TARTU ÜLIKOOL

Loodus- ja täppisteaduste valdkond

Tehnoloogiainstituut

Madis Kaspar Nigol

ÕPPEMATERJALID ROBOTPLATVORMILE ROBOTONT

Magistritöö (30 EAP)

Juhendajad:
PhD Karl Kruusamäe
PhD Veiko Vunder

Tartu 2019

Sisukord

Selede loetelu

Lühikokkuvõte

Õppematerjalid robotplatvormile Robotont

Robootika on jõudsalt populaarsust kogunud ning seetõttu on suurenenud ka nõudlus robootikainseneride järele. Paljudes robootikafirmades otsitakse ROSi (*Robot Operating System*) oskusega inimesi, seega tuleks õpetada ROSi kasutamist. Tartu Ülikoolis on arendatud robotplatvorm Robotont, mis on ehitatud ROSile, kuid millel puudusid õppematerjalid ning seda toetav tarkvara. Töö raames valmis Robotondile kaheksa praktikumitööd regulaatorite, 2D- ja 3D-kaardistamise ning autonoomse navigeerimise teemadel, seda toetav ROS tarkvara ning Gazebo simulatsioonikeskkond, mis võimaldab praktikumitöid läbida ilma robotita. Valminud praktikumitööd piloteeriti TÜ Narva kolledžis.

CERCS: T125 Automatiseerimine, robootika, control engineering

Märksõnad: õppematerjalid, roboti operatsioonisüsteem (ROS), roboti tarkvaraarendus, haridusrobot

Abstract

Study Materials for Robot Platform Robotont

As robotics has gained popularity, the need for robotics engineers has increased. Many robotics companies look for engineers with knowledge of ROS (Robot Operating System). Therefore, ROS should be taught in schools. A ROS capable robot platform called Robotont has been developed at the University of Tartu but it was missing study materials and supporting software. As a result of this effort, eight lab manuals for Robotont were created to introduce topics such as controllers, 2D and 3D mapping, and autonomous navigation. In addition, supporting ROS software was developed. Special focus was set on creating Gazebo simulation packages for Robotont to enable the use of study materials without the real robot. The study materials and robot software were piloted at Univeristy of Tartu Narva College.

CERCS: T125 Automation, robotics, control engineering

Keywords: study materials, ROS, robot software development, educational robot

Kasutatud mõisted ja lühendid

Lühend	ingliskeelne vaste	eestikeelne vaste
AR	Augmented Reality	liitreaalsus
HTML	Hypertext Markup Language	hüperteksti märgistuskeel
LiDAR	Light Detection and Ranging	laserskanneerimisseade
LTT		Loodus- ja täppisteadused
OS	Operating System	operatsioonisüsteem
PDF	Portable Document Format	arvuti riist- ja tarkvaraplatvormist sõltumatu elektrooniliste dokumentide vorming
ROS	Robot Operating System	roboti operatsioonisüsteem
SLAM	Simultaneous Localization and Mapping	samaaegne lokalisatsioon ja kaardistamine
URDF	Unified Robot Description Format	ühtne robotikirjeldusformaat
XML	eXtensible Markup Language	laiendatav märgistuskeel

Sissejuhatus

Robootika on viimase paari aastakümnega jõudsalt levinud. Paljud inimesed puutuvad kokku robootikaga seda ise teadmata — postiautomaatidest ja robottolmuimejatest isesõitvate autodeni. Kuna robotid on sedavõrd populaarsed, on vaja ka aina rohkem inimesi neid arendama. Kahjuks valitseb selles vallas suur tööjõupuudus ning üldine madal teadlikkus. Tööjõuturul on ennekõike tarvis inimesi, kes oskaksid roboteid kasutada, hallata või luua. Samuti võiks parem teadlikkus aidata maandada alusetuid hirme robootika suhtes ning seeläbi panustada robootika parimasse võimalikku rakendamisse inimkonna elukvaliteedi tõstmisel.

Robootikateadlikkuse tõstmiseks ja valdkonna jätkusuutlikuks arenguks on inimesi vaja harida juba varasest east. Sellisel viisil suureneb laste teadlikkus robotitest ning väheneb hirm robotite ees. See võib tõsta nende huvi ka tulevikus robootikaga tegeleda ning seeläbi vähendada valdkonna tööjõupuudust ja kiirendada robootika arengut.

Robootika õpetamiseks on vaja spetsiaalselt kohandatud roboteid ehk haridusroboteid. Haridusroboteid (pikemalt alapeatükis 1.1) on mitmeid ning need on mõeldud erinevatele tasemetele, alustades lasteaiast, lõpetades ülikooliga. Lasteaiatasemel on robotid lihtsad ning enamasti programmeeritavad lihtsates graafilistes keskkondades või lausa ilma arvutita [1]. Vanemas astmes on aga eelistatud tekstipõhiseid programmeerimiskeeled nagu näiteks Python, C++, Java jne.

Robotite rohkuse tõttu on tekkinud aga probleem: robotid kasutavad erinevat tarkvara. Selleks, et haridusrobotitel õpitud oskused kanduksid üle ka edaspidisesse robootikakarjääri, on vaja ühtset standardit robotite tarkvaraarendusele. Hetkel on robootikas standardiks saamas ROS (*Robot Operating System*). See laialt levinud tarkvararaamistik loob võimaluse kasutada ühtset tarkvara erinevatel robotitel, mistõttu on ROSi kasutamist mõistlik ka haridusrobotitel õpetada.

Haridusrobotite edukaks kasutamiseks on äärmiselt oluline kvaliteetsete õppematerjalide olemasolu. Õppematerjalide roll on pakkuda nii õpetajatele kui õppuritele vajalik tugi oluliste robootika teemade käsitlemiseks. Oluline on ka õppematerjalide kättesaadavus emakeeles.

Käesoleva töö eesmärk on luua õppematerjalid Tartu Ülikooli avatud robotiplatvormi Robotont kasutamiseks Eesti gümnaasiumites, kutsekoolides ja ülikoolides. Töö tulemusena loodi kaheksa eestikeelset praktikumijuhendit koos toetavate programmeerimisnäidetega ROSis. Lisaks täiustati Robotondi tarkvarafunktsionaalsust ning loodi robotile simulatsioon, et võimaldada praktikumijuhendite läbimine ka ilma roboti olemasoluta.

1. Kirjanduse ülevaade

1.1 Robotite kasutamine hariduses õppevahendina

Kuna robootika on maailmas tohutult populaarsust kogunud, siis arendatakse üle maailma robotplatvorme, mis sobivad nii robootika õpetamiseks [2–11] kui ka robotite abil teiste õppeainete õpetamiseks [12]. Roboteid arendatakse väga laiale vanusegrupile: lasteaiast [13] ülikoolini [11].

Üks populaarsemaid õpperoboteid maailmas on Lego Mindstorms EV3 (sele 1.1a) [2, 14, 15], mis on lihtsasti laiendatav robootikakomplekt, kuna roboti kuju on iga kasutaja enda teha. Selle komplektiga õpetatakse lastele inseneri-, programmeerimis- ning probleemilahendusoskusi. Chaudhary jt (2016) [16] uurisid eelnimetatud oskuste omandamist EV3 robootikakomplektide abil. Uuringus selgus, et robootikakomplektide kasutamine koos projektipõhise õppemetoodikaga annab efektiivsema tulemuse kui loengupõhine õpe.

Peale populaarsete Lego Mindstorms robotite arendatakse ka spetsiifilisemaid hariduseks mõeldud robotplatvorme. Igal robotil on oma sihtgrupp, mis varieerub lasteaiast ülikooli tasemeni. Lasteaiaealistele on loodud robot KIBO (sele 1.1b) [13], mida programmeeritakse puidust kuubikute abil. Kuubikute peale on kirjutatud inimloetav käsk koos masinloetava triipkoodiga. Neid kuubikuid ritta seades saab laps luua programme, mida robot täidab. Sellise lahenduse eesmärk on anda mänguline ja ekraanivaba viis programmeerimiseks, mis arendab nii algoritmilist, loomingulist kui ka kriitilist mõtlemist [17, 18].

Alg- ning põhikoolitasemele sobib näiteks AERobot (sele 1.1c) [3], mis on oma suhteliselt odava hinna tõttu osutunud populaarseks õppevahendiks arenguriikides. AERobot kasutab liikumiseks kahte vibreerivat mootorit ning teeb otsuseid lähedus- ja valgusanduritest saadud info põhjal. Programmeerimine käib graafilises keskkonnas või tekstipõhiselt C/C++ keeles. AERoboti pilootkursus näitas suurenenud huvi LTT (loodus- ja täppisteadused) valdkondade vastu. Teine ilma ratasteta robot on MetaBot (sele 1.1d) [4], mis kasutab liikumiseks nelja jalga. MetaBotil on üheksa vabadusastmega inertsiaalandur, kaugusandur

ning Cortex-M3 mikroprotsessor, mis teeb selle võimsamaks, kuid ka kallimaks kui AERobot. Programmeerimine toimub Blockly-põhises [19] graafilise programmeerimise keskkonnas, tehes selle sobivaks neile, kes pole varem programmeerinud. MetaBoti kasutati programmeerimise õpetamiseks ühes Prantsusmaa koolis, kus alustati programmeerimise alustaladest ning lõpetati robotitevahelise tantsuvõistlusega, kus õpilased pidid meeskondades oma robotile looma tantsuprogrammi [4].

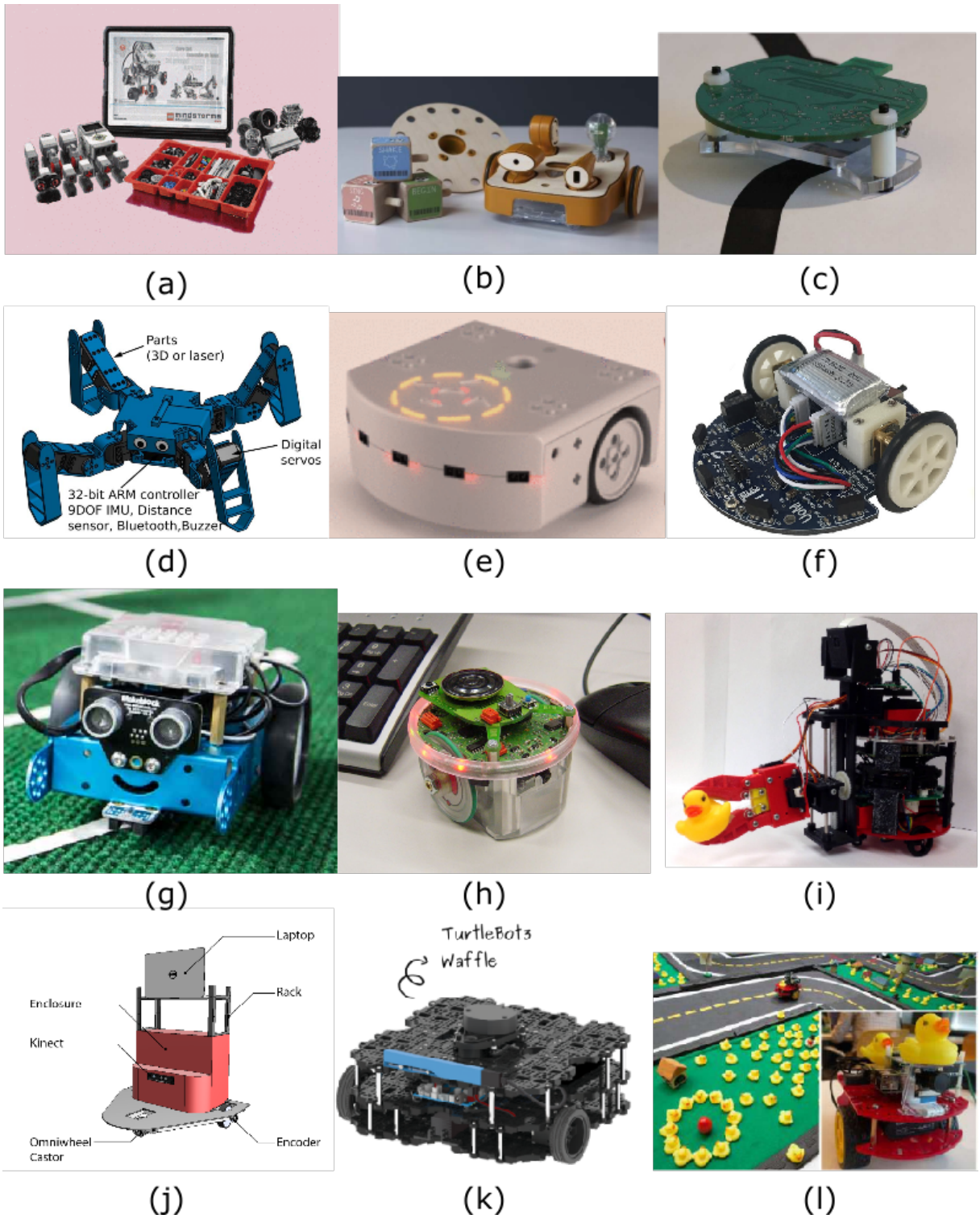
Ratastel robotitest on populaarne Thymio (sele 1.1e) [8], mille sihtgrupp on lasteaiast ülikoolini. Thymio on robustne robot suure valiku anduritega, mida saab programmeerida spetsiaalselt loodud keskkonnas Aseba. Asebas on võimalik robotit programmeerida nii visuaalprogrammeerimise kui ka skriptide kirjutamise kaudu. Suure eelisena on Aseba keskkonda võimalik ühendada ROSiga. Thymio on vabavara, mis toetab kasutajate loovust roboti kasutamise osas. Selline lähenemine tekitas Thymio ümber kogukonna, mis loob Thymiole pidevalt uusi lisasid nii riistvara, tarkvara kui ka õppematerjalide näol. Just suure kogukonna olemasolu on põhjuseks, miks Thymio õppematerjalide baas on laialdaselt levinud ja saadav erinevates keeltes. [8]

Eestis kasutatakse Tartu Ülikooli kursuse „Robootikast puust ja punaseks” [20] raames robotit mBot (sele 1.1f) [5]. Kursus on MOOC (ingl *massive open online course*) tüüpi, seega saavad seda võtta kõik huvilised olenemata asukohast.

Vanemas astmes, gümnaasiumi- ja ülikoolitasemel, on olulisemad parvrobootika (mitme roboti üheskoos kasutamine) ning ise C/C++ keeles programmeerimise võimalus. Seda võimaldavad näiteks robotid e-puck (sele 1.1g) [9], manipulaatoriga Pheeno (sele 1.1h) [6] ja vabavaraline robot Mona (sele 1.1i) [7]. E-puck keskendub programmeerimise õpetamisele, Pheeno parvrobootikale ning Mona mõlemale.

Kuna ROSi kasutamise oskus on muutunud robootikas vajalikuks, siis on seda hakatud õpetama ülikoolides [21] ning seega on vajalikud ka robotplatvormid, mis toetavad ROSi. ROSi toega robotplatvorm on näiteks RoboMuse 4.0 (sele 1.1j) [10], mille andureid ja täitureid kontrollitakse Arduino juhtplaadiga. Robotil on ka rakis, kuhu saab ühendada sülearvuti ROSi ja andmetöötlusvõimekuse tarbeks. Näiteks on RoboMuse platvormil Microsoft Kinect sügavuskaamera, mille abil on võimalik teha robotiga 3D kaarti. Üks populaarsemaid ROSi toega haridusroboteid on TurtleBot3 (sele 1.1k) [11]. Sellel robotil on 2D 360-kraadise vaateväljaga kaugusandur, mis võimaldab koostada 2D kaarte ning kaartide põhjal navigeerida. TurtleBotil on ka väga lai dokumentatsioon ning õppematerjalid, mis toetavad ROSi ja robootika õppimist inglise, hiina, korea ja jaapani keeles. Autonoomsuse õpetamiseks on välja töötatud platvorm nimega Duckietown (sele 1.1l)[22]. Duckietown on robotite linnak, kus liiguvad ROSi toel autonoomsed Duckiebotid. See platvorm on mõeldud nii õppimiseks kui

ka teadustööks ning selleks on olemas ka vajalikud õppematerjalid.



Sele 1.1. *Erinevad robotplatvormid: LEGO EV3 (a), KIBO (b), AERobot (c), MetaBot (d), Thymio (e), mBot (f), e-puck (g), Pheeno (h), Mona (i), RoboMuse 4.0 (j), TurtleBot Waffle (k) ja Duckietown (l).*

1.2 Robotite kasutamine hariduses õpetajana/õppeassistendina

Järgnev lõik põhineb Belpaeme jt ülevaateartiklil, mis ilmus 2018. a. ajakirjas Science Robotics [23]. Kui enamasti käsitletakse robotit kui õppevahendit, siis robotit on proovitud ka õpetaja või õppeassistendi rollis. Sotsiaalset õpperobotit võib käsitleda erinevalt: robot kui õpetaja, robot kui eakaaslane või robot kui algaja/õpilane. Õpetajana aitab robot õpilasi neid otse juhendades või vihjete abil. Selliseid roboteid prooviti näiteks lasteaias, kus autonoomne robot pandi sõimerühma ning selle tulemusel arenes kiiresti laste sõnavara [24]. Eakaaslasena on robot näiteks keegi, kellega rääkida võõrkeeles [25] või kes pakub hoopis motivatsioonilist tuge [26]. Kui robotit kasutada algajana, saab ära kasutada nn protežee efekti ehk õppimist läbi õpetamise [27]. Näiteks Jaapanis on lasteaiaealiste lastega kasutatud robotit, mis teeb nimelt inglise keeles vigu, et lapsed saaksid roboti keelekasutust parandada ning seeläbi ise õppida [28]. Praeguse seisuga on enamik uurimustöödest keskendunud roboti kasutamisele õpetajana. Kuigi sotsiaalse roboti kasutamine hariduses on paljulubav, on veel palju tehnilisi väljakutseid enne kui see laialtlevinuks saab. [23]

On ka välja pakutud, et sotsiaalseid roboteid võiks kasutada erivajadustega laste õpetamiseks. Itaalias tehti uurimistöö, kus uuriti erivajadustega laste õpetajate ning psühholoogiatudengite reaktsiooni sotsiaalsetele robotitele. Kuigi hüpotees oli, et sotsiaalsetesse robotitesse suhtutakse negatiivselt, siis uuringu tulemusel selgus, et arvamus robotitest oli hoopis positiivne. [29]

1.3 Miks me vajame ROSi

ROS (*Robot Operating System*) [30] on Linuxi-põhine avatud lähtekoodiga tarkvararaamistik robotitele. ROSi peamised eesmärgid on suurendada koodi taaskasutust ja toetada võimalikult palju erinevaid roboteid läbi universaalse disaini [31]. ROSi põhifunktsioon on sõnumivahetus erinevate programmide vahel. Näiteks ühes ülikoolis arendatud kaardistamisprogramm võib anda koostatud kaardi läbi ROSi sõnumite teises ülikoolis arendatud navigeerimisalgoritmile. Erinevate töögruppide koostööks on seega vaja järgida vaid ROSi sõnumivahetusstandardeid. Peale sõnumivahetuse on ROSis olemas tööriistad ning teegid koodi kirjutamiseks, kompileerimiseks, silumiseks, testimiseks ning käitamiseks ja andmete visualiseerimiseks [32].

ROSis on erinevate eesmärkidega tarkvaraüksused nimetatud kimpudeks (ingl k *ROS package*). Kimpe on võimalik eraldi kompileerida ning kimpudes asuvad käivitus- ja seadistusfailid ning programmid ehk sõlmed. Kimp on võimalikult väike ja kindla eesmärgiga, et tagada modulaarsus. Selleks, et arendajal oleks võimalik oma ROSiga töötav programm avalikkusele kättesaadavaks teha, peab ta jagama kas kimbu lähtekoodi, et kasutaja saaks kimbu ise kompileerida, või tegema oma kimbu binaarkujul kättesaadavaks läbi Linuxi pakihalduri.

Kuna ROS toetab koodi taaskasutust ja jagamist, siis on ROSi kasutajatest tekkinud suur ülemaailmne kogukond. Selle tõttu on nendel, kes tahavad ka oma robotit ROSi raamistikus arendada, olemas väga laia teadmisbaasiga tugi kogukonna poolt.

Ülikoolides liigutakse robootika õpetamisel just ROSi raamistikule, sest seal õpitavad oskused on universaalsed ja rakendatavad paljudes valdkondades [33]. Kuna ROS on muutumas robootikas standardiks, siis on ka nõudlus selle ala spetsialistide järele suur. Nõudlus tööturul motiveerib veelgi rohkem koole ja ülikoole ROSi õppetöösse lõimima.

1.3.1 Eeskujulik ROSi robot

Roboti ROSi toe all võib silmas pidada väga palju erinevaid tarkvaralahendusi. Kuid vaadates ROSi toega robotite [34] koodihoidlaid, joonistuvad välja head tavad. Näiteks võib tuua Clearpath Robotics'i Husky [35], ROBOTIS TurtleBot3 [36], Franka Emika Panda [37]. Positiivsete eeskujude põhjal saab järeldada, et roboti ROS tugikimp sisaldab funktsionaalsusi ja seonduvaid ROSi kimpe nagu toodud selel 1.1.

Sele 1.1. *Eeskujuliku ROSi roboti näited kimpude nimetamisel.*

Kimbu kirjeldus	Husky [35]	Turtlebot3 [11, 38]	Franka [37]
Kinemaatiline kirjeldus	*husky_description*	*turtlebot3_description*	*franka_description*
Roboti draiver	*husky_base*		*franka_hw*
Kiirkäitusfailid	*husky_bringup*	*turtlebot3_bringup*	
Juhtimissõlmed	*husky_control*	*turtlebot3_teleop*	*franka_control*
Robotipõhised ROS-sõnumid	*husky_msgs*		*franka_msgs*
Kaardistamis- ning navigatsioonikimp	*husky_navigation*	*turtlebot3_navigation* *turtlebot3_slam*	
Matkedraiver ja simulatsioon	*husky_gazebo* *husky_simulator*	*turtlebot3_fake* *turtlebot3_gazebo* *turtlebot3_simulations*	
Näiteprogrammid		*turtlebot3_example*	*franka_example_controllers*

Hea tava järgi on kimbud on nimetatud stiilis <*robotiNimi*>_<*funktsionaalsus*>. Järgnevates

näidetes on kasutatud roboti nimena sõna „robot”. Kokkuvõtvalt on hea nimetamistava toodud selel 1.2.

robot_description - Robootikas on paljudeks rakendusteks tähtis, et robot oleks ROSi jaoks kinemaatiliselt ära kirjeldatud [31]. Selle jaoks on XMLi-laadne süntaks URDF (*Unified Robot Description Format*) [39], kus määratakse roboti erinevad lülid ning nendevaheliste ühenduste tüübid ja asukohad. Kuna URDF faile on tülikas kirjutada ning hallata, eriti kui need hõlmavad korduvaid või sümmeetrilisi osi, siis on loodud skriptimiskeel Xacro [40]. Xacro võimaldab kirjeldada universaalsed makrod, et genereerida URDF fail robotile. Näiteks kui robotil on neli ratast, siis peab Xacro abil täielikult ära kirjeldama vaid ühe ratta makro ning hiljem asukoha parameetritega neli korda kirjeldatud makrot kasutama. Kinemaatilise kirjelduse kimbu nimi on enamasti *robot_description*. Siia alla võib minna ka roboti ja andurite andmete kuvamine ROS visualiseerimiskeskkonnas RViz, kuid vahel on visualiseerimise jaoks eraldi kimp *robot_viz*, näiteks *husky_viz* [35].

robot_driver - Draiver defineerib suhtluse roboti riistvara ja pardaarvuti vahel. Selle kimbu sisu on minimaalselt mootoridraiver, mis võimaldab robotit liigutada, kuid selles kimbus võivad asuda ka robotil olevate andurite draiverid. Seda kimpu nimetatakse enamasti stiilis *robot_base* või *robot_driver*.

robot_bringup - Hea tava on paigutada kiirkäitusfailid eraldi kimpu, kus need on kergelt kättesaadavad ja kasutatavad. Nende failide eesmärk on võimaldada kiire ja mugav roboti ülesseadmine ning kasutamine. See kimp hõlmab endas roboti seadistusskripte, et paigaldada roboti tarkvara jaoks eeldustarkvara, ning roboti andurite ning täiturite käivitusfaile. Sellist kimpu nimetatakse tavaliselt *robot_bringup* või *robot_setup*.

robot_teleop - Roboti juhtimiseks on tavaliselt kimp nimega *robot_control* või *robot_teleop*. Selles kimbus asuvad käivitusfailid ning sõlmed, mille abil robotit kaugjuhtida. Mobiilse roboti puhul asuvad siin tavaliselt kaugjuhtimismeetodid klaviatuuri ning mängupuldi jaoks.

robot_msgs - Eraldi kimp on ka oma sõnumitüüpide jaoks. Seda kimpu ei pruugi igal robotil vaja minna, sest enamasti kasutatakse ROSi standardsõnumitüüpe. Kui aga robotil on defineeritud oma sõnumitüübid, näiteks isetehtud anduri andmete edastamiseks, siis need asuvad kimbus *robot_msgs*.

robot_mapping - Kui robotil on peal visuaalsed andurid, näiteks kaamera või LiDAR (laserskanneerimisseade), siis on tõenäoliselt roboti ROSi kimbuna ka eraldi kaardistamis- ja/või navigatsioonikimp. Selles kimbus asuvad tavaliselt käivitusfailid SLAMiks (samaaegne lokalisatsioon ja kaardistamine) ning autonoomseks navigeerimiseks. Sellist kimpu nimetatakse tavaliselt *robot_mapping* või *robot_navigation*, kuid näiteks TurtleBot 3 koodihoidlas [36] on see jagatud eraldi *turtlebot3_navigation* ja *turtlebot3_slam* kimpudeks.

robot_simulation - Sageli tuleb ette, et roboti programme on vaja testida olukorras, kus robotile ligipääs puudub või kus robot on ümbrusele või iseendale ohtlik. Sellisteks olukordadeks on eeskujuliku ROSi toega roboti tarkvaras simulatsioonikimp, mis võimaldab kasutada robotit simulaatoris. Simulaatorisse tehakse robotist virtuaalne koopia, eesmärgiga saada see võimalikult reaalse roboti sarnaselt käituma. Selle tarbeks on kimbus tavaliselt matkedraiver, mis simuleerib roboti andureid ja täitureid, ning simulatsioonikeskkond, mis tehakse sarnane realistlikule töökeskkonnale. Kuna ROSi standardsimulaator on Gazebo [41], siis kimbu tavapärane nimi ongi *robot_gazebo*, mis omakorda võib olla kombineeritud metakimpu *robot_simulation* koos teiste simulatsiooniks vajaminevate kimpudega.

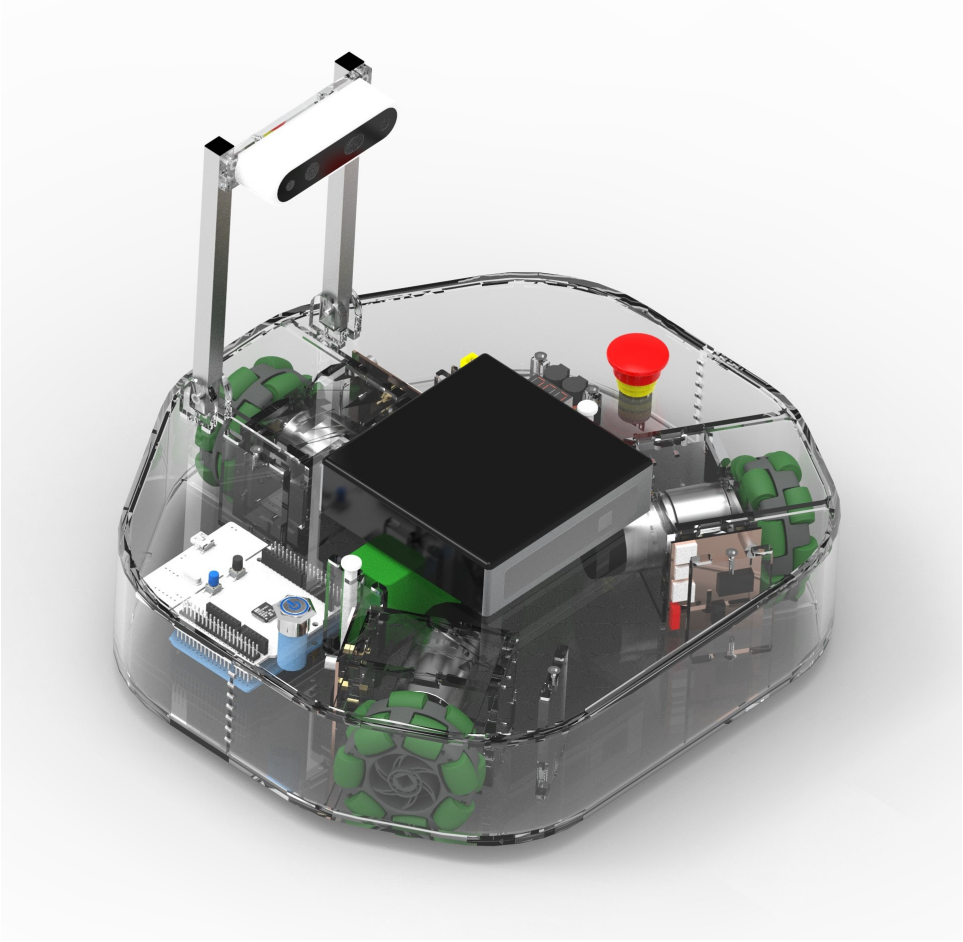
robot_example - Robotplatvormidel võib ka olla eraldi kimp *robot_example*, kus paiknevad roboti funktsionaalsust tutvustavad näiteprogrammid. Selle kimbu eesmärk on pakkuda arendajale tuge ja aidata teda robotplatvormi kasutamisel kiiremini edasi liikuma.

Sele 1.2. *Hea tava ROSi kimpude nimetamisel.*

Kimbu kirjeldus	Nimetamine
Kinemaatiline kirjeldus	*robot_description*
Andmete visualiseerimine	*robot_description* või *robot_viz*
Roboti draiver	*robot_base* või *robot_driver*
Kiirkäitusfailid	*robot_bringup* või *robot_setup*
Juhtimissõlmed	*robot_control* või *robot_teleop*
Spetsiifilised ROS-sõnumid	*robot_msgs*
Kaardistamis- ning navigatsioonikimp	*robot_mapping* või *robot_navigation*
Matkedraiver ja simulatsioon	*robot_gazebo* või *robot_simulation*
Näiteprogrammid	*robot_example*

2. Robotont

Käesolevas töös on kasutatud robotplatvormi Robotont (sele 2.1), millel on ka turustatud versioon nimega ClearBot. Magistritöös väljatöötatud õppematerjalid ja tarkvara juba osaliselt arvestab turustatud versiooni nimega, kuid sisulist erinevust kahe nimetuse vahel ei eksisteeri. Tegemist on robotplatvormiga, mis on mõeldud eeskätt robootika ja ROSi õppimiseks haridusvaldkonnas, kuid sobib kasutamiseks ka teadus- ja arendusvaldkondades.



Sele 2.1. *Robotont [42].*

2.1 Robotont/Clearbot üldkirjeldus

Robotondil on võimekas riistvara, mis sisaldab endas järgnevat:

- pardaarvuti Intel NUC7i5BNK,
- arendusplaat ST Nucleo-L476RG väliste seadmete ning mootorite jaoks,
- enkooderitega mootorid, mille otsas on omnirattad,
- sügavuskaamera Intel Realsense D435 [43].

Peale selle toetab Robotont tarkvararaamistikku ROS [30], mille eelised on välja toodud alapeatükis 1.3. Võimekas riistvara ja ROSi tugi teevad sellest robotplatvormist suurepärase õppevahendi, mille abil õpetada programmeerimist ning robootikat üldiselt. Omnirataste abil on robot võimeline liikuma igas suunas ilma pööramata. Sügavuskaamera abil on robotiga võimalik teha 2D- ja 3D-kaardistamist ning seeläbi ka autonoomset navigeerimist. Robotonti juhitakse sülearvuti abil, mis asub juhtmevabalt robotiga samas ROSi keskkonnas.

2.2 Robotondi seis magistritöö alguses

Käesoleva töö alguses oli Robotondi koodihoidlas [44] kaks ROSi kimpu ning üks seadistuskaust:

1. *robotont_description*
2. *robotont_driver*
3. *setup*

Kimbus *robotont_description* asus roboti URDF kirjeldus. See esialgne URDF kirjeldus oli genereeritud otse mudelifailist ning oli seetõttu väga halvasti loetav ja hallatav. Iga roboti lüli jaoks oli eraldi mudelifail, mis tähendas, et kuigi omniratta velg on alati samasugune, oli selle jaoks kolm eraldi faili. Samuti oli kimbus olemas käivitusfail Gazebo simulatsiooni jaoks, kuid puudusid matkedraiverid roboti juhtimiseks või andurite simuleerimiseks.

Kimbus *robotont_driver* oli Robotondi mootoridraiveri lähtekood ja käivitusfailid. Käivitusfailid võimaldasid käivitada roboti kaugjuhtimise klaviatuuri või mängupuldiga ja visualiseerida robotit RVizis.

Kaustas *setup* asus skript, mis seadistas Ubuntu terminaliaknas ROSi jaoks vajalikud globaalsed muutujad.

3. Töö eesmärk ja nõuded

Käesoleval magistritööl on kolm omavahel seotud eesmärki: Robotondi tarkvara täiustamine, õppematerjalide loomine ning loodud tarkvara ja õppematerjalide piloteerimine.

3.1 Õppematerjalid

Töö eesmärk on valmistada 6-8 praktikumijuhendit gümnaasiumitasemele koos õppejõudu toetavate töövahenditega. Õppematerjalid (lähemalt peatükis 4) käsitlevad teemasid nii programmeerimisest, ROSi kasutamisest kui ka robootikast üldiselt.

3.2 Tarkvaraarendus

Kuna Robotont on hiljuti arendatud robotplatvorm, siis puudub seal suur osa vajalikust tarkvarast (alapeatükk 2.2). Töö eesmärk on luua roboti ROSi tugi sarnaselt eeskujulikule ROSi robotile (alapeatükk 1.3.1). Seega tuleb lisada kimbud simuleerimiseks, kaardistamiseks, juhtimiseks ja õppematerjalide näiteskriptideks ning täiendada töö alguses olemasolevaid kimpe. Tarkvara on kirjeldatud peatükis 5.

3.3 Õpetamine Narvas

Koostatud õppematerjalid piloteeritakse Tartu Ülikooli Narva kolledžis õppeaine LOTI.05.080 Robotite programmeerimine ROS vahenditega (3 EAP) raames. Õppeaine eesmärk on tutvustada tudengitele ROSi, programmeerimist ning robootikat üldiselt. Kursus eeldab, et õppurite varasem kogemus selles vallas on vähene või puudub täielikult.

4. Õppematerjalid

Lõputöö raames valmis kaheksa praktikumijuhendit (lisa A). Käesolevas peatükis kirjeldatakse iga praktikumijuhendi eesmärke, sisu ja peamisi õpiväljundeid.

4.1 Praktikum 1 — Linux, ssh, klaviatuuriga juhtimine

4.1.1 Eesmärk

Esimese praktikumitöö eesmärk on tutvustada Linuxi käsurida ning õppida roboti käivitamist ja kasutamist kaugjuhtimisrežiimis.

4.1.2 Õpiväljundid

- Õppur oskab avada Linuxi terminali ning sisestada seal käske.
- Õppur kasutab Linuxis järgnevaid käske: *ls*, *mkdir*, *cd*, *pwd*, *nano*, *cat*, *mv*, *cp* ja *rm* ning saab aru nende funktsioonidest.
- Õppur oskab käsurealt luua *ssh*-ühenduse teise samasse võrku ühendatud arvutisse/robotisse.
- Õppur oskab käsurealt käivitada ROSi kiirkäivitusskripte ja juhtida robotit klaviatuuri abil.

4.1.3 Sisu

Praktikumi alguses avab õppur terminali ning õpib käsu *ls* ehk *list*, millega failisüsteemi faile näha. Seejärel õpitakse käsk *mkdir*, millega kaust luua. Kaustade vahel liikumiseks tuleb kasutada käsku *cd* ning hetke asukoha leidmine failisüsteemis sooritatakse käsuga *pwd*. Põhilise tekstiredaktorina on praktikumides kasutusel *nano*, millega õpitakse faile avama ja nendesse kirjutama ning kasutatakse käsku *cat*, et faili sisu ekraanile kuvada. Lõpuks õpitakse käskude *mv* ehk *move*, *cp* ehk *copy* ja *rm* ehk *remove* abil failide haldamist.

Peale esialgset tutvumist Linuxiga loob õppur käsurealt *ssh*-ühenduse robotiga ning käivitab roboti ROSi draiveri. Seejärel käitab õppur klaviatuuri abil roboti juhtimist võimaldava sõlme.

Praktikum loetakse edukalt sooritatuks, kui üliõpilane juhib robotit oma klaviatuuri abil.

4.2 Praktikum 2 — Programmeerimise ja robootika alused

4.2.1 Eesmärk

Teise praktikumi eesmärk on tutvustada Giti koodihoidlat, programmeerimiskeelt Python, ROS-rubriike ja ROS-sõnumeid. Pythonis õpitakse kasutama muutujaid, *for*-tsüklit ja funktsioone. Praktikumis õpitakse kirjutama Pythoni koodi, mis paneb roboti mööda ette määratud rada sõitma.

4.2.2 Õpiväljundid

- Õppur loob GitHubis endale koodihoidla ja teeb sellest arvutisse lokaalse koopia.
- Õppur oskab kirjutada Pythoni programmi, mis juhib robotit, kasutades *geometry_msgs/Twist*-tüüpi ROS-sõnumit.
- Õppur oskab kasutada Pythonis *for*-tsüklit.
- Õppur oskab kirjutada Pythonis funktsioone.

4.2.3 Sisu

Teine praktikumijuhend keskendub Pythonis programmeerimisele ja ROS-sõnumite tutvustamisele. Praktikumi alguses tuleb üles seada Giti koodihoidla. Praktikumijuhend sisaldab kolme alamülesannet.

1. Esmalt käsitletakse rubriiki „*cmd_vel*”, mis on kujunenud standardiks *geometry_msgs/Twist* liikumiskäskude edastamisel ROSis. Ülesandeks on robotiga erinevates suundades sõitmine. Selleks on õppuritel tarvis teha muudatusi etteantud Pythoni koodis.
2. Edasi on ülesandeks *for*-tsüklite õppimine ning nende abil roboti juhtimine mööda ruudukujulist trajektoori.
3. Praktikumi viimaseks ülesandeks on kirjeldada funktsioonid, millega sõita edasi-tagasi, paremale-vasakule ning pöörata päri- ja vastupäeva.

Praktikumi lõpus laadivad õppurid valminud programmid oma Giti koodihoidlasse üles.

4.3 Praktikum 3 — Andurid ja tingimuslause

4.3.1 Eesmärk

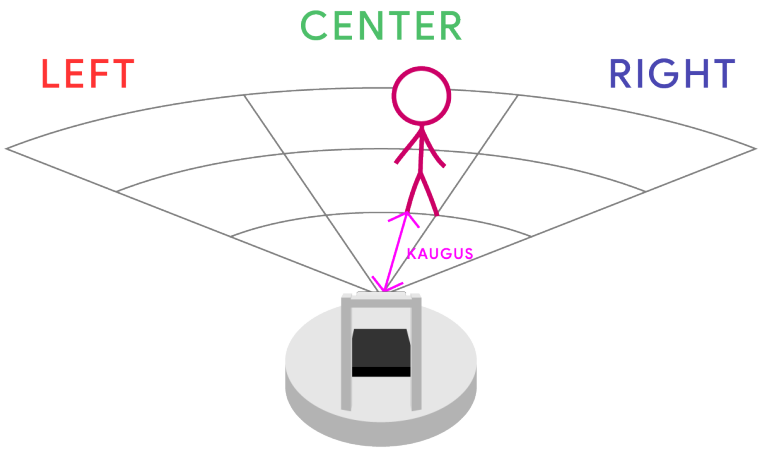
Kolmanda praktikumi eesmärk on tutvustada kaugusandurit, *bang-bang* kontrollerit ehk kaks-punkt regulaatorit ning tingimuslauset.

4.3.2 Õpiväljundid

- Õppur teab, mis on algoritmi roll robotis.
- Õppur oskab Pythonis tingimuslausete abil kirjeldada kaks-punkt regulaatori, mis paneb roboti hoidma pikivahet eespool paikneva objektiga.
- Õppur oskab koostada programmi, mille abil robot väldib takistusi ning läbib kitsaid avausi.

4.3.3 Sisu

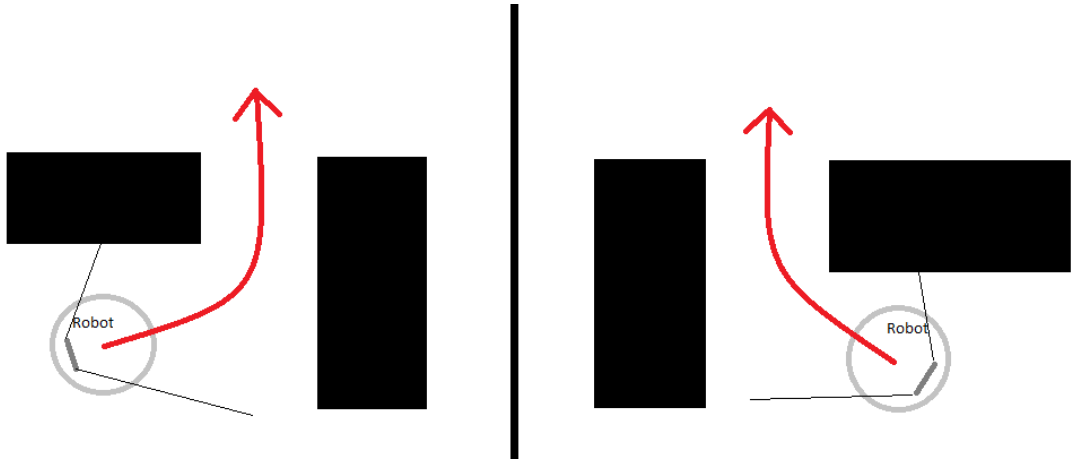
Alustuseks laadib õppur robotisse oma eelmises praktikumis loodud Git-koodihoidla, kus on ka praktikumiks vajalikud ülesandemallid, mida õppur muutma hakkab. Praktikumi alguses antakse ülevaade Robotondil olevatest anduritest. Selleks praktikumiks on ettevalmistatud spetsiaalne tarkvara, mis töötleb infot Robotondi sügavuskaamerast ja teeb sellest 2D laserskanni. Järgmise sammuna eraldatakse sellest skannist kolm sektorit (sele 4.1) ja leitakse igast sektorist robotile lähim punkt.



Sele 4.1. *Laserskann on jaotatud kolmeks sektoriks, kus tuvastatakse vahemaa lähima takistuseni.*

Praktikumitöö on jagatud kolmeks alamülesandeks.

1. Esimesena käivitab õppur faili, mis paneb tööle kaamera draiveri, mootori draiveri ning sõlme, mis teeb eelpool mainitud arvutusi. Seejärel kasutab õppur käsku „*rostopic echo*”, et näha, mis on ROS-rubriigi „*/scan_to_distance*” sisu, kuhu kuulutatakse kaugusi igas sektoris.
2. Esimeses programmeerimisülesandes kasutatakse vaid keskmise sektori kaugust, et kirjeldada kaks-punkt regulaator, mille abil üritab robot hoida end takistusest ühel kindlal kaugusel.
3. Teise programmeerimisülesande eesmärk on sõita robotiga autonoomselt läbi kitsa vahe seinte vastu põrkamata. Lisaks tuleb õppuril arvestada mitme võimaliku algolukorraga nagu on toodud selel 4.2.



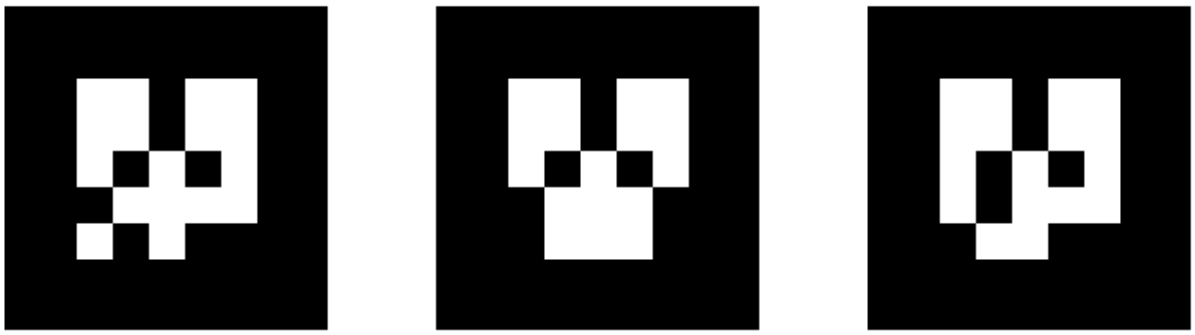
Sele 4.2. *Robot peab autonoomselt vahest läbi sõitma mõlema seadistuse puhul.*

Praktikumi lõpus laadib õppur valminud lahendusprogrammide lähtekoodid üles oma Git-koodihoidlasse.

4.4 Praktikum 4 — AR-märgised

4.4.1 Eesmärk

Neljanda praktikumi eesmärk on tutvustada AR- ehk liitreaalsuse märgiseid (sele 4.3) ja nende abil roboti juhtimist.



Sele 4.3. *AR-märgised kimbust ar_track_alvar [45].*

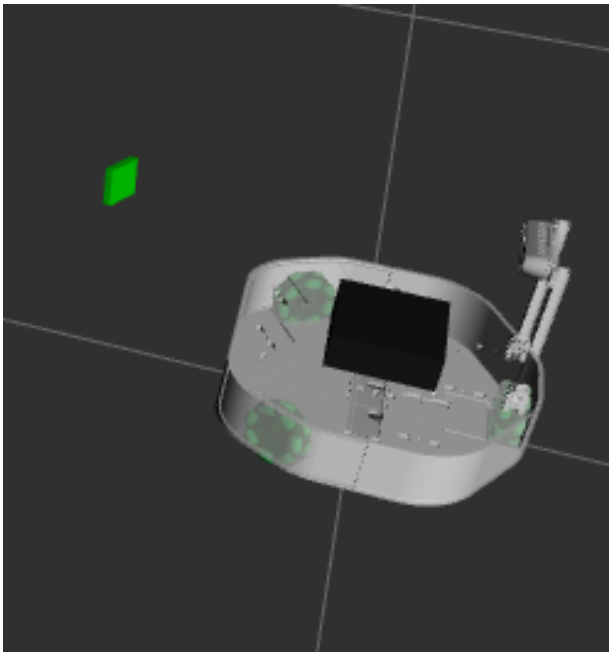
4.4.2 Õpiväljundid

- Õppur suudab mõtestada AR-märgise liikumist roboti koordinaatteljestiku suhtes.
- Õppur teab, kuidas koordinaatteljestik roboti suhtes asetseb ning kuidas toimuvad selles lineaar- ning nurkteisendused.
- Õppur kasutab kaks-punkt regulaatorit, et robot joondaks end vaateväljas paikneva AR-märgise järgi.

4.4.3 Sisu

Praktikumi alguses laadib õppur robotisse oma Git-koodihoidla. Algul tutvustatakse, mis on liitreaalsus ning AR-märgised.

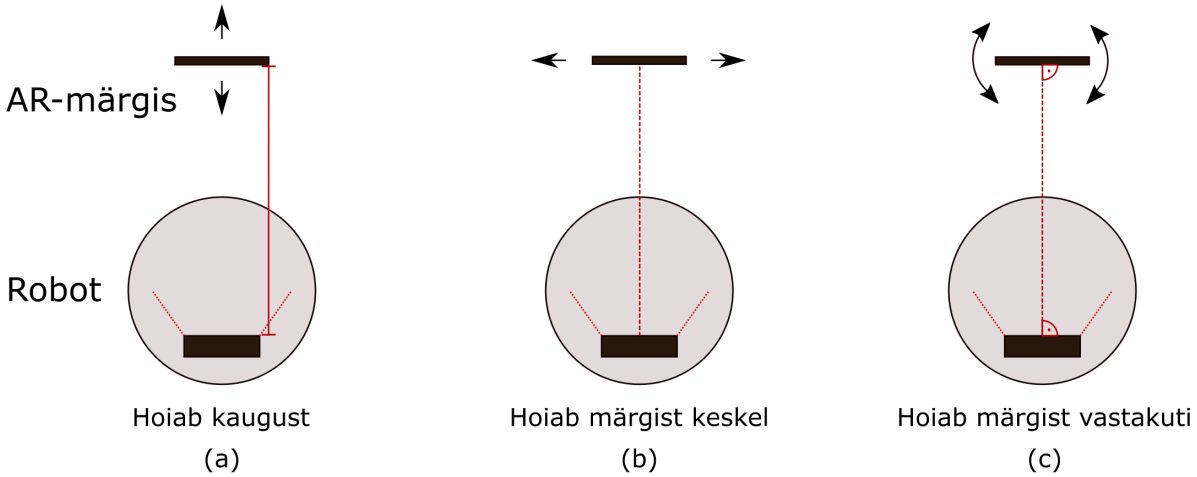
1. Õppur käivitab draiveri, mis laseb robotiga liikuda ning tuvastab AR-märgiseid. See draiver jäetakse käima kogu praktikumi ajaks.
2. Õppur käivitab ROSi visualisatsioonikeskkonna RViz, kus kuvatakse roboti mudel. Kui robot on tuvastanud AR-märgise, kuvatakse samas aknas ka märgise asukoht roboti suhtes (sele 4.4).
3. Pärast esmast tutvust AR-märgisega kirjeldatakse koordinaatteljed ning pöörded ümber telgede.
4. Seejärel käitab õppur skripti, mis esialgu prindib terminaliaknasse vaid AR-märgisele vastava numbri ja asendi kaamera koordinaatsüsteemis, eesmärgiga saada aru, kuidas markerit erinevat moodi liigutades väärtused muutuvad.



Sele 4.4. *Keskkonnas RViz kuvatud Robotondi mudel ning rohelisega AR-märgise asukoht roboti vaates.*

Edasi tuleb kirjeldada kolme funktsiooni sisu ette antud Pythoni koodifailis.

1. Esimene funktsioon hoiab robotit märgisest x-teljes teatud kaugusel (sele 4.5a).
2. Teine funktsioon hoiab robotit märgisest y-teljes teatud kaugusel (sele 4.5b).
3. Kolmas funktsioon pöörab roboti vastakuti märgisega (sele 4.5c).



Sele 4.5. *Kirjeldatavad funktsioonid: kauguse hoidmine (a), märgise keskel hoidmine (b) ja märgise vastakuti hoidmine (c).*

Kui need kolm funktsiooni on õigesti kirjeldatud, jälitab robot AR-märgist. Mittekohustusliku lisaülesandena on võimalik teha programm, mis võtab arvesse markeril oleva numbrilise info ja paneb roboti eri väärtuste korral erinevalt tegutsema. Lõpuks tuleb muudatused laadida üles oma Git-koodihoidlasse.

4.5 Praktikum 5 — 2D-kaardistamine ja navigeerimine

4.5.1 Eesmärk

Antud praktikumitöö eesmärk on tutvustada õppurile 2D-kaardistamist ning autonoomset navigeerimist. Täiendavalt demonstreerib praktikumitöö, kuidas autonoomsete sõidukite keeruline tehnoloogia on ROSi abil kõigile kergesti kättesaadav.

4.5.2 Õpiväljundid

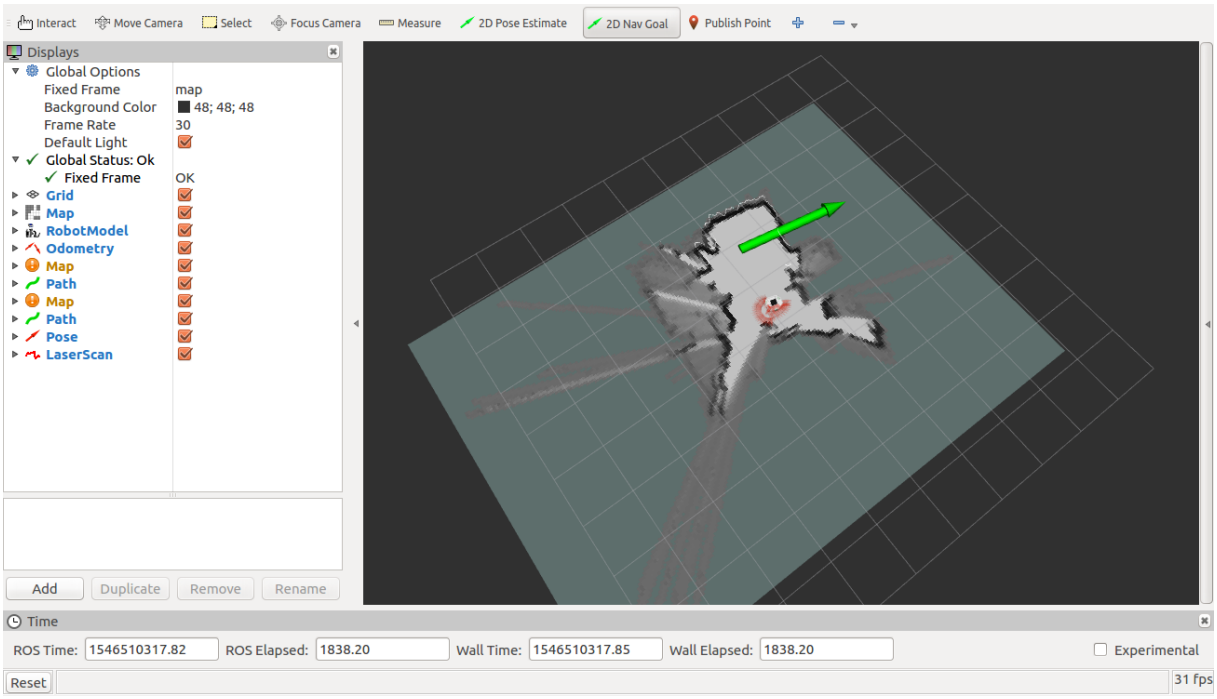
- Õppur teab, kuidas kujutatakse ühemõõtmelist laserskanni ning mõistab selle eeliseid ja puudusi roboti kaugjuhtimisel.
- Õppur oskab kasutada eelseadistatud ROSi kimpe ruumi 2D-kaardistamiseks.
- Õppur oskab RVizi ja automaatselt loodud 2D kaardi abil robotit autonoomses režiimis juhtida.
- Õppur selgitab, millised parameetrid mõjutavad roboti tegutsemist teekonna planeerimisel ja läbimisel.

4.5.3 Sisu

Praktikum tutvustab kahemõõtmelist kaardistamist roboti ja ühemõõtmelise laserskanni abil. Praktikumitöö jaguneb kolmeks ülesandeks:

1. Esmalt visualiseeritakse RVizi abil laserskann. Tuginedes vaid ekraanil kujutatud laserskanni hetkväärtusele, tuleb õppuril robotiga läbida varjatud takistusrada.
2. Õppuri järgmine ülesanne on tundmatu ala kaardistamine, kasutades eelseadistatud ROSi kiirkävitusfaile Google Cartographer [46] tarkvara kasutamiseks. Ülesande vältel juhib õppur arvutiklaviatuuri abil vaateulatusest väljas paiknevat robotit ning jälgib ekraanil roboti ümber tekkivat ja aina täpsustuvat kaarti. Kui kaart tundub terviklik, siis kasutatakse RVizis graafilise kasutajaliidese elemente, et käivitada kaardipõhine rajaplaneerimine ning roboti autonoomne liikumine kahe punkti vahel (sele 4.6).
3. Järgmiseks uuritakse eksperimentaalselt, kuidas mõned olulisemad parameetrid (pöörd- ja lineaarsuunalised kiirenduspiirangud ning lõpppunkti pöörd- ja lineaartolerantsid)

mõjutavad roboti tegutsemist autonoomses režiimis. Õppur täidab nende parameetrite vähendamise ning suurendamise mõju kohta tabeli ja analüüsib roboti käitumist iga parameetri muutmisel.



Sele 4.6. *Keskkonnas RViz kuvatud Robotondi mudel, koostatud kaart ning rohelise noolega märgitud sihtpunkt (ingl k 2D Nav Goal).*

4.6 Praktikum 6 — 3D-kaardistamine

4.6.1 Eesmärk

Praktikumitöö eesmärk on tutvustada õppurile punktipilve ning 3D-kaardistamist. Täiendavalt demonstreerib praktikumitöö, kuidas tundmatute alade inspekteerimisel kasutatav keeruline tehnoloogia on ROSi abil kõigile kergesti kättesaadav.

4.6.2 Õpiväljundid

- Õppur teab, mis on punktipilv, ning kogeb selle eeliseid roboti ümbruse hindamisel võrreldes kaamerapildi või laserskanniga.
- Õppur oskab kasutada eelseadistatud ROSi kimpe ruumi 3D-kaardistamiseks.
- Õppur mõistab olulisemate parameetrite mõju 3D kaardi kvaliteedile.

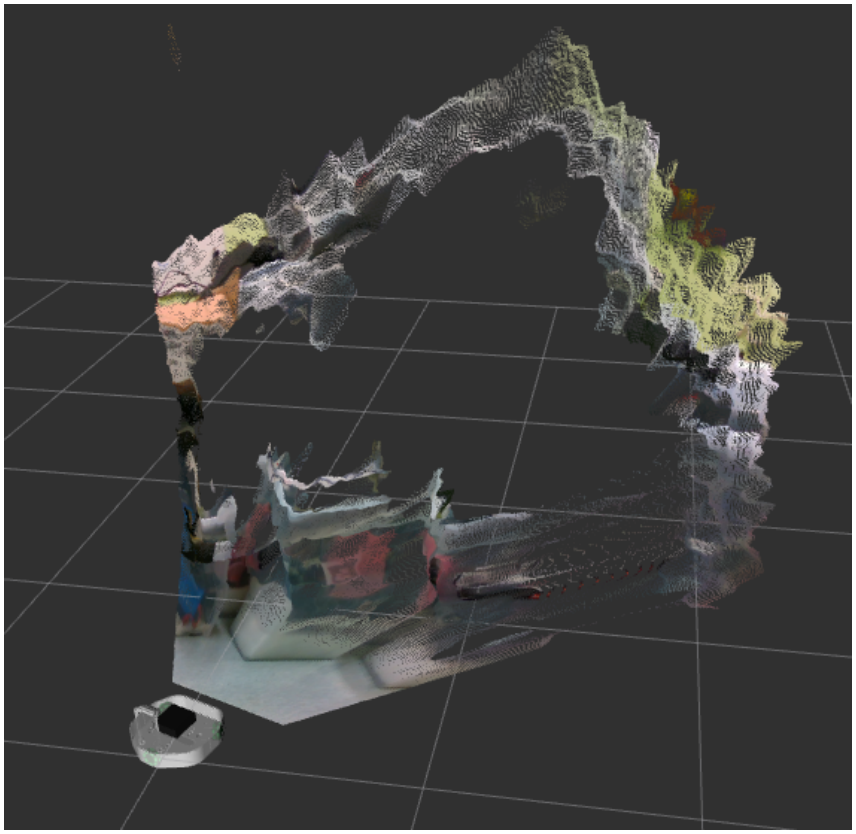
4.6.3 Sisu

Praktikumitöö jaguneb kolmeks ülesandeks: punktipilve ja selle omadustega tutvumine, 3D-kaardistamine töövahendiga *RTAB-Map* ning 3D-kaardi põhjal roboti kaugjuhtimine.

1. Punktipilve olemusega tutvumiseks visualiseerib õppur RVizis punktipilve, mida väljastab robotondi Realsense D435 sügavuskaamera. Antud punktipilvel on peale tavapärase sügavusteabe iga punkt värvitud lähtuvalt sama kaamera 2D pildist (sele 4.7). Selliselt värvitud punktipilvest on õppuril võrdlemisi lihtne mõista roboti vaatevälja, sest info on esitatud inimloetaval kujul.

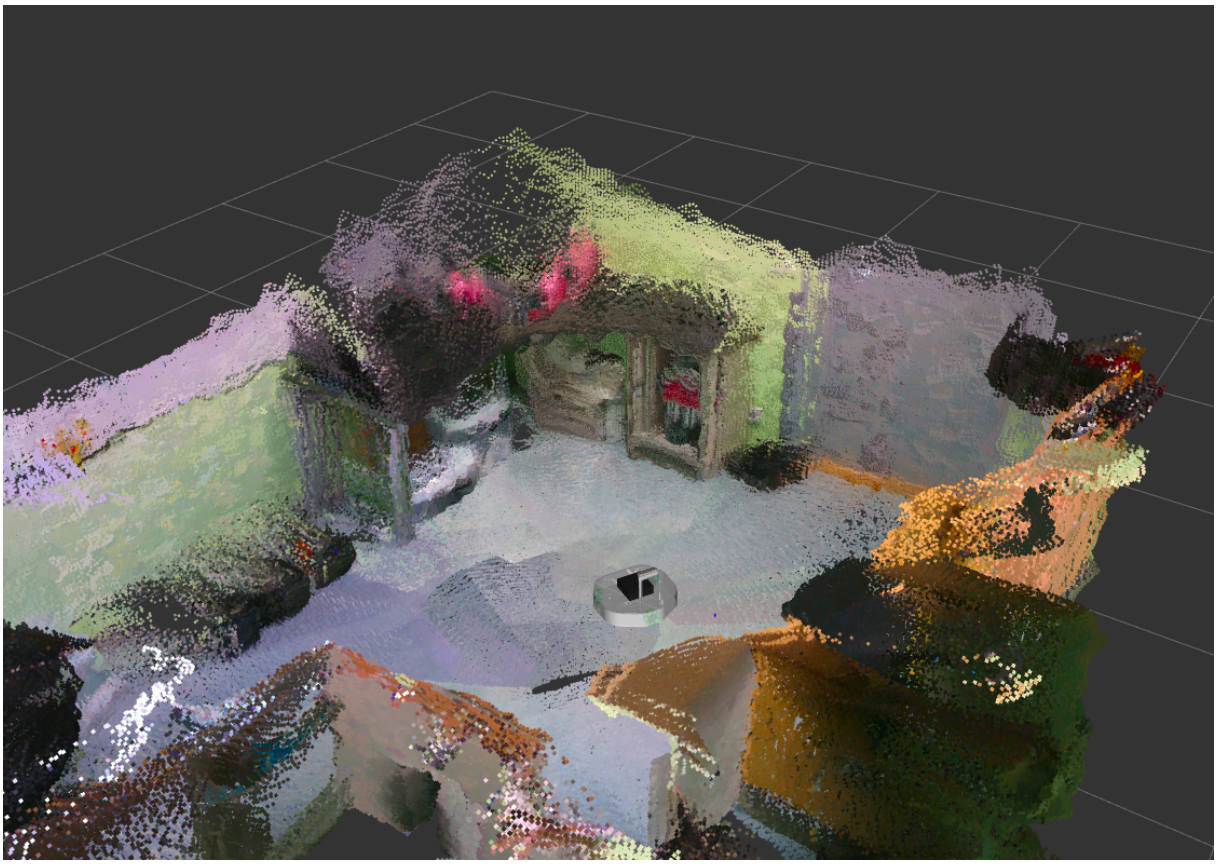
2. Järgmiseks käivitab õppur 3D-kaardistamise kimbu *RTAB-Map* [47, 48]. Eelseadistatud kaardistamise parameetrid annavad tulemuseks äratuntava, aga ebakvaliteetse kaardi. Õppuri ülesanne on eksperimentaalselt uurida, kuidas mõjutavad lineaar- ja nurkuuendussagedus ning punktide vahekaugus loodava kaardi kvaliteeti. Eesmärk on leida selline parameetrite kombinatsioon, mis annab tulemuseks piisavalt kvaliteetse kaardi.

3. Õppur kaardistab ruumi, kus ta ise ei viibi, ning suudab seal ohutult robotit juhtida ja erinevaid esemeid tuvastada.



Sele 4.7. *Keskkonnas RViz kuvatud Robotondi mudel ning värvitud punktipilv.*

Sele 4.8 näitlikustab, milline näeb välja piisavalt kvaliteetne ruumi kaart.



Sele 4.8. *Näide RTAB-Mapiga kaardistatud ruumist.*

4.7 Praktikum 7 — Regulaatorid

4.7.1 Eesmärk

Praktikumi eesmärk on tutvustada õppurile nii kolm-punkt kui ka proportsionaalset regulaatorit ning võrrelda neid varasemalt õpitud kaks-punkt regulaatoriga.

4.7.2 Õpiväljundid

- Õppur teab ja oskab Pythonis kirjeldada kolm-punkt regulaatorit.
- Õppur teab ja oskab Pythnois kirjeldada proportsionaalset regulaatorit.

4.7.3 Sisu

See praktikum on mõeldud eelnevate teemade kinnistamiseks ning kolmandas ja neljandas praktikumis tehtud regulaatorite täiendamiseks. Praktikumi alguses laadib õppur roboti pardaarvutisse oma Git-koodihoidla. Edasine praktikumitöö on jaotatud kolmeks ülesandeks.

1. Esimene ülesanne on muuta oma kolmandas praktikumis tehtud kaks-punkt regulaatori koodi, et tavalise kaks-punkt regulaatori asemel oleks kirjeldatud kolm-punkt regulaator.
2. Teises ülesandes tuleb samas failis asendada kolm-punkt regulaator proportsionaalsega.
3. Kolmandas ülesandes tuleb õppuril muuta oma neljandas praktikumis valminud koodi ja viia AR-märgiste järgi sõitva roboti liikumine sujuvamaks, kasutades proportionaalset regulaatorit.

4.8 Praktikum 8 — Robotsimulatsioon

4.8.1 Eesmärk

Praktikumi eesmärk on tutvustada Gazebo simulatsioonikeskkonda ning demonstreerida selle kasutuslihtsust ROSis loodud programmidega.

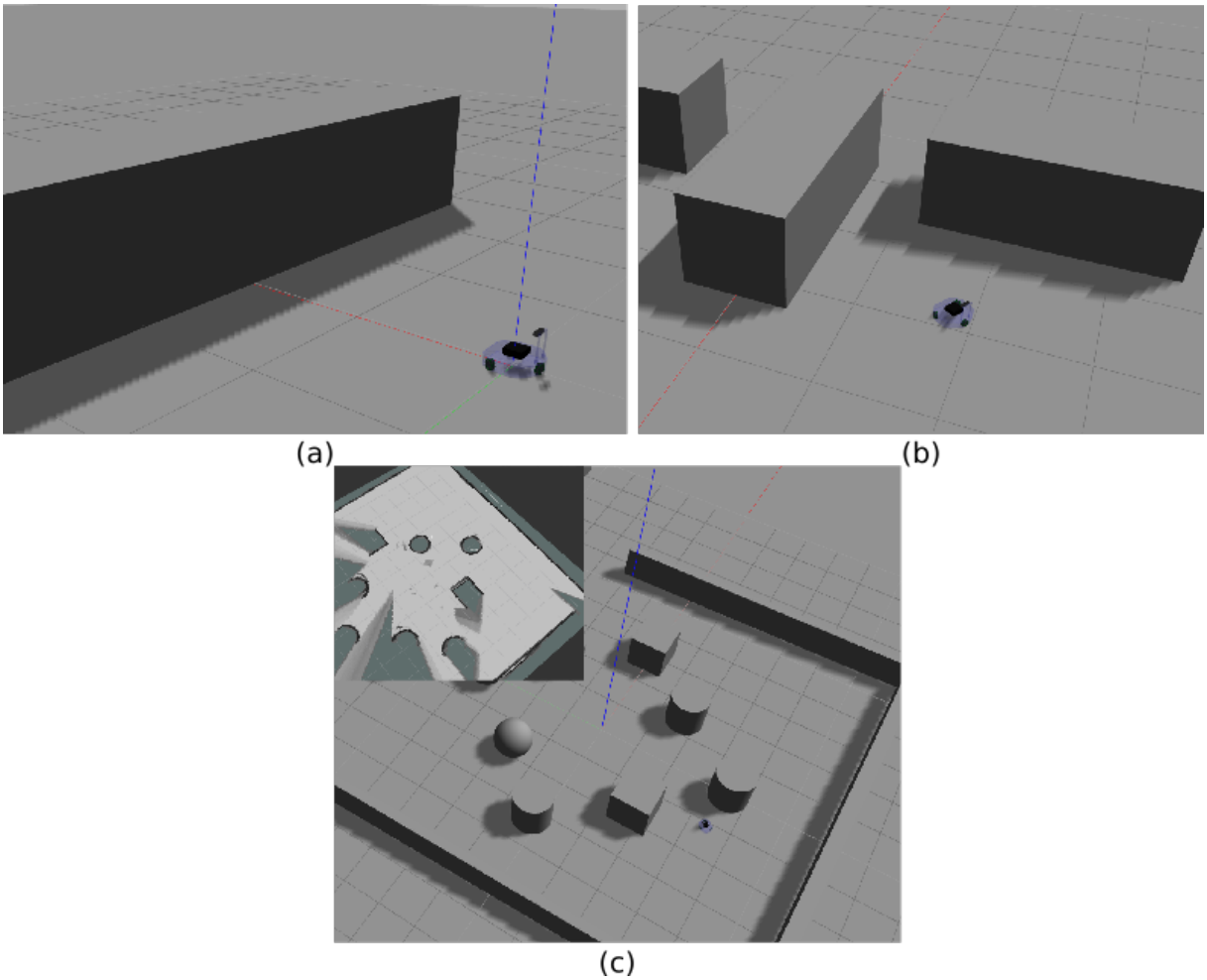
4.8.2 Õpiväljundid

- Õppur oskab käivitada Gazebo simulatsioonikeskkonda eelseadistatud roboti ja maailmaga.
- Õppur oskab Gazebo kasutajaliidese abil liigutada erinevaid objekte simuleeritud maailmas.
- Õppur oskab testida roboti juhtimiseks loodud ROSi programme Gazebo simulatsioonikeskkonnas.

4.8.3 Sisu

Antud praktikumis tehakse tutvust simulatsioonikeskkonnaga Gazebo. Praktikumi edukaks sooritamiseks tuleb õppuril käivitada Gazebo ning testida kolme oma varasemat praktikumilahendust simuleeritud näidismaailmades. Praktikumijuhendis on näitena toodud kolm järgmist ülesannet, kuid õppur võib vabalt valida, millist oma lahendustest soovib testida.

1. Proportsionaalse regulaatoriga seinast ühtlase kauguse hoidmine (7. praktikumitöö 2. ülesanne, sele 4.9a).
2. Autonoomne takistusraja läbimine kaks-punkt regulaatoriga (3. praktikumitöö 3. ülesanne, sele 4.9b).
3. 2D-kaardistamine ja kaardi järgi sõitmine analoogselt 5. praktikumitööle (sele 4.9c) .



Sele 4.9. *Gazebo näidismaailmad: seinast kauguse hoidmine (a), takistusraja läbimine (b) ja kaardistamine (c).*

5. Tarkvara

Praktikumitöödes seatud eesmärkide saavutamiseks tuli oluliselt laiendada Robotondi tarkvaralist funktsionaalsust ning luua nii alusfailid kui ka hõlpsasti kasutatavad ROSi kiirkäivitusskriptid.

5.1 Nõuded tarkvarale

5.1.1 Tarkvarastruktuur

Tarkvarastruktuur ROSi tasandil peab järgima eeskujulikule ROSi robotile (alapeatükk 1.3.1) analoogset struktuuri.

5.1.2 Simulaator

Eeskujulikul ROSi robotil (alapeatükk 1.3.1) on olemas ka Gazebo [41] simulatsioonikeskkond. Vajadus simulatsioonikeskkonna järele tuleneb sellest, et riistvaratõrke korral peab olema võimalus juhendeid läbida simulaatoris. Seda eriti juhul, kui käimas on õppetöö ning rikke kõrvaldamiseks ei ole aega või võimalust. Robotondi puhul on vaja simulaatoris roboti simuleerimiseks kirjeldada:

1. roboti väljanägemine,
2. roboti põrkepiirkond,
3. värvi- ning sügavuskaamera liides,
4. viis, kuidas robotit liigutada.

Kuna töö üks eesmärkidest on piloteerida õppematerjale TÜ Narva kolledžis, siis tuleb simulaatori arendamisel arvestada õppetöös kasutatavate sülearvutitega HP Elitebook Folio 9470m [49]:

- Intel Core i5-3437U @ 1.9 GHz neljatuumaline protsessor
- integreeritud graafikakaart Intel Ivybridge Mobile

- 8 GB muutmälu

Valminud simulatsiooni käitamise reaalajafaktor peab olema vähemalt 1. Reaalajafaktor iseloomustab suhet simuleeritud ning reaalaja vahel.

5.2 Tarkvaraarendus

5.2.1 Arenduspõhimõtted

Olemasolevate ROSi kimpude kasutamine

Kuna ROSi üks ideoloogiaid arenduses on kimpude taaskasutus [50], siis kasutatakse sama põhimõtet ka käesolevas töös. Kui mingi funktsiooni täitmiseks on võimalik kasutada kellegi poolt arendatud vabavaralist kimpu, siis seda ka tehakse.

ROSi kimpude struktuur

Robotondi tarkvara arendamisel lähtutakse eeskujuliku ROSi roboti (alapeatükk 1.3.1) struktuurist.

5.2.2 Kasutatud töövahendid

Versioonihaldustarkvara

Tänapäeval on versioonihaldus tarkvaraarenduse keskne osa. Seda on vaja, et koodimuudatusest oleks säilinud ajalugu, ning samuti lihtsustab see märkimisväärselt arendajate koostööd. Käesolevas töös on kasutatud versioonihaldustarkvara Git [51] keskkonnas GitHub [52].

Operatsioonisüsteem (OS)

Selles töös kasutati operatsioonisüsteemi Ubuntu 16.04, mis valiti, et toetada Robotontide peal kasutatud ROSi versiooni Kinetic. Seda OSi kasutati nii tarkvara kui ka õppematerjalide arenduseks.

Tekstiredaktor

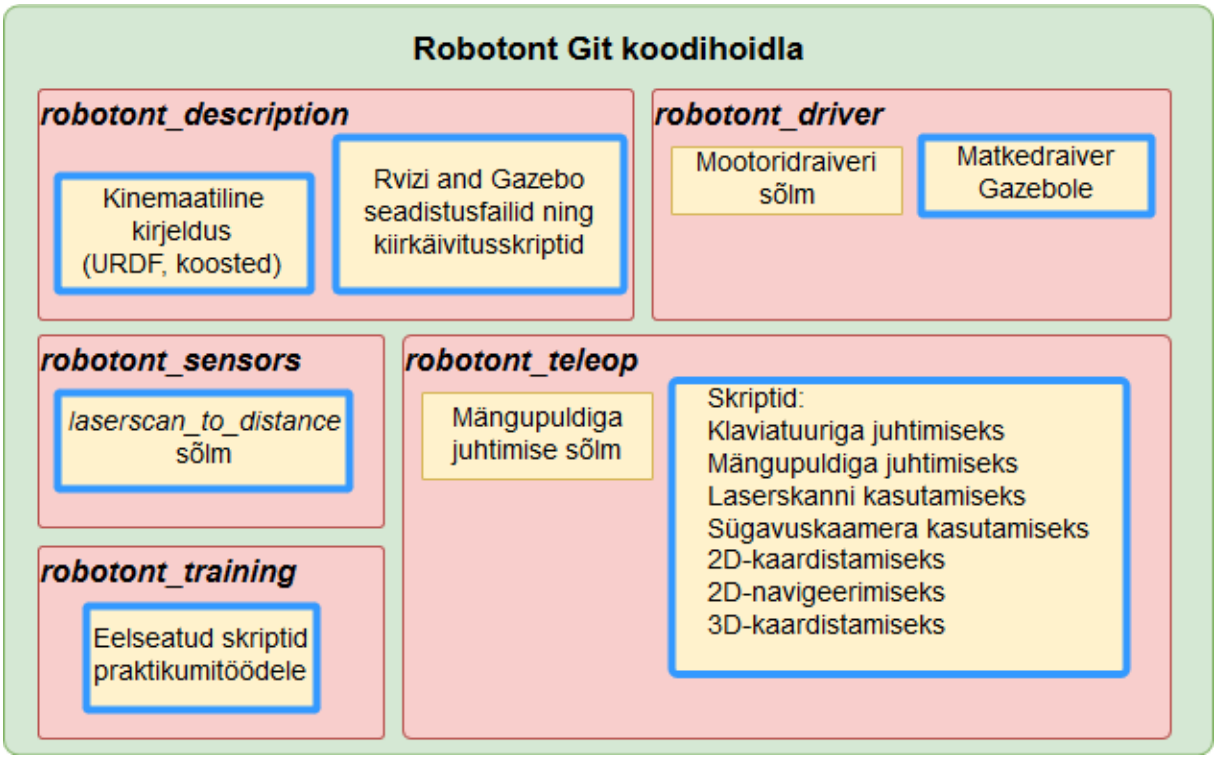
Käesolevas töös kasutati peamise tekstiredaktorina Visual Studio Code'i [53]. Selle laiendatava funktsionaalsusega tekstiredaktori eelis tavalise tekstiredaktori ees on programmeerimiskeele süntaksi markeerimine, arukas koodi lõpetamine ja paljud muud programmeerimist hõlbustavad abivahendid. Visual Studio Code võimaldab liideste abil toetada erinevaid keeli ja on toetatud erinevatel OSidel. Seetõttu on see sobiv valik antud töös, kus on vaja redigeerida erinevat tüüpi tekstifaile ning kohati võib toimuda õppematerjalide arendus ka erineval operatsioonisüsteemil, näiteks Windowsil.

Õppematerjalide publitseerimine

Kuna eesmärk on õppematerjalid avalikult kättesaadavaks teha, siis tuli leida selleks sobiv platvorm. Varasemalt olid Robotondile tehtud õppematerjalid Google Docs [54] keskkonnas ning eksporditud PDF-formaati. PDFis tehtud juhendid kopeeriti õppearvutitesse. Kuna sellisel kujul õppematerjalide uuendamine ja haldamine on pigem tülikas, loodi käesolevas töös õppematerjalid reStructuredText märgistuskeeles [55], millest dokumentatsioonigeneraator Sphinx [56] loob HTML-veebilehe. Genereeritud veebileht on tehtud kättesaadavaks GitHub Pages [57] teenuse kaudu. Samuti on võrguühenduse puudumisel võimalik vaadata juhendeid lokaalselt Giti koodihoidlast.

5.3 Tarkvarastruktuur

Töö raames arendati Robotondile tarkvara, järgides eeskujuliku ROSi roboti (alapeatükk 1.3.1) tarkvarastruktuuri. Valminud tarkvarastruktuur on näidatud selel 5.1. Osa tarkvarast tuleb käivitada robotil (mootoridraiver, kaameradraiver, kaardistamistarkvara) ja teine osa, mis vajab graafilist keskkonda, sülearvutis (RVizis visualiseerimine, navigeerimiskäskude andmine, simulatsioon). Kuid on ka sõlmed, mis töötavad nii roboti kui sülearvuti peal (roboti klaviatuuriga juhtimine, eelseatud skriptid praktikumitööle).



Sele 5.1. *Koodihoidla [44] struktuur. Sinise raamiga on esile toodud antud magistritöö raames arendatud komponendid.*

robotont_description

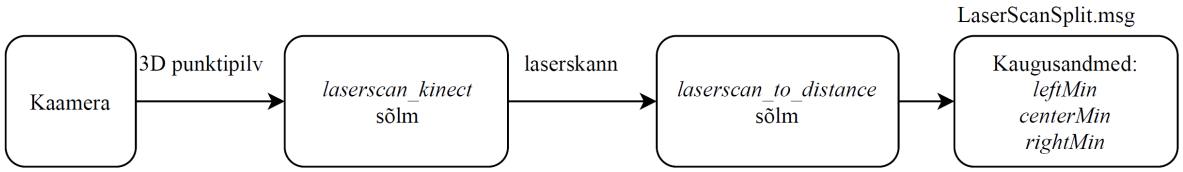
Kimbu *robotont_description* (sele 5.1) alla kuuluvad roboti kinemaatiline kirjeldus, välimus ja Gazebo simulatsioonikeskkond. Simulatsioonikeskkonda ning selleks vajalikke komponente on kirjeldatud alapeatükis 5.4. Visualiseerimiseks on selles kimbus RVizi seadistusfailid, mis võimaldavad käivitada RVizi kindlaks otstarbeks eelseadistatud kujul. Seadistusfailidest saab määrata RVizis laaditavad pistikprogrammid ja nende parameetrid. Näiteks kui kasutada seadistusfaili „robotont_show_depth.rviz", siis laaditakse vastavalt seadistatud moodulid roboti kinemaatiline kirjelduse ja sügavuskaamera pildi kuvamiseks (sele 4.7).

robotont_driver

Kimbus *robotont_driver* asub tavaline ning matkedraiver roboti rataste kontrollimiseks. Tavaline draiver saadab üle jadaliidese roboti STM32 arendusplaadile edasi sõnumeid, mis suunas robot liikuma peaks. Iga ratta madalama taseme kontrolli eest vastutab arendusplaat. Matkedraiveri roll on simuleerida madalama taseme kontrolli. Selle abil saadetakse Gazebosse iga ratta kiiruse väärtused, et liikuda soovitud suunas. Hetkeseisuga ei ole matkedraiver kasutuses, sest selline lähenemine ei osutunud piisavalt efektiivseks madalama võimsusega arvutitel (vt peatükk 5.4).

robotont_sensors

Kimbus *robotont_sensors* asub ROS-sõlm, mis jälgib laserskanni sõnumeid ning kuulutab tingliku kaugusanduri näite (sele 5.2). Iga laserskann jaotatakse kolmeks sektoriks ning määratakse lähima objekti kaugus antud sektoris. Sõlm on loodud kolmanda praktikumijuhendi jaoks (alapeatükk 4.3). Sõlmele on loodud eraldi sõnumitüüp, mille sisus seatakse iga sektori minimaalne väärtus.



Sele 5.2. *Punktipilve muutmine kaugusanduri näitudeks.*

robotont_teleop

Kimbus *robotont_teleop* asuvad:

- sõlmed, mis täiustavad mängupuldiga roboti juhtimise funktsionaalsust,
- käivitusfailid roboti juhtimiseks,
- käivitusfailid roboti andurite käitamiseks,
- käivitusfailid 2D-kaardistamiseks ja navigeerimiseks ning 3D-kaardistamiseks,
- seadistusfailid kaardistamiseks.

Käivitusfailid võimaldavad käitada palju sõlmi korraga, peites seeläbi õppurite eest „köögipoole".

2D-kaardistamiseks testiti kolme kimpu: *Google Cartographer* [46], *hector_slam* [58] ja *gmapping* [59]. Kuna *Google Cartographer* oli kõige lihtsamini seadistatav ning andis kvaliteetseima kaardi, siis kasutatakse seda ka Robotondiga 2D-kaardistamisel. Koostatud kaardi abil autonoomseks navigeerimiseks kasutati kimpu *move_base* [60]. 3D-kaardistamiseks kasutati tarkvara *RTAB-Map* [47, 48].

robotont_training

Selles kimbus sisalduvad õppuritele vajalikud Pythoni alusskriptid, mida nad praktikumitööde raames täiendavad. See kimp on õppe-eesmärgil viidud eraldi koodihoidlasse nimega *robotont_blank_scripts* [61]. Õppur kopeerib eraldi koodihoidla enda GitHubi kontole, et tal oleks võimalik hoida süstematiseeritud ajalugu ja muudatusi praktikumides tehtud täiendustest. Skriptides on ettevalmistatud ROSi raamistik ja muu vajalik, et õppur saaks keskenduda programmeerimisülesandele.

Praktikum 2 skriptide teemad on:

- ROSi kiirussõnumi saatmine,
- *for*-tsükli õppimine,
- funktsioonide kirjeldamine.

Praktikum 3 skriptid on sarnase struktuuriga nagu 2. praktikumi skripid, aga juurde on lisatud võimalus kasutada kaugusanduri andmeid. Kolmanda praktikumi skriptide teemad on:

- seinast kaugust hoidev kaks-punkt regulaator.
- kokkupõrke vältimise regulaator.

Praktikum 4 skriptid on teistsuguse arhitektuuriga. Skriptides on ettevalmistatud raamistik nii AR-märgiste sõnumite tellimiseks kui ka märgiste info lahtipakkimiseks. Samuti on skriptis kirjeldatud turvameetmed, mis peatavad roboti, kui märgise sõnum ei ole teatud aja jooksul saabunud. Siin on skriptis kirjeldatud kolm funktsiooni, mille õppur ise kirjeldab, kuid mille välja kutsumine on eeldefineeritud:

- funktsioon märgisest kauguse hoidmiseks,
- funktsioon märgise roboti jaoks keskel hoidmiseks,
- funktsioon roboti märgisega vastakuti hoidmiseks.

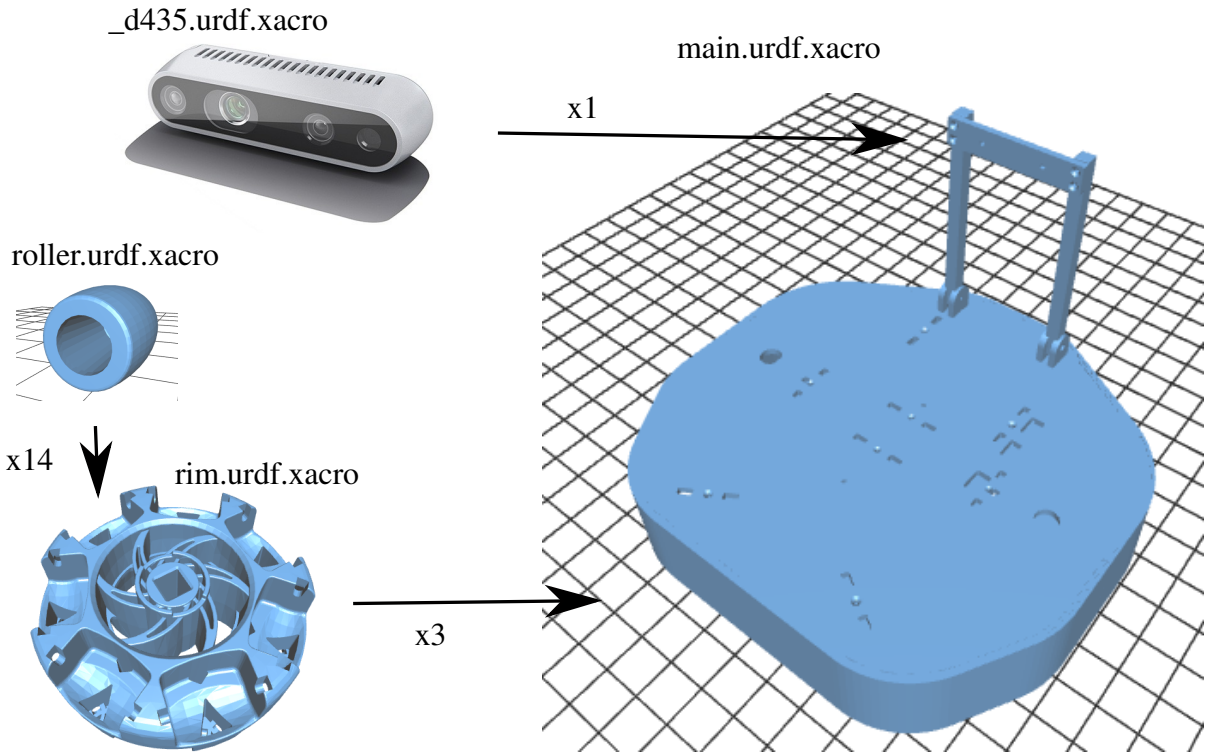
5.4 Simulatsioon

Nagu kirjeldatud alapeatükis 5.1.2, oli üheks Robotondi tarkvaranõudeks simulatsioonikeskkond, mis käesoleva töö raames arendati simuleerimistarkvaras Gazebo [41].

Roboti Gazebo simulatsiooni loomiseks on tarvis luua:

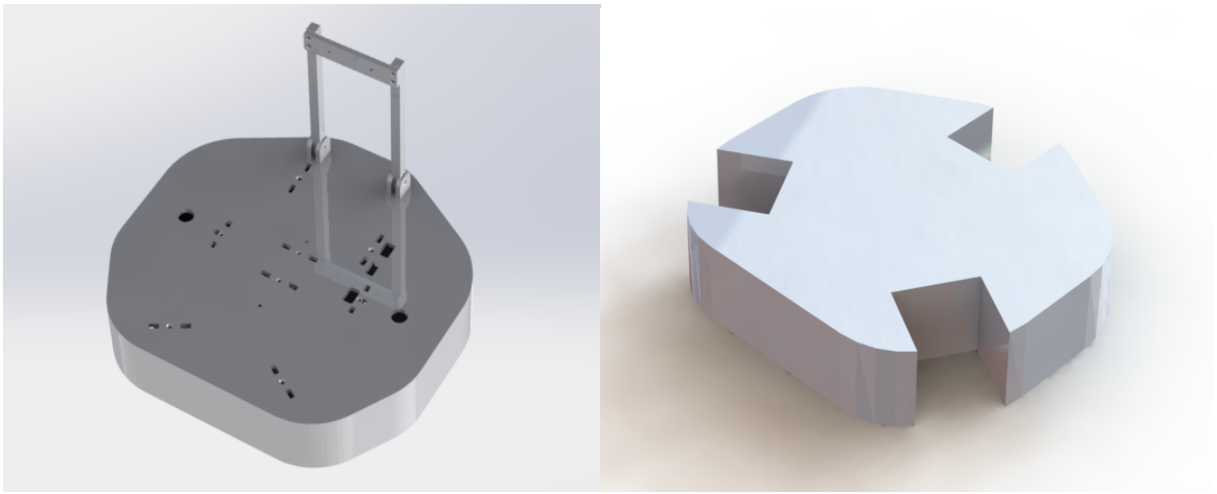
1. Roboti URDF mudel (soovitatavalt dünaamilises Xacro formaadis). Mudelis peavad olema kirjeldatud:
   (a) iga lüli asukoht teiste suhtes,
   (b) iga lüli väljanägemine,
   (c) iga lüli põrkepiirkond,
   (d) iga lüli jaoks Gazebo materjal,
   (e) liidesed, mille abil simuleeritakse sensoorikat ning aktuaatoreid.
2. Gazebo simuleeritav maailm, kuhu virtuaalne robot paigutub.

Esimene väljakutse oli Robotondi mudel kirjeldada URDFis ja/või Xacros [40]. Robotondi mudeli jaoks on neli erinevat Xacro-faili (sele 5.3): omniratta velg, omniratta rullik, kaamera ning peafail, mis kirjeldab ära kogu roboti, kaasates eelnevaid Xacro-faile.



Sele 5.3. *Xacro failide omavaheline arhitektuur. Omniratta rullik, omniratta velg, Intel Realsense D435 ja kere.*

Intel Realsense D435 [43] kaamerafail on pärit Realsense'i ametlikust koodihoidlast [62]. Omniratta rulliku failis kirjeldatakse rulliku visuaalne väljanägemine mudelifaili abil ning parameetrite abil defineeritakse rulliku asukoht. Rullikul füüsilist piiret ei ole. Omniratta velje failis kirjeldatakse velje visuaalne väljanägemine mudelifaili abil, füüsilise piirdena on defineeritud silinder ja ratta asukoht määratakse parameetrite abil. Põhifailis on kirjeldatud kere väljanägemine, kus on mudelina kasutatud Robotondi originaalset 3D joonist. Selleks, et lihtsustada füüsika simuleerimist, kasutatakse füüsilise piirdena lihtsustatud mudelit (sele 5.4), millelt on eemaldatud kõik üleliigsed nurgad ja augud. Samuti defineeritakse rataste asukohad kere suhtes ning ühekordsed mittesümmeetrilised teisendused, näiteks kere ja arvuti ning kere ja kaamera vahelised teisendused.



(a) (b)

Sele 5.4. *Detailne mudel (a) ja lihtsustatud mudel simulatsiooni füüsika jaoks (b).*

Gazebo liidestena on Robotondi puhul peamiselt vaja värvi- ja sügavuskaamera ning roboti liikumise simuleerimist. Värvi- ja sügavuskaamera jaoks on kasutatud Gazebo liidest „gazebo_ros_openni_kinect" [63], mille parameetrid on seadistatud olema sarnased Intel Realsense D435 omadega. Robotondi liigutamiseks prooviti esmalt kasutada sama lähenemist nagu reaalse roboti puhul. Kirjutati regulaator, mis vastavalt sisendile liigutab iga ratast eraldi. Kuigi võimsamal arvutil oli selle regulaatori kasutamine sarnane reaalse roboti liikumisele, siis sellise regulaatori kasutamine viis õppearvutil reaalajafaktori liiga madalale (~0.6) ning liikumine muutus aeglaseks ja mittekontrollitavaks. Selle parandamiseks tuli leida alternatiivne viis roboti liigutamiseks. Sobiva lahenduse pakkus liides „gazebo_ros_planar_move" [63]. See liides tõlgib ROSi *geometry_msgs/Twist*-tüüpi sõnumi otse liikumiseks tasandil, arvestamata füüsikalisi omadusi nagu inerts ja hõõrdejõud. Selle liidese abil töötab simulatsioon reaalajas, kuid kaotab liikumisel realistlikkust — roboti kiirendus on hetkeline.

Tavalisi URDF failis kirjeldatud lülide väljanägemise värve ei loeta sisse Gazebos. Selleks, et robotil Gazebos realistlikum väljanägemine oleks, tuleb URDFis kirjeldada eraldi iga lüli jaoks Gazebo materjal. Kui tahta kasutada materjali, mida pole Gazebos defineeritud, siis tuleb kirjeldada selleks eraldi fail. Näiteks on Robotondi kere jaoks defineeritud sinakas materjal selel 5.5.

Simulatsiooni üks eesmärk on katsetada robotit olukordades, kus see päriselus ei ole võimalik või on ohtlik. Gazebo abil on võimalik valmis teha simulatsioonimaailm ja see koos Robotondi mudeliga Gazebos laadida ning seejärel roboti käitumist virtuaalses keskkonnas simuleerida. Õppematerjalide ühe osana arendati välja Gazebo maailmad kaks-punkt regulaatori, takistuste vältimise algoritmi ja kaardistamislahenduste testimiseks (sele 4.9).

```
material Robotont/BlueishTransparent
{
  technique
  {
    pass
    {
      scene_blend alpha_blend
      depth_write off

      ambient 0.0 0.0 0.7 0.01
      diffuse 0.1 0.1 0.8 0.1
      specular 0.1 0.1 0.6 1.000000

      texture_unit
      {
        colour_op_ex source1 src_current src_current 0 1 0
        alpha_op_ex source1 src_manual src_current 0.7
      }
    }
  }
}
```

Sele 5.5. *Läbipaistev sinakas material Robotondi kere jaoks.*

6. Arutelu ja järeldused

6.1 Tarkvara

Töö raames arendatud tarkvara on suur edasiminek algsest seisust. Valmis simulaator, mis võimaldab robotit kasutamata robotplatvormile programme arendada. Simulaator arendati eesmärgiga, et see töötaks madalama võimsusega arvutitel ning see eesmärk ka täideti. Koodihoidlasse jäi alles ka realistlikuma simulatsiooni tarkvara, kuid selle käitamiseks peab olema piisavalt suure jõudlusega arvuti. Samuti panustati tarkvarastruktuuri arendusse, järgides ROSi häid tavasid. Siiski on mõningate kimpude sisu võimalik muuta veelgi läbipaistvamaks. Näiteks on hetkel juhtimissõlmede kaustas ka kogu kaardistamis- ning navigeerimisfunktsionaalsus, mis keerukuse kasvades tasuks paigutada eraldi kimpu nimega *robotont_mapping*. Samuti saaks kinemaatilise kirjeldusega kimbust eraldada visualiseerimise *robotont_viz* ning simulatsioonikeskkonna *robotont_simulation* kimbu.

6.2 Õppematerjalid ja pilootkursus Narva kolledžis

Pilootkursusega töötati läbi valminud õppematerjalid sihtgrupi peal ning seeläbi lihviti juhendidest välja suuremad vead. Pilootkursuse käigus selgus, et juhendite maht on valitud sihtgrupile piisav. Aine läbimise tingimuseks oli läbida 75% praktikumitöödest ning esitada digitaalne õpipäevik, kus tudeng koostas lühikese enesereflesiooni läbitud praktikumi kohta. Ainele registreerus 18 tudengit. Tulemused kirjutamise seisuga (17.05.2019) on:

- 6 — arvestatud,
- 3 — mittearvestatud.
- 8 — läbinud vähemalt 75% praktikumitöödest, kuid pole veel esitanud õpimappi,
- 1 — mitteilmunud.

Praktikumide läbimine on toodud selel 6.1.

Sele 6.1. *Läbinute arv iga praktikumi kohta.*

Praktikum	Läbinute arv
Praktikum 1	17
Praktikum 2	15
Praktikum 3	16
Praktikum 4	15
Praktikum 5	12
Praktikum 6	11
Praktikum 7	6
Praktikum 8	3

6.3 Edasised plaanid

Kuna tarkvarastruktuur ei jälgi ideaalselt eeskujuliku ROSi roboti mudelit, siis üks ülesanne on viia Robotondi koodihoidla eeskujuliku mudeliga vastavusse, et Robotonti oleks lihtsam kasutada. Lisaks tuleb süübida koodi kvaliteeti failide tasemel — korrastada ning dokumenteerida kood. Kuna Robotonti turustatakse nimega ClearBot, siis võiks kaaluda nimevahetust ka koodihoidlas, et vähendada potentsiaalseid segadusi.

Et tagada Robotontide laiahaardeline kasutus, tuleb õppematerjalid tõlkida ka teistesse keeltesse, nt inglise keelde.

Juhendites võiks olla eraldi nupp, millega saab valida kas päris roboti või simuleeritud roboti juhendite vahel.

Kohandatud õppematerjale on järgmisena kavas kasutada Eesti õpetajatele suunatud koolituses, et õpetada neile ROSi ning robootikat, et nad saaksid seda edaspidi gümnaasiumis õpetada.

7. Kokkuvõte

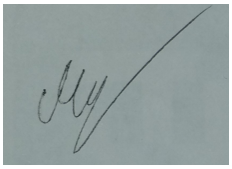
Käesoleva töö raames valmis haridusrobot Robotondile kaheksa praktikumitööd, mis on avalikult kättesaadaval veebis. Praktikumitööd tutvustavad ROSi kasutamist, programmeerimist keeles Python, koodihaldustarkvara Git kasutamist ning Linuxi keskkonna kasutamist. Praktikumijuhendid katavad muuhulgas teemasid nagu regulaatorid, liitreaalsuse markerid, 2D- ja 3D-kaardistamine ning autonoomne navigeerimine.

Praktikumitööde toetamiseks valmis Robotondile suur osa ROSi-põhisest tarkvarast, mis arendati lähtuvalt praktikumitööde teemadest. Robotondile lisandus hulgaliselt tarkvarafunktsionaalsust, näiteks 2D-kaardistamine ning autonoomne navigeerimine ja 3D-kaardistamine. Samuti valmis Robotondi jaoks simulatsioonikeskkond Gazebos. Gazebo simulatsioon on optimeeritud töötama madalama võimsusega sülearvutitel, et toetada selle kasutamist klassiruumides. Simulatsioon võimaldab kasutada kõiki Robotondi andureid ning täitureid. Samuti valmis kolm erinevat näidismaailma, kus simuleeritud robotit kasutada. Koodihoidla on struktureeritud, pidades silmas ROSi kimpude struktuuri välja kujunenud häid tavasid.

Valminud praktikumitööd piloteeriti Tartu Ülikooli Narva kolledžis kursusel LOTI.05.080 „Robotite programmeerimine ROS vahenditega”. Pilootkursuse käigus testiti edukalt tarkvara ning tagasiside põhjal parandati juhendmaterjale.

Tänuavaldused

Autor soovib tänada oma juhendajaid, Karli ja Veikot, suurepärase juhendamise ja abi eest. Lisaks soovib tänada Hannat toe eest kogu magistritöö jooksul, Anetet keeletoimetuse eest ning kõiki teisi, kes on kogu töö käigus seltsiks ja abiks olnud.



Kirjandus

[1] Theodosios Sapounidis, Stavros Demetriadis, and Ioannis Stamelos. Evaluating Children Performance with Graphical and Tangible Robot Programming Tools. *Personal Ubiquitous Comput.*, 19(1):225–237, January 2015.

[2] About Lego Mindstorms EV3. `https://www.lego.com/en-us/mindstorms/about-ev3`. Kasutatud 28.04.2019.

[3] M. Rubenstein, B. Cimino, R. Nagpal, and J. Werfel. AERobot: An affordable one-robot-per-student system for early robotics education. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6107–6113, May 2015.

[4] G. Passault, Q. Rouxel, F. Petit, and O. Ly. Metabot: A Low-Cost Legged Robotics Platform for Education. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 283–287, May 2016.

[5] Makeblock mBot. `https://www.makeblock.com/steam-kits/mbot`. Kasutatud 16.05.2019.

[6] S. Wilson, R. Gameros, M. Sheely, M. Lin, K. Dover, R. Gevorkyan, M. Haberland, A. Bertozzi, and S. Berman. Pheeno, A Versatile Swarm Robotic Research and Education Platform. *IEEE Robotics and Automation Letters*, 1(2):884–891, July 2016.

[7] Farshad Arvin, Jose Espinosa, Benjamin Bird, Andrew West, Simon Watson, and Barry Lennox. Mona: an Affordable Open-Source Mobile Robot for Education and Research. *Journal of Intelligent & Robotic Systems*, May 2018.

[8] F. Mondada, M. Bonani, F. Riedo, M. Briod, L. Pereyre, P. Retornaz, and S. Magnenat. Bringing Robotics to Formal Education: The Thymio Open-Source Hardware Robot. *IEEE Robotics Automation Magazine*, 24(1):77–85, March 2017.

[9] Paulo J.S. Gonçalves, Paulo J.D. Torres, Carlos M.O. Alves, Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and A Martinoli. The e-puck, a Robot Designed for Education in Engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, 1, 01 2009.

[10] Ayush Shukla, Rishabjit Singh, Rishabh Agarwal, Muhammad Suhail, Subir K. Saha, and Santanu Chaudury. Development of a Low-Cost Education Platform: RoboMuse 4.0. In *Proceedings of the Advances in Robotics*, AIR '17, pages 38:1–38:6, New York, NY, USA, 2017. ACM.

[11] TurtleBot3. `http://emanual.robotis.com/docs/en/platform/turtlebot3/`

overview/. Kasutatud 28.04.2019.

[12] M. E. Karim, S. Lemaignan, and F. Mondada. A review: Can robots reshape K-12 STEM education? In *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 1–8, June 2015.

[13] Amanda Sullivan, Mollie Elkin, and Marina Umaschi Bers. KIBO Robot Demo: Engaging Young Children in Programming and Engineering. In *Proceedings of the 14th International Conference on Interaction Design and Children*, IDC '15, pages 418–421, New York, NY, USA, 2015. ACM.

[14] Özgen Korkmaz. The Effect of Scratch- and Lego Mindstorms Ev3-Based Programming Activities on Academic Achievement, Problem-Solving Skills and Logical-Mathematical Thinking Skills of Students. *MOJES: Malaysian Online Journal of Educational Sciences*, 4(3):73–88, 2018.

[15] A. Takacs, G. Eigner, L. Kovacs, I. J. Rudas, and T. Haidegger. Teacher's Kit: Development, Usability, and Communities of Modular Robotic Kits for Classroom Education. *IEEE Robotics Automation Magazine*, 23(2):30–39, June 2016.

[16] V. Chaudhary, V. Agrawal, P. Sureka, and A. Sureka. An Experience Report on Teaching Programming and Computational Thinking to Elementary Level Children Using Lego Robotics Education Kit. In *2016 IEEE Eighth International Conference on Technology for Education (T4E)*, pages 38–41, Dec 2016.

[17] M. U. Bers. Coding, Playgrounds and Literacy in Early Childhood Education: the Development of KIBO Robotics and ScratchJr . In *2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 2094–2102, April 2018.

[18] Amanda A Sullivan, Marina Umaschi Bers, and Claudia Mihm. Imagining, Playing, and Coding with KIBO: Using Robotics to Foster Computational Thinking in Young Children . *Siu-cheung KONG The Education University of Hong Kong, Hong Kong*, 110, 2017.

[19] Blockly. https://developers.google.com/blockly/. Kasutatud 15.05.2019.

[20] Robootikast puust ja punaseks. https://sisu.ut.ee/robot/avaleht. Kasutatud 16.05.2019.

[21] Andrej Zdešar, Sašo Blažic, and Gregor Klančar. Engineering Education in Wheeled Mobile Robotics. *IFAC-PapersOnLine*, 50(1):12173 – 12178, 2017. 20th IFAC World Congress.

[22] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard, and A. Censi. Duckietown: an Open, Inexpensive and Flexible Platform for Autonomy Education and Research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*,

pages 1497–1504, May 2017.

[23] Tony Belpaeme, James Kennedy, Aditi Ramachandran, Brian Scassellati, and Fumihide Tanaka. Social robots for education: A review. *Science Robotics*, 3(21), 2018.

[24] Javier Movellan, Micah Eckhardt, Marjo Virnes, and Angelica Rodriguez. Sociable robot improves toddler vocabulary skills. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 307–308. ACM, 2009.

[25] Takayuki Kanda, Takayuki Hirano, Daniel Eaton, and Hiroshi Ishiguro. Interactive Robots as Social Partners and Peer Tutors for Children: A Field Trial. *Human–Computer Interaction*, 19(1-2):61–84, 2004.

[26] Nichola Lubold, Erin Walker, and Heather Pon-Barry. Effects of voice-adaptation and social dialogue on perceptions of a robotic learning companion. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 255–262. IEEE, 2016.

[27] Catherine C. Chase, Doris B. Chin, Marily A. Oppezzo, and Daniel L. Schwartz. Teachable Agents and the Protégé Effect: Increasing the Effort Towards Learning. *Journal of Science Education and Technology*, 18(4):334–352, Aug 2009.

[28] F. Tanaka, K. Isshiki, F. Takahashi, M. Uekusa, R. Sei, and K. Hayashi. Pepper learns together with children: Development of an educational application. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 270–275, Nov 2015.

[29] Daniela Conti, Santo Di Nuovo, Serafino Buono, and Alessandro Di Nuovo. Robots in Education and Care of Children with Developmental Disabilities: A Study on Acceptance by Experienced and Future Professionals. *International Journal of Social Robotics*, 9:51–62, 02 2017.

[30] ROS - Robot Operating System. `https://www.ros.org/`. Kasutatud 25.04.2019.

[31] Morgan Quigley, Ken Conley, Brian P Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. volume 3, 01 2009.

[32] ROS Introduction. `http://wiki.ros.org/ROS/Introduction`. Kasutatud 25.04.2019.

[33] Stefano Michieletto, Stefano Ghidoni, Enrico Pagello, Michele Moro, and Emanuele Menegatti. Why teach robotics using ROS? *Journal of Automation, Mobile Robotics & Intelligent Systems*, 8:60–68, 01 2014.

[34] ROS Robots. `https://robots.ros.org/`. Kasutatud 16.05.2019.

[35] Clearpath Robotics Husky Git koodihoidla. `https://github.com/husky/husky`. Kasutatud 23.04.2019.

[36] TurtleBot3 Git koodihoidla. `https://github.com/ROBOTIS-GIT/turtlebot3`. Kasutatud 23.04.2019.

[37] Franka Emika Research Robots ROS Git koodihoidla. https://github.com/frankaemika/franka_ros. Kasutatud 23.04.2019.

[38] Simulations for TurtleBot3 Git koodihoidla. https://github.com/ROBOTIS-GIT/turtlebot3_simulations. Kasutatud 24.04.2019.

[39] Unified Robot Description Format. http://wiki.ros.org/urdf. Kasutatud 09.05.2019.

[40] Xacro. http://wiki.ros.org/xacro. Kasutatud 17.04.2019.

[41] Gazebo. http://gazebosim.org/. Kasutatud 17.04.2019.

[42] ClearBot. https://clearbot.eu/. Kasutatud 25.04.2019.

[43] Intel Realsense Depth Camera D435. https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435.html. Kasutatud 21.04.2019.

[44] Robotont Git koodihoidla. https://github.com/ut-ims-robotics/robotont. Kasutatud 24.04.2019.

[45] ar_track_alvar ROS package. http://wiki.ros.org/ar_track_alvar. Kasutatud 13.05.2019.

[46] Google Cartographer. https://google-cartographer.readthedocs.io/en/latest/#. Kasutatud 23.04.2019.

[47] RTAB-Map. http://introlab.github.io/rtabmap/. Kasutatud 23.04.2019.

[48] Mathieu Labbé and François Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[49] HP EliteBook Folio 9470m Notebook PC Product Specifications. https://support.hp.com/ee-en/document/c03559797. Kasutatud 22.04.2019.

[50] Pablo Estefó, Jocelyn Simmonds, Romain Robbes, and Johan Fabry. The Robot Operating System: Package Reuse and Community Dynamics. *Journal of Systems and Software*, 151, 02 2019.

[51] Git. https://git-scm.com/. Kasutatud 20.04.2019.

[52] Github. https://github.com/. Kasutatud 20.04.2019.

[53] Microsoft Visual Studio Code. https://code.visualstudio.com/. Kasutatud 20.04.2019.

[54] Google Docs. https://docs.google.com/document/u/0/. Kasutatud 21.04.2019.

[55] reStructuredText. http://docutils.sourceforge.net/rst.html. Kasutatud 21.04.2019.

[56] Sphinx. http://www.sphinx-doc.org/en/master/. Kasutatud 21.04.2019.

[57] GitHub Pages. https://pages.github.com/. Kasutatud 21.04.2019.

[58] hector_slam ROS package. http://wiki.ros.org/hector_slam. Kasutatud 07.05.2019.

[59] OpenSlam's Gmapping ROS Wrapper. http://wiki.ros.org/gmapping. Kasutatud 07.05.2019.

[60] move_base ROS package. http://wiki.ros.org/move_base. Kasutatud 07.05.2019.

[61] robotont_blank_scripts koodihoidla. https://github.com/ut-ims-robotics/robotont_blank_scripts. Kasutatud 07.05.2019.

[62] Intel Realsense ROS Wrapper for D400 series and SR300 Camera. https://github.com/intel-ros/realsense. Kasutatud 21.04.2019.

[63] Gazebo Plugins Git koodihoidla. https://github.com/ros-simulation/gazebo_ros_pkgs/tree/kinetic-devel/gazebo_plugins. Kasutatud 18.04.2019.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Madis Kaspar Nigol,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Õppematerjalid robotplatvormile Robotont,

   mille juhendajad on PhD Karl Kruusamäe ja PhD Veiko Vunder,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. olen teadlik, et punktis 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Madis Kaspar Nigol,
Tartu, 20. mai 2019. a.

Lisa A. Praktikumitööd

Praktikum 1 — Linux, ssh, klaviatuuriga juhtimine

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum1.html

Praktikum 2 — Programmeerimise ja robootika alused

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum2.html

Praktikum 3 — Andurid ja tingimuslause

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum3.html

Praktikum 4 — AR-märgised

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum4.html

Praktikum 5 — 2D-kaardistamine ja navigeerimine

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum5.html

Praktikum 6 — 3D-kaardistamine

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum6.html

Praktikum 7 — Regulaatorid

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum7.html

Praktikum 8 — Robotsimulatsioon

https://ut-ims-robotics.github.io/robotont/html/labs/praktikum8.html