TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Marti Kingisepp 185682IACB

MASINNÄGEMISEGA ISESÕITEV VÕISTLUSSÕIDUROBOT

Bakalaureusetöö

Juhendaja: Uljana Reinsalu
Teadur

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Marti Kingisepp
16.05.2022

Annotatsioon

Käesoleva bakalaureusetöö raames on ehitatud masinnägemisega isesõitev võistlussõidurobot. Masinnägemine tuleneb roboti rajatuvastusmeetodist ehk robot navigeerib rajal kasutades videokaamerat. Videokaamera tuvastab HSV (*Hue-saturation-value*) värviruumis rada, et lävendi põhjal eristada seda muust keskkonnast. Robot peab olema võistlussõitudel konkurentsivõimeline ehk võistlusel osaledes sõitma vähemalt ühe ringi, et tõestada rajatuvastusmeetodi toimimist.

Lõputöös on jaotatud roboti ehitus ja programmeerimine kolme peatüki vahel, mis hõlmavad roboti mehaanilisi, riistvaralisi ja tarkvaralisi osasid. Robot on realiseeritud Raspberry Pi 4B sardsüsteemiga, millel on kaks mootorit, videokaamera, ultrahelisensor, mootorite juhtmoodul, RGB (*Red-green-blue*) diood ja juhtpult. Kere detailid on 3D-prinditud ja nende failid on saadaval avatud kaustas (Lisa 2).

Roboti masinnägemise programm on kirjutatud C++ programmeerimiskeeles kasutades OpenCV teeki ja kasutatud Pythonis kirjutatud abistavaid skripte. Kõik roboti loomisel kasutatud programmid asuvad autori avatud Git repositooriumis (Lisa 3).

Töö tulemusena valmis videokaameral põhinev rajatuvastuse algoritmi kasutav isesõitev võistlussõidurobot, mis osales *Robotex International* rahvasõidu võistlusel 2021 aastal. Võistlusel saavutas robot tingimuse, et sõita vähemalt üks ring.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 6 peatükki, 31 joonist, 1 tabelit.

Abstract

Self-Driving Racing Robot with Machine Vision

As part of this bachelor's thesis, a self-driving racing robot based on machine vision has been built. Machine vision comes from the robot's track detection method, that is that the robot navigates the track using a video camera. The image from the video camera is converted to the HSV (Hue-saturation-value) color space to distinguish the track from the rest of the environment based on the threshold set by the HSV filter. The robot must be competitive while racing, which means that the robot has to drive at least one lap while participating in a race to prove that the track detection method works.

The thesis divides the construction and programming of the robot into four chapters, which cover the mechanical, hardware, and software parts of the robot. The robot is implemented on a Raspberry Pi 4B embedded system, with two motors, a video camera, an ultrasonic sensor, a motor control module, an RGB (Red-green-blue) diode, and a gaming controller. Body details are 3D printed and their files are available in a publicly accessible folder (Appendix 2).

The robot's machine vision program is written in C++ programming language using the OpenCV library and uses supportive scripts written in Python. All the program code to create the robot are in the author's open-source Git repository (Appendix 3).

The thesis resulted in a self-driving racing robot that drives based on a video camera input, using a lane detection algorithm. The robot participated In the Robotex International competition in 2021, where the robot participated in a folkrace competition and successfully achieved the condition to drive at least one lap.

The thesis is in Estonian and contains 35 pages of text, 6 chapters, 31 figures, 1 tabel.

Lühendite ja mõistete sõnastik

ABS	Akrüülnitriilbutadieenstüreen
GPIO	*General-Purpose input/output*, mitmeotstarbeline sisend/väljund
HSV	*Hue-saturation-value,* värvitoon-küllastus-väärtus
I2C	*Inter-integrated circuit*, suhtlusprotokoll ehk mitme võimaliku ülemaga jadasiin
MAC	*Media access control address,* meediumipöörduse juhtimise aadress
PLA	*Polylactic acid*, polülaktiid
PWM	*Pulse-width modulation,* pulsilaiusmodulatsioon
RGB	*Red-green-blue,* punane-roheline-sinine
SSH	*Secure shell protocol*, krüptograafiline võrguprotokoll turvaliseks kaugühendamiseks
USB	*Universal Serial Bus,* universaalne jadasiin
VNC	*Virtual network computing*, Virtuaalse töölaua jagamise süsteem

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Autonoomsed sõidukid, mida tuntakse isejuhtivate sõidukitena, on tunnustatud kui praegune suundumus teadusuuringutes ja arendustes, kus paljud suured uurimiskeskused, autotööstusettevõtted ja akadeemilised asutused panustavad sellesse valdkonda igapäevaselt [1]. Käesoleva töö eesmärgiks on ehitada autonoomne sõiduk ehk masinnägemisega isesõitev võistlussõidurobot, mis navigeerib sõidurajal kasutades videokaamerat.

Töös käsitletud robot on arendatud TTÜ Robotiklubis ja sealsete vahenditega piiratud. Ehitatav robot peab vastama *Robotex International folkrace* reeglistikule [2] ehk võistlusel läbima tehnilise kontrolli. Töö autor osales lõputöös käsitletava robotiga TTÜ Robotiklubi projektis projektijuhina.

Käesolev töö võrdleb kahe erineva programmeerimiskeele kasutust valitud masinnägemise algoritmiga, annab ülevaate roboti mehaanilistest ja elektroonika komponentidest ning nende komplekteerimisest, kirjeldab kogu tarkvara paigaldamist, seadistamist ning loomist ja esitab töö tulemused. Töö esimeses osas tutvustatakse roboti mehaanika ja elektroonika komponente, võrreldakse erinevaid sardsüsteemi tuuma valikuid ning komplekteeritakse robot.

Töö teine osa keskendub roboti sardsüsteemi seadistamisele ja tarkvara arendamisele. Paigaldatakse sardsüsteemi tuuma operatsioonisüsteem, kompileeritakse masinnägemiseks vajalik OpenCV teek [3] ning jagatakse programm lõimedeks. Valmis roboti tulemused on esitatud peatükis 5.2.

Väljaspool lõputööd on roboti nimeks „ICU“, mis on inspireeritud inglisekeelsest väljendist „*I see you*“. Nimi tuleneb roboti rajatuvastusmeetodist, kuna seda tuvastatakse videokaameraga.

2 Töömeetodid ja -tehnoloogiad

Lõputöö teostamisel on valdavalt kasutatud vaba ja avatud lähtekoodiga tööriistu ning vahendeid, tänu millele on selle töö teostus võimalikuks osutunud. Projekti pilditöötlus on kirjutatud C++ programmeerimiskeeles kasutades OpenCV raamistikku. Pilditöötluse optimeerimiseks ja üldise programmi töö jagamiseks on kasutatud C++ teeke.

Mootorite ja sensoritega suhtlemiseks on kasutatud wiringPi [4] ja pigpio teeke [5]. Mõlemad nimetatud teegid on kirjutatud C programmeerimiskeeles. Kasutusel on wiringPi teek juhtmooduli jaoks ning pigpio teek servomootori juhtimiseks.

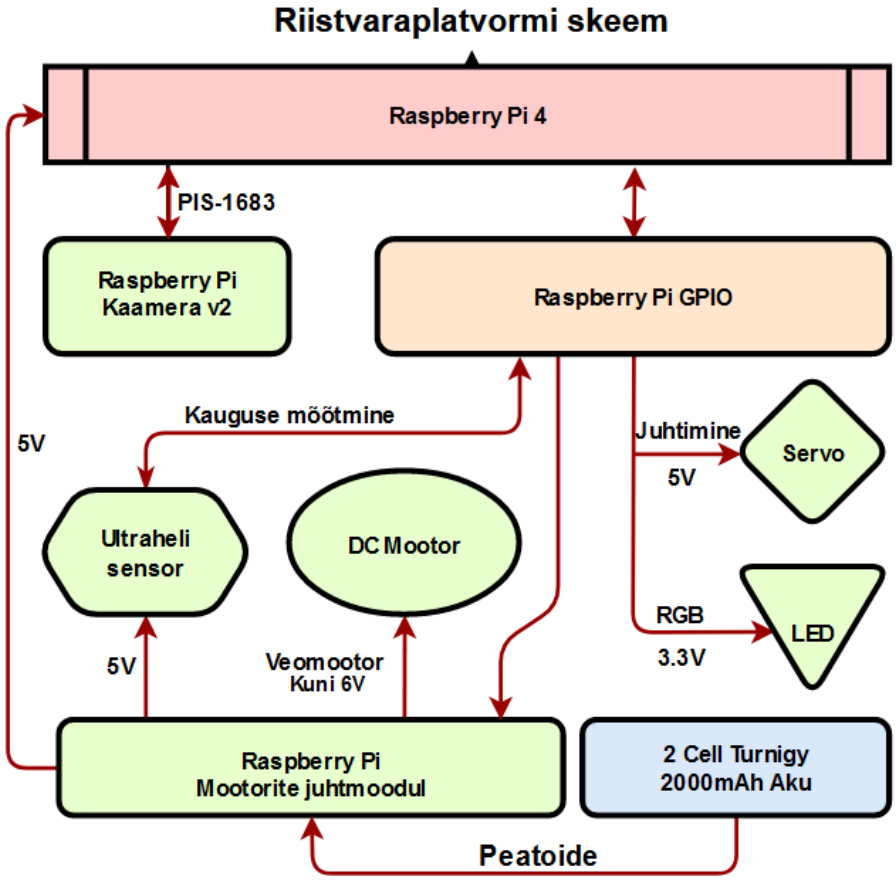
Roboti koostud on prinditud kasutades TTÜ Robotiklubis olevaid 3D printereid: Ender 3 [6] ja Ultimaker 2. Koostude failid on loodud kasutades Solidworks [7] projekteerimistarkvara.

Lõputöös on kasutatud OpenCV raamistiku versiooni 4.5.3. Roboti operatsioonisüsteem on Raspberry Pi OS versioon 10 buster [8]. Pythoni skriptid on kirjutatud kasutates versiooni 3.7.3.

3 Riistvaraplatvormi ehitamine

Selle peatüki eesmärk on kirjeldada lõputöö raames isesõitva roboti ehitamiseks kasutatud metoodikat, tehnikat ja komponente. Kaalutud on mitmeid alternatiivseid komponente ning iga komponendi valik on põhjendatud.

Lõplik roboti riistvara skeem on esitatud joonisel (Joonis 1). allpool oleval joonisel on välja toodud kõik riistvaraplatvormiga seotud elektroonika komponendid. Iga joonisel olev komponent on vastavas alampeatükis detailsemalt kirjeldatud.



Joonis 1. Lõplik riistvaraplatvormi skeem, kus nooled viitavad komponentide ühenduse (ja suhtluse) suunale (värvid ja kujundid on juhuslikult valitud).

3.1 Roboti mehaanika

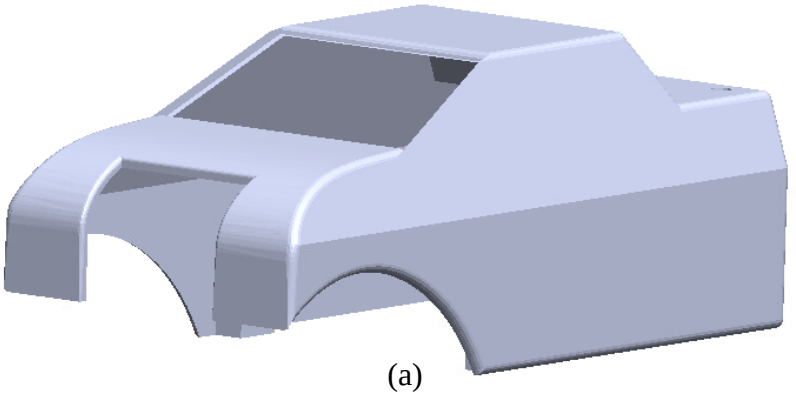
Esialgne kavand oli osta mudelauto, kus oleks olemas kogu mehaanika, elektroonika ja juhtimiskomponent, et lisada mudelautole juhtimissüsteemiks ainult sardsüsteemi tuum. Autor valmistas roboti koos kahe Tallinna Tehnikaülikooli tudengi abiga, kuna sobiva mudelauto leidmine osutus keeruliseks ning roboti ehitamiseks vajalikud komponendid ja tootmisvõimekus olid TTÜ Robotiklubis olemas.

Roboti koostud koostas kaastudeng Sander Kajak, kes tegeles ka koostude printimisega. Korpus on koostatud kursusekaaslase Helen Ennoki poolt. Kõik roboti koostud on saadaval avalikus kaustas, mis on toodud lisas (Lisa 2).

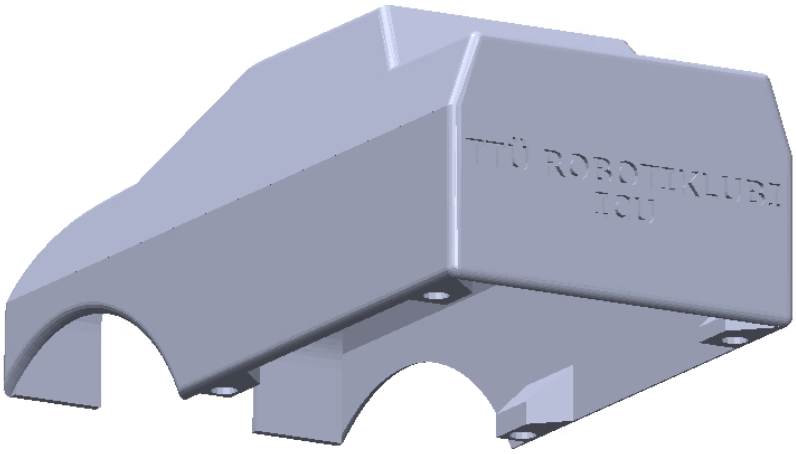
3.1.1 Roboti kere

Kere loomisel oli eesmärk ära mahutada kõik vajalikud komponendid mudelauto suurusega platvormile, mis vastaks võimalikult paljudele võistluste reeglistikele ja mahutaks ära kõik vajalikud elektroonikakomponendid. Kere disainimise ajal kõige detailsem reeglistik oli *Robotex International* ürituse eeskiri [2], mille piirangutest lähtudes sai roboti kere loodud.

Lahtiste elektroonikakomponentide kaitseks on koostatud robotile korpus (Joonis 2), mis kinnitub korpuse ja kere põhja liimitud magnetitega. Korpusele on tehtud esiakna juurde ava, kust saaks robotit mootorite juhtmoodulist mugavalt sisse-välja lülitada. Kaamera jaoks on tehtud roboti ette avaus.



(a)

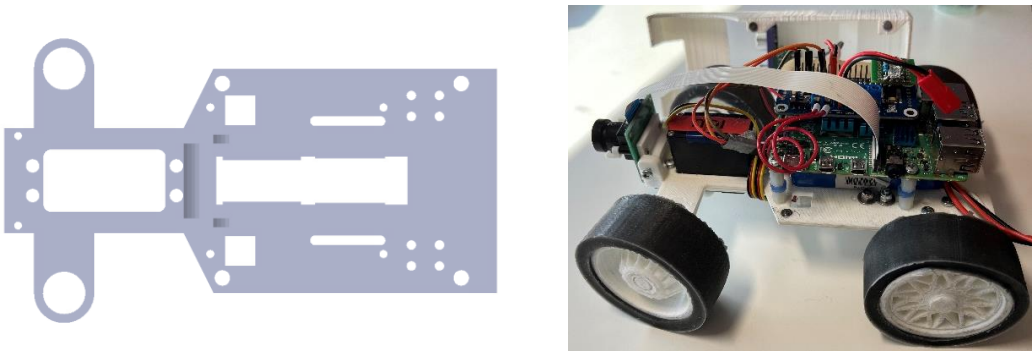


(b)

Joonis 2. Roboti korpus: (a) korpuse eestvaade, (b) korpuse tagant- ja altvaade.

Roboti koostud on 3D-prinditud enamasti termoplastilisest polüestrist ehk polülaktiidist (edaspidi PLA), kuid PLA madala sulamistemperatuuri tõttu on kasutatud printimisel ka akrüülnitriilbutadieenstüreeni (edaspidi ABS). Kuna ABS sulamistemperatuur on kõrgem kui PLA [9], on ABS plastikut kasutatud kohtades, kus võib temperatuur tõusta liiga kõrgeks, et plastikusse hakkaksid tekkima moonutused, mis nihutaksid komponendid paigast. Roboti veomootor oli elektroonikakomponent, mis läks proovisõitudel kõige kuumemaks ning plastiku korduva moonutuste tõttu sai mootori kinnitus hiljem prinditud ABS materjalist.

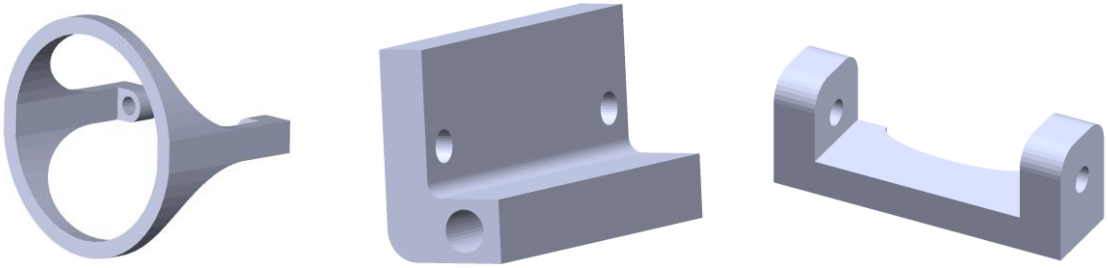
Sardsüsteemi tuum asetseb kere (Joonis 3) keskel kerest 2.3 cm kõrgusel. Selle alla on mahutatud aku, mis on kinnitatud roboti põhja külge kahepoolse teibiga. Väiksema aku puhul ei ole seda roboti külge vaja kinnitada, kuna korpuse lisamisel on aku liikumine piiratud.



(a) (b)

Joonis 3. Roboti kere: (a) kere koost, (b) komplekteeritud kere koos komponentidega.

Kaamera hoidmiseks on 3D-prinditud reguleeritav kaamera jalg, mille külge on kruvitud kaamera (Joonis 4). Kaamera objektiivi vigastuste vältimiseks on objektiivi kinnitusele lisatud kaitsevahend.



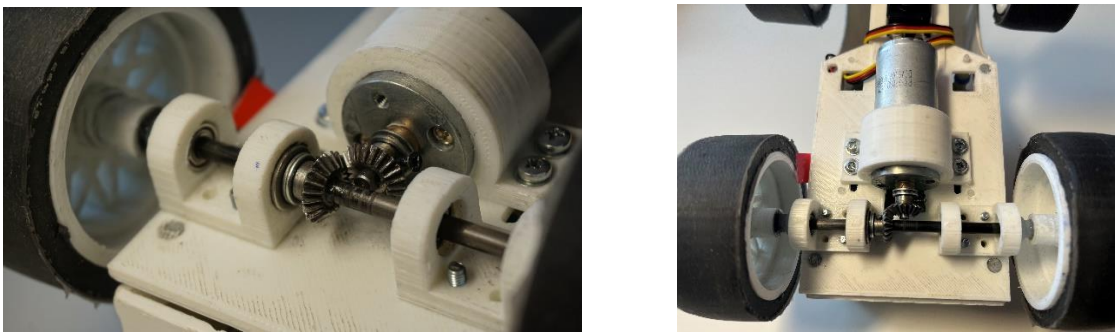
(a) (b) (c)

Joonis 4. Roboti kaamera koostud: (a) objektiivi kaitsevahend, (b) kaamera kinnitus kaamera jalaga, (c) kaamera jala kinnitus kerega.

3.1.2 Roboti veomootor

Mootorite valikul lähtusime TTÜ Robotiklubis olemasolevatest mootoritest. Esialgne otsus oli kasutada kahte veomootorit [10] tagumisel teljel, kuid testrajal proovimise käigus selgus, et mootoritel puudub 20-kraadise ülesmäkke sõidu korral piisav inerts ja veojõud.

Valitud mootoriks sai üksik mootor [11] (Joonis 5), mis oli samuti TTÜ Robotiklubis saadaval. Selle mootori eeliseks oli võimsus ja inerts, kuid mootori suuruse tõttu pidime kasutama ainult ühte veomootorit ning roboti kere põhja uuesti modelleerima. Kuna kasutusel oli ainult üks mootor, oli lisaks vaja juurde ehitada tagumisele teljele võll koos hammasrattaülekandega. Võll oli tehtud süsinikkiust, mis oli TTÜ Robotiklubist saadaval. Võll toetus laagritele, mis oli omakorda kinnitatud PLA-st prinditud laagrite kinnitustele (Joonis 5).

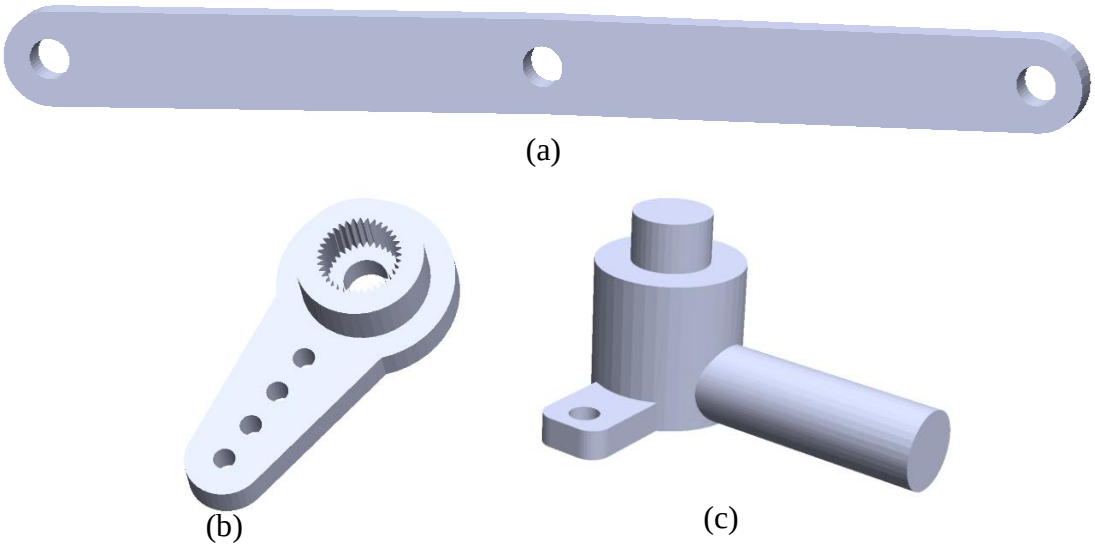


(a) (b)

Joonis 5. Roboti veomootori kinnitus ja ülekanne: (a) hammasrattaülekanne võllile, (b) mootori kinnitus.

3.1.3 Servomootor ja rattad

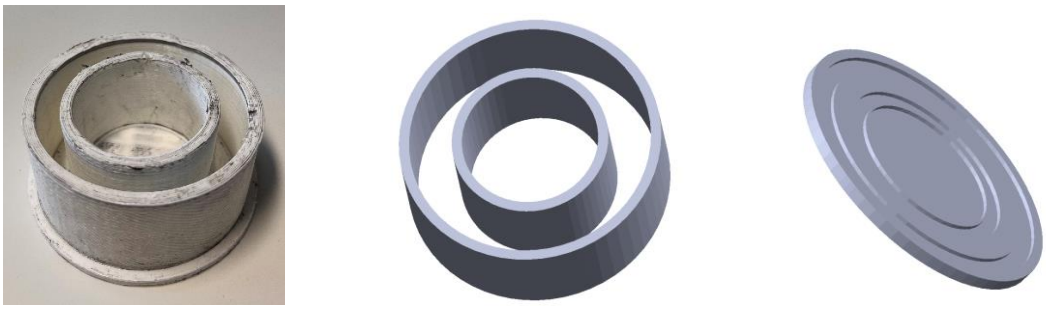
Roboti esiteljel paiknes servomootor, mille abil toimus esitelje pööramine rajal manööverdamiseks. Rataste jaoks 3D-prinditi esiratta kinnitused, mis olid ühendatud ratta ja esisilla laagrite külge. Servomootori kasutamiseks oli 3D-prinditud hoob, rataste ühendused ja ühendussild, mille abil ühendus servomootor ratastega (Joonis 6). Kõik servomootori jaoks prinditud ühendused olid üksteise külge kruvidega kinni keeratud (Joonis 8).



(a) (b) (c)

Joonis 6. Servomootori ja rataste ühendamise koostud: (a) hoobi ühendussild ratastega, (b) servomootori hoob, (c) ratta ühendus sillaga.

Kõik rattad olid prinditud PLA-st ja rataste rehvid olid valatud läbipaistvast silikoonist, kuhu oli lisatud värvipigment. Rehvide valamiseks koostati eraldi kestad, kuhu kuulus kaks rõngast ning alusplaat (Joonis 7). Kaks rõngast sobitusid alusplaadile, kus välisele ringile valades moodustusid ratastele õige suurusega rehvid.



(a) (b) (c)

Joonis 7. Kestad rehvide valamiseks: (a) valmis prinditud rehvide vorm, (b) kesta rõngaste mudel, (c) kesta alusplaadi mudel.

Servomootoriga ühendatud ratastel olid otstes laagrid, kuhu kinnitusid sillaga ratta ühendused (Joonis 8). Võlli küljes olevad rattad (Joonis 8) ühendati ja liimiti võlli külge.

(a) (b) (c)

Joonis 8. Roboti rattad: (a) tagumised võlli külge liimitud rattad, (b) servomootoriga ühendatud rattad, (c) servomootoriga ühendatud rattad peale komplekteerimist.

3.2 Roboti elektroonika

Selle peatüki eesmärk on kirjeldada roboti sardsüsteemi komponentide valikut. Elektroonika komponentide valikul oli siht valida komponendid, mida oleks võimalikult lihtne omavahel ühendada.

Selles peatükis oli komponentide valik piiratud TTÜ Robotiklubis oleva elektroonikaga. Vaatamata piirangule oli uuritud teisi võimalikke sardsüsteeme, juhul kui ehitada tulevates projektides töös käsitletud robotit teise sardsüsteemi põhjal.

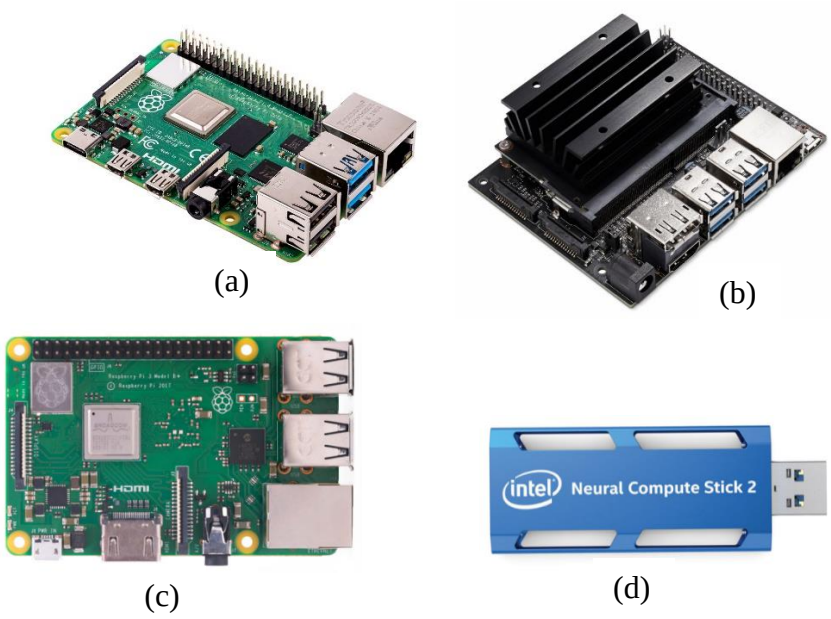
3.2.1 Sardsüsteemi tuum

Kõige olulisem riistvara komponent oli roboti sardsüsteemi tuum, mille peal töötas kogu programm. Tuum töötles ja juhtis kõikide teiste elektroonikakomponentide tööd ning samal ajal tegi autonoomseks sõitmiseks vajalikku pilditöötlust.

Sardsüsteemi tuuma oli võimalik valida mitmelt eri tootjalt, mis kõik erinesid võimekuse, sisendite, väljundite ja ühilduvuse poolest. Kõik valikus olnud alternatiivid on toodud allpool tabelis ja esitatud joonisel (Joonis 9). Kiibipuuduse tõttu ei ole tabelisse lisatud hinna veergu, kuna hind oli pidevas muutumises ja saadavust ei olnud [12].

Tabel 1. Sardsüsteemi tuuma valikud.

Sardsüsteem	**Protsessor**	**Videokaart**	**Mõõdud**	**Sisendid ja väljundid**	**Kaugühendus**
Raspberry Pi 4B	Cortex-A72 1.5 GHz	Broadcom VideoCore VI 500 MHz	85mm x 56mm x 16mm	Standard 40-pin GPIO header	2.4 GHz, 5 GHz WiFi, Bluetooth 5.0
NVIDIA Jetson Nano Developer Kit	Cortex-A57 1.43 GHz	128-core Maxwell 640 MHz	100mm x 80mm x 29mm	40-pin GPIO header	Puudub, kuid võimalik lisada
Raspberry Pi 3B+	Cortex-A53 1.4 GHz	Broadcom Videocore IV 300 MHz	85mm x 56mm x 16mm	Standard 40-pin GPIO header	2.4 GHz, 5 GHz WiFi, Bluetooth 4.2
Intel Neural Compute Stick 2	Movidius Myriad X VPU 700 MHz	Puudub	72.5mm x 27mm x 14mm	USB 3.1	Puudub



Joonis 9. Tabelis loetletud sardsüsteemid: (a) Raspberry Pi 4B, (b) NVIDIA Jetson Nano Developer Kit, (c) Raspberry Pi 3B+, (d) Intel Neural Compute Stick 2.

Tabelis (Tabel 1) on välja toodud roboti valmistamiseks kõige vajalikumad sardsüsteemi tuuma atribuudid: protsessor ja videokaart on elemendid, mis mõjutasid kogu masinnägemise programmi töökindlust ja kiirust, mõõdud olid olulised, kuna kogu roboti sardsüsteem pidi ära mahtuma 3.1.1 peatükis viidatud *Robotex International* reeglistikus olevatele mõõtudele. Ühendada oli vaja ka kogu elektroonika, mis vajasid mitmeotstarbelisi sisend- ja väljundviike. Sardsüsteemi tuumas kaugühenduvuse abil oli võimalik kaugjuhtida sardsüsteemi ning visuaalselt üle võrguühenduse kontrollida. Seda oleks saanud teha ka ilma kaugühenduseta, kasutades vastavat kaablit, kuid kuna tegu oli rajal sõitva robotiga, millele polnud võimalik alati videokaablit järgi panna, oli kaugühenduvus aja kokkuhoidmiseks oluline.

Esialgu sai roboti jaoks valitud Raspberry Pi 3B+ (Joonis 9), kuna autoril oli see juba eelmisest projektist olemas ning kasutusvalmis. See oli küll aeglasem esimesest kolmest ülalpool välja toodud sardsüsteemist, kuid piisav esialgseks algoritmi katsetamiseks. Roboti testimise jooksul avaldati soovi TTÜ Robotiklubile varuda ka Raspberry Pi 4B juhuks, kui Raspberry Pi 3B+ protsessor jääks programmi jaoks aeglaseks.

Nvidia Jetson Nano Developer Kit oli valikus olnud sardsüsteemidest kõige kiirema videokaardiga. Programmi loomisel kasutatav OpenCV platvorm pakub tuge

matemaatiliste operatsioonide delegeerimist videokaardile, mis kiirendaksid erinevaid etappe pilditöötluses [13]. Jetson Nano on hea valik isesõitva roboti ehitamiseks, kuid kiibipuuduse tõttu oli Jetson Nano sardsüsteemi kohalikelt tarnijatelt keeruline leida.

Lisaks on viimasena tabelis välja toodud (Tabel 1) Intel Neural Compute Stick 2 (edaspidi Intel NCS), mis erandlikult teistest eelnevalt mainutud süsteemidest ei oma mitmeotstarbelisi sisend/väljund viike. Selle lõputöö raames see kasutust ei näinud, kuid autor tõi selle välja, kuna see oleks hea võimalus edaspidiseks lõputöö arendamiseks. Intel NCS on mälupulga suurune arenduskomplekt tehisintellekti arendamiseks [14], mis võimaldaks lisada robotile objektituvastuse, et vältida võimalikke takistusi võistlusrajal sõites. Kuna Intel NCS'il puuduvad vajalikud ühendused muu elektroonika ühendamiseks, siis sobib see ehitatava roboti põhjal ainult masinõppe võimekuse lisamiseks.

Viimasena sai paigaldatud sardsüsteemi tuumaks Raspberry Pi 4B. Ilma korpuseta proovisõitu tehes lühistasid lahtised juhtmed Raspberry Pi 3B+ 5V ja 3.3V viigud omavahel kokku ning Raspberry Pi 3B+ lakkas töötamast.

3.2.2 Kaamera

Robotile valiti videokaameraks Raspberry Pi kaamera versioon 2.0 (Joonis 10), kuna seda oli lihtne ühildada *PIS-1683* ribakaabli abil sama tootja sardsüsteemidega. Kaamerale lisati lainurkobjektiiv, mis andis moonutatud perspektiiviga kujutise. Moonutus suurendas kaamera vaatevälja, et sõidurajal oleksid piirseinad laialt nähtaval.

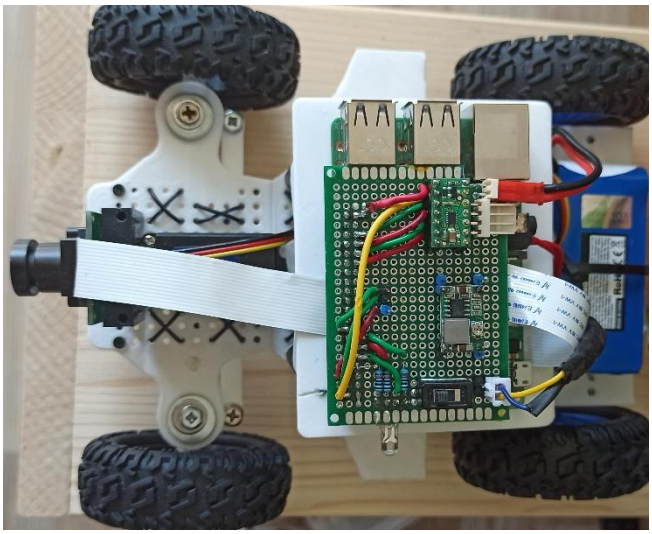
Joonis 10. Raspberry Pi 2.0 kaamera robotile kinnitatud ja ühendatud.

Roboti ehitamisel ei ole vajalik selles töös valitud kaamerat kasutada, selleks sobib ka USB (*Universal Serial Bus*) ühendusega kaamera. Autor oli eelnevas projektis kasutanud USB ühendusega kaamerat, kuid roboti ühilduvuse, kompaktsuse ja kaamera olemasolu tõttu oli kasutatud Rasperry Pi kaamerat.

3.2.3 Servomootor ja mootorite juhtmoodul

Roboti manööverdamiseks vajaliku servomootori valiku tingimusteks oli töökindlus ja lihtne ühilduvus. Servomootori ülesandeks oli keerata jäiga esitelje abil mõlemat esiteljel olevat ratast. Valituks servomootoriks sai Hitec HS-485HB [15], kuna see oli TTÜ Robotiklubis saadaval ning täitis kõik vajalikud tingimused.

Mootori juhtmooduli valikul lähtuti samadest tingimustest nagu servomootoril mainitud. TTÜ Robotiklubi poolt pakuti kahte juhtmoodulit: DRV8835 ja L298N. Need juhtmoodulid osutusid probleemseks, kuna nende kasutamiseks oli vaja eraldi trükkplaati joota, et komponent sardsüsteemi lisada ja lakkasid mitu korda töötamast teadmata põhjuse pärast. Trükkplaadi koostamine DRV8835 juhtmooduli jaoks osutus väga tülikaks ja aeganõudvaks protsessiks ning probleemide esinemisel vea tuvastamine oli väga keeruline (Joonis 11). Varem mainitud tingimuste põhjal kumbki nendest juhtmoodulitest ei sobinud ning otsustati leida alternatiiv.



Joonis 11. DRV8835 juhtmooduli ja teiste komponentide kasutamiseks koostatud trükkplaat.

Mootorite juhtmooduliks valiti Waveshare Motor Driver HAT [16], kuna selle jaoks ei olnud vaja teha eraldi trükkplaati ning ühilduvus Raspberry Pi sardsüsteemiga oli valikus olnud juhtmoodulitest kõige lihtsam. Selle juhtmooduliga tuli tootja poolt kaasa juhend, mis selgitas, kuidas seda paigaldada ja programmis kasutada [17].

3.2.4 Mälu

Raspberry Pi vajas operatsioonisüsteemi ja tarkvara hoiustamiseks mälukaarti [18]. Mälukaardiks sai valitud Samsung EVO mälukaart [19], mis oli autoril olemas ning lähtudes tootja poolt toodud nõuetest [18] sobilik.

Oli võimalik kasutada ainult mälupulka või mälukaardi ja mälupulga kombinatsiooni, kuid mälupulga kasutamine vajas rohkem ruumi, aga annaks robotile lisa mäluressurssi. Suurem mäluressurss ei olnud üheski programmi osas vajalik ning vajalikuks osutus ainult mälukaart.

3.2.5 Aku

Aku valik sõltus suurusest ja mahutavusest. Algul oli plaanis kasutada kahte eraldiseisvat toidet – üks sardsüsteemi jaoks ja teine mootoritele, kuid kuna mootorite juhtmooduli ühildus sardsüsteemi tuumaga oli otsene ning ühe liitiumpolümeeraku mahutavus oli piisavalt suur, et toita ära kogu robotil olev elektroonika vähemalt pooleks tunniks, piisas ainult ühest akust. Liitiumpolümeerakud Turnigy nano-tech 2000 mAh 2S (Joonis 12) [20] ja Turnigy 1600 mAh 2S [21] olid TTÜ Robotiklubis saadaval ning kiiresti taaslaetavad liitiumpolümeeraku [22] laadija kaudu.

Liitumpolümeerakud võivad osutuda ohtlikuks [23] ning liigse tühjenemise vältimiseks oli aku küljes robotiga sõitmise ajal alati ühendatud LiPo *alarm* (Joonis 12) [24], mis aku liigse tühjenemise korral annab helihäire. Heli teavituse läve saab muuta, kasutades LiPo *alarmi* peal olevat nuppu, mis seadistab heli teavituse pingeläve.

Lõputöös kirjeldatud robot kasutab ühte akut ning aku toitekaabel on ühendatud otse motoori juhtmoodulisse, et juhtmooduli kaudu toita ära kogu sardsüsteem.

(a) (b)

Joonis 12. Robotiga kasutatud LiPo alarm (a) (AmericanAirsoftClub) ja liitiumpolümeeraku (b) (Hobbyking)

3.2.6 Wi-Fi ja juhtpuldi ühendus

Raspberry Pi 3B+ ja 4B toetavad mõlemad 2.4 GHz ja 5 GHz traadita internetiühendust. Internetiühenduse peamine eesmärk oli võimaldada robotiga kaugühendust, et laadida alla programmis tehtud muudatusi ja jälgida sardsüsteemi programmi tööd otse SSH (*Secure shell protocol*) ja VNC (*Virtual network computing*) ühenduse kaudu.

Kohtvõrguühendus oli võistlusel alati 5 GHz sageduse peal, et võimalikult vähe segada Bluetooth suhtlust juhtpuldiga, mis toimub 2.4 GHz sagedusel.

Roboti mugavaks kaugjuhtimiseks oli kasutuses Playstation DualShock 4 [25] juhtpult, kuna sellel oli Bluetooth protokolli kasutades väga täpne analoogsisend ning lihtne ühildumine sardsüsteemiga.

3.2.7 Muu elektroonika

Lisaks kaamerale kasutati ka ultraheliandurit ja valiti ultrahelisensoriks HC-SR04 [26]. Ultrahelisensori jaoks oli TTÜ Robotiklubi andnud korpuse (Joonis 19).

Robotile oli lisatud tavaline RGB diood, et seda programmis kasutada tagasiside andmiseks. Näiteks: diood põles roheliselt, kui robot oli autonoomne ja diood põles punaselt, kui roboti juhtimine oli manuaalne.

Kuna autoril puudus varasem kogemus güroskoopanduriga ja kiirendusmõõturiga, ei ole kasutatud neid selle roboti ehitamisel, kuid oleks hea võimalus edasiarendamiseks, et tuvastada õiget sõidusuunda ja vältida kokkupõrkeid.

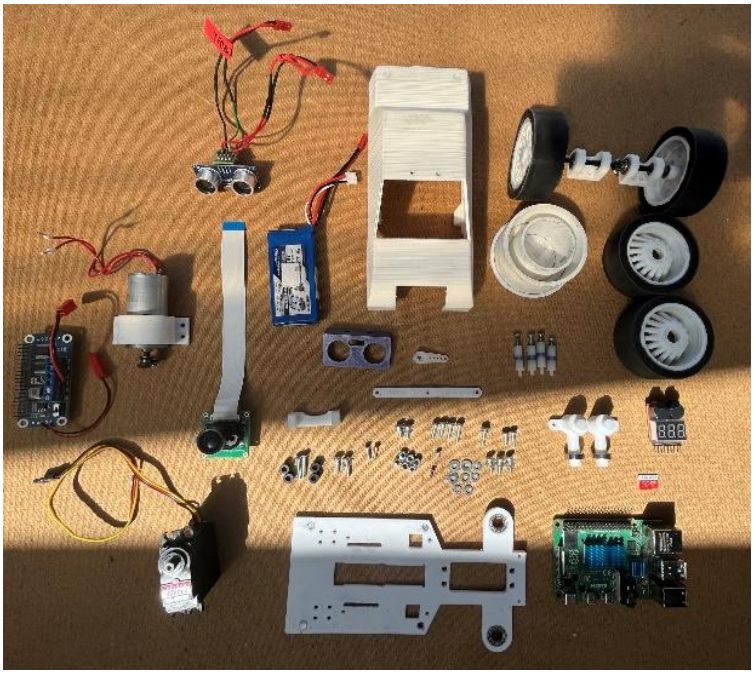
3.3 Roboti ehitamine

See alampeatükk täpsustab, kuidas on kõik ülalpool mehaanika- ja elektroonikapeatükkides kirjeldatud koostud ning komponendid omavahel kokku pandud ja ühendatud. Komponentide kokkupanek ja ühendused on toodud kahes eraldi alampeatükis.

Roboti komplekteerimine algas peale koostude printimist. Viimasena, peale roboti komplekteerimist, ühendati elektroonika komponendid.

3.3.1 Koostude kokkupanek

Kõik komponendid roboti kokkupanekuks on toodud joonisel (Joonis 13). Osad komponendid olid varasemalt kokku liimitud ja komplekteeritud: mootori kinnitus mootoriga, korpuse ja kere küljes olevad magnetid, tagumine telg koos telje kinnituse, võlli ning ratastega.

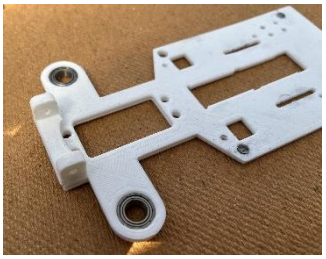


Joonis 13. Roboti komponendid.

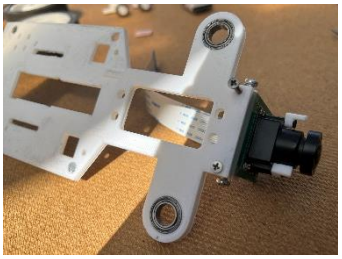
Roboti komplekteerimiseks ei pea järgima täpselt käesolevas töös toodud järjekorda. Komponentide kinnitamiseks oli kasutatud tööriistu, mis olid TTÜ Robotiklubis saadaval.

Esimesena kinnitati kaamera külge kaamera tagus ning kaamera jala kinnitus, seejärel paigaldati kaamera jalg roboti kere külge (Joonis 14). Kaamera jala kinnitamisel kere külge tuli kontrollida, et jala kaar jääks kere poole, muidu ei olnud võimalik servomootorit mõlemalt poolt kinnitada.

(a)



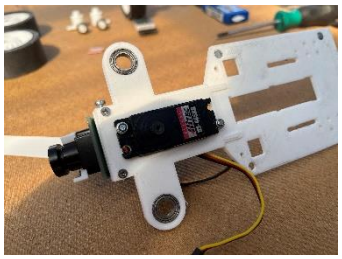
(b)



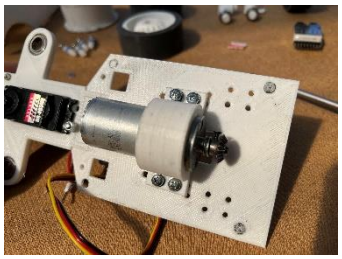
(c)

Joonis 14. Kaamera kinnitus: taguse ja jala kinnitus kaameraga (a), kaamera jala kinnitus (b), kaamera kinnitus kerega (c).

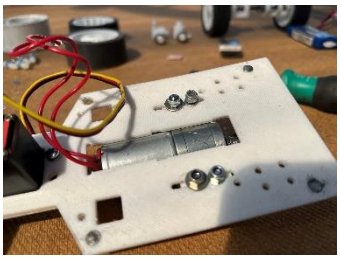
Järgmisena olid lisatud mootorid (Joonis 15). Veomootor oli kinnitusega kokku liimitud, et mootor oleks paremini kinnitatud. Kruvid veomootori kinnitamiseks tuli kinnitada kere alt, et vajadusel mootorit nihutada, kuna mootori kohale tuli sardsüsteemi tuum.



(a)



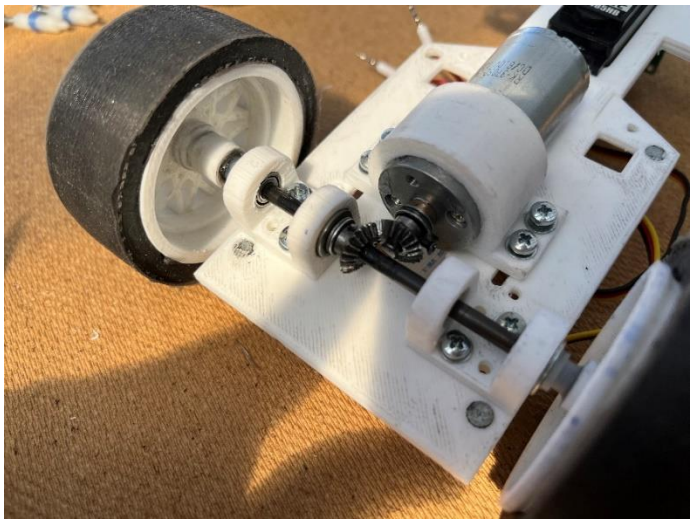
(b)



(c)

Joonis 15. Mootorite kinnitused kerega: servomootori kinnitus (a), veomootori kinnitus altvaatest (b), veomootori kinnitus pealtvaatest (c).

Peale veomootori kinnitamist oli lisatud võll koos ratastega (Joonis 16). Võlli keskele kinnitati hammasratas, mis paigutati veomootori hammasrattaga risti. Rattad liimiti võlli otstesse ning võll asetati võlli hoidjates olevatesse laagritesse.



Joonis 16. Võlli kinnitus kere külge.

Seejärel kinnitati esirataste kinnitused kere küljes olevatesse laagritesse ning nende vahele servomootori hoobi kinnitus (Joonis 17). Servomootori hoob kruviti servomootori enda ja kinnituse külge. Seejärel lisati esirataste kinnituste laagrite külge rattad.

(a) (b)

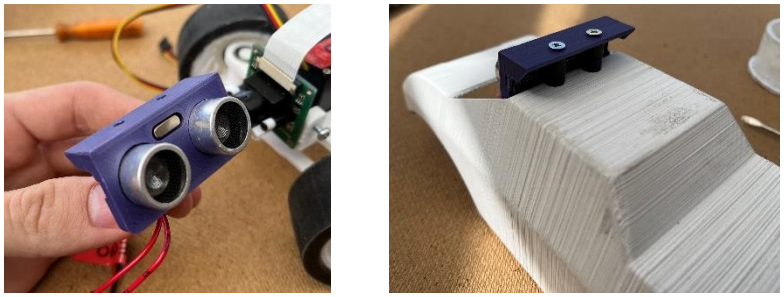
Joonis 17. Esitelje komplekteerimine: servomootori hoobi kinnitus (a), esirataste kinnitus (b).

Sardsüsteemi tuuma lisamiseks kinnitati kere külge tõstejalad. Tõstejalgade peale kruviti Raspberry Pi 4, mille peale kinnitati mootorite juhtmoodul (Joonis 18).

(a) (b) (c)

Joonis 18. Sardsüsteemi tuuma ja juhtmooduli paigaldamine: jalgade kinnitus (a), Raspberry Pi 4 kinnitus (b), juhtmooduli kinnitus (c).

Viimasena lisati ultrahelisensorile korpus ning kinnitati see roboti korpuse külge (Joonis 19). Lõpuks ühendati kõik elektroonika komponendid sardsüsteemiga, nagu on toodud allpool olevas peatükis või muudatuste korral vastavalt programmile. Korpuse kinnitamiseks kerega magnetite puudumisel võib puurida augud läbi magnetite kinnituskohtade, et oleks võimalik korpus kere külge kruvidega kinnitada.

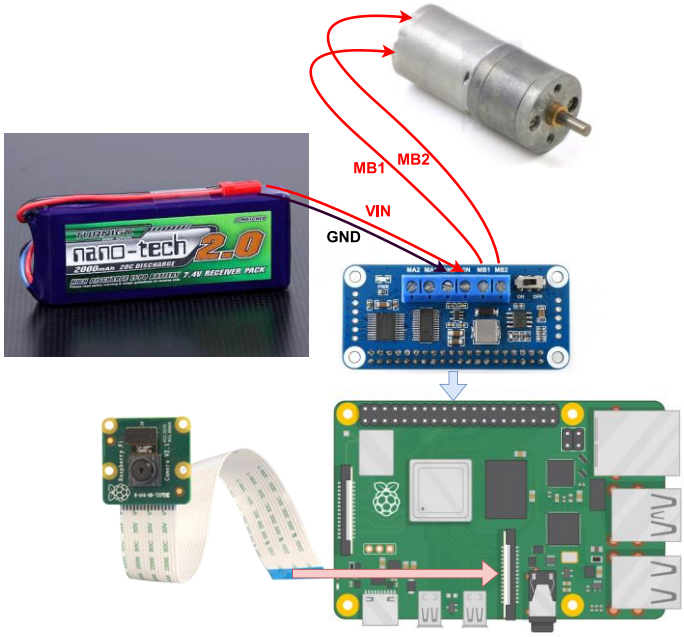


(a) (b)

Joonis 19. Ultrahelisensori komplekteerimine: sensori korpus (a), sensori kinnitus roboti kerega (b).

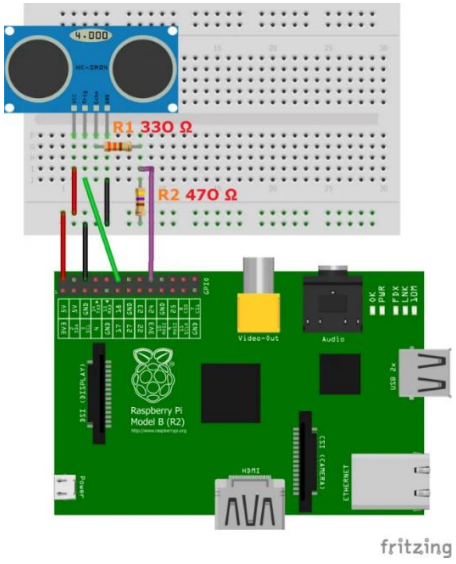
3.3.2 Elektroonika paigaldamine ja ühendamine

Raspberry Pi ja juhtmoodul ühendusid üksteistega otse, kinnitudes juhtmooduli Raspberry Pi sisend- ja väljundviikude külge. Kaamera oli ühendatud sardsüsteemi tuuma kaamera siini (Joonis 20). Juhtmooduli külge lisandus aku ning juhtmooduli viikudest oli juhtmetega ühendatud kõik muu elektroonika.



Joonis 20. Kaamera, Raspberry Pi, juhtmooduli, aku ja mootori ühenduste skeem sardsüsteemis.

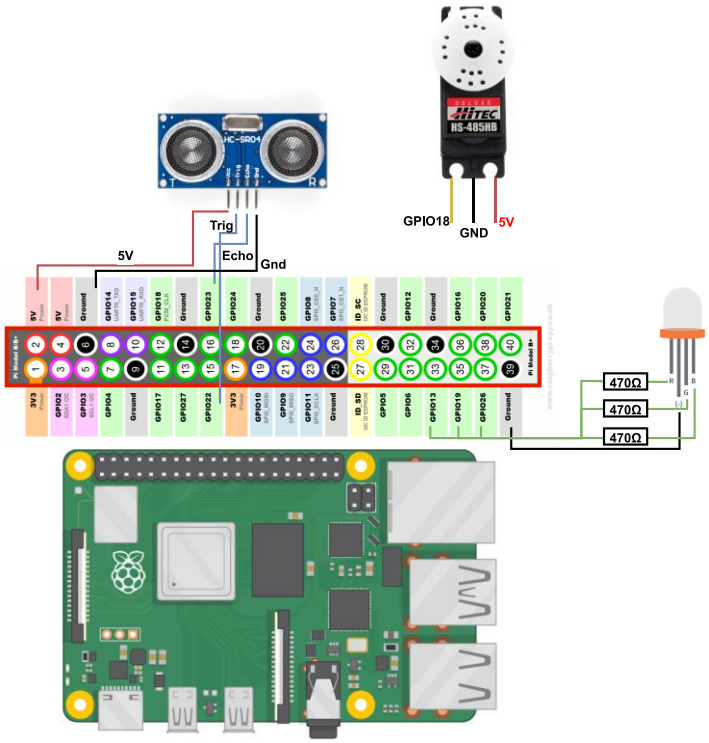
Ultrahelisensori jaoks oli vaja eraldi trükkplaati, kuna sensori loogika töötas 5V pingega [26], aga Raspberry Pi 3.3V pingega [27]. Raspberry Pi loogikaga ühendamiseks oli vaja alandada ultraheli sensorist tulev 5V Raspberry Pi'le sobivaks 3.3V, mis saavutati kasutades kahte takistit (Joonis 21). Trükkplaadi koostamiseks kasutati [28] juhendit.



Joonis 21. Ultrahelisensori skeem Raspberry Pi'ga ühendamiseks.

Servomootori jaoks valiti GPIO (*General-Purpose input/output*) 18, mille kasutus täpsustati peatükis 4.2.3. Ultrahelisensori, servomootori ja RGB dioodi ühendused sardsüsteemis on toodud joonisel (Joonis 22). Joonisel pole välja toodud ultrahelisensori

trükkplaadi ühendusi, kuna ülalpool oleval joonisel on need olemas. Ultrahelianduri *Trigger* ja *Echo* GPIO viigud valiti juhuslikult ning on isesõitvas programmis defineeritud kui GPIO 23 ja GPIO 22.



Joonis 22. Ultrahelisensori, servomootori ja *RGB* dioodi ühendused sardsüsteemis.

4 Tarkvara

See peatükk kirjeldab kogu tarkvara paigaldamist ning loomist. Loodud tarkvara on vajalik sardsüsteemi kasutamiseks, kaamerapildi töötlemiseks, juhtpuldiga ühendamiseks ja dioodi vilgutamiseks.

Isesõitva programmi algoritm põhineb HSV värviruumis raja tuvastuse kursusest [29]. Kursusel olev Pythoni kood on võimalikult palju ümber kirjutatud C++ programmeerimiskeelde, kuna kooli õppekavast on autor tuttavam C++ programmeerimiskeelega ja peatükis 4.4 väidete põhjal on C++ pilditöötlusel kaks korda kiirem.

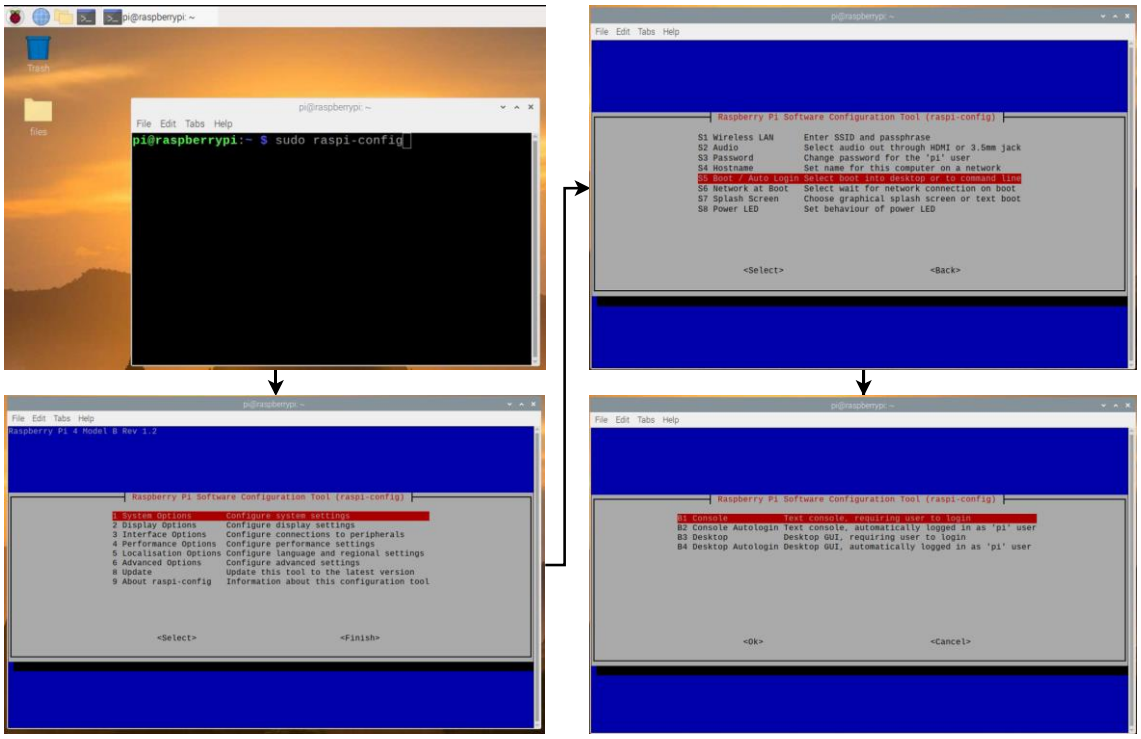
4.1 Sardsüsteemi seadistamine programmi tööks

Lisaks programmi tööle oli vaja võimaldada programmi ja elektroonika vaheline suhtlus, millel on mitu erinevat etappi. Esialgu oli vaja paigaldada ning seadistada Raspberry Pi operatsioonisüsteem ning seejärel alla laadida ja kompileerida OpenCV pilditöötluse teek.

Selle töö jaoks on kasutatud sardsüsteemi tuumal Rasperry Pi enda operatsioonisüsteemi. Raspberry Pi’le sobivad ka teised operatsioonisüsteemid, nagu näiteks Ubuntu [30], kuid kuna autor pole seda operatsioonisüsteemi varem kasutanud koos valitud sardsüsteemiga, ei ole seda kasutatud selle töö raames.

4.1.1 Operatsioonisüsteemi paigaldamine ja seadistamine

Raspberry Pi OS on spetsiaalselt Raspberry Pi jaoks loodud, mis andis seadistamiseks vajaliku operatsioonisüsteemi [8]. Operatsioonisüsteem sai paigaldatud koos graafilise liidesega, mida on võimalik sisse- ja väljalülitada. Graafilise liidese sisse- ja välja lülitamiseks saab kasutada käsuakent, sisestades vastava käsu ning navigeerides avanenud menüüd nagu on joonisel (Joonis 23) demonstreeritud.



Joonis 23. Graafilise liidese sisse- ja väljalülitamise juhend.

Operatsioonisüsteemi paigaldamiseks laeti operatsioonisüsteemi pilt mälukaardile, milleks kasutati Raspberry Pi Imager tarkvara [13]. Peale seda lisati traadita võrguühenduse konfiguratisoon [31] ning SSH kaugühendus [32] Raspberry Pi'le järgides tootja poolt vastavaid juhendeid. Sisestades mälukaardi sardsüsteemi tuuma, andes arvutist toite, ühendades ekraani ja klaviatuuri, paigaldati operatsioonisüsteem järgides ekraanil olevaid juhiseid.

Peale operatsioonisüsteemi paigaldamist tuli aktiveerida vajalikud seaded kaamera [33] ja valitud mootorite juhtmooduli I2C (*Inter-integrated circuit*) [34] liidese ühendamiseks. Lisaks SSH kaugühendusele võeti kasutusele programm VNC Connect [35], et kontrollida programmi tööd ning saada graafilist pilti otse Raspberry Pi'lt. Arvutist pildi jälgimiseks kasutati VNC Viewer'it [36].

4.1.2 OpenCV paigaldamine

OpenCV on käesolevas töös peamine tööriist masinnägemise algoritmi loomisel. See on avatud lähtekoodiga reaalaja aplikatsioonide teek, mis toetab C++, Python ja Java programmeerimiskeeli [3].

Teegi paigaldamisel lähtuti [37] juhendist. Juhendi järgi laeti alla kõige uuem teegi versioon ning kompileeriti sardsüsteemi tuuma peal. Juhendi järgi toimus Raspberry Pi nelja peal kompileerimine kasutades kõiki tuumi. Kasutades Raspberry Pi nelja tuuma, võttis autoril kompileerimise protsess umbes kaks tundi. Selle lõputöö raames käsitletava isesõitva programmi jaoks ei ole vajalikud Pythoni jaoks alla laetavad laiendused, näitekoodid, TBB [38] ja Qt [39], kuid võivad osutuda kasulikuks, kui proovida uusi masinnägemise lahendusi, katsetada näidisprogramme otse arendataval platvormil või optimeerida programmi paralleeltöötlust.

4.2 Programmi loomine

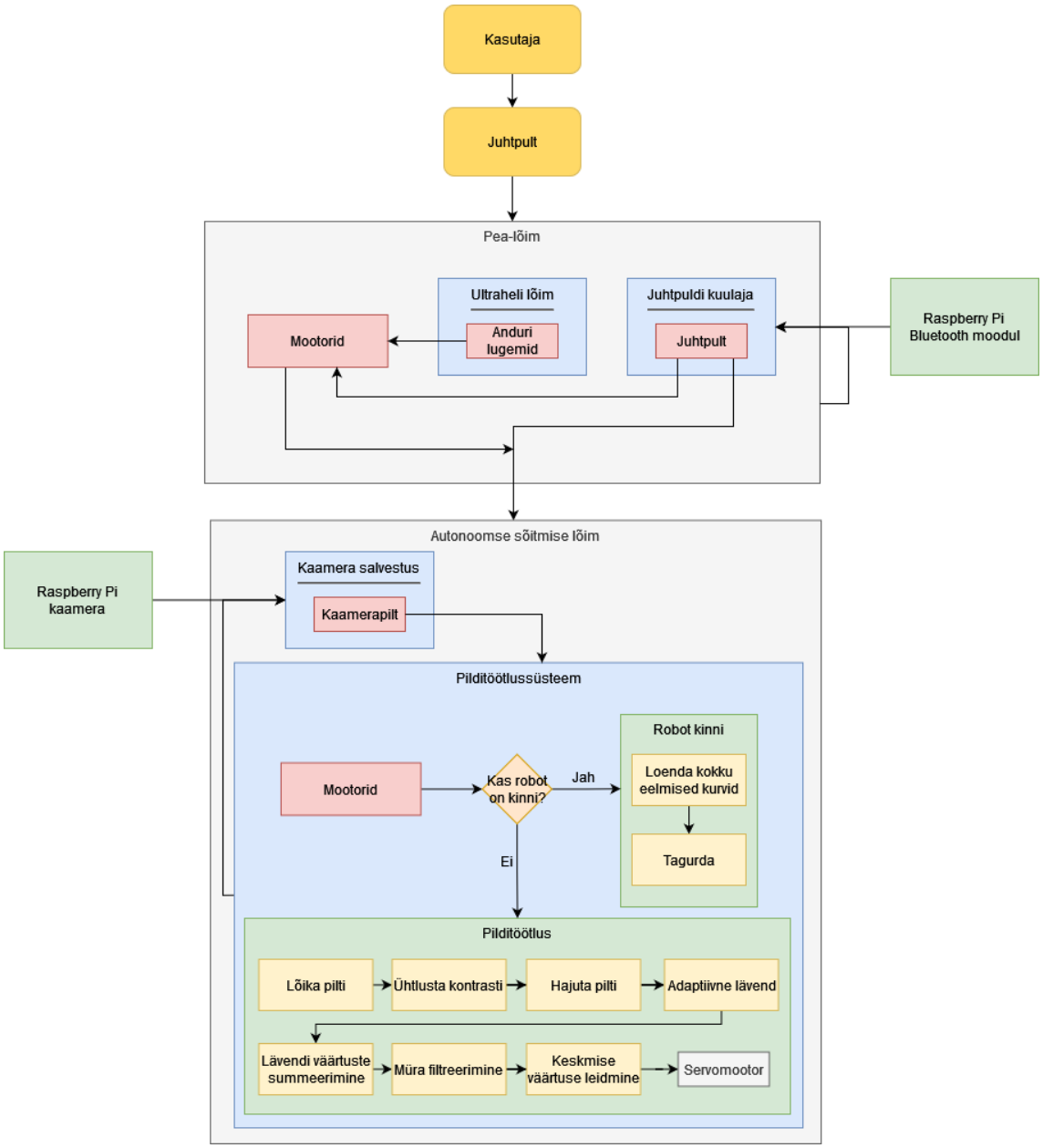
Roboti põhiprogramm jaguneb mitmeks etapiks, mis on välja toodud selles peatükis. Esimesena alustati kaamerapildi töötlusega, et tööle saada osa programmist, mis ei sõltu sellest, kus arvutis seda arendatakse ehk ei oleks pidevat vajadust roboti sardsüsteemi kasutamiseks.

Programmi loomisel on kasutatud *git* versioonihaldustarkvara [40] ning repositooriumina *GitHub*'i [41]. Arendus on toimunud kahes harus ehk põhiharu, kus on viimane töökindel kood, millega saab robotit igal hetkel testida ning arendusharu, milles toimus uue koodi testimine. Kui arendusharus probleeme ei esinenud, liideti arendusharu põhiharuga kokku.

Koodi kompileerimiseks kasutati koosteprogrammi *make* [42]. Selle jaoks loodi fail nimega *makefile*, kus on koosteprogrammi jaoks kirjutatud juhendid, kuidas programmi kompileerida. Koosteprogrammi kasutamiseks tuli eelnevalt kontrollida, et kõik kirjeldatud *makefile* sisu vastaks kasutatava süsteemi juurstruktuurile.

4.2.1 Programmi struktuur

Üldine programmi struktuur on toodud joonisel (Joonis 24). Joonisel on esitatud, kuidas on programm jaotatud eraldi osadeks ning millises programmi osas toimub autonoomne sõitmine ja juhtpuldilt signaali saamine. Juhtpuldilt signaali saamiseks on kasutusel Raspberry Pi Bluetooth moodul.



Joonis 24. Roboti jaoks loodud masinnägemise programmi struktuur.

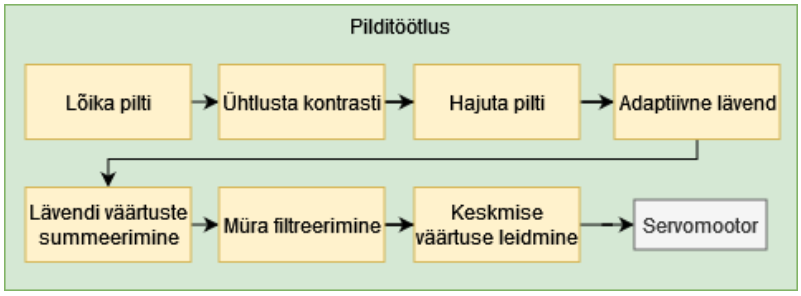
Roboti jaoks loodud masinnägemise programm algab, luues mootori, juhtpuldi ja dioodi objektid. Dioodi tuli pannakse põlema punaselt, et anda märku programmi käivitumisest. Ultrahelisensori töö jaoks luuakse eraldi lõim, millele antakse kaasa mootori objekt, et vajadusel lisada funktsionaalsust või piirata mootori kiirust. Peale ultrahelisensori lõime loomist jääb programm juhtpuldilt sisendit kuulama.

Vastava juhtpuldi sisendi korral loob peamine-lõim autonoomse sõitmise lõime ning seadistab dioodi siniselt põlema. Autonoomse sõitmise lõime loomisel initsialiseeritakse vajalikud muutujad ning sisestatakse mootorile kiirus. Sõitmise ajal loetakse massiivi viimase kahesaja servomootori pööramisnurga väärtused ning selle massiivi põhjal

otsustab robot, mis suunas on vaja välja tagurdada kinnijäämise korral. Ülejäänud autonoomse sõitmise lõime töö on kirjeldatud allpool olevas peatükis.

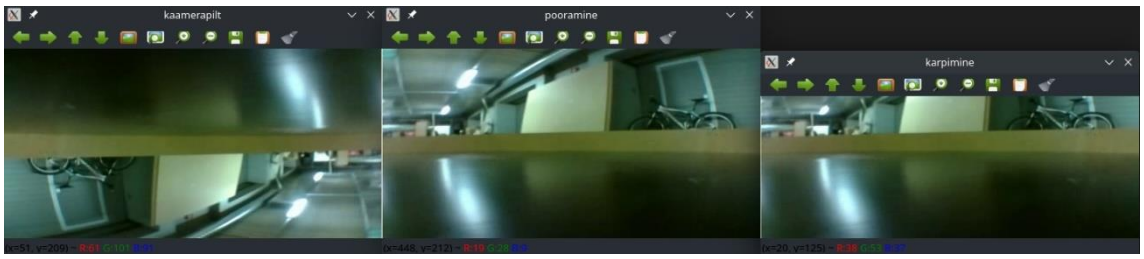
4.2.2 Kaamerapildi töötlus

Pilditöötluse algoritm on toodud programmi struktuurist suurendatuna joonisel (Joonis 25), kus on kirjeldatud üldistatult pilditöötluse järjekord. Pilditöötlus asub eraldi failis nimega detection.cpp.



Joonis 25. Pilditöötluse protsessi järjekord.

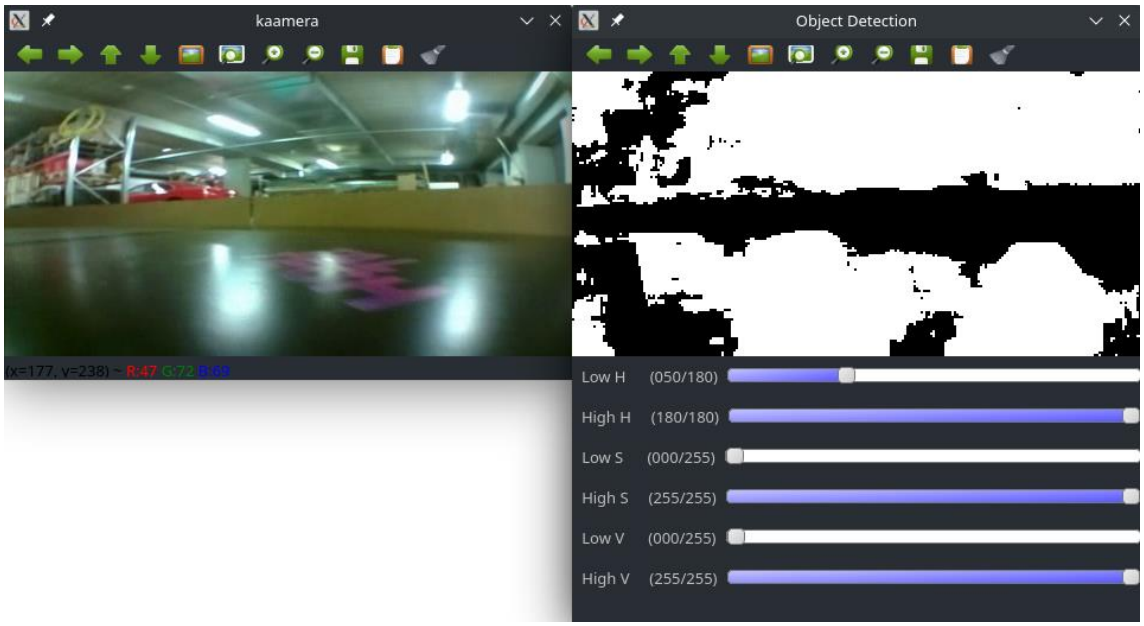
Kaamerapildi töötlus algab kaamerapildi võtmisega kaamerast ning eeltöötlusega maapinna tuvastamiseks (Lisa 3). Kaamerapilt võetakse, luues muutuja Mat, mis sisaldab endas viidet n-mõõtmelisele massiivile, kuhu salvestatakse kõik videokaamerast tulevad pikslid *BGR* formaadis, mis on võetakse OpenCV kaamera objektist [43]. Loodud muutujast kärbiti välja pildi osa, mis jääb raja vaateväljast välja, et vähendada pilditöötluse mahukust. Enne kärpimist toimub kaamerapildi otseks pööramine, kuna kaamera on roboti ette paigutatud tagurpidi. (Joonis 26). Viimaseks on kasutatud Gauss hägustamist [44], et vähendada pildile jäänud müra ning detaile [45]. Robotiga testimise käigus tekkis vajadus müra vähendamiseks veel ühele eeltöötlusprotsessile, mis on kirjeldatud peatükis 5.



Joonis 26. Kaamerapildi pööramine ja kärpimine programmis.

Pilditöötluse esimene samm on sõiduraja maapinna leidmine. Maapinna leidmiseks kasutati HSV värviruumi, et määrata maapinna tuvastuslävend. Lävendi määramiseks

kasutati abiprogrammi, mis on täpsemalt kirjeldatud punktis 6.2. Seades HSV värviruumile lävendi, on sellega võimalik eristada maapinda teistest objektidest. Lävendi lisamisest saadakse binaarne pilt, kus on ainult mustad ja valged pikslid (Joonis 27).



Joonis 27. Maapinna tuvastuse jaoks lävendi leidmine.

Varem saadud binaarselt pildilt, kus valge piksli väärtus on 255 ja musta piksli väärtus on 0, kasutatakse kurvi leidmiseks. Selle jaoks liidetakse kokku igas veerus olevad väärtused omavahel, kasutades selle jaoks OpenCV funktsiooni *reduce* [46], mis annab vastuseks ühe rea, kuhu on summeeritud kõik binaarse pildi veerud. Summeritud reast saab moodustada histogrammi, mille visualiseerimisel saab liita kaamerast tuleva pildi kaamera pildiga, et saada visuaalselt kinnitada, kas maapinna tuvastus on maapinnaga ühtne (Joonis 28). Varem leitud summeeritud reast leitakse kõige suurema väärtusega veerg ning selle veeru väärtuse põhjal filtreeritakse välja ülejäänud reast müra, mis ei ületa suurima väärtuse poolt loodud piirväärtust. Lävendi kordajaks on võetud 0.8, kuna robotiga sõitmise käigus osutus see lävend kõige täpsemaks.

(a) (b)

Joonis 28. Histogrammi visualiseerimine koos keskmise väärtusega (kollane ring): kaamerapilt koos histogrammiga (a), histogramm üksi visualiseerituna (b).

Viimaseks arvutab programm eelnevalt summeritud ja filtreeritud vektorist keskmise väärtuse ehk liidetakse kõik vektori väärtused kokku ning jagatakse vektoris olevate liikmete arvuga. Saadud keskmine edastatakse servomootorile peale eeltöötlust.

4.2.3 Mootorite jutimine

Mootoritega suhtlemiseks on loodud eraldi C++ fail nimega motors.cpp. See fail kasutab oma mootoritele käskluste andmiseks kolme abifaili: DEV_Config.cpp, MotorDriver.cpp ja PCA9685.cpp, mis on juhtmooduli tootja poolt koostatud [17].

Fail motors.cpp mis sisaldab klassi nimega *Motors*, mille abil saab mootoreid initsialiseerida, seadistada mootori kiirust ja juhtida servomootorit. Veomootorile kiiruse andmiseks genereeritakse riistvara peal PWM (*Pulse-width modulation*) signaal, kasutades juhtmoodulil olevat PCA9685 kiipi [17]. Kasutades pigpio teeki, antakse PWM juhtsignaal servomootorile. Lisaks PCA9685 kiibil PWM signaali genereerimisele, on võimalik genereerida sardsüsteemi tuuma peal tarkvaralise PWM signaali või riistvaraga ajastatud PWM signaali. Olenedes, mis viigu külge on servomootor ühendatud, oskab pigpio automaatselt valida PWM signaali generatsioonitüübi [47].

4.2.4 Programmi automaatkäivitus

Programmi automaatkäivitus vajadus tekkis siis, kui puudus võimalus luua kaugühendus sardsüsteemiga. Automaatne käivitus võimaldab juhtpuldiga robotit juhtida ilma arvutiga kaugühenduse loomiseta, mis teeb roboti käivitamise ja programmi testimise lihtsaks ka nendele, kes ei ole robotiga tuttavad, kuid soovivad robotit juhtida.

Programmi automaatseks käivitamiseks on loodud Pythoni programm, mis ootab kuni juhtpult ühendub sardsüsteemi operatsioonisüsteemiga. Pythoni programm käivitatakse automaatselt, kasutades Linux'is olevat cron töö planeerijat, mis on Raspberry Pi operatsioonisüsteemis tagataustal olev protsess, mis käivitub automaatselt peale operatsioonisüsteemi [48]. Cron protsessi konfiguratsiooni on lisatud varem mainitud Pythoni programm.

Pythoni programmi sisuks on oodata, kuni juhtpult ühendub sardsüsteemiga. Peale puldiga ühendumist käivitatakse roboti isesõitmise programm ning Pythonis kirjutatud programm sulgub. Automaatse käivitamise programm on kirjutatud Pythonis lihtsuse tõttu, kuna see ei vaja eraldi kompileerimist, seda programmi on lihtne muuta ning Python sisaldub koos Raspberry Pi operatsioonisüsteemiga. Selle programmi kasutamiseks teise juhtpuldiga on vaja muuta programmi koodis olev juhtpuldi MAC *(Media access control address)* aadress, et see ühilduks uue juhtpuldiga. Enne selle programmi kasutamist on vaja vastava juhtpult vähemalt ühe korra sardsüsteemi tuumaga ära ühendada, et salvestada selle juhtpuldi ühendus.

4.2.5 RGB diood ja ultrahelisensor

Robotile on lisatud RGB diood, mida kasutab isesõitmise programm märguande andmiseks. Autor kasutas märguandetuld, et roboti kasutajat ja pealtvaatajaid teavitada, millal robot sõidab autonoomselt ning millal juhitakse robotit manuaalselt.

RGB tule jaoks on kasutusel kolm GPIO viiku, mille kaudu edastab programm signaale dioodile. Dioodi ühendus sardsüsteemiga on täpsustatud alampeatükis 3.3.2. Programmis dioodi kasutamiseks on loodud led.cpp fail, kus on autori poolt defineeritud kolm värvi: punane, roheline, sinine. Programmi poolt kasutatavad viigud on defineeritud led.hpp failis.

Ultraheli sensorit ei olnud alguses kavas kasutada, kuid kuna ainult kaameraga sõites jäi robot pidevalt takistuste taha kinni, otsustas autor lisada ultrahelisensori, et tuvastada roboti kaugust objektidest. Ultrahelisensori kood [28] on autori poolt kirjutatud ümber C++ programmeerimiskeelde ning asub kahes failis: sonic.cpp, mis sisaldab ainult ultrahelisensori koodi ja main.cpp, kus on sensor roboti programmi integreeritud.

4.2.6 Juhtpuldi kasutamine

Sardsüsteemi Bluetooth mooduliga ühendatud juhtpuldi sisendi lugemiseks on kasutatud avatud lähtekoodiga C++ faili, mis on modifitseeritud isesõitva programmi jaoks. Kuna kasutatud fail on üldine ning peaks oskama lugeda iga juhtmevaba puldi sisendit, on selle faili kasutamiseks loodud eraldi juhtpuldi tootja spetsiifiline fail. Loodud failis on muutuja klass nimega *Controller*, mis defineerib saadud lugemid, et need ühilduksid Dualshock 4 juhtpuldi nuppude ja käskudega.

Isesõitva programmi juhend koos juhtpuldiga kasutamiseks on toodud joonisel (Joonis 29). Enne juhtpuldi kasutamist on vaja ühildada juhtpult Raspberry Pi süsteemiga, nagu on kirjeldatud peatükis 5.2.4. Pulti võib konfigureerida vastavalt vajadusele, lisades või muutes tingimusi *switch* käskluses main.cpp failis. Dualshock 4 puldist erineva puldi kasutamiseks tuleks muuta controller.hpp ja joystick.cpp failid panna vastavusse puldi sisendiga.

Joonis 29. Juhend Dualshock 4 puldi kasutamiseks robotiga.

4.3 Programmi lõimedeks jagamine

Roboti ehitamise ja testimise käigus lisandus aina enam programmiga juhitavaid komponente, mistõttu suurenes ka programmi keerukus. Kõikide komponentidega toimus suhtlus ühes protsessis, kus kuulati juhtpuldi signaale, loeti ultrahelisensori lugemeid, juhiti mootoreid, töödeldi kaamerapilti ning toimis kogu ülejäänud programmi töö. Juhul kui mõni komponent lakkas töötamast, kas elektroonika probleemi tõttu või katkes

juhtpuldiga ühendus, jäi kogu programm ootele või lõpetas juhtpuldile reageerimise. Probleemi lahendamiseks on programmi etapid jagatud eraldi lõimedeks.

Põhiprogrammist eraldi protsessideks on jaotatud ultrahelisensori kasutamine ja autonoomne sõitmine koos pilditöötlusega. Kõikide protsesside vahel on jagatud üks viide mootorite objektile, mis luuakse enne lõimede tekitamist. Funktsioonide lõimedeks jagamine toimub kasutades C++ *thread* teeki [49]. Ultrahelisensori lõim alustab enne juhtpuldi kuulamise algust ning jääb tsüklisse. Sensori lõime lõppu on lisatud paus, et vältida sensori ülekoormust. Autonoomse sõitmise lõim algab, kui juhtpuldilt saadakse vastav signaal ning lõppeb, kui muudetakse mootorite objekti juhtimistüüpi. Mootoritele käskluste andmiseks kasutavad kõik lõimed mootori objekti viidet, millele on loodud käskude täitmiseks vastavad funktsioonid.

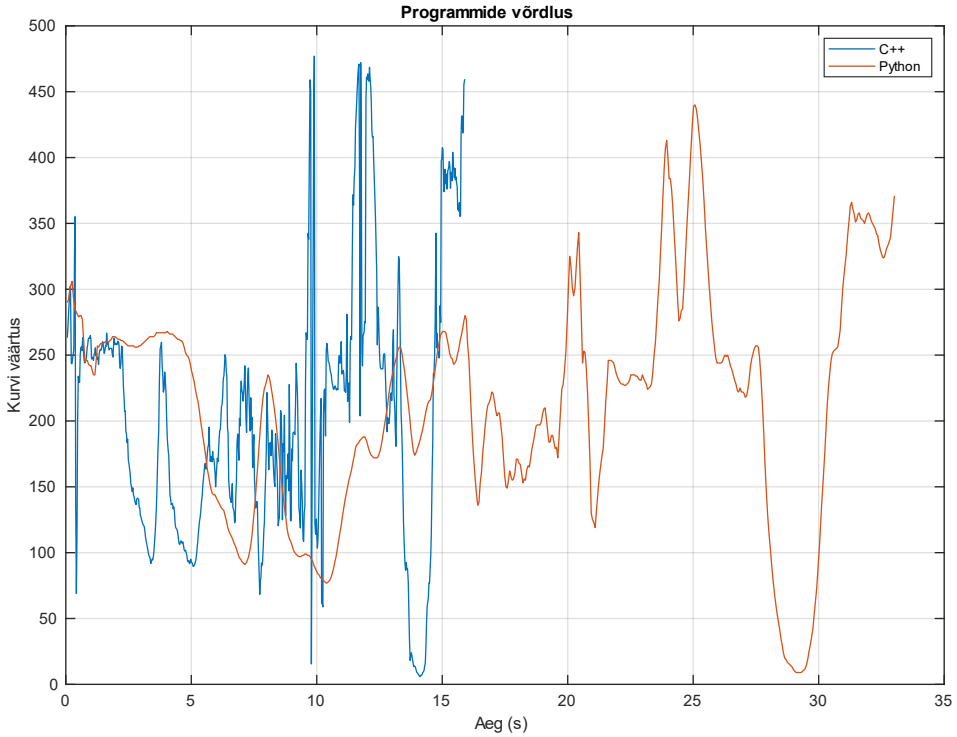
4.4 Raja tuvastuse metoodikad ja programmeerimiskeelte erinevused

Lõputöö autor on varem katsetanud masinnägemisega võistlussõidu roboti ehitamist, kuid mitte edukalt. Varasem projekt põhines maapinna ja seinte äärte tuvastusel, kasutades selleks *Canny* ääretuvastus algoritmi ja *Hough Line Transform* sirgete tuvastamist [50], [51]. Autori poolt tehtud varasemas projektis kasutatud sardsüsteemi tuum Raspberry Pi 3B+ ei olnud piisavalt võimas, et olla võistlusel kiirusega konkurentsivõimeline. Eelneva projekti kood ja lühitutvustus on avalikult kättesaadavad [52], [53].

Eelneva projekti tõttu otsustas autor kirjutada uue roboti programmi C++ programmeerimiskeeles, kuna internetist leitud allikate põhjal on Pythoni programmeerimiskeel OpenCV raamistikus aeglane [54], [55], [56]. Väidete kontrollimiseks tegi autor mõlema programmeerimiskeele puhul kiirustestid.

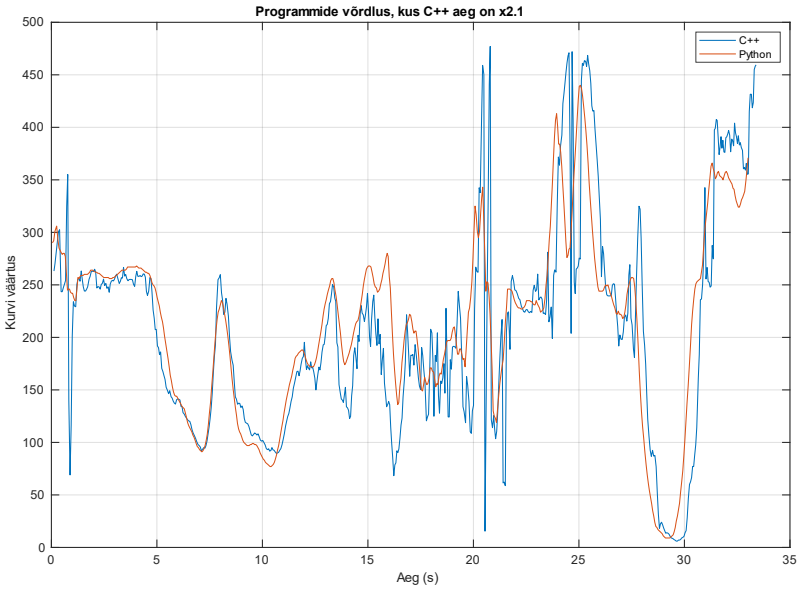
Peale programmi osaliselt ümberkirjutamist C++ programmeerimiskeelde, on läbi viidud programmide kiiruste võrdlused. Programmide lähtekoodid ei ole täpselt vastavuses, kuid kasutatavad algoritmid ja meetodid on ühtivad. Kiiruste mõõtmiseks on kasutatud programmeerimiskeelte omi teeke ja OpenCV teegis olevaid vahendeid [57], ning mõlema programmi tulemused on saadud Raspberry Pi 4B platvormil. Testimisel on kasutatud Raspberry Pi OS operatsioonisüsteemi koos graafilise liidesega. Mõlema programmeerimiskeele testimiseks on kasutatud sama videot [58], mille kestus on 31 sekundit ja filmitud testrajal töös käsitletava robotiga.

Programmis aja mõõtmine algab hetkest, mil edastatakse videokaamerast tulev pilt programmi muutujasse ning lõppeb siis, kui programm on jõudnud tsükli lõppu. Leitud kurvide väärtused on kirjutatud failidesse koos arvutusajaga, et neid kujutada võrdluseks graafikul (Joonis 30, Joonis 31).



Joonis 30. C++ ja Pythoni pilditöötluse kiiruste võrdlus, kus on esitletud arvutustulemuste sõltuvus ajast. Kiiruste võrdlusteks on kasutatud eelsalvestatud videot, mis on 32 sekundit pikk.

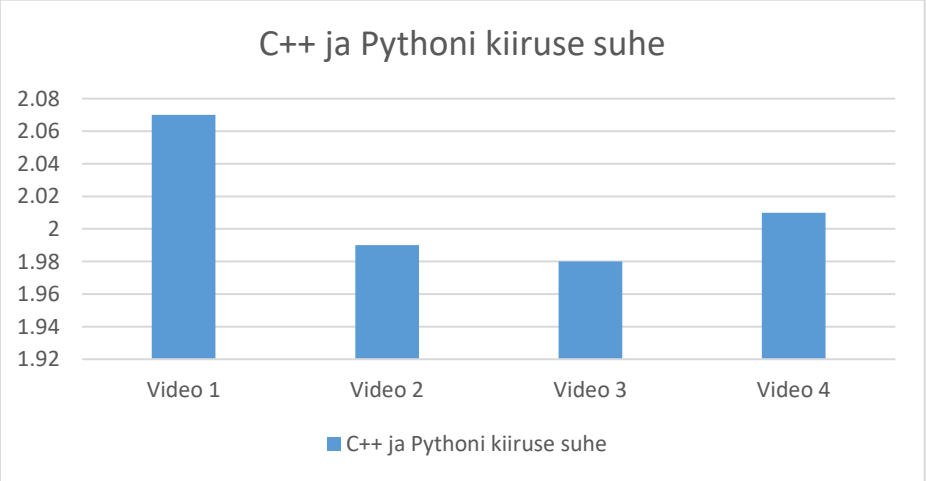
Joonisel (Joonis 30) on kujutatud sinisena C++ kurvide arvutuste sõltuvus ajast ja sama Pythonis. C++ kood töötleb videot kaks korda kiiremini, lõpetades videotöötlemise 16 sekundil, kuid Pythonis kirjutatud analoogne programm lõpetas töötlemise 33 sekundil.



Joonis 31. C++ aja väärtused korrutatud väärtusega 2.1, et ühtlustada kurvide graafikuid.

Viies mõlemad graafikud omavahel ajaliselt peaaegu ühtima, võib graafikult näha kurvide arvutuste üldist kattuvust. Mõõtmiseks kasutatud koodfailid on avalikult kättesaadavad [59], [60]. Varem mainitud viited, kus väidetakse, et C++ on kiirem kui Python OpenCV raamistikus, vastab tõele selle lõputöö raames käsitletud sõidurajatuvastamise programmis.

Kontrolliks testis autor sama metoodikaga nelja juhuslikku videofaili. Testide tulemused olid sarnased eelnevaga ehk C++ programm on ligikaudu kaks korda kiirem, kui Pythonis kirjutatud programm (Joonis 32). Videote resolutsiooniks oli kõigil testimiseks kasutatud videotel 480x240 pikslit.



Joonis 32. C++ Ja Pythoni programmide kiiruste suhted kasutades sama videofaili.

5 Täiendamine ja tulemused

Maapinna eristamisel tekkisid raskused juhul, kui maapind on peegelduv. Lampidest tulenevad valgusvihud peegelduvad maapinnal ning tekitavad vältimatu segaja. Peegelduvat valgust on võimalik maapinnast eristada lävendiga, kuid sellega kaotab teiste objektide tuvastamiseks kasutatav lävend oma täpsuse. Peegelduste vähendamiseks on kasutatud kontrasti adaptiivset ühtlustamist [61].

Kontrastiga piiratud adaptiivne histogrammiga ühtlustamine on protsess, kus jagatakse pilt väikesteks plokkideks ning seejärel ühtlustatakse iga ploki kontrastsust. Kui mõni histogrammi veerg on üle määratud kontrasti piiri (see on vaikimisi 40 OpenCV's), siis need pikslid kärbitakse ja jaotatakse ühtlaselt teistesse veergudesse enne histogrammi tasandamise rakendamist. Pärast ühtlustamist rakendatakse bilineaarset interpolatsiooni, et eemaldada artefaktid plokkide piirides [62].

Git projekti *tools* kaustas on mitu üldprogrammi tööd toetavat faili. HSV värviruumi lävendi leidmiseks kasutatav abiprogramm on ümberkirjutatud Pythoni programmeerimiskeelest C++ programeerimiskeelde, et seda vajadusel põhiprogrammiga integreerida.

5.1 Abiprogrammid

Tööriistade kaustas asub kaks Pythoni faili ja eraldi kaust, kus on C++ failid. Ultrahelisensori testimiseks ja töökäigu arusaamiseks on loodud us.py Pythoni fail, mille sisu põhineb [28] juhendist. Selle programmi kasutamiseks tuleb ühendada sensori ühendused vastavusse faili sisuga või muuta programmis kasutatavaid viike.

Peatükis 4.2.4 kirjeldatud automaatkäivituse programmi kasutamiseks tuleb lisada programmikoodi juhtpuldi MAC aadress ning sisestada käsklus, mida operatsioonisüsteem täidab. Vaikimisi puudub programmis korrektne MAC aadress ning programm on seadistatud käivitama masinnägemise programmi autori poolt defineeritud kaustas.

Pilditöötluse individuaalseks testimiseks on loodud detection.cpp fail. Selles failis on ainult pilditöötluse osa ilma mootorite juhtimiseta, kus saab katsetada pilditöötluse täiendamist ja muudatusi enne roboti testimist. Testimise sisendina võib kasutada vabalt valitud videot või videokaamerat, kuid reaalsema olukorra simuleerimiseks on kasutatud robotiga salvestatud videot, mis asub *git* projekti *test_footage* kaustas.

Korrektselt installeeritud OpenCV kontrollimiseks on kirjutatud threadCheck.cpp fail. Seda faili kompileerides ja käivitades väljastab see programm OpenCV versiooni, mitu loogilist protsessorit saab OpenCV kasutada ning kas installeeritud OpenCV toetab NEON tehnoloogiat. NEON tehnoloogia ei ole oluline laiendus OpenCV teegi jaoks selle lõputöö raames, kuid on kasulik kui lisada robotile objektituvastus [63].

HSV värviruumis objektide eristamiseks on hsvspace.cpp, mis on kopeeritud OpenCV juhendist [64], kus on ka täpsem kasutusjuhend ja programmi töö kasutamise juhend. Selle programmi tööd on näha joonisel (Joonis 27), kus on seda kasutatud maapinna eristamiseks.

5.2 Tulemused

Robotiga võisteldi *Robotex International* võistlusel aastal 2021. Võistlusest on video, kus võistleb lõputöös käsitletud robot (alustab sõitmist esimesena) [65]. Võistlusel sõitis robot alagrupis ühe terve ringi tagurpidi, mistõttu jäi alagrupis kolmandaks ning ei pääsenud järgmisesse vooru [66]. Vaatamata sellele oli ehitatud robot võistlusel ainus, mis kasutas videokaamerat rajal autonoomselt sõitmiseks ning oli konkurentsivõimeline.

Võistlusest salvestatud video põhjal saab analüüsida roboti probleemseid kohti. Robot sõltub palju keskkonnas olevatest värvidest ehk kui on seadistatud maapinna lävend, siis muud objektid, teised robotid ja takistused võivad jääda seatud lävendist välja või sisse. Video alguses on näha, kuidas robot ei ole võimeline eesolevale hallile takistusele reageerima, kuna see jääb HSV värviruumis seatud lävendisse sisse, mistõttu ei pea programm seda takistuseks.

Teine probleemne koht on see, kui robot sõidab seina poole mis on temaga risti ning eristusobjekti nähtavus on mõlemal pool roboti vaateväljas võrdne. Sel juhul edastatakse servomootorile 0-väärtus ehk käsklus jätkata otse sõitmist.

Vaatamata probleemidele on näha head koostööd sardsüsteemis, kus ultrahelisensor annab robotile käskluse tagurdada, küll alati mitte õiges suunas, kuid suutis robotit hoida pidevas liikumises. Robot sõidab sujuvalt mööda rada, jälgides maapinda ja kohati ka teisi roboteid.

Joonisel (Joonis 27) on näha, et testimiseks kasutatud rada erineb võistlusel olevast rajast. Uuteks tingimusteks pilditöötluse ettevalmistamine toimis võistlusel kohapeal, kasutades HSV värviruumi objektide eristamiseks abiprogrammi (Lisa 4). Roboti peal abiprogrammi kasutades koguti vajalikud lävendi väärtused, mille põhjal robot võistlusel sõitis.

6 Kokkuvõte

Töö eesmärgiks oli ehitada masinnägemisega isesõitev võistlussõidurobot, mille rajatuvastus põhineks videokaameral ja sõidaks võistlusel vähemalt ühe ringi. Roboti ehitamisel peamised tingimused olid:

- kasutada TTÜ Robotiklubis olemasolevaid vahendeid;
- roboti ehitamisel lähtuda võimalikult palju võistluste reeglistikest;
- robot peab olema võimeline võistlusel sõitma vähemalt ühe ringi.

Sobiva algoritmi valikul lähtuti varasemast kogemustest erinevate programmeerimiskeeltega ning masinnägemise algoritmidega ja lõputöös leitud programmerimiskeelte võrdlustest. Lõplik lahendus kasutab HSV värviruumi maapinna tuvastamiseks ja Raspberry Pi 4B sardsüsteemi tuuma roboti juhtimiseks ning algoritmi töötluseks.

Ehitatud robot täidab ettenähtud tingimusi. See võistles *Robotex International* 2021 üritusel, kus robot läbis edukalt tehnilise kontrolli ja sõitis ühe ringi, kuid vastassuunas. Robot ehitati TTÜ Robotiklubi poolt varutud vahenditega ning kasutas autonoomseks sõitmiseks videokaameral põhinevat rajatuvastusalgoritmi.

Kasutatud kirjandus

[1] J. Kocič, N. Jovičić and V. Drndarević, "Sensors and Sensor Fusion in Autonomous," *26th Telecommunications forum TELFOR 2018,* no. doi: 10.1109/TELFOR.2018.8612054, pp. 420-425, 2018.

[2] Robotex International, "Folkrace," [Online]. Available: https://robotex.international/folkrace. [Accessed 14 May 2022].

[3] OpenCV team, "OpenCV," [Online]. Available: https://opencv.org/. [Accessed 12 January 2022].

[4] Wiring Pi, "WiringPi," [Online]. Available: http://wiringpi.com/. [Accessed 25 April 2022].

[5] Pigpio, "The pigpio library," [Online]. Available: https://abyz.me.uk/rpi/pigpio/. [Accessed 2 May 2022].

[6] Shenzhen Creality 3D Technology Co., Ltd, "Ender 3," [Online]. Available: https://www.creality.com/products/ender-3-3d-printer. [Accessed 25 April 2022].

[7] Dassault Systèmes, "Solidworks," [Online]. Available: https://www.solidworks.com. [Accessed 25 April 2022].

[8] Raspberry Pi Ltd, "Raspberry Pi OS," [Online]. Available: https://www.raspberrypi.com/software/. [Accessed 13 May 2022].

[9] S.K. Dhinesh, Prakash S. Arun, Kumar K.L. Senthil, A. Megalingam, "Study on flexural and tensile behavior of PLA, ABS and PLA-ABS materials," 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785320323270. [Accessed 5 March 2022].

[10] Hong Kong feng tai co., LTD, "1:90 110RPM 3V-36V Metal DC Gear Motor," [Online]. Available: https://www.aliexpress.com/item/1005002280965951.html. [Accessed 27 04 2022].

[11] TME, " 34:1 METAL GEARMOTOR 25DX52L MM HP 6V POLOLU," [Online]. Available: https://www.tme.eu/en/details/pololu-1573/dc-motors/pololu/34-1-metal-gearmotor-25dx52l-mm-hp-6v/?fbclid=IwAR0M4Z7NnofDBlfV74OZNM5_Ml36eMVctnZq4FKeVXcGz689xUgehs9ZOIc. [Accessed 27 April 2022].

[12] E. Upton, "Production and supply-chain update," [Online]. Available: https://www.raspberrypi.com/news/production-and-supply-chain-update/. [Accessed 2 May 2022].

[13] B. Hangün and Ö. Eyecioğlu, "Performance Comparison Between OpenCV Built in CPU and GPU Functions on Image Processing Operations," 2017. [Online]. Available: https://www.researchgate.net/publication/341251073_Performance_Comparison_Between_OpenCV_Built_in_CPU_and_GPU_Functions_on_Image_Processing_Operations.

[14] Intel, "Intel® Neural Compute Stick 2," [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/neural-compute-stick/overview.html. [Accessed 6 March 2022].

[15] Hitec RCD, "HS-485HB Deluxe HD Ball Bearing Standard Servo," [Online]. Available: https://hitecrcd.com/products/servos/analog/sport-2/hs-485hb/product. [Accessed 27 April 2022].

[16] Waveshare, "Motor Driver HAT for Raspberry Pi," [Online]. Available: https://www.waveshare.com/motor-driver-hat.htm. [Accessed 2 May 2022].

[17] Waveshare, "Motor Driver HAT," [Online]. Available: https://www.waveshare.com/wiki/Motor_Driver_HAT. [Accessed 2 May 2022].

[18] Raspberry Pi Ltd, "Raspberry Pi 4 Tech Specs," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. [Accessed 6 March 2022].

[19] Samsung, "EVO Plus microSD Memory Card 32GB," [Online]. Available: https://www.samsung.com/us/computing/memory-storage/memory-cards/microsdhc-evo-plus-memory-card-w--adapter-32gb--2017-model--mb-mc32ga-am/. [Accessed 2 May 2022].

[20] HobbyKing, "Turnigy nano-tech 2000mAh 2S1P," [Online]. Available: https://hobbyking.com/en_us/turnigy-nano-tech-2000mah-2s1p-20-40c-lipo-receiver-pack.html. [Accessed 2 May 2022].

[21] HobbyKing, "Turnigy 1600mAh 2S 20C Lipo Pack," [Online]. Available: https://hobbyking.com/en_us/turnigy-1600mah-2s-20c-lipo-pack-losi-mini-compatible.html. [Accessed 2 May 2022].

[22] HobbyKing, " HobbyKing Quattro 4x6S Lithium Polymer Multi Charger," [Online]. Available: https://hobbyking.com/en_us/hobbykingtm-quattro-4x6s-lithium-polymer-multi-charger.html?gclid=EAIaIQobChMIgMrYv4jB9wIVNgWiAx2MugTnEAAYASAAEgJ89PD_BwE&___store=en_us. [Accessed 2 May 2022].

[23] J. Wen, Y. Wen, Y. Chunhua and C. Chunhua, "A Review on Lithium-Ion Batteries Safety Issues: Existing Problems and Possible Solutions," *Materials Express*, vol. 2, no. 3, pp. 197-212, 2012.

[24] HorizonHobby, "LiPo Voltage Checker Warning Alarm," [Online]. Available: https://www.horizonhobby.com/product/lipo-voltage-checker-warning-alarm/DYNF0002.html. [Accessed 2 May 2022].

[25] Sony, "Dualshock 4 Wireless Controller," [Online]. Available: https://www.playstation.com/en-us/accessories/dualshock-4-wireless-controller/. [Accessed 2 May 2022].

[26] SparkFun Electronics, "Ultrasonic Distance Sensor - HC-SR04," [Online]. Available: https://www.sparkfun.com/products/15569. [Accessed 2 May 2022].

[27] Raspberry Pi Ltd, "Raspberry Pi Documentation," [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html. [Accessed 5 May 2022].

[28] Tutorials for Raspberry Pi, "Using a Raspberry Pi distance sensor," [Online]. Available: https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/. [Accessed 5 May 2022].

[29] Vizdx LLC, "Self driving car using Raspberry Pi," [Online]. Available: https://www.computervision.zone/courses/self-driving-car-using-raspberry-pi/. [Accessed 22 April 2022].

[30] Canonical Ltd, "Enterprise Open Source and Linux | Ubuntu," [Online]. Available: https://ubuntu.com/. [Accessed 13 May 2022].

[31] Raspberry Pi Ltd, "Raspberry Pi Documentation," [Online]. Available: https://www.raspberrypi.com/documentation/computers/configuration.html. [Accessed 10 February 2022].

[32] Raspberry Pi Ltd, "Raspberry Pi Documentation," [Online]. Available: https://www.raspberrypi.com/documentation/computers/remote-access.html. [Accessed 10 February 2022].

[33] Raspberry Pi Ltd, "Introducing the Raspberry Pi Cameras," [Online]. Available: https://www.raspberrypi.com/documentation/accessories/camera.html. [Accessed 2 May 2022].

[34] Raspberry Pi Ltd, "The raspi-config tool," [Online]. Available: https://www.raspberrypi.com/documentation/computers/configuration.html. [Accessed 2 May 2022].

[35] RealVNC, "VNC Connect," [Online]. Available: https://www.realvnc.com/en/raspberrypi/. [Accessed 2 May 2022].

[36] RealVNC, "VNC Connect," [Online]. Available: https://www.realvnc.com/en/connect/download/viewer/. [Accessed 2 May 2022].

[37] S. Vishwesh, "Install OpenCV 4 on Raspberry Pi," [Online]. Available: https://learnopencv.com/install-opencv-4-on-raspberry-pi/. [Accessed 5 May 2022].

[38] Intel, "oneTBB," [Online]. Available: https://github.com/oneapi-src/oneTBB. [Accessed 5 May 2022].

[39] The Qt Company, "Qt," [Online]. Available: https://www.qt.io/. [Accessed 5 May 2022].

[40] S. Chacon, "Git," [Online]. Available: https://git-scm.com/. [Accessed 5 December 2022].

[41] GitHub, "GitHub," [Online]. Available: https://github.com/. [Accessed 2 December 2022].

[42] GNU, "GNU make," [Online]. Available: https://www.gnu.org/software/make/manual/make.html. [Accessed 25 Aprill 2022].

[43] B. Gábor, "Mat - The Basic Image Container," [Online]. Available: https://docs.opencv.org/4.x/d6/d6d/tutorial_mat_the_basic_image_container.html. [Accessed 2 May 2022].

[44] OpenCV, "Smoothing Images," [Online]. Available: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html. [Accessed 2 May 2022].

[45] E. S. Gedraite and M. Hadad, "Investigation on the Effect of a Gaussian Blur in Image Filtering and Segmentation," [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6044249. [Accessed 12 February 2022].

[46] OpenCV, "Operations on arrays," [Online]. Available: https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga4b78072a303f29d9031d56e5638da78e. [Accessed 2 May 2022].

[47] Pigpio, "pigpio C Interface," [Online]. Available: https://abyz.me.uk/rpi/pigpio/cif.html. [Accessed 2 May 2022].

[48] Cornell University, "Linux - cron and crontab," [Online]. Available: https://compbio.cornell.edu/about/resources/linux-cron-and-crontab/. [Accessed 13 May 2022].

[49] cppreference, "std::thread," [Online]. Available: https://en.cppreference.com/w/cpp/thread/thread. [Accessed 5 May 2022].

[50] OpenCV, "Canny Edge Detection," [Online]. Available: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html. [Accessed 13 May 2022].

[51] OpenCV, "Hough Line Transform," [Online]. Available: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html. [Accessed 13 May 2022].

[52] M. Kingisepp, "Robotiklubi projekt ICU," [Online]. Available: https://gitlab.com/Fyoxy/icu_old. [Accessed 12 May 2022].

[53] M. Kingisepp and S. Kajak, "Icu," [Online]. Available: http://robotiklubi.ee/doku.php?id=projektid:voistlusrobotid:folkrace:icu. [Accessed 12 May 2022].

[54] Tron777, "Python vs C++ (personal experiences with using both)," 25 June 2021. [Online]. Available: https://forum.opencv.org/t/python-vs-c-personal-experiences-with-using-both/4016. [Accessed 12 May 2022].

[55] Quora, "What are the pros and cons of developing in OpenCV Python versus OpenCV C++?," [Online]. Available: https://www.quora.com/What-are-the-pros-and-cons-of-developing-in-OpenCV-Python-versus-OpenCV-C++. [Accessed 12 May 2022].

[56] E. Gölge, "Does performance differ between Python or C++ coding of OpenCV?," 17 November 2012. [Online]. Available: https://stackoverflow.com/questions/13432800/does-performance-differ-between-python-or-c-coding-of-opencv. [Accessed 12 May 2022].

[57] OpenCV, "cv::TickMeter Class Reference," [Online]. Available: https://docs.opencv.org/3.4/d9/d6f/classcv_1_1TickMeter.html. [Accessed 14 May 2022].

[58] M. Kingisepp, "ICU test_footage," [Online]. Available: https://github.com/Fyoxy/ICU/tree/dev/test_footage/output.avi. [Accessed 13 May 2022].

[59] Vizdx LLC, "Python lane detection main program," [Online]. Available: https://github.com/Fyoxy/ICU/tree/dev/python_performance/program.py. [Accessed 13 May 2022].

[60] Vizdx LLC, "Python lane detection supportive program," [Online]. Available: https://github.com/Fyoxy/ICU/tree/dev/python_performance/utlis.py. [Accessed 13 May 2022].

[61] OpenCV, "cv::CLAHE Class Reference," [Online]. Available: https://docs.opencv.org/3.4/d6/db6/classcv_1_1CLAHE.html. [Accessed 5 May 2022].

[62] OpenCV, "Histograms - 2: Histogram Equalization," [Online]. Available: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html. [Accessed 12 May 2022].

[63] A. Rosebrock, "Optimizing OpenCV on the Raspberry Pi," [Online]. Available: https://pyimagesearch.com/2017/10/09/optimizing-opencv-on-the-raspberry-pi/. [Accessed 12 May 2022].

[64] L. García and R. Surti. [Online]. Available: https://docs.opencv.org/4.x/da/d97/tutorial_threshold_inRange.html. [Accessed 12 May 2022].

[65] H. Ennok and M. Kingisepp, "Project ICU at Robotex International 2021," 2021. [Online]. Available: https://www.youtube.com/watch?v=LKv6RdX_fBk. [Accessed 12 May 2022].

[66] Robotex International, "Tulemused / Results," [Online]. Available: https://robotex.international/et/tulemused/. [Accessed 12 May 2022].

[67] Raspberry Pi Ltd, "Raspberry Pi 3 Model B+," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. [Accessed 6 March 2022].

[68] NVIDIA, "Jetson Nano Developer Kit," [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed 6 March 2022].

[69] Jkollerup.dk, "Raspberry Pi 4 Model B – 4 GB," [Online]. Available: https://raspberrypi.dk/en/product/raspberry-pi-4-model-b-4-gb/?src=raspberrypi. [Accessed 6 March 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Marti Kingisepp

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Masinnägemisega isesõitev võistlussõidurobot“, mille juhendaja on Uljana Reinsalu
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh 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 käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

16.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Roboti koostude kaust

https://livettu-my.sharepoint.com/:f:/g/personal/mkingi_ttu_ee/EtvjBGtB_M1Hj3w4fqkXqqcBVHQE3K1q2umgFtgPsy0rMw?e=hubRoL

Lisa 3 – Pilditöötluse programmi lähtekood

https://github.com/Fyoxy/ICU

Lisa 4 – HSV värviruumi lävendi seadmise programmi lähtekood

https://github.com/Fyoxy/ICU/blob/main/tools/obj/hsvspace.cpp