kui 10 cm, peatab see roboti.

```
if (!isMoving)
{
  // Initialize movement only if not already moving or if the target changes
  isMoving = true;
  reverse = false;

  switch (targetPoint)
  {
  case 1: //Upper lest
    stopTime = 2300; // 3 seconds to reach target A1
    setMotorSpeeds(speed, speed);
    Serial.println("Moving to A1");
    stepperDist = 200;
    break;

  case 2: //Upper right
    stopTime = 2000;  // 0.5 seconds to reach target A2
    setMotorSpeeds(speed, speed);
    Serial.println("Moving to A2");
    stepperDist = 200;
    break;
  case 3: //Lower left
    stopTime = 2300; // 0.5 seconds to reach target A2
    setMotorSpeeds(speed, speed);
    Serial.println("Moving to A2");
    stepperDist = 100;
    break;
  case 4: //Lower right
    stopTime = 2000; // 0.5 seconds to reach target A2
    setMotorSpeeds(speed, speed);
    Serial.println("Moving to A2");
    stepperDist = 100;
    break;

  default:
    Serial.println("Invalid target");
    isMoving = false;
    return; // Exit function for invalid target
  }
```

Joonis 3.14 moveTo() funkstsiooni



Joonis 3.15 moveTo() loopis

See algoritm tuleb rakendada *ROS* abil. Selle jaoks tuleks kirjutada oma *Node* ja siis saab seda täielikult kasutada. Praegu on algoritm rakendatud loop-is, et näha töövõimet.

Koodi kirjutamisel, õige struktuuri loomisel ja roboti ehitamisel kasutati materjale saidilt, kus autor lõi video-ja tekstivormingus juhised ROS-i programmeerimiseks ja robotite loomiseks. See aitas palju ja kiirendas arendusprotsessi.[13][14]

3.4 Roboti töö demonstreerimine

Pärast kõiki tehtud töid õnnestus tulemusi saavutada. Robot saab teostada toimingute algoritmi ja seda saab juhtida ka klaviatuurilt.

https://drive.google.com/file/d/1xsIL1tn9lebdz4O4jDWn2DeJ3u48T1d5/view?usp=drive_link

See video näitab demonteeritavalt roboti tegevuse algortimi, et haarata kasti riiulilt. Kood käivitatakse, kui sisestate terminali käsu, näidates riiuli. Sõltuvalt käsust edastatakse muutujate erinevad parameetrid, näiteks mootorite tööaeg või platvormi tõstmise kõrgus.

https://drive.google.com/file/d/1xmuklb6OBV2jbz9yop6flDBcMZY1yxpN/view?usp=sharing

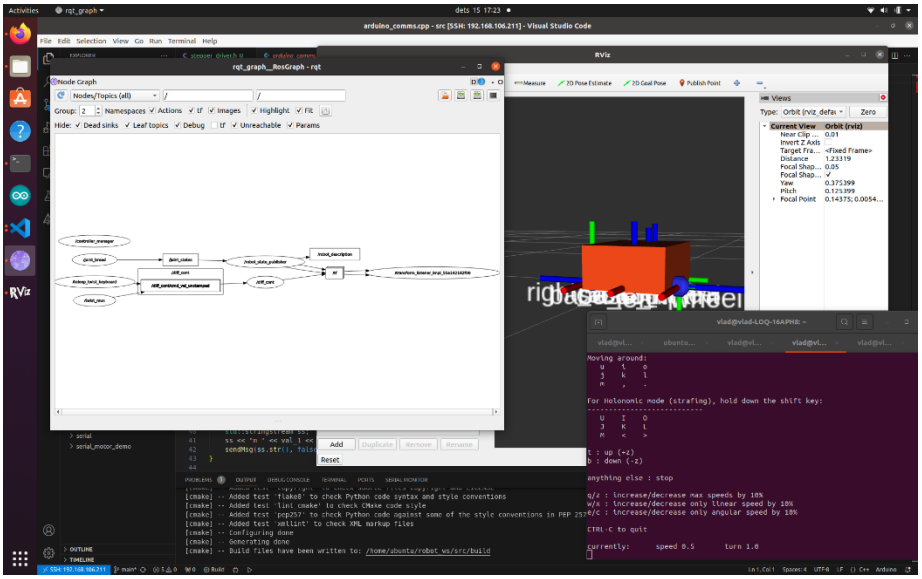
Samuti töötab ultraheliandur, u peatab roboti, kui tuvastatakse prits.

https://drive.google.com/file/d/1xa_aiFiy5870xSxv20PRVrLmXKxwCCIe/view?usp=sharing

Järgmine video näitab "manuaalset režiimi". Klaviatuuri abil saab autor juhtida mootoreid ja seega robotit kaugjuhtida. Selles režiimis töötavad kõik ROS-ühendused,

iga varem registreeritud Node avaldab andmeid või kuulab teisi sõlme.

Joonis 3.16 näitab sülearvuti ekraani roboti juhtimise ajal "manuaalses režiimis". Vasakpoolne aken näitab kommunikatsiooni ROS-i sees. Topic / teleop_twist_keyboard edastab vajutatud klahvi andmed Node diff_cont. Selles Nodes toimub vajutamise teisendamine kiiruseks ja ühenduste liikumiseks. Nii saab ROS aru, milliseid andmeid kontrollerile edastada. Ekraanil on ka Rviz. See programm visualiseerib roboti ja selle ühenduste liikumist. Kui kooderid on ühendatud, kordab Rviz roboti liikumisi reaalajas. See võimaldab teil simulatsiooni käivitada ja selgelt näha, mis robotiga toimub. Paremal küljel on terminal, kus töötab käsk teleop_twist_keyboard. Nimetatud klahvide abil saab robotit reaalajas juhtida.



Joonis 3.16 Rviz, Graphs, teleop

KOKKUVÕTE

Lao robotiseerimine on töö efektiivsuse suurendamise lahutamatu osa. Laos olevad robotid suurendavad tehtud tööde arvu, vähendavad nende teostamise aega, võimaldavad mastaapsust ja säästavad inimesi monotoonsest ja ohtliku töö hetkedest.

ROS-kasulik robotisüsteemide arendamisel, see võimaldab teil rakendada erinevaid ideid, jätta välja palju tehnilisi punkte ja keskenduda põhiülesande täitmisele.

Praegu ei ole prototüübis võimalik kõiki soovitud funktsioone rakendada, samuti on vajalik juba olemasolev kood. Vaatamata kõigele sellele õnnestus saavutada teatud tulemusi. Koostatakse tööriistade ja pakettide andmebaas, mida saab ainult täiendada ja täiustada. Robot on võimeline liikuma kaugjuhtimise abil, samuti saab sammumootoreid käivitada kaugjuhtimisega. Samuti, kuigi demonstratiivselt, on robot võimeline sooritama toimingute komplekti, järgides antud loogikat.

Füüsiliselt pole robot ka täielikult valmis, kuid on juba hea alus, mida saab ka edasi arendada ja täiustada. Autori lähiajal on see rakendada hästi struktureeritud kood, mis suhtleb otse ROS-iga ja kasutab kõiki selle võimalusi. Samuti paigaldatakse robot riiulitelt kastide haaramise mehhanismiga.

Kõiki eesmärke ei saavutatud ega täidetud 100%. Selle põhjuseks on selliste süsteemidega töötamise aja ja kogemuste puudumine. Autor kohtus esimest korda roboti arendamisega ja iga element oli uus. Seetõttu kulus materjali uurimiseks, ettevalmistamiseks ja rakendamiseks palju aega. Mõnikord võib esmapilgul lihtne ülesanne võtta nädala, et viia see tööseisundisse. See kehtib nii töö programmilise osa kui ka roboti enda füüsilise kokkupaneku kohta. Näiteks draiveri kirjutamine võttis vabal ajal paar nädalat tööd. Kuigi lõpuks on see väga sarnane juba esitatud draiveriga, oli vaja täielikult mõista, milline koodirida mille eest vastutab.

Töö kokkuvõtmisel on autor tulemusega rahul. Kõik ei õnnestunud realiseerida, kuid ees on veel palju plaane ja ideid roboti täiustamiseks ning prototüübi muutmiseks täieõiguslikuks robotiks.

SUMMARY

Robotization of a warehouse is an integral part of increasing work efficiency. Robots in stock increase the number of works performed, reduce the time of their execution, allow scalability and save people from monotonous and dangerous moments of work.

ROS-useful in the development of robotic systems, it allows you to implement various ideas, omit many technical points and focus on performing the main task.

At the moment, it is not possible to implement all the desired functions in the prototype, and already existing code is also required. Despite all this, we managed to achieve certain results. A database of tools and packages is compiled, which can only be supplemented and improved. The Robot is able to move remotely, and the stepper motors can also be started remotely. Also, although demonstratively, the robot is able to perform a set of actions, following the given logic.

Physically, the robot is also not completely ready, but there is already a good base that can also be further developed and improved. In the near future of the author, it is to implement a well-structured code that communicates directly with ROS and uses all its capabilities. Also, the robot is installed with a mechanism for grabbing boxes from shelves.

Not all goals were achieved or met 100%. This is due to the lack of time and experience of working with such systems. The author first met the development of the robot, and each element was new. Therefore, it took a lot of time to study, prepare and implement the material. Sometimes a simple task at first glance can take a week to bring it to a working state. This applies to both the programmatic part of the work and the physical assembly of the robot itself. For example, writing a driver took a few weeks of work in his spare time. Although in the end it is very similar to the driver already presented, it was necessary to fully understand which line of code is responsible for what.

Summing up the work, the author is satisfied with the result. Not everything was realized, but there are still many plans and ideas ahead to improve the robot and turn the prototype into a full-fledged robot.

KASUTATUD KIRJANDUSE LOETELU

[1] AI-Driven Warehouse Automation 11.12.2024 [Veeb]

https://gsconlinepress.com/journals/gscarr/content/ai-driven-warehouse-automation-comprehensive-review-systems#:~:text=AI-driven%20warehouse%20automation%20systems,adapt%20to%20changing%20customer%20needs.

[2] Robot path optimization in warehouse management system 11.12.2024 [Veeb]

https://link.springer.com/article/10.1007/s12065-021-00614-w

[3] A Cyber-Physical Warehouse Management System Architecture in an Industry 4.0 Context 11.12.2024 [Veeb]

https://www.researchgate.net/publication/344046960_A_Cyber-Physical_Warehouse_Management_System_Architecture_in_an_Industry_40_Context

[4] The Amazing Ways Amazon Is Using AI Robots 11.12.2024 [Veeb]

https://www.forbes.com/sites/bernardmarr/2024/09/20/the-amazing-ways-amazon-is-using-ai-robots/

[5] Warehouse Automation: Types, Challenges, and Benefits 11.12.2024 [Veeb]

https://novushitech.com/warehouse-automation-types-challenges-and-benefits/#:~:text=Warehouse%20automation%20is%20transforming%20the,benefits%20far%20outweigh%20these%20obstacles.

[6] Warehouse Robotics Market: E-commerce increased the Need for More Efficient Warehouse Operations to Handle the High Volume of Orders and Drive the Demand for Robotics 11.12.2024 [Veeb]

https://www.maximizemarketresearch.com/market-report/global-warehouse-robotics-market/33454/#:~:text=Global%20Warehouse%20Robotics%20Market%20size,at%20a%20CAGR%20of%2015.7%20%25.

[7] The Robot Operating System (ROS1 &2): Programming Paradigms and Deployment 14.12.2024 [Veeb]

https://www.researchgate.net/publication/363842243_The_Robot_Operating_System_ROS1_2_Programming_Paradigms_and_Deployment

[8] ROS concepts 14.12.2024 [Veeb]

https://wiki.ros.org/ROS/Concepts

[9] ROSArduinoBridge [Veeb] 14.12.24

https://github.com/joshnewans/ros_arduino_bridge

[10] BTS7960 [Veeb] 14.12.24

https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf

[11] What is a Raspberry Pi? [Veeb] 14.12.24

https://opensource.com/resources/raspberry-pi

[12] About Arduino [Veeb] 14.12.24

https://www.arduino.cc/en/about

[13] ROS and Robotics tutorials [Veeb] 14.12.24

https://articulatedrobotics.xyz/

[14] ROS codes [Veeb] 14.12.24

https://github.com/joshnewans

[15] Aku 24v [Veeb] 25.12.24

https://www.aliexpress.com/item/1005007523291712.html?spm=a2g0o.productlist.main.11.2b10WEjYWEjYkH&algo_pvid=51ad0f28-f567-4f5c-b780-0dc422c96b7e&algo_exp_id=51ad0f28-f567-4f5c-b780-0dc422c96b7e-5&pdp_npi=4%40dis%21EUR%2157.13%2128.57%21%21%21424.11%21212.05%21%40211b80f717351244797773000ed085%21112000041136279967%21sea%21EE%211836261364%21X&curPageLogUid=Kbi6VGibYB7a&utparam-url=scene%3Asearch%7Cquery_from%3A

[16] TB6600 [Veeb] 25.12.24

https://www.kiwi-electronics.com/en/tb6600-stepper-motor-driver-3752

[17] TB6600 ühendus [Veeb] 25.12.24

https://www.reddit.com/r/arduino/comments/17dgy69/help_stepper_motor_wont_move/?rdt=45363

[18] Nema 17 [Veeb] 25.12.24

https://robolabor.ee/en/stepper-motors/1270-stepper-motor-nema-17-with-30cm-lead-screw.html

[19[Stepdown toitumodul [Veeb] 25.12.24

https://www.oomipood.ee/category/robootika_ja_konstruktorid/moodulid_muundurid/toitemoodulid

[20] Alumiiniumprofiil 2020 [Veeb] 25.12.24

https://aliexpress.ru/item/1005002537611699.html?utm_referrer=https%3A%2F%2Fwww.google.com%2F&sku_id=12000021048744251