Tartu Ülikool

Loodus- ja täppisteaduste valdkond

Tehnoloogiainstituut

Raid Vellerind

Avatud robotiarendusplatvormi ROS võimekuse loomine Tartu Ülikooli Robotexi robootikaplatvormile

Bakalaureusetöö (12 EAP)

Arvutitehnika eriala

Juhendaja:

Robootika dotsent Karl Kruusamäe

Tartu 2017

Resümee/Abstract

Avatud robotiarendusplatvormi ROS võimekuse loomine Tartu Ülikooli Robotexi robootikaplatvormile

Tartu Ülikooli Robotiklubis on seoses aastatepikkuse osalemisega rahvusvahelisel robootikavõistlusel Robotex, arendatud välja iseseisev robotiplatvorm. Platvorm on jõudnud sedavõrd stabiilsesse arengujärku, et lisaks robootikavõistlustele on platvormil potentsiaali ka uurimis- ja haridusvaldkonnas. Laialdasema kasutusevõttu võimaldamiseks tuleb antud robotiplatvormile luua standardiseeritud modulaarne tarkvaraliides

Käesoleva lõputöö eesmärgiks on arendada platvormile sobilik tarkvaraline tugi, mis lubaks platvormi liita ka ROSi (*Robot Operating System*) ökosüsteemiga. ROS on tarkvara raamistik, mis sisaldab endas tööriistu ja teeke keerulise funktsionaalsuse lihtsaks implementeerimiseks töökindlatesse robotisüsteemidesse.

Lõputöös antakse ülevaade ROSi ajaloost, struktuurist ja funktsionaalsusest. Töö raames arendati välja funktsionaalne draiver, mis vahendab informatsiooni riistvara ja ROSi kõrgema taseme loogika vahel.

Väljaarendatud draiver loob võimaluse ROSi navigatsiooni teegi ja roboti riistvara vaheliseks suhtluseks, lubades realiseerida keerukaid rajaplaneerimise algoritme. Seejuures lubab draiver robotiplatvormiga liita mistahes platvormile loodud ROSiga ühilduvaid tarkvara pakke.

Tarkvara testiti kahe juhtimismeetodiga: arvutiklaviatuuri ning nutitelefoni näol realiseeritud juhtimispuldiga. Testimise käigus töötas draiver ootuspäraselt ning loob sellega hea baasi edasisteks arendustöödeks.

CERCS: T125 Automatiseerimine, robootika, control engineering

Märksõnad: ROS, robootika, Robotont, draiver, navigatsiooni teek, liikumisbaas, baaskontroller, Robotex

ROS driver development for the University of Tartu's Robotex robotics platform

A robotics platform optimized for competing at international robotics competition Robotex has long been developed within the Robotics Club of the University of Tartu. Throughout years of development, the platform has grown stable enough to look towards new goals. It is not merely a football robot platform anymore, but a research and educational platform as much. The only downside of the platform is a weak support for well-structured modular software.

The purpose of this thesis is to develop such solution for previously stated problems that would also be compatible with ROS (Robot Operating System). ROS provides extensive tools and flexible solutions for complicated robotics tasks.

This thesis provides a detailed description of the history, functionality and structure of ROS. Within the thesis a functional driver was developed that operates as an intermediate software link that connects the high-level packages of ROS with the robot's hardware.

The developed driver provides a way for ROS Navigation stack to communicate with the hardware and make intelligent decisions regarding motion-planning. It also lets the robot interact with any kind of software developed for any other platform working with ROS.

The software was tested with two different types of controllers: a keyboard and a virtual joystick displayed on a smartphone. Both tests were a success and give a good starting point for further development.

CERCS: T125 Automation, robotics, control engineering

Keywords: ROS, robotics, Robotont, driver, navigation stack move_base, base controller, Robotex

Sisukord

Resümee/Abstract 2
Jooniste loetelu 6
Lühendid, konstandid, mõisted 7
1 Sissejuhatus 8
2 Ülevaade probleemist 9
2.1 Robotex 9
2.2 Tartu Ülikool Robotexil 10
2.3 Riistvaraline platvorm 11
2.4 Tarkvaraline platvorm 12
2.5 Varsemad arendused Robotexi platvormil 13
2.6 Projekt „Robotont“ 14
2.7 Käesoleva töö eesmärk 15
3 Robotont ROSi draiver 16
3.1 ROS 16
3.1.1 Ajalugu 17
3.1.2 ROSi tööpõhimõte 17
3.1.3 ROSi modulaarsus 19
3.1.4 Navigatsiooni teek 19
3.2 Tarkvaralised võimalused jadapordi ühenduse realiseerimiseks 20
3.3 ROSi draiveri arhitektuur 21
3.3.1 Driver_node 22
3.3.2 Serial_com_node 24
4 Tulemused 25
4.1 Algne olukord 25
4.2 Esmane funktsionaalne testimine 26
4.2.1 Robotondi juhtimine arvutiklaviatuuriga 26
4.2.2 Robotondi juhtimine nutitelefoniga 28
5 Tulemuste analüüs ja järeldused 31
5.1 Peamised tulemused 31
5.2 Järgmised arendustegevused 31
5.2.1 Jadaliides 32
5.2.2 Odomeetria 32

6 Kokkuvõte 33
Viited 34
Lisad 39
Lisa 1 39
Lihtlitsents 40

Jooniste loetelu

Joonis 1: Liikumisvõimekusega robotiplatvorm 10
Joonis 2: Robotiplatvormi sisemus, liikumismoodulid ja elektroonika 12
Joonis 3: ROSi navigatsiooni teegi struktuur 20
Joonis 4: Robotiplatvormi arhitektuur 22
Joonis 5: Draiver node sisemine matemaatika pseudokoodis väljendatuna koos vastavate sõnumitüüpidega 23
Joonis 6: Serial_com_node sisemine andmete vahendamine pseudokoodis väljendatuna ... 24
Joonis 7: Klaviatuuriga Robotondi juhtimine. 27
Joonis 8: Turtlebot robotiplatvormi kasutajaliides roboti klaviatuuriga juhtimiseks 28
Joonis 9: DroidTick serveri tarkvara kasutajaliidese kuvatõmmis [47] 29
Joonis 10: DroidTick mobiilirakenduse kasutajaliidese kuvatõmmis 30
Joonis 11: Juhtpuldiga Robotondi juhtimine 30

Lühendid, konstandid, mõisted

API – *Application programming interface* – rakendusliides ehk reeglistik valmisprogrammiga suhtlemiseks.

Baaskontroller – *Base_controller* – tarkvaraliides, mis seob ROSi kõrgema taseme loogika roboti riistvaraga.

Jälgija – *Subscriber* – sõlm, mis jälgib sõnumeid mõnel teema.

Avaldaja – *Publisher* – sõlm, mis avaldab sõnumeid mõnel teemal.

Liikumisbaas – *Move_base* – ROSi pakk, mis väljastab roboti liikumiskiiruse rajaplaneerimisalgoritmide põhjal. Liikumisbaas on osa navigatsiooni teegist.

Mbed – ARM Cortex-M mikrokontrollerite baasil põhinev elektroonika platvorm.

Navigatsiooni teek – *Navigation Stack* – ROSi pakk, mis tegeleb navigeerimisega (sh rajaplaneerimisega).

NUC – *Next Unit of Computing* – Inteli pisiarvuti platvorm.

Odomeetria – *Odometry* – kirjeldus roboti liikumise kohta otsese mootorite tagasiside põhjal.

Omniratas – *Omniwheel* – ratas, mille pind on kaetud rullikutega. Võimaldab liikuda igas suunas sõltumata ratta pöörlemisest.

Pakk – *Package* – ROSi tarkvara alamüksus. Üks pakk võib sisaldada teisi pakke.

ROS – *Robot Operating System* – modulaarne robootikaarendus platvorm.

Sõltuvuspakid – *Dependencies* – ROSi pakid, mille olemasolu on vastava ROSi programmi käitamiseks vajalik.

Sõnum – *Message* – informatsioonikandja ROSi siseseks suhtluseks.

Teema – *Topic* – tarkvaraline vahend sõnumite haldamiseks. Iga sõnum avaldatakse konkreetsel teemal.

1 Sissejuhatus

Tartu Ülikool on rahvusvahelise robootikavõistluse Robotex üks korraldajaid ning suurimaid osalejaid. Iga-aastase osalemise tagajärjel on Tartu Ülikooli Robotiklubis arenendatud modulaarne robotiplatvorm. Robotiplatvorm on jõudnud arenguga piisavalt stabiilsesse seisundisse, et lisaks robootikavõistlustele on sellel potentsiaali ka teadusuuringutes ning haridusvaldkonnas olulist mõju avaldada. Senine robotiplatvormi arendus on keskendunud riistvara modulaarsele arengule. Tarkvaraline standardiseeritud tugi robotiplatvormil sisuliselt puudub.

Tarkvaraliseks lahenduseks pakub sobivat võimekust avatud robotiarendustarkvara ROS (*Robot Operating System*), mis oma modulaarse raamistikuga lubab läheneda igale ülesandele paindlikult. ROSi ökosüsteem sisaldab rohkem kui 3000 tarkvarapakki, mis on vabalt kasutatavad ning ühilduvad iga ROSil töötava süsteemiga.

Käesoleva töö eesmärgiks on luua väljaarendatud robootikaplatvormile võimekus ühenduda ROSi ökosüsteemiga. ROSi kasutamiseks on vaja välja arendada robotiplatvormiga ühilduv baaskontroller ehk draiver, mis vahendaks suhtlust roboti riistvara ja ROSi kõrgema taseme loogika vahel.

Käesolev lõputöö on jaotatud kaheks suuremaks peatükiks, millest esimeses (pt 2) antakse ülevaade probleemist, kirjeldatakse juba valmis robotiplatvormi ning sõnastatakse töö eesmärk. Töö teises pooles (pt 3) antakse põhjalik ülevaade ROSist, selle ajaloost ja struktuurist. Pärast mida kirjeldatakse detailselt töö käigus väljaarendatud draiveri struktuuri ning komponente. Töö lõpuosas (pt 4 ja 5) võetakse kokku töös saavutatu ning arutletakse arenduse problemaatilisi külgi ning nende võimalike lahendusi.

2 Ülevaade probleemist

Tartu Ülikool on aastaid korraldanud ja võistelnud rahvusvahelisel robootikavõistlusel Robotex [1,2]. Selle aja jooksul on väljaarendatud mobiilne robotiplatvorm, millel iga-aastaste osalejate robotid põhinevad. Kui riistvaraline pool platvormist on võrdlemisi terviklik [3] ning lubab kiiresti jõuda töötava prototüübini, siis tarkvaraliselt tuleb tudengitel alustada iga aasta nullist. Kuna õppimiskõver selle juures on võrdlemisi järsk, siis kulub esmaselt roboti tööle saamiseks võrdlemisi palju ressurssi. Efektiivsemaks muudaks roboti valmimise olukord, kus lisaks riistvaralisele baasile oleks olemas ka tarkvaraline baas, millele üles ehitada kõrgema taseme loogika.

Seesuguse tarkvaralise lahenduse olemasolu suurendaks ka platvormi sihtgruppi. Lisaks robootikavõistlustele on tekkinud ka ambitsioon kasutada platvormi õppe-eesmärgil üldhariduskoolides ning katsevahendina erinevates robootikaga tegelevates uurimislaborites.

2.1 Robotex

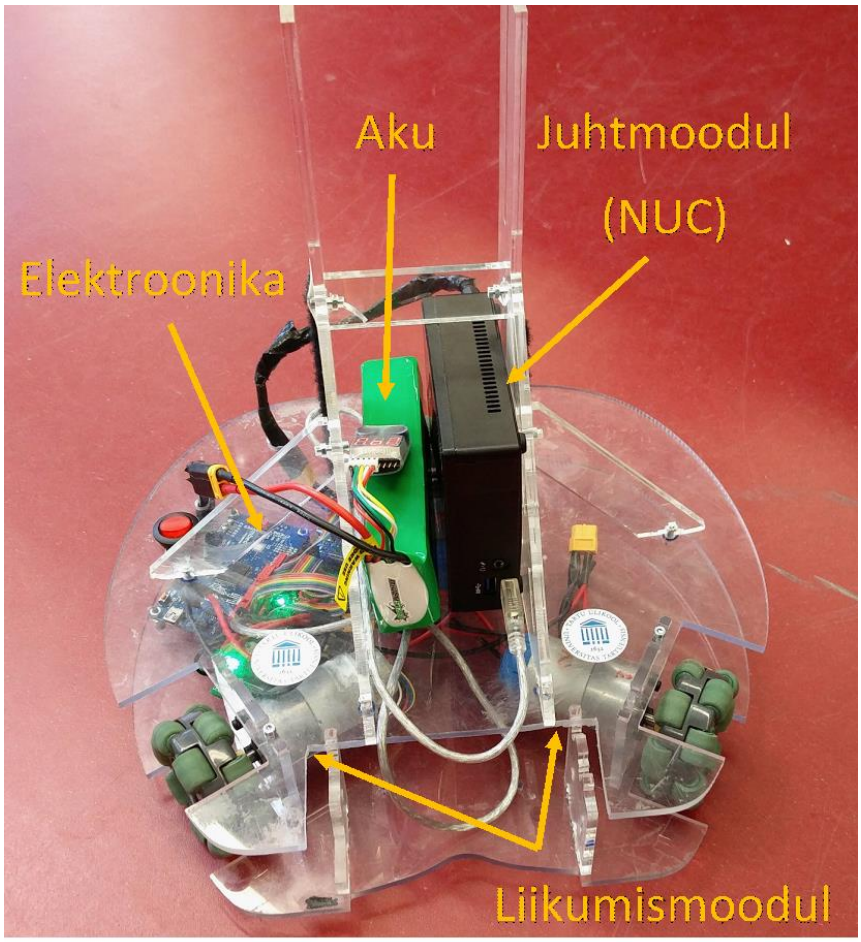
Robotex on Tallinna Tehnikaülikooli, Eesti Infotehnoloogia Kolledži ja Tartu Ülikooli poolt korraldatud rahvusvaheline robootikavõistlus. Robotexi korraldatakse iga-aastaselt sügiseti (sõltuvalt aastast novembri lõpus või detsembri alguses) alates 2001. aastast. Kuni 2009. aastani kuulutati igal aastal välja uus võistlusülesanne, mis tähendas, et igal aastal tuli osalejatel ehitada uuele võistlusülesandele vastav robot. Selline lähenemine soosis igal aastal nullist uute robotite ehitamist. [2,4]

Alates 2009. aastast on Robotexi võistlusalade tuumik püsinud suures osas muutumatuna. Iga-aastaselt on esindatud nii jalgpall, sumo kui ka joonejälgimine. Neist esimesele on põhjalikumalt keskendunud ka Tartu Ülikooli Robotiklubi. [1,5,6]

2017. a. sügisel on taaskord plaanis põhiülesannet muuta. Käesoleva lõputöö valmimise ajaks pole teada uue ülesande kohta rohkem, kui et see saab olema korvpall, mida mängitakse seinatennise pallidega. [6]

2.2 Tartu Ülikool Robotexil

Tartu Ülikooli tudengitel on võimalus Robotexil osaleda õppeaine Robootika praktikum (LOFY.03.033) raames. Ainet loetakse õppeaasta sügissemestril ning eduka osalemise korral Robotexi võistlusel teenib tudeng aine läbimise eest 6 Euroopa ainepunkti (EAP). Kursus valmistab algajad robootikud ette võistluseks kogenud juhendajate käe all. Kursuse raames jagatakse tudengid võistkondadesse. Võistkondade ülesandeks on valmis ehitada robot (joonis 1), millega tuleb osaleda robotivõistlusel Robotex Tallinna Tehnikaülikoolis. Kursuse raames saadakse kogemusi nii mehaanika, elektroonika ja programmeerimise alalt kui ka meeskonnatöö ja projektijuhtimise valdkonnast. [1,7]



Joonis 1: Liikumisvõimekusega robotiplatvorm

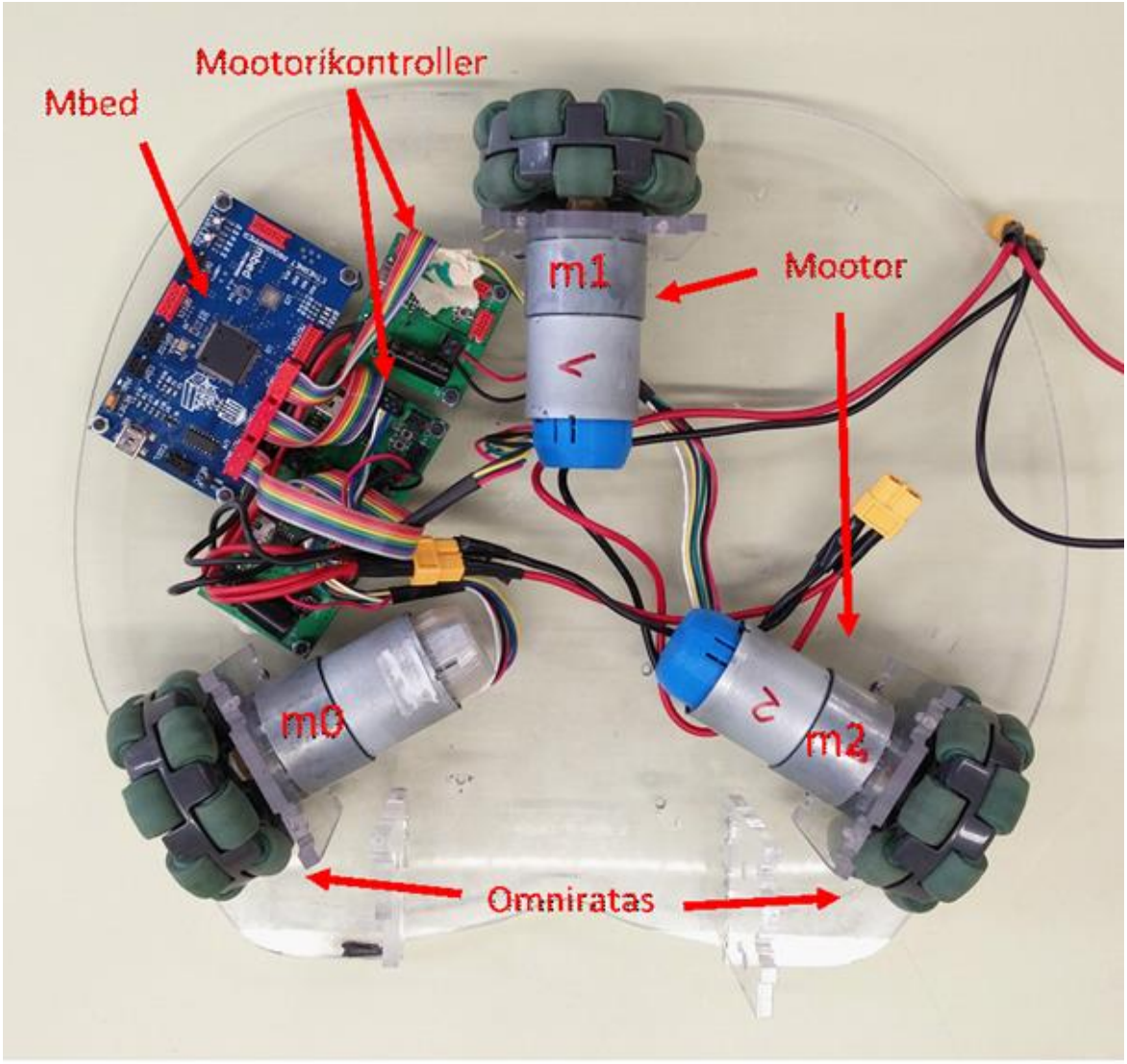
Robotexi ja sügissemestri alguse vahele jääb umbes kolm kuud (sõltuvalt aastast). Sellest robotite ehitamiseks jääb tudengitele umbes kaks kuud, mis tõttu jääb võistkondadel tihti

robotite ehitamisel ajast puudu. Traditsiooniliselt ei ole tudengite kujutlusvõimet ega ehitamisvabadust kuidagi piiratud ning iga võistkond on võinud robotiehitamisele lähenda oma soovidele vastavalt. Selline käitumine viib aga olukorrani, kus valmib palju täiesti erinevaid roboteid, mille valmistamisprotsess on olnud täiesti erinev. See omakorda tähendab, et erinevate vigade avaldumisel on nende üles leidmine ja kõrvaldamine palju keerulisem, kui kindlat normi jälgiva arhitektuuri korral. Robotite ehitamise hõlbustamiseks tutvustati 2012. aastal kursusel osalevatele tudengitele osaliselt valmislahendustest koosnevat roboti kontseptsiooni. Sellist modulaarset robotiplatvormi on igal aastal aina edasi arendatud. Sisuliselt on jõutud disainini, kus tudengid saavad kätte erinevad robotimoodulid (valmislahendus mingist roboti komponendist) ja/või detailse juhendi, kuidas vastavat moodulit valmistada. Igal võistkonnal on võimalus oma mooduleid muuta vastavalt oma võistkonna vajadustele, kuid ühtne baas lubab gruppidel kergemini jõuda potentsiaalsete terviksüsteemis esinevate probleemide kõrvaldamiseni. Samuti lubab selline lähenemine panustada rohkem aega roboti testimisele ja peenhäälestusele, kuna esimene töötav versioon robotist saavutatakse suhteliselt kiiresti.[1,3,8]

2.3 Riistvaraline platvorm

Platvormi riistvara (joonised 1 ja 2) on disainitud võimalikult modulaarseks, et suurendada töökindlust ning säilitada võimekus sobivate täienduste lisamiseks. Roboti struktuuri moodustab keremoodul, mis koosneb kahest sarnasest polükarbonaadist plaadist. Üks plaatidest moodustab roboti põhja ja teine pealmise tasandi. Kahe kihi vahele on lisatud ristipidised neljakandilised mootorikinnitusplaadid, mis lisavad platvormile jäikust ja tugevust. Mootorikinnitusplaatidele kinnituvad kolm liikumismoodulit, mille moodustab mootor [9], selle külge kinnituv omniratas [10] ning vastav kinniti [11] ja ka mootorit käitav kontroller [12]. Mootorikontrollerid ühenduvad kõik keskse mbed kontrolleri [13] külge, mis suhtleb läbi USB realiseeritud jadaliidese platvormi juhtarvutiga. Mbed kontrollerile on võimalik juurde ühendada roboti ülesandest sõltuvalt sobivaid lisamooduleid. Juhtarvutina on kasutusel Inteli *Next Unit of Computing* platvorm ehk NUC [14]. Kuna NUCis tuleb sõltuvalt roboti ülesandest teostada suurt arvutusvõimsust vajavaid ülesandeid (näiteks

videotöötlus), siis on NUCi protsessorina kasutusel Inteli Core-i5 seeria protsessor. Juhtarvuti külge on vajadusel USB liidesega ühendatud ka kaamera, mida kasutatakse rajaplaneerimiseks ja navigeerimiseks.



Joonis 2: Robotiplatvormi sisemus, liikumismoodulid ja elektroonika

2.4 Tarkvaraline platvorm

Modulaarse lähenemise esimestele aastatel anti TÜ Robotiklubi juures Robotexil võistlevatele tiimidele ette riistvaralised moodulid roboti valmistamiseks. Kogu tarkvaraline pool tuli tudengitel iseseisvalt välja töötada. See jättis küllaltki vähe aega keerukama loogika

implementeerimiseks ja programmi silumiseks. Hilisematel aastatel on ette antud ka teatud tarkvaralised moodulid, mis hõlbustasid riistvara poole pöördumist, kuid jätsid siiski kogu tarkvaralise poole arhitektuuri võrdlemisi lahtiseks. Selline lähenemine tingib väga palju erinevalt ülesehitatud tarkvaralisi lahendusi, mis ei toeta parimate lahenduste taaskasutamist. [1,8]

Töövoo muudaks oluliselt lihtsamaks ja optimaalsemaks, kui kogu tarkvara arendus käiks juba eelnevalt paika pandud tarkvara arhitektuuri järgi ja oleks võimalikult modulaarne. Avatud robootikaarendusplatvormi ROS (*Robot Operating System*) kasutuselevõtt oleks üks võimalik lahendus. [15]

2.5 Varsemad arendused Robotexi platvormil

Varasemalt on Tartu Ülikoolis korduvalt teostatud uurimusi Robotexi robotivõistluse ja sellega seoses ka Tartu Ülikooli robotiplatvormiga seonduvatel teemadel. Kõige otsesemalt puudutab terviklikku robotiplatvormi 2013. aastal Kalle-Gustav Kruusi bakalaureusetöö teemal Jalgpalliroboti löögimehhanismi elektroonikalahendus. [8]

Oma töös annab Kruus põhjaliku ülevaate rahvusvahelisest robootika võistlusest Robotex ning tutvustab laiemalt ka robotitevaheliste jalgpallivõistluste ajalugu. Kuna 2012. aastast alates on Tartu Ülikooli siseselt arendatud modulaarset jalgpallirobotiplatvormi, siis on platvorm, millel uurimus põhineb, suures osas sama, millel baseerub ka käesolev bakalaureuse töö. Muutumatuna on püsinud keremoodul ning liikumismoodul. Põhjaliku uuenduse on saanud roboti pardaelektroonika, mis on läbinud mitu erinevat arendusfaasi ning ei ole tänaseni lõpliku kuju saavutanud. Kuigi Kruus annab suure detailsusega ülevaate roboti moodulitest, ei ole ta puudutanud ei keskse pardaarvuti ega roboti optiliste sensorite spetsifikatsiooni. Töö teises pooles kirjeldab autor põhjalikult erinevaid löögimehhanisme ning palli triblajaid. Kuna Kruus töötas välja ka spetsifikatsiooni roboti löögimehhanismi ja triblaja jaoks, siis kirjeldatakse tema lõputöös põhjalikult vajalikku elektroonikat kirjeldavaid suurusi. [8]

Lisaks Kruusi tööle on Robotexil osalemisest inspireeritud veel ka Janno Jõgeva 2012. aastal koostatud bakalaureusetöö ja 2013. aastal koostatud Priit Kallase magistritöö. Mõlemad autorid lahkavad oma töödes peamiselt tarkvaraspetsiifilisi probleeme. [16,17]

Jõgeva uurib oma töös erinevaid võimalusi hüperboloidpeegliga roboti kaamera pildi kalibreerimiseks. Ta toob välja põhjused, miks seesuguse kaamerasüsteemi kasutamine on kasulik, ning ka mitmed meetodid soovitava saavutamiseks. Töös puudub detailne kirjeldus kasutatud robotiplatvormi riistvara ja ka tarkvara kohta, keskendutakse konkreetse kaameramooduli tarkvaralisele realiseerimisele. Töö on valminud ajal, mil modulaarne Tartu Ülikooli Robotiklubi platvorm oli veel lapsekingades, kuid ülevaatlikult kirjeldatud platvormil on märgatavaid sarnasusi (eeskätt kere- ja liikumismoodul) tänaseks väljaarendatud platvormiga. [16]

Priit Kallas keskendub oma töös roboti lokaliseerimisele etteantud jalgpalli väljaku kontekstis. Detailselt kirjeldatakse erinevaid võimalusi sensorite informatsiooni põhjal roboti asukohta jälgida ja roboti edasist käitumist selle põhjal määrata. Muuhulgas antakse ülevaade ka kasutatavast riistvarast. Kallas on arenduseks kasutanud Robotexil korduvalt edukalt esinenud Telliskivi robotit. Telliskivi kasutab kolme omniratta asemel nelja ning on varustatud kahe kaameraga. Mitme kaamera kasutamine lubab korraga koguda rohkem informatsiooni võistluskeskkonna kohta ning seeläbi piirduda keskkonna õppimisel vähemate manöövritega. [17]

Kõik kolm lõputööd pakuvad informatsiooni eelnevalt Tartu Ülikooli Robotiklubis tehtud jalgpalliplatvormi arendustegevusest. Kalle-Gustav Kruusi töö on ainus, mis on keskendunud konkreetse robotiplatvormi modulaarsele arengule.

2.6 Projekt „Robotont“

Tänaseks päevaks on Robotexi platvormi areng jõudnud seisu, kus roboti põhivõimekust – tasasel pinnal liikumine – saab rakendada ka muudes valdkondades. Platvorm sobib kasutuseks tegevusaladele, kus tegeletakse rajaplaneerimist ja kaardistamist puudutavate probleemide lahendamisega. Lisaks uurimisvaldkonnale on roboti platvormil potentsiaali ka

haridusvaldkonnas. Sarnaselt juba turul olevatele platvormidele nagu Turtlebot, saab robotiplatvormi kasutada ka mitmesuguste töötubade läbiviimiseks ja muul viisil robootika propageerimiseks noorema põlvkonna seas [18].

Platvormi arengu hõlpsamaks haldamiseks on alates 2017. aasta kevadest robotiplatvormi arendus koondatud koodnime Robotont alla. Robotont platvormi arenduses viimistletakse ja standardiseeritakse platvormi riistvaraline ning tarkvaraline spetsifikatsioon. Esimeses etapis keskendutakse platvormi liikumisvõimekusele ja keskkonnas leiduvate takistuste vältimisele.

2.7 Käesoleva töö eesmärk

Robotont platvorm on kasutamiseks nii robootikavõistlustel (robotitevaheline jalgpall, korvpall jms), uurimislaborites (näiteks rajaplaneerimine) kui ka robootika propageerimiseks noorte seas. Eelnevalt loetletud kasutusjuhtudest lähtuvalt on platvormi baasfunktsionaalsus liikumine. Modulaarse disaini tõttu on baasfunktsionaalsust võimalik hiljem täiendada keerulisema funktsionaalsuse lisamisega. Käesoleva töö eesmärgiks on luua Robotont platvormile juhtimis- ja liikumisvõimekuse tagav Linuxi draiver, mis ühildub ROSi ökosüsteemiga.

3 Robotont ROSi draiver

Püstitatud probleemile pakub sobiliku lahenduse avatud robotiarendusplatvorm ROS. Töö eesmärgiks on võimaldada ROSi kasutamine Robotont robotiplatvormi juhtimiseks (joonis 4). Selleks tuleb luua jadaliidese vahendusel suhtlev vahelüli (draiver) roboti riistvara (ning selle peal töötava madalama taseme tarkvara) ja juhtmoodulis töötava ROSi vahele. Draiver teeb võimalikuks ROSi move_base paki kasutamise [19] antud platvormil ning võimaldab realiseerida ROSi navigatsiooni teegi funktsionaalsust (mh kõrgema taseme rajaplaneerimist) [20]. Draiveri näol on tarvis luua move_base arhitektuuri sobituv liikumibaasi kontroller, mis jälgib roboti kiirussõnumeid teemal ''/cmd_vel'' ja avaldab roboti odomeetriat teemal ''/odom''. Roboti riistvaraga suhtlemiseks kasutab draiver jadaliidese ühendust. [21]

3.1 ROS

ROS on robotite tarkvara loomise raamistik, mis sisaldab endas vajalikke teeke ja tööriistu, et lihtsustada keerukate, kuid töökindlate robotisüsteemide loomist. [15]

ROSi näol ei ole tegemist otseselt operatsioonisüsteemiga, nagu tarkvara nimetusest võiks järeldada. ROS loob struktureeritud suhtluskihi Linux operatsioonisüsteemi peale. Erinevate alamprogrammide ehk sõlmede [22] omavaheliseks suhtluseks on kasutusel standardiseeritud sõnumitüübid [23]. ROSi tarkvara võib jagada kolmeks: programmeerimiskeelest ja platvormist sõltumatud tööriistad, mida kasutatakse ROSil põhineva tarkvara kompileerimiseks ja jagamiseks; ROSi teegid erineva ROSi funktsionaalsuse realiseerimiseks; konkreetse ülesande realiseerimiseks vajaminevat koodi sisaldavad pakid (*packages*) [24], mis sõltuvad erinevatest ROSi teekidest (*dependencies*). [25–27]

ROS on programmeerimiskeeleülene ning kasutab suhtluseks struktureeritud sõnumeid. ROS toetab mitmeid programmeerimiskeeli (Python, C++, LISP), kuid kaks levinumat on Python ja C++. Keeleülesus tähendab, et ühes ROSil põhinevas süsteemis saab samaaegselt käitada

erinevates programmeerimiskeeltes kirjutatud programme (sõlme), mis omavahel ROSi sõnumite kaudu suhtlevad. [25,26]

3.1.1 Ajalugu

ROSi arendus sai alguse 2007. aastal, kui Stanfordi Ülikooli Tehisintellektuaalsuse laboris töötati välja tehisintellekt-robotit nimega STAIR. Roboti tarkvara pidi ühendama endas nii masinõppe, arvutinägemise, navigeerimise ja planeerimise, kaalutletud otsuste tegemise kui ka keelelise suhtluse tööriistad. [28,29]

Aastatel 2008-2013 tegeles peamiselt ROSi arendamisega robootikalabor Willow Garage. 2013. aastal võttis ROSi arenduse enda kanda OSRF (*Open Source Robotics Foundation* - Avatud Lähtekoodiga Robootika Sihtasutus). [30,31]

Käesoleva töö kirjutamise hetkel on välja antud 10 ROSi distributsiooni. Iga ROS-i distributsioon on vaikimisi seotud sama perioodi Ubuntu Linuxi versiooniga. Näiteks, Robotondi tarkvara arendamiseks valitud ROS Indigo ühildub ja on testitud Ubuntu 14.04-ga. Tarkvara valimisel peeti silmas distributsioonis leiduvate võimalike vigade minimaalset arvukust ning jätkuvat tuge ROSi arendajate poolt. [30,32]

3.1.2 ROSi tööpõhimõte

ROSi sisesest tööpõhimõttest arusaamiseks tuleb alustada terminoloogiast ning selgitada lahti järgnevad mõisted: sõlm, sõnum, teema ja teenus. [26]

Sõlmed on iseseisvad programmid, mis lahendavad ühte konkreetset ülesannet. Keerulisemaid ülesandeid lahendavad süsteemid koosnevad sageli mitmest sõlmest. Iga sõlm on iseseisev tarkvaraline üksus, mida võib käivitada teistest sõltumatuna. Sõlmed saavad küll omavahel suhelda, kuid kui üks peaks lakkama töötamast, funktsioneerivad teised iseseisvalt edasi. [22,25,26]

Sõlmed suhtlevad omavahel läbi sõnumite. Sõnumeid on mitut eri tüüpi ning igal tüübil on oma kindel struktuur ja süntaks. Sõnumid võivad olla algelised (täisarv, ujukomaarv, tõeväärtus jne) või koosneda kõigist teistest eksisteerivatest sõnumitüüpidest. Sobiva sõnumitüübi puudumisel on võimalik vajalik sõnumitüüp kasutaja poolt defineerida ning süsteemis kasutusele võtta. [22,23,25,26]

Sõnumitega suhtlus käib kindlatel teemadel. Kui sõlm tahab edastada infot (*Publisher* – avaldaja), siis avaldab ta sõnumi kindlal teemal (näiteks teemal ‘‘/odom’’ või ‘‘/cmd_vel’’). Sõlm, mis ootab teatud tüüpi sõnumit (*subscriber* – jälgija), jälgib pidevalt vastavat teemat. Ühele teemale võib avaldada korraga mitu sõlme samamoodi nagu ühte teemat võib korraga jälgida mitu sõlme. Üks sõlm võib samuti avaldada ja/või jälgida) sõnumeid korraga mitmel erineval teemal. Sisuliselt ei ole avaldajad ja jälgijad teineteise olemasolust teadlikud - nad on teadlikud vaid sõnumite olemasolust. Kõik ROSi sõnumid liiguvad üle TCP/IP protokolli. [25,26,33,34]

Lisaks ühesuunalisele sõnumsidele (avaldamine ja jälgimine) on ROSis ka teenused. Teenus koosneb kahest sõnumist: päringust ja vastusest. Erinevalt teemadest, saab konkreetse teenuse välja reklaamida korraga vaid üks sõlm. Teenus reklaamitakse välja teenuse nimega (sarnane teemale). Teenuse väljakutsumiseks on tarvis saata päring vastava teenuse nimele ning oodata vastust. Teenused töötavad analoogiliselt veebiteenustele, kus teenus on defineeritud unikaalse identifikaatoriga ning omab päringu ja vastuse dokumenti. [25,26,35,36]

Selleks, et erinevad sõlmed saaksid käivituda ja omavahel suhelda, peab olema käivitatud ROSi põhiprogramm *roscore* ehk tuum. ROSi tuum peab arvet, millised sõlmed käivitunud on, millistel aadressidel sõlmed asuvad ning milline sõlm millisele teemale sõnumeid avaldab ja milliste teemade sõnumeid jälgib. [25,26,37,38]

3.1.3 ROSi modulaarsus

ROS on disainitud pidades silmas ühilduvust erinevate platvormide vahel. Kogu ROSi sõnumiside baseerub SI-süsteemi ühikutel (kiirused ühikutes m/s, nurkkiirused ühikutes rad/s jne). Nii on võimalik taaskasutada eri robotitele loodud sõlmesid ja teeke ning integreerida neid uude arendusse. Määravaks on siinkohal, et sõlmed suudaksid omavahel suhelda üheselt defineeritud sõnumitega kindla nimetusega teemal. [25,26]

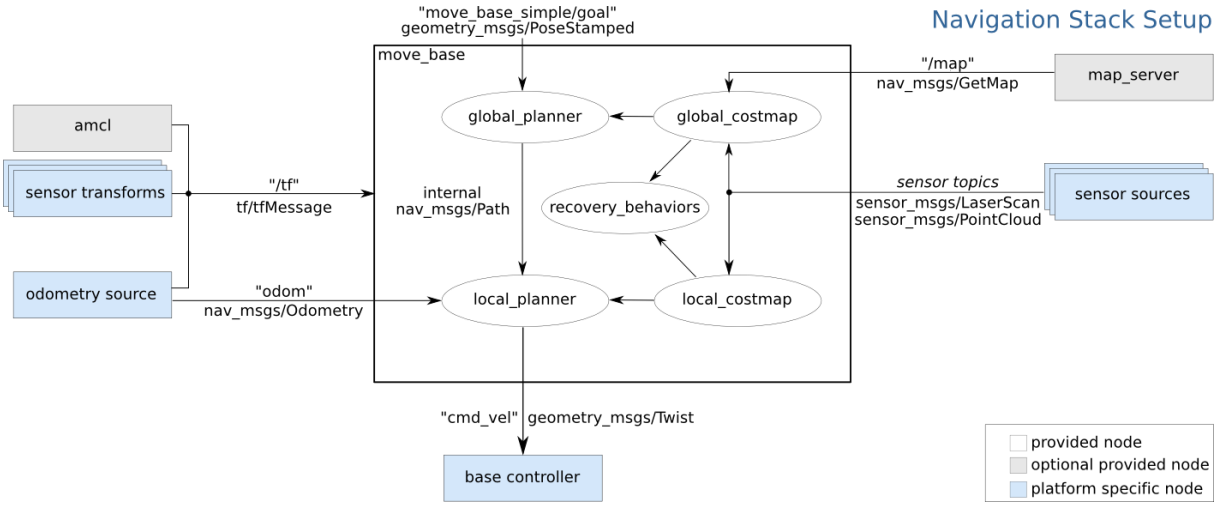
ROSi sõlmed jooksevad iseseisvalt ehk on omavahel vaid käitusaegses sõltuvuses (kompilleeruvad sõltumatult). See lubab erinevaid ROSi sõlmi jooksvalt käivitada, peatada ja välja vahetada ilma, et teised sõlmed oma töö peatama peaksid. Kui sõlme töös peaks tekkima käitusaegne viga, on võimalik sõlm peatada ja taaskäivitada, ilma et kogu süsteem tuleks selleks välja lülitada. [22,25,26]

Need omadused teevad ROSi kasutamise mugavaks. Tihti on uue roboti arendamisel võimalik jooksvalt erinevaid programme täiendada või kasutada juba kellegi teise poolt varasemalt arendatud ROSi teeke ja tööriistu [39]. [25]

3.1.4 Navigatsiooni teek

Üks ROSi olulisemaid osasid on navigatsiooni teek (*Navigation Stack*), mis koosneb paljudest sõlmedest ning kannab hoolt selle eest, kuidas robot liigub. Teek võtab sisendiks odomeetria, sensorite väärtused ning soovitava lõpp-positsiooni (joonis 3) [21]. Sisendi põhjal arvutatakse välja sobiv kiirus roboti liikumiseks (sõnum(id) teemal ‘‘/cmd_vel’’), et jõuda vastavasse sihtpunkti. Keskse osa navigatsiooni teegist moodustab ROSi pakk move_base ehk liikumisbaas, mis tegeleb roboti rajaplaneerimisega. Navigatsiooni teegis väljaarvutatud roboti kiiruse riistvaraliseks realiseerimiseks on tarvis luua draiver. Seda draiverit nimetatakse navigatsiooni teegi kontekstis liikumisbaasi kontrolleriks (*base controller*). Draiveri esmane ülesanne on teisendada roboti liikumiskiirus (‘‘/cmd_vel’’ teema sõnum) riistvara poolt toetatud käskudeks. Käskude tulemusena pannakse roboti mootorid

tööle panna seesuguse kiirusega, et robotiplatvorm tervikuna liiguks etteantud kiirusega. [19,20]



Joonis 3: ROSi navigatsiooni teegi struktuur

3.2 Tarkvaralised võimalused jadapordi ühenduse realiseerimiseks

Esimestes arendusfaasides realiseeriti jadaliidese ühendus Pythonis kasutades selleks Pyserial teeki. Sellisel viisil jadapordi ühenduse lahendamine ei olnud soovitud töökindlusega ning põhjustas korduvalt roboti töö seiskumise. Et draiveri lõplik versioon on planeeritud realiseerida C++ programmeerimiskeeles, siis katsetati järgmise alternatiivina ROSi pakki nimega rosserial [40]. Rosserial võimaldab soovitud seadmega suhelda justkui oleks tegemist mõne teise ROSi sõlmega [40]. Selleks tuleb roboti riistvarale - käesolevas projektis mbed kontroller - laadida programm, mis suhtleb üle jadaliidese ROSi sees jooksva sõlmega [40]. Sellise skeemiga saab ROSis suhelda ROSi sõnumitega pöördudes otse riistvara poole ning vahepealse info edastamise teeb ära juba rosserialisse sisse-ehitatud funktsionaalsus [40]. Pärast mitmeid edutuid katsetusi rosserial pakiga ning hoolikat süvenemist rosseriali lähtekoodi, tuli välja, et rosserial kasutab jadapordi ühenduseks protsessori TX ja RX viike, mitte läbi USB korraldatavat jadaliidese ühendust. Riistvaraliselt oleks seesuguste viikude ühendamine juhtarvuti külge äärmiselt keerukas, mis tõttu ei õigusta rosseriali pakk enda kasutust.

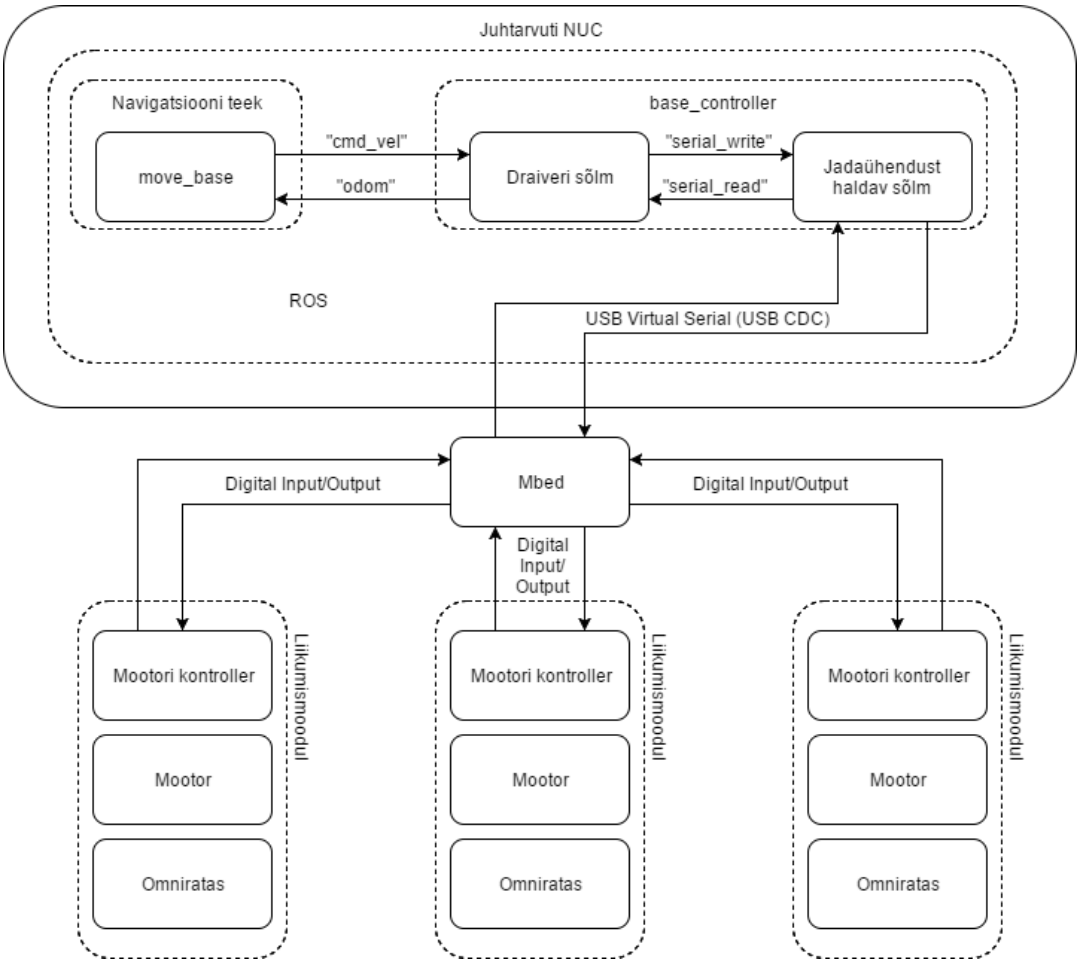
Alternatiivset lahendust pakub ROSi pakk nimega serial. Serial pakk kujutab endast teeki, mis sisaldab lihtsalt kasutatavaid meetodeid jadaliidese ühenduse kasutamiseks. Seriali esmakordsel kasutamisel Linuxis tuleb täpsustada vastava jadapordi fail ning suhtluskiirus (*baudrate*). Samuti tasub täpsustada ka ühenduse aegumise ajad (*timeout*) ning püüda kinni võimalik erindid. Suhtluse realiseerimiseks on serial teegis lihtsalt implementeeritavad funktsioonid (*open, read, write, close* jt), mis ei vaja toimimiseks välistest pakkidest sõltuvust (*dependencies*). Serial pakki kasutamine on nähtav serial_com_node sõlmes (lisa 2). [41]

3.3 ROSi draiveri arhitektuur

Efektiivsema ressursi kasutuse tõttu ajakriitiliste probleemide lahendamisel võeti vastu otsus luua draiver C++ programmeerimiskeeles.

Navigatsiooni teegi spetsifikatsioonist (joonis 4) tuleneb, et roboti liikumist kirjeldavad käsud edastatakse teemale ‘‘/cmd_vel’’ (teema nimetus on lühend sõnapaarist *command velocity*). Seega peab draiver jälgima pidevalt ‘‘/cmd_vel’’ teemat. Teemalt saadud informatsiooni põhjal tuleb välja arutada sobivat liikumist kirjeldavad kiirused iga individuaalse mootori jaoks. Arvutustulemustes kogutud informatsioon tuleb edastada üle jadaliidese roboti riistvarale (mbed kontrollerile). [20,21]

Et lugeda mootorikontrolleritelt ka odomeetria tarbeks sisendit, peab jadaliidese suhtlus olema kahepoolne. Seesuguse disaini lihtsamaks ja modulaarsemaks implementeerimiseks otsustati paralleelselt panna tööle kaks iseseisvat sõlme, millest üks haldab jadaliidese ühendust ning teine tõlgib geometry_msgs/Twist sõnumeid [42] mootorikontrollerile edastavateks käskudeks. Sõlmed suhtlevad omavahel kasutades ROSi sõnumeid.



Joonis 4: Robotiplatvormi arhitektuur

3.3.1 Driver_node

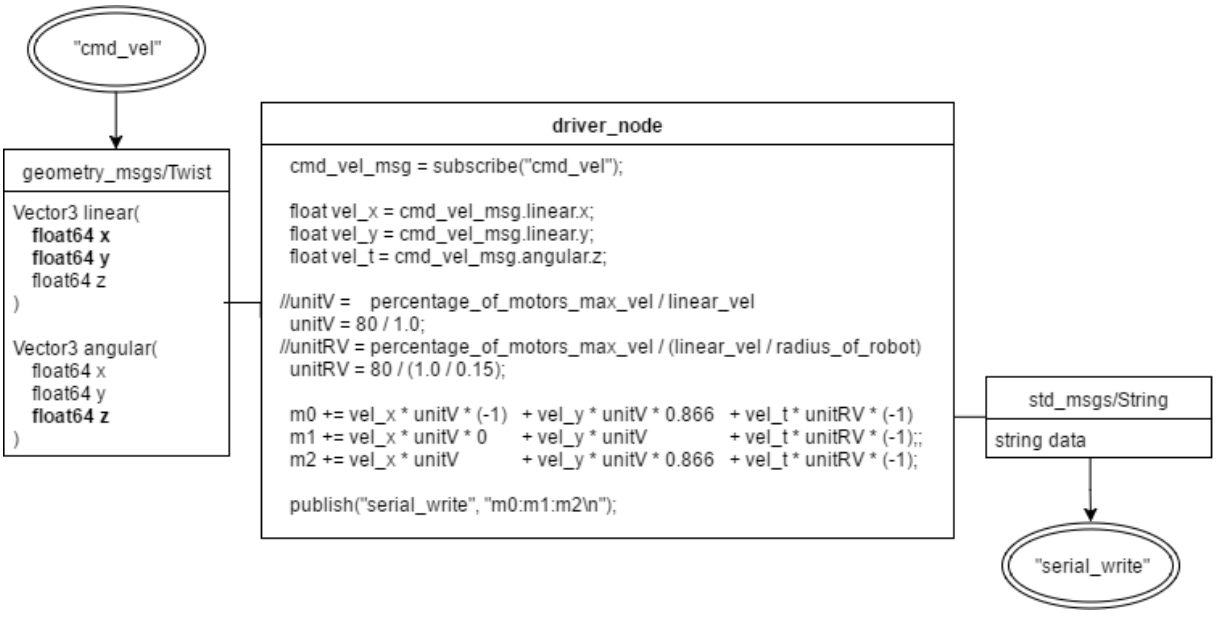
Driver_node jälgib ‘’/cmd_vel’’ teemal edastatavaid sõnumeid, mis kirjeldavad roboti liikumise kiirust. Sõlmesisene loogika on näidatud joonisel 5.

Teemal ‘’/cmd_vel’’ edastatavad sõnumid on geometry_msgs/Twist tüüpi sõnumid, mis kirjeldavad liikumist nii joon- kui ka nurkkiirustega. Üks sõnum koosneb kahest kolmemõõtmelisest vektorist: *linear* ja *angular* [42]. Kumbki vektor sisaldab kolme 64-bitist ujukomaarvu, mis kirjeldavad vastavalt x, y ja z telje suhtes liikumist. Joonkiirust kirjeldav vektor *linear* koosneb iga telje sihilistest kiirustest ning nurkkiirust kirjeldav vektor *angular* vastavalt iga telje ümber pöörlemise kiirust. Et roboti riistvara võimaldab liikumist kahes

teljes (*linear.x* ja *linear.y*) ning pöörlemist ümber kolmanda (*angular.z*), siis teistel sõnumiväljadel edastatud kiirusi arvutustes arvesse ei võeta.

Mootorite kiirused tuleb väljendada arbitraarses suuruses vahemikus -100 – 100. Mootorite maksimaalne kiirus on määratud mbed kontrolleri püsivara autori tunnetuse järgi ning ei ole täpselt kalibreeritud. Draiveri sõlme on sisse kirjutatud muutujad kalibratsiooni teostamiseks. Tarkvaras tuleb ette anda maksimaalne väärtus, millega mootorit käitada soovitakse ning sellele vastav (mootori külge kinnitatud) omniratta liikumise joonkiirus. Nende andmete põhjal leitakse ühikjoonkiirus ja ühiknurkkiirus. Teemal ''/cmd_vel'' avaldatud sõnumite väärtuste läbikorrutamisel ühikkiirustega, leitakse reaalne mootoritele edstatav väärtus. Kiiruste leidmise algoritmid on leitavad sõlmest driver_node (lisa 2).

Formaat, milles mootorite kiirused tuleb mbed kontrollerile edastada on „m0:m1:m2\n“, kus m0, m1 ja m2 tähistavad vastavalt joonisel 2 märgitud mootorite kiirusi. Kiiruste jada tuleb lõpetada reavahetussümboliga. Valmiskujul käsk avaldatakse ''/serial_write'' teemal kasutades sõnumitüüpi std_msgs/String [43]. Kogu driver_node sõlme lihtsustatud ülesehitus on kujutatud joonisel 5. Driver_node alusfaili driver_node.cpp lähtekood on leitav käesoleva töö lisas 1.



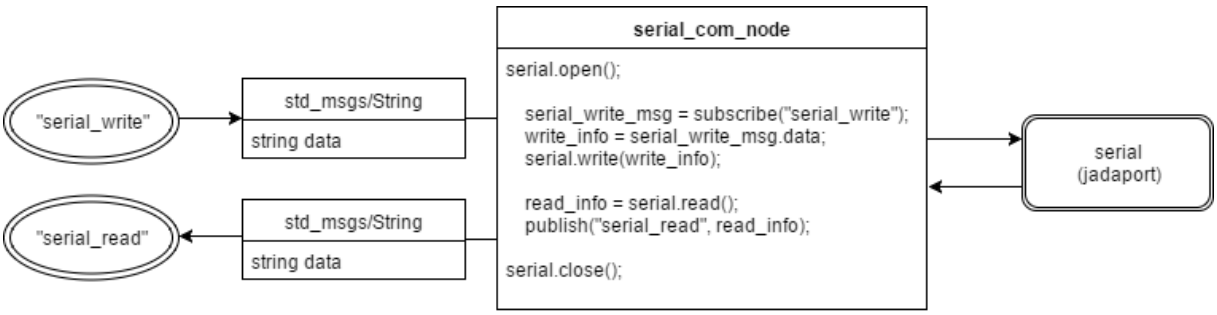
Joonis 5: Draiver node sisemine matemaatika pseudokoodis väljendatuna koos vastavate sõnumitüüpidega

3.3.2 Serial_com_node

Serial_com_node sõlme ülesandeks on hallata läbi jadapordi toimuvat suhtlust. Sõlm kasutab suhtluseks serial paki [41] APIt, mis lubab jadapordi lihtsalt üles seada. Esmakordsel kasutamisel tuleb täpsustada vastava jadapordi fail ning suhtluskiirus (*baudrate* – mbed kontrolleri püsivara on seadistatud kasutama kiirust 9600 bps), muude parameetrite korral kasutada serial paki vaikeväärtusi. Vältimaks programmi soovimatut käitumist tuleb seadistada ka aegumise ajad ning püüda kinni võimalikud erindid. Serial paki API lubab suhtluse avamiseks ja lõpetamiseks kasutada vastavalt *open()* ja *close()* meetodeid ning lugemiseks ja kirjutamiseks vastavalt *read()* ja *write()* meetodeid [41].

Serial_com_node lihtsustatud ülesehitus on toodud joonisel 6. Sõlm jälgib ‘’/serial_write’’ teemat, millele antud rakenduses avaldab informatsiooni driver_node. Sisendsõnum edastatakse muutmata kujul läbi jadaliidese mbed kontrollerile.

Et ROS saaks navigeerimisel kasutada ka mootorite tagasisidet, siis on jadaliidese ühendus realiseeritud kahepoolselt. Mbed kontroller saadab sagedusega 50 Hz juhtarvutile läbi jadapordi mootorite enkooderite väärtusi. Enkooderite väärtused on samas vormingus, mis mootoritele edastatud kiirused – „m0:m1:,m2“, kus vastavad suurused tähistavad vastavate mootorite hetkkiirusi. Jadapordist loetud informatsiooni avaldab sõlm otse teemale ‘’/serial_read’’. Serial_com_node sõlme alusfaili serial_com_node.cpp lähtekood on leitav käesoleva töö lisas 1.



Joonis 6: Serial_com_node sisemine andmete vahendamine pseudokoodis väljendatuna

4 Tulemused

Käesolevas peatükis tutvustatakse testimiseks kasutatud riistvaraplatvormi ja demonstreeritakse loodud draiveri funktsionaalsust. Draiveri funktsionaalsuse ja modulaarsuse demonstreerimiseks kasutatakse roboti juhtimiseks arvutiklaviatuuri ja nutitelefonirakendust. Töö tulemusena on loodud tugi modulaarse tarkvaralise lahenduse realiseerimiseks Robotont platvormile.

4.1 Algne olukord

Käesolevas töös valminud ROSi draiverit testiti 2016. aastal Robotexil osalenud võistkonna Frankenstein robotil [44]. Robot sisaldas riistvaraliselt funktsionaalseid liikumismooduleid, keremoodulit, kaameramoodulit, juhtmoodulit ning liikumismoodulite juhtimiseks vajalikku elektroonikat ja võistlusspetsiifilist riistvara (triblaja ja löögimehhanism). Energiaallikana kasutati kahte kolmeelemendilist liitiumpolümer (3S LiPo) akut. Üks aku toitis elektroonikat ning mootoreid ja teine läbi *boost*-muunduri juhtmoodulit. Informatsiooni vahendamiseks kasutataval mbed kontrolleril olnud püsivara ei tõestanud end katsetuste käigus usaldusväärsena. Juhtmoodulina oli kasutusel Inteli NUC, millel käitati Windows 10 operatsioonisüsteemi ja selle sees omakorda vajalikke Pythoni programme.

Arendustegevuse käigus asendati NUCi operatsioonisüsteem Ubuntu 14.04-ga, millele paigaldati ROS Indigo [45]. Igal paarisarvulisel aastal väljaantud ROSi distributsioon on nn pikaaegse toega (*LTS – Long Term Support*), millele OSRF pakub tuge järgeva viie aasta jooksul. Nii ROS Indigo Igloo (2014) kui ka ROS Kinetic Kame (2016) on LTS distributsioonid. Arendustöö alustamise hetkel oli ROS Indigo juba kaks aastat avalikkuse käes testimisel olnud ning võis eeldada oluliselt vähem pisivigu, mis arendust pidurdada oleks võinud. [32]

Esialgsete katkestuste käigus andis mbed-kontrolleri püsivara väga ebastabiilseid tulemusi, seetõttu tuli edasiseks arenduseks luua mbed-kontrollerile uus püsivara. Riistvaraliselt sai eemaldatud võistlusspetsiifiline riistvara ning kaameramoodul. Samuti võeti kasutusele kahe

3S LiPo aku ja *boost*-muunduri asemel üks 4S LiPo aku. Suurema pingega aku toidab samaaegselt juhtarvutit, liikumismooduleid ning pardaelektroonikat.

4.2 Esmane funktsionaalne testimine

Töö tulemusel on robotiplatvormile loodud ROSi tugi ning esmane liikumisvõimekus. Koostatud draiver koosneb kahest sõlmest (draiver-sõlm ning jadapordi kaudu suhtlemist korraldav sõlm). Draiver jälgib ROSi teemat ‘‘/cmd_vel’’ ning on võimeline selle põhjal robotit liigutama. Samuti suudab draiver lugeda roboti riistvaralt tagasisidet roboti liikumise kohta ning seda avaldada.

Sisuliselt saab arendatud draiverit kasutades Robotonti juhtida mistahes ROSi tarkvaraga, mis avaldab teemal ‘‘/cmd_vel’’ sobivat tüüpi kiirusesõnumeid. Käesoleva töö raames valiti näitlikustamiseks universaalne arvutiklaviatuur ja Android nutitelefon, mille abil on võimalik robotit juhtida.

4.2.1 Robotondi juhtimine arvutiklaviatuuriga

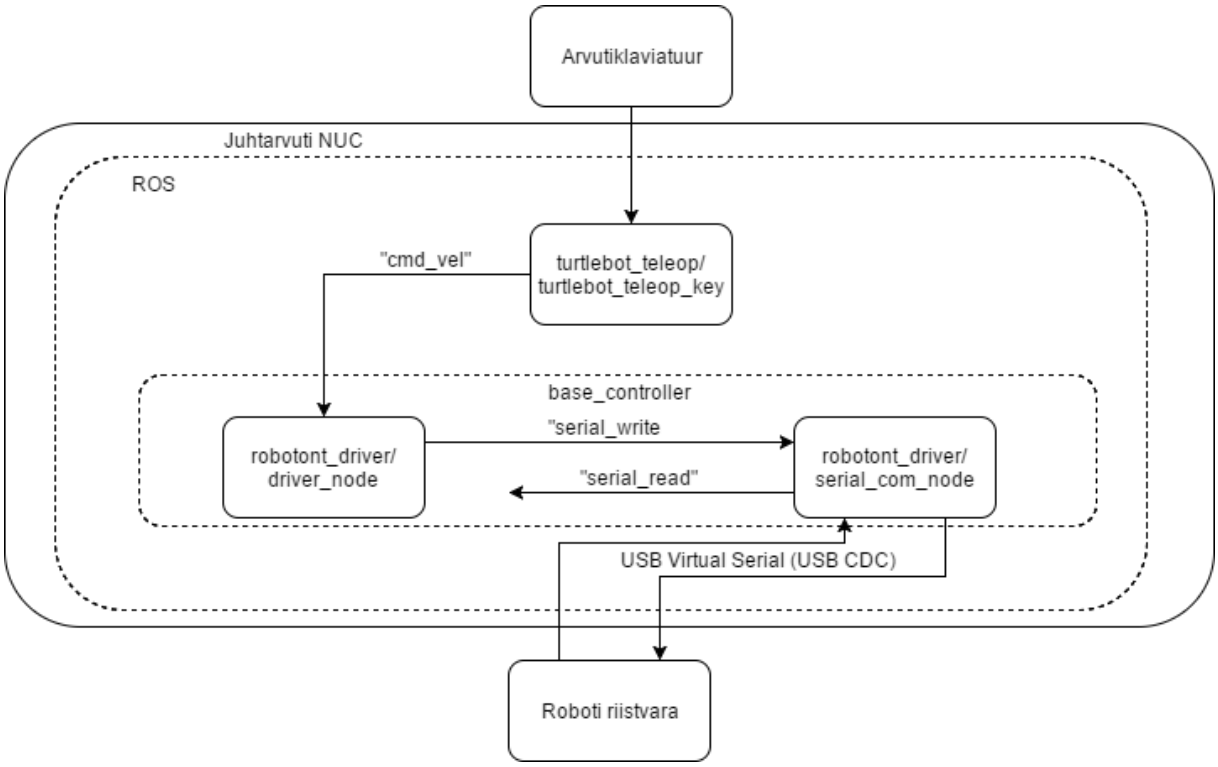
Lihtsaim moodus Robotondi juhtimiseks on kasutades klaviatuuri. Selleks tuleb luua või kasutada mõnd olemasolevat sõlme, mis suudaks klaviatuuri sisendi põhjal luua sobiva vorminguga sõnumi teemal ‘‘/cmd_vel’’. ROSi modulaarsuse tõttu on lihtne Robotont süsteemiga liita mõne olemasoleva platvormi (näiteks Turtlebot) juhtimise pakk või sõlm. [18]

Turtleboti platvorm on vabavaraline ning kogu lähtekood on internetist vabalt leitav ja kasutatav. Lihtsaim moodus roboti kontrollimiseks on kasutada näiteks sõlme turtlebot_teleop_key ROSi pakist turtlebot_teleop. [46]

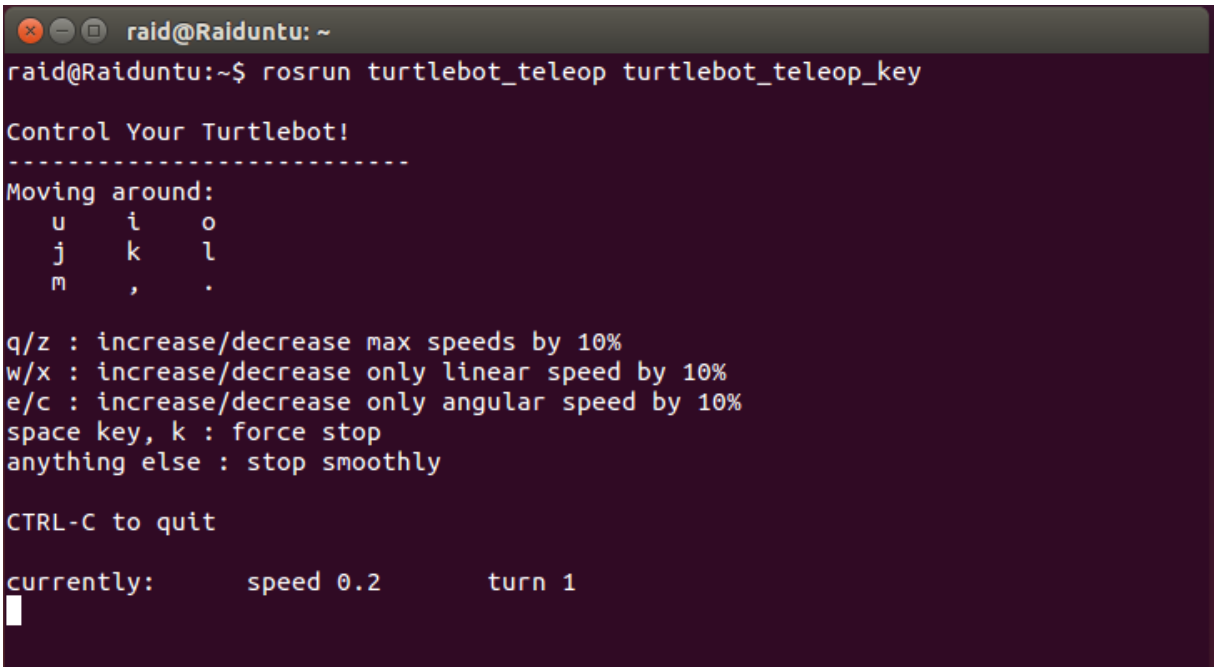
Turtlebot_teleop_key loeb käsurealt kasutajapoolseid klahvivajutusi ning genereerib nende põhjal sobiva geometry_msgs/Twist tüüpi sõnumi. See sõnum avaldatakse teemal

‘’/cmd_vel’’. Klaviatuuriga Robotondi kontrollimise struktuur on nähtav joonisel 7 ning turtlebot_teleop_key sõlme kasutajaliides on nähtav joonisel 8. [46]

Seesuguse lahenduse juures tuleb arvestada, et juhtmega klaviatuuri kasutades ei ole robotit kuigi mugav juhtida. Probleemi lahenduseks sobib juhtmevaba klaviatuuri kasutamine või mõni kaugühendumise meetod juhtarvutisse. Näiteks võib vastava sõlme käivitada ka juhtarvutisse SSH protokolliga ühendudes ning seeläbi robotit teise arvuti ekraanilt juhtida. Arvutiklaviatuuriga roboti juhtimisele vastav tarkvaraline arhitektuur on näha joonisel 8.



Joonis 7: Klaviatuuriga Robotondi juhtimine.



Joonis 8: Turtlebot robotiplatvormi kasutajaliides roboti klaviatuuriga juhtimiseks

4.2.2 Robotondi juhtimine nutitelefoniga

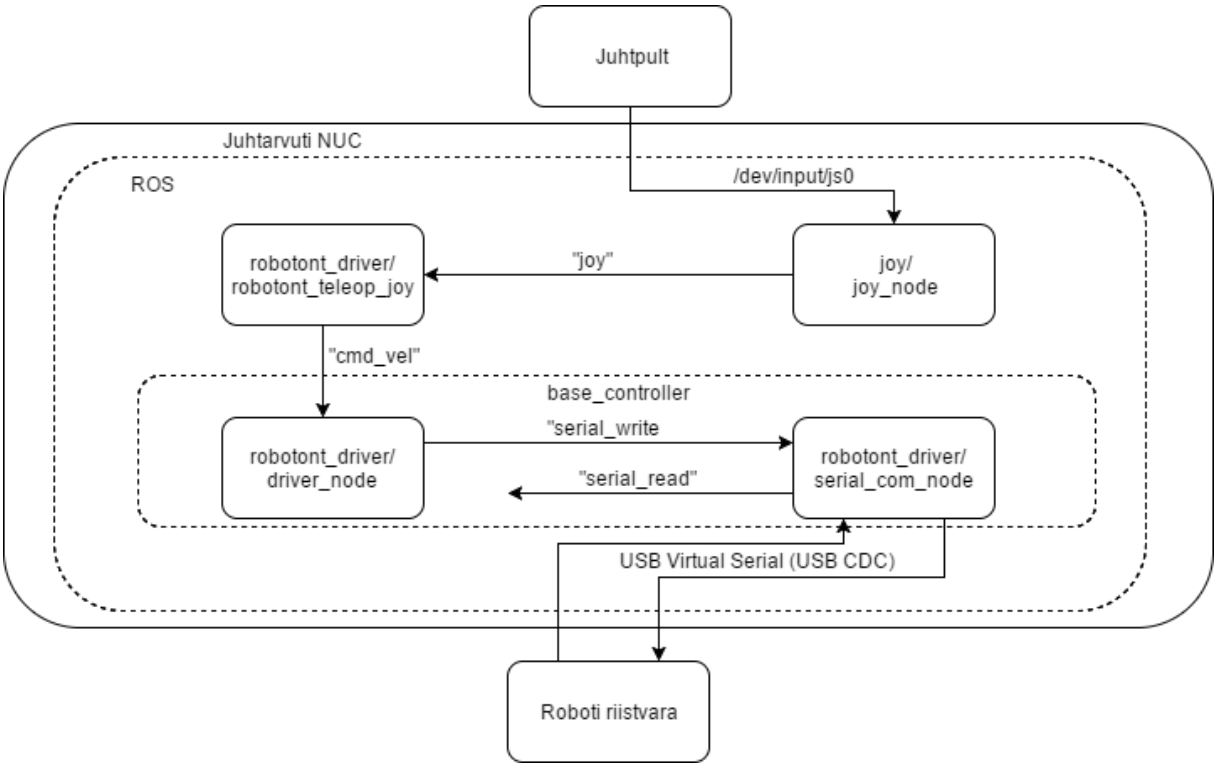
Sarnaselt klaviatuuri kasutamisele, on nutitelefoniga roboti juhtimiseks tarvis vaid ‘‘/cmd_vel’’ teemale õiges vormingus sõnum edastada. Et juhtimismeetod oleks võimalikult intuitiivne, luuakse nutitelefoni ekraanile virtuaalne juhtkangidega juhtimispult.

Juhtpuldi realiseerimiseks on kasutusel tarkvarakomplekt nimega DroidTick. DroidTicki käitamiseks on tarvis juhtarvutisse installeerida serverprogramm. Serverprogramm loob Arvutisse virtuaalsed seadmed, mida on võimalik nutitelefonist kontrollida. Nutitelefonis on vajalik DroidTick mobiilirakenduse olemasolu (toetatud on Android platvorm [47]). Rakenduses on kasutajal võimalik valida nelja erineva kujundusega juhtpuldi vahel, mis toetavad ka juhtkangi (joystick) funktsionaalsust. DroidTick serveriprogrammi ja mobiilirakenduse kasutajaliidesed on toodud joonistel 9 ja 10. [48]

Joonis 9: DroidTick serveri tarkvara kasutajaliidese kuvatõmmis [47]

DroidTick töötab arvuti jaoks kui iga teine füüsiline juhtpult. Juhtpuldi signaalide lugemiseks ROSis on olemas pakk nimega joy [49]. Juhtpuldi kasutamiseks tuleb käivitada ROSi sõlm joy_node. Joy_node sõlm avaldab juhtpuldist saadud signaalid sensor_msgs/Joy tüüpi sõnumina teemal ‘‘/joy’’. Avaldatud sõnum tuleb teisendada geometry_msgs/Twist formaati ning avaldada teemal ‘‘/cmd_vel’’. Selle funktsionaalsuse jaoks on töö raames loodud ROSi sõlm Robotont_teleop_joy. Sõlm jälgib sisendina ‘‘/joy’’ teemat ning teisendab sisendinformatsiooni reaalseks kiiruseks, mille avaldab teemal ‘‘/cmd_vel’’. Robotont_teleop_joy sõlme alusfaili Robotont_teleop_joy.cpp lähtekood on leitav lisas 1. Juhtpuldiga robotiplatvormi kontrollimisele vastav tarkvaraline arhitektuur on nähtav joonisel 11.

Joonis 10: DroidTick mobiilirakenduse kasutajaliidese kuvatõmmis



Joonis 11: Juhtpuldiga Robotondi juhtimine

5 Tulemuste analüüs ja järeldused

Käesolevas peatükis kirjeldatakse lühidalt töös saavutatut ning arenduse käigus esinenud probleeme. Probleemidele pakutakse välja ka võimalikud lahendused, mis edasises arenduses draiveri tööd parandaksid.

5.1 Peamised tulemused

Käesoleva töö tulemusena on realiseeritud võimekus kasutada Robotont platvormi ROSi tarkvaraga. Välja on arendatud ROSi navigatsiooni teegi ja riistvara vahelist suhtlust koordineeriv draiveri pakk. Draiver lubab robotiplatvormi siduda ROSi ökosüsteemiga ning integreerida ükskõik millise teise platvormi jaoks arendatud ROSi tarkvara Robotont platvormiga. Robotiplatvormi liikumisvõimekust ja draiveri tööd testiti olemasolevate ROSi rakendustega nii klaviatuuri kui ka nutitelefoni abil.

5.2 Järgmised arendustegevused

Käesolevas töös valminud draiveris ei ole realiseeritud efektiivselt täisfunktsionaalsust, mida Robotont platvorm vajaks. Arendusfaasis osutus problemaatiliseks jadaliidese ühendus, mis järjepideva testimise tagajärjel osutus äärmiselt ebastabiilseks ning ei andnud soovitud tulemusi. Samuti ei ole täielikult täidetud tingimused ROSi navigatsiooni teegiga ühendumiseks. Draiveri efektiivsemaks kasutamiseks tuleb implementeerida efektiivsem algoritm kiirusteisendusteks.

5.2.1 Jadaliides

Robotiplatvormi arenduse algfaasidest peale on problemaatiliseks osutunud stabiilse jadaliidese ühenduse realiseerimine. Esialgsete katsetuste käigus kadus läbi jadapordi ühendus ka lihtsate testprogrammide jooksutamisega. Eeldades, et tegemist on tarkvaralise veaga, prooviti mitmesuguseid erinevaid tarkvara variante jadaliidese ühenduse kasutamiseks. ROSi draiverina prooviti läbi kolm erinevat meetodit jadaliidese ühenduse seadistamiseks. Esialgsete katsetuste põhjal tundus probleem viimase tarkvaraga olevat kõrvaldatud. Hilisemate katsetuste käigus osutus ühendus läbi jadapordi siiski kasutuskõlbmatult ebastabiilseks. Kahtlustades endiselt tarkvaralist probleemi, sai uuendatud ka mbed kontrolleri püsivara. Ka pärast kõikide tarkvaraliste komponentide väljavahetamist ei olnud ühendus läbi jadapordi rahuldava stabiilsusega. See jätab õhku hüpoteesi, et jadaliidese ebastabiilsus on tingitud riistvaralistest häiretest. Suure tõenäosusega tekitab probleeme isevalmistatud mbed kontrollerelektroonika [13]. Robotondi järgmistes versioonides on vastav riistvara komponent planeeritud välja vahetada kaubandusliku valmistootega [50].

5.2.2 Odomeetria

Liikumisbaasi ühendmaiseks robotiplatvormi riistvaraga on tarvis ka odomeetriat edastada teemale ‘‘/odom’’. Kuna käesolevat draiverit kirjutades oli prioriteet saada robotiplatvorm esmajärjekorras kontrollitult liikuma ROSi vahendusel., siis ei pandud rõhku odomeetria avaldamisele. Antud töös koostatud draiver loeb küll riistvaralt odomeetriat kirjeldavaid väärtusi, kuid ei ole lõpuni realiseeritud nende andmete edastamine vastavale teemale. Andmed avaldatakse praegu std_msgs/String-tüüpi sõnumina teemal ‘‘/serial_read’’, kuid navigatsiooni teegi ja liikumisbaasiga ühildumiseks oleks tarvis nad omakorda edastada ja avaldada nav_msgs/Odometry-tüüpi sõnumina teemal ‘‘/odom’’ [21,51].

6 Kokkuvõte

Seoses iga-aastase rahvusvahelise robootikavõistlusega Robotex on Tartu Ülikooli Robotiklubis väljaarendatud riistvaraliselt funktsionaalne robotiplatvorm. Standardiseeritud riistvaramoodulite kõrvalt puudub modulaarselt struktureeritud ja töötav tarkvaraline lahendus.

Selle töö eesmärgiks oli välja arendada eelnimetatud robotiplatvormile ROSi ökosüsteemiga ühilduv draiver. Eesmärgi saavutamiseks uuriti põhjalikult ROSi tööpõhimõtet ning pandi paika arhitektuur vajaliku tarkvaraliidese disainiks, mis seejärel realiseeriti.

Töö tulemusena arendati välja ROSI arhitektuuri sobituv baaskontroller, mis lubab ROSi programmidel suhelda jadaliidese vahendusel roboti riistvaraga. Valminud tarkvara testiti Robotexil osalenud robotil kahe erineva juhtimismeetodiga (arvutiklaviatuur ja nutitelefon). Draiver töötas ootuspäraselt ning võimaldas realiseerida testitud ROSi funktsionaalsust.

Väljaarendatud draiver jälgib ROSi teemal ’’/cmd_vel’’ avaldatavaid sõnumeid ning arvutab nende põhjal välja igale mootorile vastava kiiruskäsu. Leitud käsud edastatakse üle jadapordi roboti riistvarale, kus need läbi mootorikontrollerite realiseeritakse.

Viited

1. Robotex [Internet]. Digilabor. 2016 [tsiteeritud 2. mai 2017]. Salvestatud: http://digi.physic.ut.ee/mw/index.php?title=Robotex&oldid=26447
2. Robotex MTÜ. Mis on Robotex? [Internet]. Robotex. 2017 [tsiteeritud 9. mai 2017]. Salvestatud: http://robotex.ee/
3. Football Robot Guide [Internet]. Digilabor. 2016 [tsiteeritud 11. mai 2017]. Salvestatud: http://digi.physic.ut.ee/mw/index.php/Football_Robot_Guide
4. Arak M. Rahvusvaheline robotivõistlus Robotex 2014 [Internet]. 2014. Salvestatud: http://tehnoloogia.ee/wp-content/uploads/2014/10/Robotexi-tutvustav-materjal-originaal-2.pdf
5. Robotex MTÜ. Võistlused ja reeglid [Internet]. Robotex. 2016 [tsiteeritud 30. aprill 2017]. Salvestatud: http://www.robotex.ee/voistlused-ja-reeglid
6. Robotex MTÜ. Võistlused [Internet]. Robotex. 2017 [tsiteeritud 2. mai 2017]. Salvestatud: http://robotex.ee/voistlused/
7. Tartu Ülikool. Õppeaine üldandmed - Robootika praktikum (LOTI.05.023) [Internet]. Tartu Ülikooli Õppeinfosüsteem. 2016 [tsiteeritud 9. mai 2017]. Salvestatud: https://www.is.ut.ee/rwservlet?oa_aine_info.rdf+1222218+HTML+23912199120783572237+text/html
8. Kruus K-G. Jalgpalliroboti löögimahhanismi elektroonikalahendus. 2013; Salvestatud: http://dspace.ut.ee/handle/10062/32440
9. Pololu Corporation. 19:1 Metal Gearmotor 37Dx68L mm with 64 CPR Encoder [Internet]. Pololu. 2017 [tsiteeritud 9. mai 2017]. Salvestatud: https://www.pololu.com/product/2822
10. WEX Robotics. Wheels [Internet]. WEX EDR. 2017 [tsiteeritud 9. mai 2017]. Salvestatud: https://www.vexrobotics.com/vexedr/products/accessories/motion/edr-wheels.html

11. Technobots Ltd. Universal Coupling Insert - 6mm [Internet]. Tehnobots Online Group. 2017 [tsiteeritud 11. mai 2017]. Salvestatud: https://www.technobotsonline.com/universal-coupling-insert-6mm.html

12. Jaaniste A, Adamson T. Kommunikatsioonimoodul [Internet]. Digilabor. 2015 [tsiteeritud 9. mai 2017]. Salvestatud: http://digi.physic.ut.ee/mw/index.php/Kommunikatsioonimoodul

13. Jaaniste A, Lember O. Mbed mainboard [Internet]. Digilabor. 2016 [tsiteeritud 11. mai 2017]. Salvestatud: http://digi.physic.ut.ee/mw/index.php/Mbed_mainboard

14. Intel Corporation. Intel NUC [Internet]. Intel. 2017 [tsiteeritud 11. mai 2017]. Salvestatud: http://www.intel.com/content/www/us/en/products/boards-kits/nuc.html

15. Open Source Robotics Foundation. About ROS [Internet]. ROS. 2017. Salvestatud: http://www.ros.org/about-ros/

16. Jõgeva J. Hüperboloidpeegliga roboti kaamera pildi kalibreerimine. 2012; Salvestatud: http://dspace.ut.ee/handle/10062/32815

17. Kallas P. Probabilistic Localization of a Soccer Robot. 2013; Salvestatud: http://dspace.ut.ee/handle/10062/33050

18. Open Source Robotics Foundation. What is a TurtleBot? [Internet]. TurtleBot. 2017 [tsiteeritud 11. mai 2017]. Salvestatud: http://www.turtlebot.com/

19. Open Source Robotics Foundation. move_base [Internet]. ROS wiki. 2016 [tsiteeritud 30. aprill 2017]. Salvestatud: http://wiki.ros.org/move_base

20. Open Source Robotics Foundation. navigation [Internet]. ROS wiki. 2017 [tsiteeritud 30. aprill 2017]. Salvestatud: http://wiki.ros.org/navigation

21. Open Source Robotics Foundation. Setup and Configuration of the Navigation Stack on a Robot [Internet]. ROS wiki. 2015 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/navigation/Tutorials/RobotSetup

22. Open Source Robotics Foundation. Nodes [Internet]. ROS wiki. 2012 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/Nodes

23. Open Source Robotics Foundation. Messages [Internet]. ROS wiki. 2016 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/Messages

24. Open Source Robotics Foundation. Packages [Internet]. ROS wiki. 2015 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/Packages

25. Morgan Quigley, Brian Gerkey WDS. Robots with ROS. 2015. 447 lk.

26. Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, et al. ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software [Internet]. 2009. Salvestatud: http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf

27. Open Source Robotics Foundation. Managing System dependencies. ROS wiki. 2016.

28. Quigley M, Berger E, Ng AY. Stair: Hardware and software architecture. AAAI 2007, Robot Work [Internet]. 2007;31–7. Salvestatud: http://www.aaai.org/Papers/Workshops/2007/WS-07-15/WS07-15-008.pdf

29. Ng AY. STAIR: STanford Artificial Intelligence Robot [Internet]. [tsiteeritud 30. aprill 2017]. Salvestatud: http://stair.stanford.edu/

30. Open Source Robotics Foundation. History [Internet]. ROS. 2016 [tsiteeritud 30. aprill 2017]. Salvestatud: http://www.ros.org/history/

31. Open Source Robotics Foundation. Robotics Projects [Internet]. Open Source Robotics Foundation. 2017 [tsiteeritud 30. märts 2017]. Salvestatud: https://www.osrfoundation.org/osrf-projects/

32. Open Source Robotics Foundation. Distributions [Internet]. ROS wiki. 2017 [tsiteeritud 30. aprill 2017]. Salvestatud: http://wiki.ros.org/Distributions

33. Open Source Robotics Foundation. Topics [Internet]. ROS wiki. 2014 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/Topics

34. Open Source Robotics Foundation. Writing a Simple Publisher and Subscriber (C++) [Internet]. ROS wiki. 2016 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

35. Open Source Robotics Foundation. Services [Internet]. ROS wiki. 2012 [tsiteeritud 14.

mai 2017]. Salvestatud: http://wiki.ros.org/Services

36. Open Source Robotics Foundation. Examining the Simple Service and Client [Internet]. ROS wiki. 2016 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/ROS/Tutorials/ExaminingServiceClient

37. Open Source Robotics Foundation. roscore [Internet]. ROS wiki. 2016 [tsiteeritud 30. aprill 2017]. Salvestatud: http://wiki.ros.org/roscore

38. Open Source Robotics Foundation. Master [Internet]. ROS wiki. 2012 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/Master

39. Open Source Robotics Foundation. Browsing packages for Indigo [Internet]. ROS. 2017 [tsiteeritud 11. mai 2017]. Salvestatud: http://www.ros.org/browse/list.php?package_type=package&distro=indigo

40. Open Source Robotics Foundation. rosserial [Internet]. ROS wiki. 2015 [tsiteeritud 11. mai 2017]. Salvestatud: http://wiki.ros.org/rosserial

41. Open Source Robotics Foundation. serial [Internet]. ROS wiki. 2014 [tsiteeritud 11. mai 2017]. Salvestatud: http://wiki.ros.org/serial

42. geometry_msgs/Twist Message [Internet]. ROS docs. 2017 [tsiteeritud 9. mai 2017]. Salvestatud: http://docs.ros.org/kinetic/api/geometry_msgs/html/msg/Twist.html

43. No Titlestd_msgs/String Message [Internet]. ROS docs. 2017 [tsiteeritud 17. mai 2017]. Salvestatud: http://docs.ros.org/kinetic/api/std_msgs/html/msg/String.html

44. Sattar A, Shumejko P, Siilak K-J, Peensoo KM, Yuschenko I. Frankenstein 2016 [Internet]. Digilabor. 2016 [tsiteeritud 17. mai 2017]. Salvestatud: http://docs.ros.org/kinetic/api/std_msgs/html/msg/String.html

45. Open Source Robotics Foundation. Ubuntu install of ROS Kinetic [Internet]. ROS wiki. 2017 [tsiteeritud 11. mai 2017]. Salvestatud: http://wiki.ros.org/kinetic/Installation/Ubuntu

46. Open Source Robotics Foundation. turtlebot/turtlebot [Internet]. GitHub. 2016 [tsiteeritud 17. mai 2017]. Salvestatud: https://github.com/turtlebot/turtlebot

47. MiguelAngel_LV. DroidTick [Internet]. Google Play. 2013 [tsiteeritud 17. mai 2017]. Salvestatud: https://play.google.com/store/apps/details?id=com.cc.droidtick&hl=et

48. López MÁ. DroidTick (English) [Internet]. Androides y Pingüinos. 2012 [tsiteeritud 17. mai 2017]. Salvestatud: https://miguelangellv.wordpress.com/droidtick-en/

49. Open Source Robotics Foundation. joy [Internet]. ROS wiki. 2016 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/joy

50. STMICROELECTRONICS NUCLEO-L476RG Development Board, STM32L476RG Arduino/mbed Nucleo, On-board STLINK/V2-1 [Internet]. Farnell. 2017 [tsiteeritud 17. mai 2017]. Salvestatud: http://ee.farnell.com/stmicroelectronics/nucleo-l476rg/dev-board-arduino-mbed-nucleo/dp/2493816

51. Open Source Robotics Foundation. Publishing Odometry Information over ROS [Internet]. ROS wiki. 2016 [tsiteeritud 17. mai 2017]. Salvestatud: http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom

Lisad

Lisa 1

Robotondi ROSi draiveri arendamise haldamiseks kasutakse GitHub platvormi. Käesoleva töö raames loodud kood on leitav GitHubi repositooriumist. Konkreetse töö raames kirjutatud kood asub täiemahuliselt kaustas /robotont_driver. Sõlmede alusfailide lähtekoodid on leitavad alamkaustast /src.

https://github.com/ut-ims-robotics/robotont/releases/tag/v0.1

Lihtlitsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Raid Vellerind, annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Avatud robotiarendusplatvormi ROS võimekuse loomine Tartu Ülikooli Robotexi robootikaplatvormile“, mille juhendaja on robootika dotsent Karl Kruusamäe

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

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

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

Tartus 17.05.2017