Tartu Ülikool
Loodus- ja täppisteaduste valdkond
Tehnoloogiainstituut

Sven-Ervin Paap

ROS2 draiver õpperobotile Robotont

Bakalaureusetöö (12 EAP)
Arvutitehnika eriala

Juhendaja:

PhD Veiko Vunder

Tartu 2024

ROS2 draiver õpperobotile Robotont

Lühikokkuvõte:

Tartu Ülikooli tehnoloogiainstituudis on olemas ROS1 toega mobiilne haridusrobootikaplatvorm Robotont. Robotil puudus ROS2 draiver, mistõttu ei olnud see funktsionaalsuselt päevakajaline ROS2-ga töötamisel nii teadus- kui õppetöö eesmärkidel.

Antud töö eesmärgiks oli uuendada Robotont draiverit ROS2 versioonile ning kaasajastamise käigus parendada pistikprogrammide integreerimist, verifitseerida draiveri arhitektuur ja avaldada draiver pakihaldusüsteemis.

Töö tulemusena loodi uus draiver ROS2 versioonile, mille arhitektuur verifitseeriti teiste robootikaplatvormide draiverite näidetel. Lisati võimalus käivitada pistikprogramme vastavalt parameetritele ja avaldati draiver *ROS index*i abil pakihaldusüsteemidesse.

Võtmesõnad:

ROS, ROS2, Robotont, robotics

CERCS:

T120 Süsteemitehnoloogia, arvutitehnoloogia; T125 Automatiseerimine, robootika,

control engineering

ROS2 driver for the educational robot Robotont

Abstract:

At the University of Tartu's Institute of Technology, there is a mobile educational robotics platform called Robotont, which supports ROS1. The robot lacked a ROS2 driver, making it outdated in terms of functionality for working with ROS2 for both research and educational purposes.

The goal of this work was to update the Robotont to the ROS2 version, and during the update process, to improve the integration of plugins, verify the driver's architecture, and publish the driver in the package management system.

As a result of this work, a new driver was created for the ROS2 version, and its architecture was verified using examples from other robotics platform drivers. The capability to run plugins according to parameters was added, and the driver was published in package management systems using ROS index.

Keywords:

ROS, ROS2, Robotont, robotics

CERCS:

T120 Systems engineering, computer technology; T125 Automation, robotics, control engineering

Sisukord

Jooniste loetelu

1. Sissejuhatus

Haridusrobootika on valdkond, mis kasutab õpetamisprotsessis roboteid ja täidab seeläbi olulist rolli robootikainseneride puuduse lahendamisel. Haridusrobotplatvormi ja õppematerjalide olemasolu langetab robootikaga alustamise barjääri ning õpetab vajalikke robootikaalaseid oskuseid. [1]

Robotont on Tartu Ülikooli tehnoloogiainstituudi poolt juhitav robootikaplatvormi arendusprojekt. Platvormi kasutatakse robootika-alaseks õppe- ja teadustööks. Koolitusi on Robotondi põhjal läbi viidud nii Tartu Ülikoolis kui ka muudes õppeasutustes ning mitmed bakalaureuse- ja magistritööd toetuvad Robotondi olemasolule. [2, 3]

ROS (*Robot Operating System*) on standardiseeritud tarkvararaamistik, mille peamine eesmärk on toetada tarkvaraarendust erinevatele robotitele. Selle laiapõhjaline kogukond tagab ROS-i kasutamise kõikides võimalikes valdkondades, mis robootikale toetuvad ja ROS on muutunud robootikas standardiks. [4]

Robotondil oli olemas ROS1 versioonil baseeruv draiver, kuid selleks, et õppetöö saaks toimuda võimalikult kaasaegsete õppevahenditega, oli vaja roboti ajakohastamiseks luua ROS2 draiver. Draiveri olemasolu võimaldab ülejäänud riist- ja tarkvarakomponentide ning õppematerjalide moderniseerimist.

Käesoleva töö eesmärgiks on töötada välja Robotondi ROS2 draiver ning parendada selle käigus pistikprogrammide kasutamist ja valideerida draiveri arhitektuuriline ülesehitus teiste haridusrobotplatvormide ROS2 draiverite põhjal.

2. ROS

Roboti Operatsioonisüsteem ehk ROS on avatud lähtekoodiga vahevara ehk tarkvara, mis seob kaht iseseisvat rakendusprogrammi. Kuigi nimi sellele viitab, ei ole ROS siiski operatsioonisüsteem vaid tarkvararaamistik, mille eesmärk on toetada tarkvaraarendust erinevatele robotitele. ROSi populaarsuse on taganud selle multifunktsionaalsus ja iseseisvus arendusplatvormist. Koodi kirjutamine on võimalik mitmes eri keeles ja aitab hõlpsalt siduda eri riist- ja tarvarakihte. [4]

2.1 Ajalugu

ROSi eelkäija sai alguse Stanfordi Ülikooli robootikalaboris: kaks doktoranti, Eric Berger ja Keenan Wyrobek leidsid, et mitmed nende kolleegid, kes on väga head tarkvaraarendajad, ei saa oma täit potentsiaali rakendada, sest neil puudub robootikas vajalik laiapõhjaline teadmine. Keegi, kes on võimeline arendama tipptasemel trajektoori planeerimise algoritmi, ei pruugi olla piisavalt pädev masinnägemises selleks, et oma loodud algoritmi praktiliselt rakendada. Lahenduseks sellele probleemile hakatigi planeerima ROSi eelkäijat. [5, 6] Esimene koodiversioon laeti üles novembris 2007 [7].

Teadusinkubatsioonikeskuses Willow Garage jätkati tööd ROSiga. Ehitati robot PR2 (*Personal Robot 2*), mis kasutas juba ROS tarkvara, kuhu panustasid enam kui 20 erinevat institutsiooni. Panustati nii tuum-tarkvarasse kui ka tugikimpudesse [8]. Kuna ROSi arendasid ka teised peale Willow Garage, tähendas see, et ROS oli juba algusest multi-roboti platvorm ehk see disainiti töötama mitmel robotil, mitte ühel konkreetsel.

2009. aasta augustis loodi ROS.org veebileht, kuhu postitati esmaseid juhendeid, mis aitasid valmistuda ROS 1.0 avalikustamiseks 2010 jaanuaris. Toodeti suur kogus dokumentatsiooni ja juhendeid kolme aasta jooksul loodud tarkvaralise võimekuse kohta. Andmaks hoogu enda platvormi juurutamisele, jagas Willow Garage kümnele kõige võimekamale ülikoolile PR2 riistvaraplatvormi. Lisaks sellele oli nende praktikaprogramm ülimalt edukas, mis aitas levitada ROSi kohta teadmisi. [9-12]

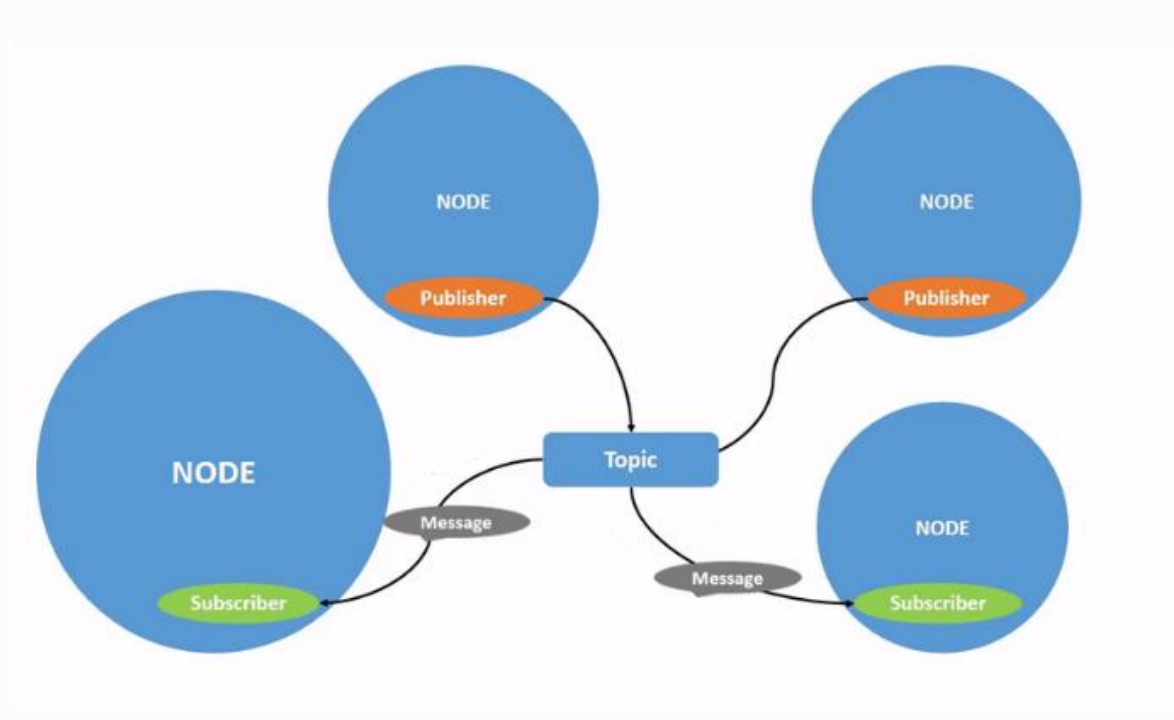
2011 oli ROSi jaoks plahvatusliku populaarsuse kasvu aasta. 15. veebruaril loodi ROS Answers – küsimuste ja vastuste foorum ROSi kasutajatele. 18. aprillil tutvustati väga edukat TurtleBot robotikomplekti, tänu millele kasvas ROSi erinevate harude arv koodihoidlates üle saja 5. maiks. [13-15]

2012. aastal lõi Willow Garage organisatsiooni OSRF, millest sai aastast 2013 ROSi peamine tarkvara haldaja [16]. Selleks hetkeks oli avalikustatud seitse peamist distributsiooni ja ROSil oli kasutajaid üle kogu maailma. Aastast 2013, pärast OSRFi poolt projekti haldamise üle võtmist, on igal aastal avalikustatud uus ROS distributsioon. Erandiks oli aasta 2019, mil uut peamist distributsiooni ei avaldatud. [17]

2014. aastal kuulutati välja ROS2, mis on suuremahuline rakendusliidese ümberdisain. ROS2 on mõeldud toetama reaalajalist programmeerimist, laiemat tuge erinevatele arvutuskeskkondadele ja kasutama modernsemaid tehnoloogiaid. Esimene üldkasutusse mõeldud ROS2 distributsioon, *Ardent Apalone* avaldati 8. detsembril 2017. [18 - 20]

2.2 Struktuur

ROS on arendatud pidades silmas avatud lähtekoodi põhimõtteid, et kasutajad saaksid ise valida, millises konfiguratsioonis ja milliseid tööriistu ning kimpe kasutatakse ROSi komponentide vahelisel suhtlemisel ja info töötlemisel. Selle eesmärk on võimaldada kasutajal kohandada oma ROSi installatsiooni vastavalt vajadusele ja robotplatvormile. Suuremahulise kohandatavuse võimaldamiseks on olemas komponendid: tuum (ROS1), sõlm, rubriik, teenus ja parameetrid. Komponentide omavaheline suhtlus on kujutatud joonisel 1 ning järgnevalt kirjeldatakse komponentide olemust [21].



Joonis 1. ROS komponentide struktuur

Tuum (ingl *core, roscore*) on kogum sõlmedest ja programmidest, mis on eeltingimuseks ROSil põhineva süsteemi käivitamiseks. Tuum peab olema käivitatud selleks, et sõlmed saaksid omavahel suhelda. Tuum eksisteeris vaid ROS1 versioonis. ROS2 ei vaja sõlmede juhtimiseks tuuma. [22, 23]

Sõlm (ingl *node*) on ROSi struktuuris käivitatud ja hallatud protsess. Igal sõlmel on nimi, millega see ROS tuumas registreeritakse. Unikaalne nimi on oluline erinevate sõlmede vahelisel suhtlusel. Sõlmed on ROS programmeerimise põhiline komponent. Enamus ROS kliendi koodist on sõlme kujul, mis sooritab tegevusi vastavalt teistest sõlmedest pärit informatsioonile, saadab selle teistele sõlmedele või saadab ning võtab vastu tegevuspäringuid muudelt sõlmedelt. [24]

Rubriik (ingl *topic*) on kanal, mida sõlmed sõnumite edastamiseks kasutavad. Rubriikidel peab samuti olema unikaalne nimi. Sõnumite saatmiseks rubriiki peab sõlm sellesse rubriiki sõnumeid avaldama. Sõnumite vastuvõtmiseks peab sõlm rubriiki tellima. Saatmine (kuulutamine) ja vastuvõtmine (tellimine) on anonüümsed. Ükski sõlm ei tea, millisest rubriigist sõnum saadetakse või vastu võetakse. Sõnumite tüübid ei ole fikseeritud, vaid võivad varieeruda ning on kasutajate poolt modifitseeritavad. Rubriikide abil jagatakse üldiselt näiteks sensorite mõõteandmeid, mootorite kontrollsõnumeid ja oleku informatsiooni. Sõnumi sisu ei ole aga mingil määral piiratud. [25]

Lisaks rubriikidele saab iga sõlm saab välja kuulutada **teenuseid** (ingl *service*). Teenus on tegevus, mida sõlm saab teha, millel on üks kindel tulem. Sellest tulenevalt kasutatakse teenuseid tihti tegevusteks, millel on kindel algus ja lõpp, näiteks sensorilt ühe mõõtetulemuse lugemine. Sõlmed avaldavad ja kutsuvad omavahel teenuseid. [26]

Parameetrid (ingl *parameters*) on sõlmede vahel jagatud muutujad, kus hoiustatakse staatilist või poolstaatilist infot, mis ei muutu tihti ja mille lugemiskiirus on madal. Näiteks roboti kaal või distants kahe fikseeritud punkti vahel on sobilikud parameetrina hoiustamiseks. [26]

Graafi struktuuris kujutatakse ROSi protsesse sõlmedena, mis on ühendatud servadega, mida kutsutakse rubriikideks. Sõlmed saavad saata üksteisele sõnumeid läbi rubriikide, teha teenuse kutseid, teenuseid pakkuda ja kirjutada või lugeda parameetreid. ROS1 versioonis eksisteerib ROS tuum, mis toimib serverina ja teab keskselt kõigest, mis graafis toimub. Tuum registreerib kõik sõlmed, seab üles sõlmvõrgu rubriikide abil suhtlemiseks ja kontrollib parameetrite serveri uuendusi. Sõnumite saatmine ja teenuste kutsed ei käi läbi

tuuma. Tuum konfigureerib partnervõrgu kõikide sõlmede vahel pärast seda, kui need tuumas registreeritakse. Selline detsentraliseeritud arhitektuur sobib hästi just robotitele, mis koosnevad tihti nii arvutiriistvarast kui ka välistest manussüsteemidest, mis omavahel suhtlevad. [27]

2.3 Erinevused ROS 1 ja ROS2 vahel

Kuna ROS1 loodi juba 2007. aastal, oli seda arendanud meeskond kümme aastat hiljem oluliselt kogenum selle osas, mis on valesti ja puudu või mida oleks võinud hoopis teistmoodi üles ehitada. Selliste arhitektuuriliselt erinevate muudatuste tegemiseks ei olnud mõistlik muuta ROS1 lähtekoodi, vaid luua nullist uus ROS, sest ROS1 muutmine oleks lõhkunud mitmeid seni kasutusel olnud funktsionaalsuseid ja muutnud selle ebastabiilseks. [28]

ROS1 kasutas erinevaid teeke C++ ja Pythoni koodi jaoks, mis tähendas, et neis võis esineda erinevusi, sest need olid mõlemad nullist ehitatud eraldiseisvad teegid. Arendajatele tähendas see, et rakendusliides ja funktsionaalsus ei olnud mõlemal teegil tingimata samasugune. ROS2 rakendab rohkem tarkvaralisi kihte ja kasutab mõlemas programmeerimiskeeles üht „rcl“ nimelist teeki, mis on implementeeritud C-keeles ja on aluseks kõikidele ROS2 funktsioonidele. ROS2 implementeerimiseks ei kasutata rcl teeki otse vaid kasutatakse klienditeeki. C++ puhul rclcpp ja Pythoni puhul rclpy. See võimaldab Willow Garage'il uute funktsioonide implementeerimiseks või vigade parandamiseks muuta ainult rcl teeki. Arendajatele, kes ROS2'te kasutavad, tähendab see, et rakendusliides erinevates keeltes kirjutatud komponentide vahel on identsem ja kui avaldatakse uus funktsionaalsus, on see kõikides keeltes samaaegselt saadaval. [29]

ROS1 tuuma loomisel oli eesmärgiks kasutada C++ versiooni 03 ja Pythoni versiooni 2. See tähendab, et ROS1 ei kasuta ära C++ 11, Python 3 ja uuemate versioonide funktsionaalsust oma rakendusliideses, mis tõstis ROS1 kasutamisel märgatavalt implementeerimise keerukust, muutis seda aeglasemaks ja lisas turvariske. ROS2 on ehitatud selliselt, et see kasutab C++ 11 ja 14 ning Python 3 funktsionaalsust vaikimisi. See suurendab saadavalolevat funktsionaalsust, muudab arendamise kaasaegsemaks ning turvalisemaks. [28, 29]

Lisaks erinevatele tehnoloogilistele uuendustele kaasnes uue versiooniga ka arendamise reeglistik. Kui varasemalt võis igaüks kirjutada oma sõlmesid nii nagu soovis, siis uue objektorienteeritud programmeerimise põhimõtetel põhineva reeglistikuga pandi paika,

kuidas tuleb sõlmesid luua. Selline kindel reeglistik on oluline, et muuta koodibaasi puhtamaks ja selgemaks ning võimaldada erinevatel osapooltel, kes konkreetse projekti kallal töötavad, asju lihtsamini haarata. [28]

ROS1 kasutas ülem-alluv-süsteemi ja XML-RPC protokolli. Seda kasutati sõlmede ja protsesside vaheliseks suhtluseks (*inter-process communication (IPC)*). XML abil kodeeriti sõnumid ja neid transporditi HTTP abil. See lisas keerukust mitme robotiga hajussüsteemi ehitamisel, sest vajas keskset serverit, kuhu kõik robotid ühendusid. Sellise konfiguratsiooni puhul tekib üksik rikkepunkt, mille rikke või sellega ühenduse kaotamise korral on sinna ühendunud robotid juhtimiseta. Selle vältimiseks on võimalik kasutada näiteks ROS-i kimpu *multimaster_fkie*, mis vahendab sõnumeid samas võrgus asuvate robotite tuumade vahel [30]. Selline implementatsioon tõstab aga veelgi enam ROS-i implementeerimise keerukust. [23]

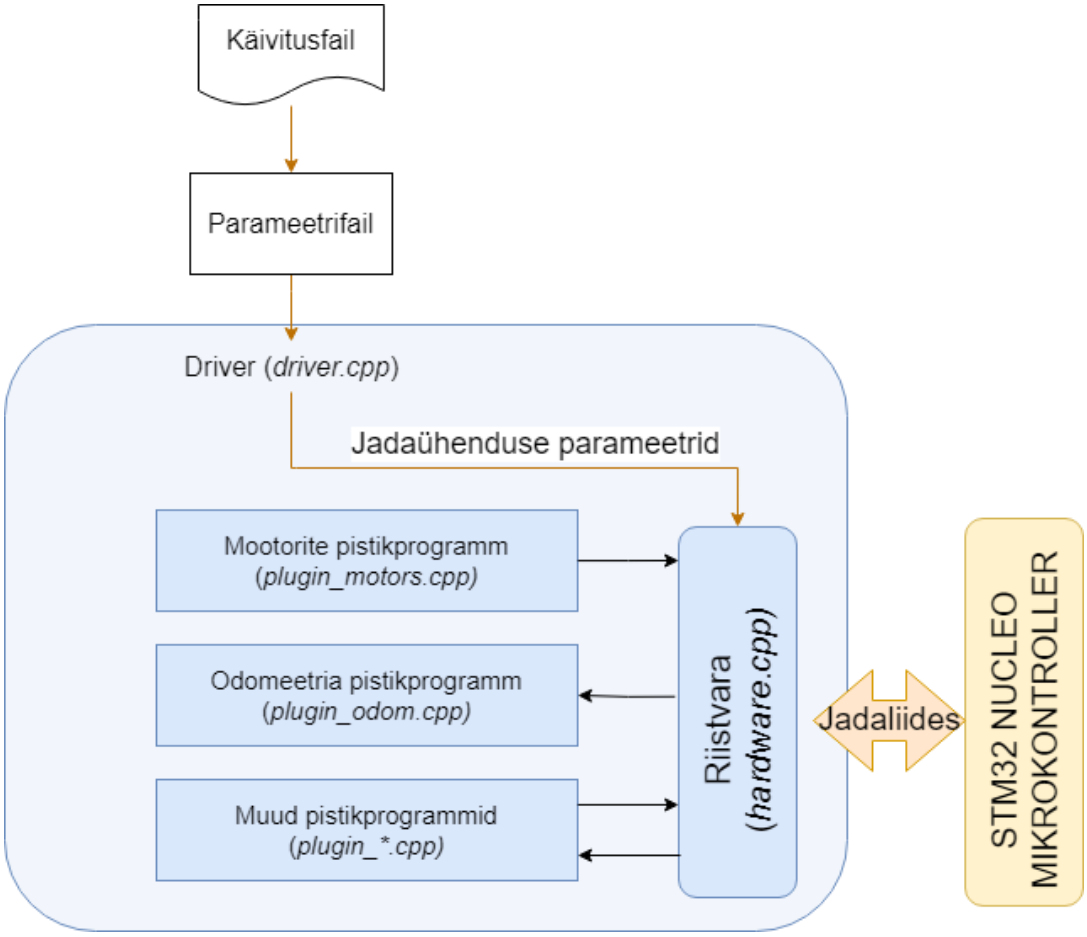
ROS2 kasutab andmejaotusteenust (ingl *Data Distribution Service (DDS)*), mis on disainitud olema efektiivsem, töökindlam, madalama latentsusega ja skaleeritavam. DDS on uus protokoll, mis asendab XML-RPC sõlmede vahelises suhtluses. Kuigi XML-RPC on parem lihtsate protseduuriliste väljakutsete jaoks, toetab DDSi lisanduv kompleksus paremini reaalajatöötlusega tegelevaid süsteeme. Tänu sellele kaob ka ROS2 implementeerivatest süsteemidest ROS1-le omane üksik rikkekoht, sest ROS2 toetab mitme sõlme samaaegset käivitamist. See kasutab paremini ära modernseid mitme tuumaga protsessoreid. [23]

3. Robotont

Robotont on avatud platvorm robootika õppimiseks ja teadustööks. See on omniliikuv mobiilne robot, mis toetab ROS tarkvara. Robotondi arendust juhib Tartu Ülikooli tehnoloogiainstituut. Tarkvara arendus toimub kogukonnapõhiselt *GitHub*is ja on kättesaadav kõigile. Nii riistvara- kui ka elektroonika joonised on olemas *GitHub*is, mis tähendab, et igaüks võib endale Robotondi ehitada [31, 32]. Füüsilise Robotondi olemasolu ei ole hädavajalik, sest roboti digikaksik on võimaldab opereerida ka virtuaalselt Gazebo simulatsiooni keskkonnas. [2]

3.1 Robotont ROS1 draiver

Robotondi draiver koosneb mitmetest modulaarsetest komponentidest. Pardaarvutil käivitatav draiver loob ROS sõlme ja draiveri sõlme, käivitab vajalikud pistikprogrammid ning haldab üle jadaliidese suhtlust mikrokontrolleriga. Draiveri ülesehitus on kujutatud joonisel 2. [33]



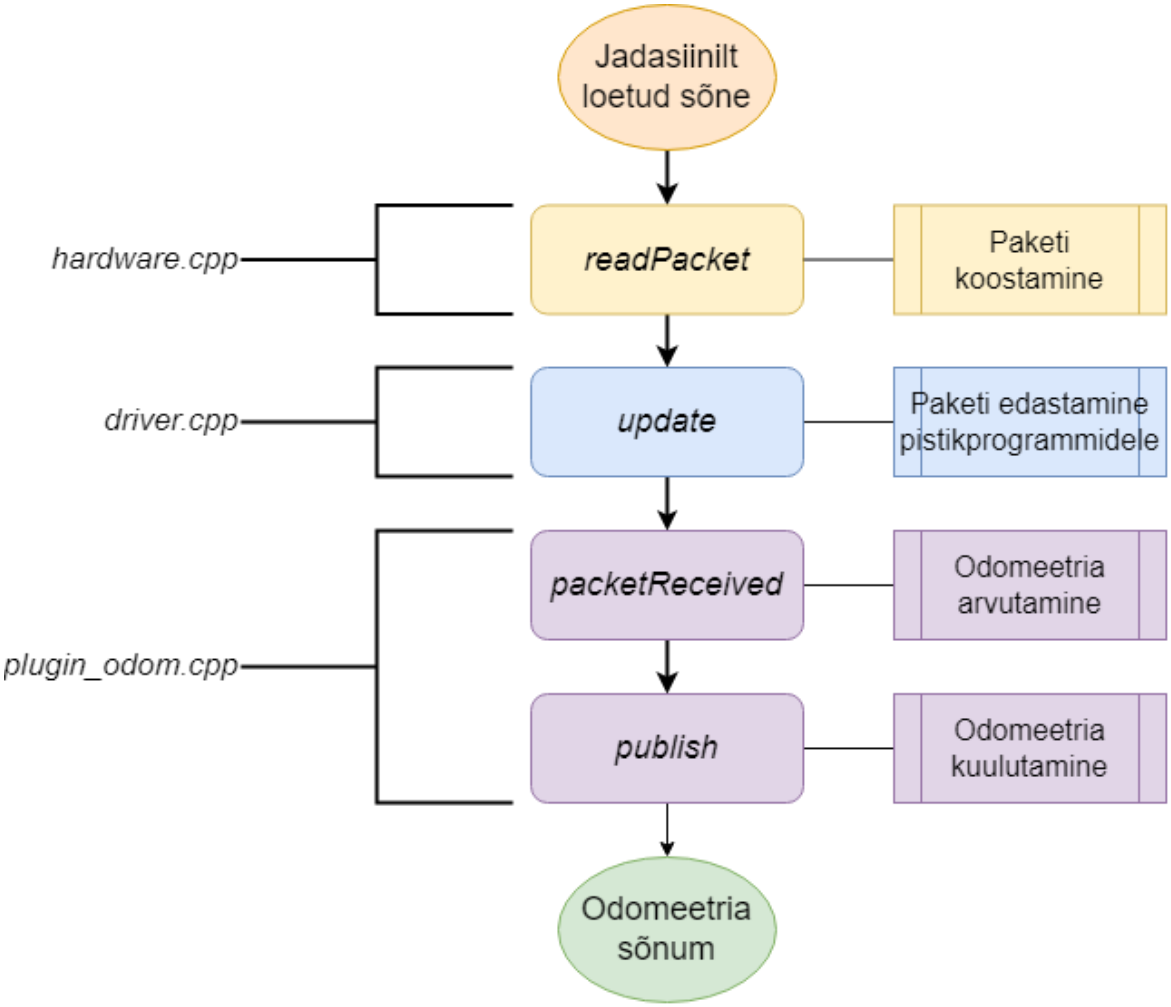
Joonis 2. Robotont ROS1 draiveri arhitektuur

Modulaarsuse saavutamiseks on draiver jagatud mitmeks eri komponendiks, mis täidavad spetsiifilisi ülesandeid. Järgnevalt kirjeldatakse erinevaid draiveri komponente, nende ülesandeid ja viise, kuidas neid ülesandeid täidetakse.

Draiver (*driver.cpp*) on keskne klass, mis koordineerib kõiki pistikprogramme ja haldab riistvaraga suhtlemist. Draiver käivitatakse *driver_node.cpp* failis, mis registreerib ROS-sõlme ja käivitab põhilõime. Peale käivitamist loob draiver pistikprogrammide instantsid ja seejärel edastab nendele jadaliidese kaudu saabunud infot. Iga paketi saabumisel riistvarast edastab draiver selle igale pistikprogrammile, mille loogika otustab, kas paketis olevat infot on vaja töödelda või mitte. [33]

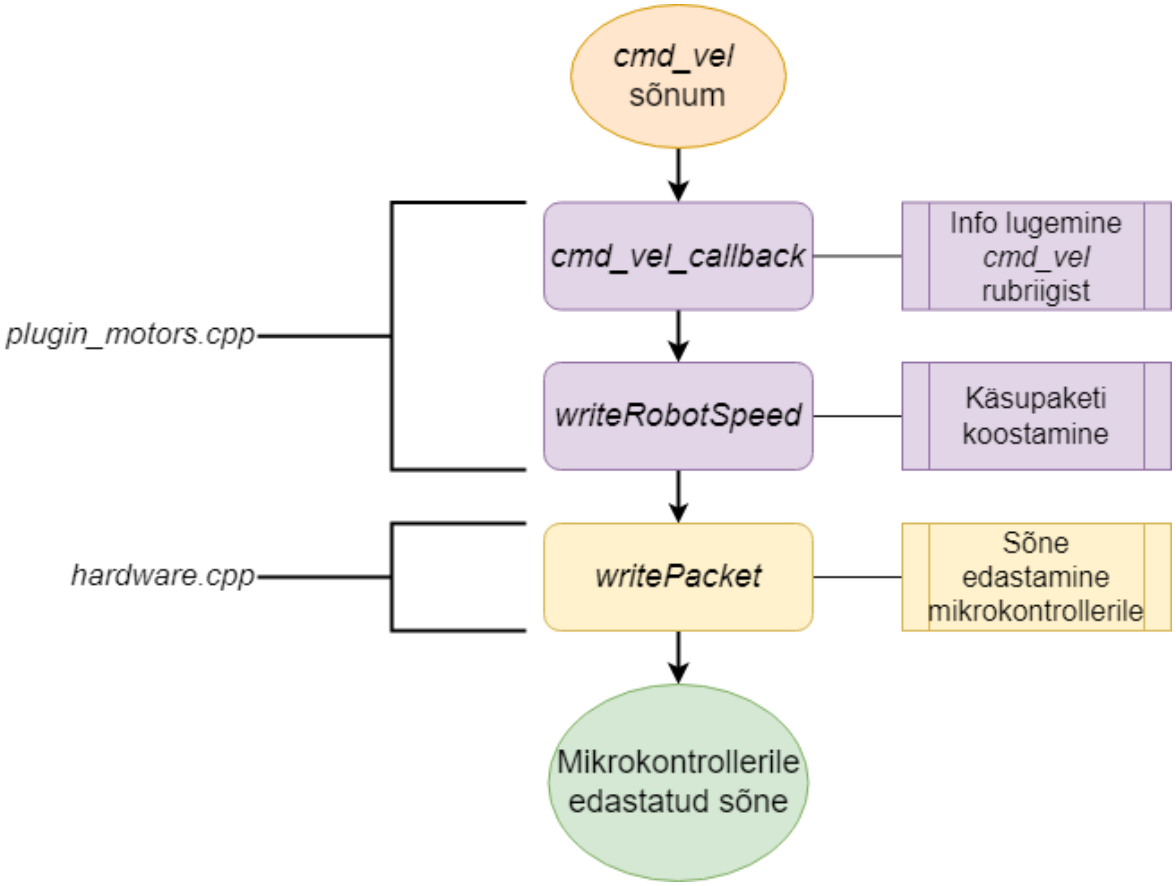
Riistvara klass (*hardware.cpp*) haldab madalatasemelist suhtlust riistvaraga, kasutades selleks jadaühendust, mis luuakse Robotondi mikrokontrolleriga üle USB-liidese. *Hardware.cpp* tegeleb andmete vastuvõtmise ja saatmisega ning vigade ja ühenduse probleemide haldamisega. Andmete lugemisel arvestatakse jadasiini levinumate probleemidega, kontrollides andmete terviklikkust ja sisend-väljund erindeid. Klassis on funktsioon *checkConnection*, mis kontrollib ROS taimerit kasutades regulaarselt, kas ühendus on katkenud või on vajalik taasühendamine. Taasühendamise vajadusest antakse märku *reconnect_requested* lipu tõeseks seadmisega, misjärel proovitakse uuesti samade parameetritega (*port, baud rate, timeout*) ühendus taastada. [33]

Odomeetria (*plugin_odom.cpp*) on oluline komponent roboti liikumise jälgimiseks ja selle ruumilise asendi mõistmiseks. See pistikprogramm kasutab riistvara klassi poolt kogutud andmeid roboti asukoha ja liikumise arvestamiseks ning avaldab odomeetria sõnumeid *odom* rubriigis. Odomeetria sõnumi koostamiseks on kasutusel funktsioon *packetReceived*, mis saab parameetriks riistvara klassilt kogutud sõne. Selle sõne põhjal väärtustatakse positsiooni ja kiiruse muutujad, arvutatakse kvaternioon ning lisatakse kõik saadud muutujad koos ajaga odomeetria sõnumisse. Joonis 3 annab visuaalse ülevaate informatsiooni liikumisest odomeetria sõnumi avaldamiseks. [33]



Joonis 3. Infovahetus odomeetria sõnumi avaldamiseks

Mootorite pistikprogramm (*plugin_motors.cpp*) tegeleb mootorite juhtimisega ja võimaldab Robotondil liikuda vastavalt saadud liikumiskäskudele. Käivitamisel loob konstruktor tellija rubriigile *cmd_vel*, kuhu edastatakse *geometry_msgs::Twist* tüüpi sõnumeid, mis sisaldavad lineaarset ja nurkkiirust. Rubriiki saabuvate sõnumite põhjal määratakse igale mootorile sobilik kiirus ja edastatakse see riistvara klassi, mis seejärel kiirused mikrokontrollerile saadab. Joonisel 4 on visualiseeritud liikumiskäskude lugemine, sõnede loomine ja mikrokontrollerile edastamine. [33]



Joonis 4. Info liikumine läbi draiveri funktsioonide mootori kiiruste seadmiseks

Matkedraiver (*fake_driver_node.cpp*) registreerib ROS-sõlme, mis simuleerib roboti draiverit. See imiteerib riistvara käitumist, et võimaldada süsteemi testimist ja arendamist ilma füüsilise robotita. Matkedraiver sisaldab endas riistvara, odomeetria ja mootorite funktsionaalsust. Integreeritakse saabunud kiiruse sõnumeid ja väljastatakse nende põhjal ideaalne odomeetria, sest ei arvestata roboti massiga ega seetõttu ka kiirendusega. [33]

Käivitusfaili kasutamisel kasutatakse konfiguratsioonifailis asuvaid jadaühenduse ja odomeetria päise parameetreid, mis laetakse parameetriserverisse ja seejärel kasutatakse sõlme või klassi käivitamisel. **Konfiguratsioonifail** (*robotont_params.yaml*) sisaldab parameetreid jadaühenduse (port ja sümbolikiirus (*baud rate*)) ja odomeetria päise jaoks (*frame* ja *child_frame*). [33]

Käivitusfaile (*driver_basic.launch*, *fake_driver.launch*) kasutatakse eelseatud parameetritega draiveri sõlme käivitamiseks. [33]

4. Olemasolevad ROS2 draiverid

4.1 Micro-ROS-i kasutamine

ROS1 puhul ei olnud spetsiifiliselt eelistatud viisi manusüsteemidega suhtlemiseks. ROS2 integreerib Micro-ROS-i, mis on spetsiifiliselt mikrokontrollerite jaoks disainitud ROS-i versioon. Selle arendamist aitas rahastada Euroopa Liidu projekt OFERA [34]. Kasutades Micro-ROS-i on manusüsteemi ja ROS rakendusliides tihedamalt integreeritud, sest see võimaldab kasutada rubriike ja sõlmesid, mis oskavad süsteemiüleselt üksteisega suhelda. [35]

Micro-ROS-i kasutamiseks tuleks Robotondi mikrokontrollerile kirjutada uus püsivara, sest Micro-ROS ei ole tagasiulatuvalt ROS1-ga ühilduv [35]. Robotondil peab olema käivitatav nii ROS1 kui ka ROS2 draiver ilma vajaduseta mikrokontrolleri püsivara ümber kirjutada. Eriti oluline on see teise põlvkonna Robotontide uuendamisel, mida praktilises osas uue draiveri loomiseks kasutati. ROS2 draiver peab Robotondi manusüsteemiga suhtlema samamoodi nagu ROS1 draiver – üle jadapordi. Selliselt integreerituna on võimalik erinevate ROS tuuma versioonide vahel vahetada, valides vaid käivitatavat draiverit ja vastavat ROS versiooni.

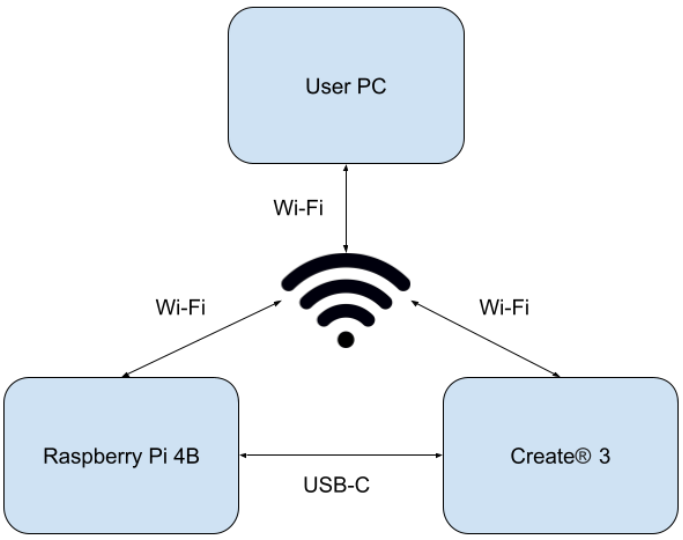
4.2 Husarion Panther

Husarion on Poolas asuv ettevõte, mis pakub erinevaid mobiilseid roboti arendusplatvorme, tarkvara- ja ühenduvuslahendusi oma klientidele, et võimaldada neil ehitada ROS ja ROS2-l baseeruvaid autonoomseid süsteeme lihtsalt ja efektiivselt. Nende tootevalikusse kuulub 4 erinevat platvormi: Panther, ROSbot XL, ROSbot 2 PRO ja ROSbot 2R. [36, 37]

Panther kasutab riistvaraga suhtlemiseks kontrollervõrku *CAN (Controller Area Network)*, ülejäänud kolme roboti ROS2 draiverid toetuvad Micro-ROSile [38, 39]. Panther on Husarioni kõige vanemat tüüpi mudel ja selle esimene draiver kirjutati ROS1-te silmas pidades, mistõttu on selle arhitektuur erinev teistest Husarioni toodetest ja tõenäoliselt ei ole seetõttu kasutusel ka Micro-ROS. Pantheri draiveri arhitektuur on sarnane Robotondi ROS draiverile, mis kasutab üht klassi, mis on pühendatud suhtlusele riistvaraga. Ülejäänud tarkvara komponendid toetuvad sealt toodavale infole ning kasutavad seda andmete saatmiseks mikrokontrollerile. [40]

4.3 Clearpath Robotics TurtleBot 4

Kanada ettevõtte Clearpath Robotics mobiilne roboti arendusplatvorm TurtleBot 4 ja selle variatsioonid (TurtleBot 4 Standard ja TurtleBot 4 Lite) kasutavad iRobot® Create® 3 Educational Robot platvormi. Create 3 platvorm on sensorite, mootorite ja akuga varustatud robot, mis suhtleb pardaarvutiga kasutades üle USB-C kaudu loodava kohtvõrgu liidese (visualiseeritud joonisel 5) [41]. Create® 3 tegeleb mootorite juhtimise ja andurite info edastamisega mikrokontrolleril oleva püsivara abil, mis loob ROS2 sõlme ja kuulab ning edastab vastavates rubriikides sõnumeid. Seega on TurtleBot® 4 draiver lihtsakoelisem, sest ei pea tegelema spetsiifiliselt riistvara suhtlusega, vaid saab toetuda Create® 3 roboti püsivarale. Sellegipoolest on TurtleBot 4 arhitektuuris komponent *turtlebot4.cpp*, mis loob vastavate rubriikide kuulajad ja kuulutajad ning kasutaja loodavad sõlmed peaksid kasutama seal deklareeritud funktsioone. [41, 42]



Joonis 5. TurtleBot 4 pardaarvutite ühenduvus

4.4 Järeldused teistest ROS2 draiveritest

Analüüsides erinevaid ROS2 draivereid, mis kasutavad erinevaid füüsilisi ja tarkvaralisi kihte, saab järeldada, et levinud on arhitektuur, kus üks klass haldab riistvaralist suhtlust

ning loob sellega võimaluse integreerida kasutaja loodava koodi, kuhu kirjutatakse roboti juhtimiseks vajalik loogika. See on sarnane olemasoleva Robotondi ROS1 draiveri arhitektuuriga, kuhu on lisaks sellele loodud ka modulaarsed pistikprogrammid, mis tegelevad vastavate riist- või tarkvaraliste komponentidega, võimaldades arendajal mitte mõelda andmete edastamise võimekusele ja keskenduda pistikprogrammi loogikale.

5. Töö eesmärk ja nõuded

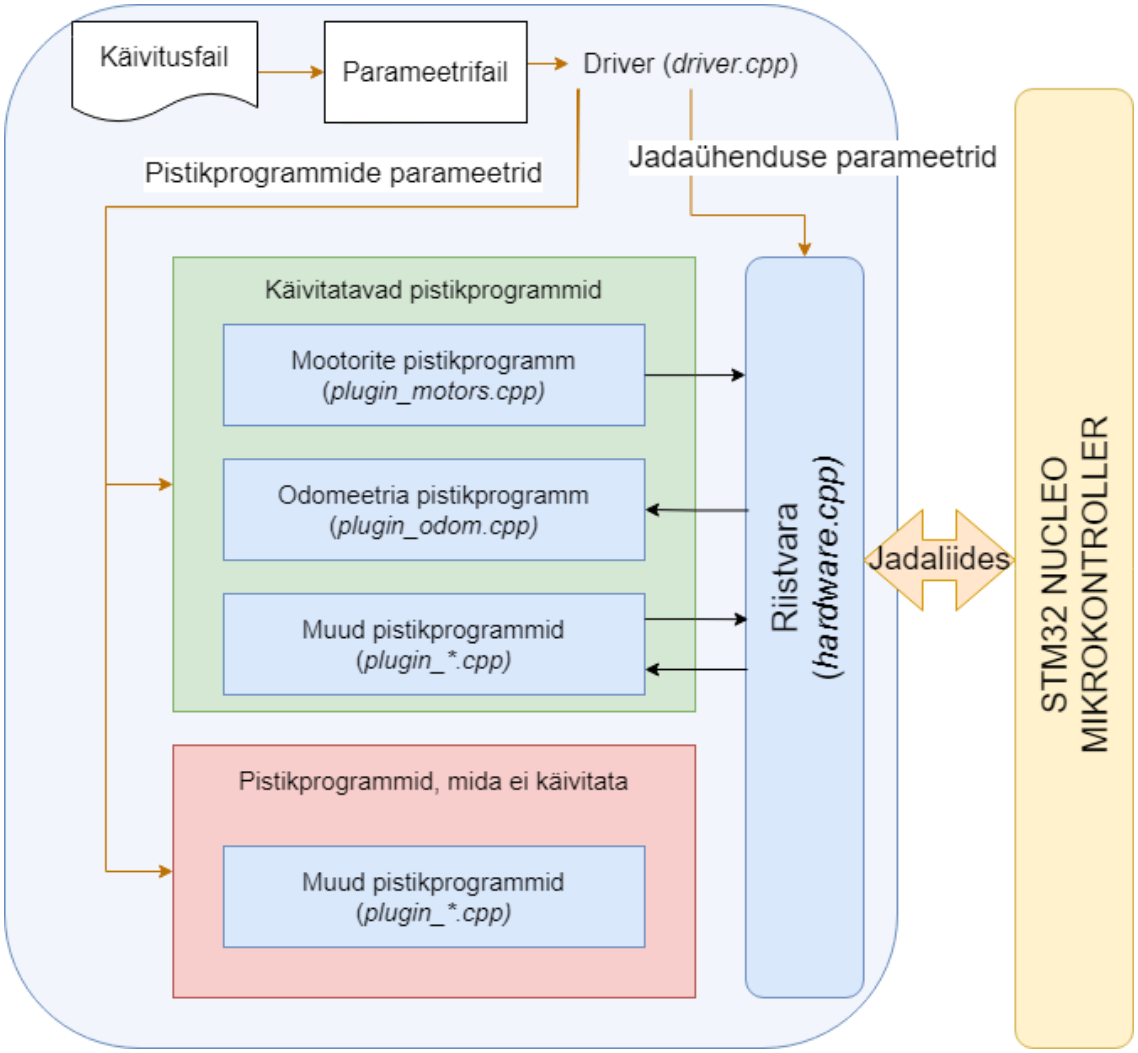
Käesolev lõputöö keskendub Robotondi ROS2 draiveri väljatöötamisele, tuginedes varasemalt arendatud ROS1 draiverile. Uue draiveri eesmärk on säilitada kõik olulised baasfunktsionaalsused, mis olid olemas ROS1 draiveris ning kohandada need ROS2 arhitektuurile vastavaks. Uue draiveri loomise käigus võeti ka eesmärgiks parendada pistikprogrammide arendamist ja käivitamist, ning teha draiver pakihaldussüsteemi kaudu saadavaks. ROS2 draiverile sõnastati järgnevad nõuded:

- Draiver realiseerib ühenduse roboti mikrokontrolleriga USB jadaliidese kaudu kasutades selleks ROS-i või operatsioonisüsteemi sisseehitatud teeke.
- Ühenduse katkemisel (nt. kaabli eemaldamisel) proovitakse ühendust kindlaksmääratud intervalliga taasluua.
- Draiver tellib *Geometry_msgs::Twist* tüüpi sõnumeid rubriigist *cmd_vel* ja koostab nende põhjal mikrokontrollerile edastamiseks sobiva mootorite kiiruste paketi
- Draiver interpreteerib mikrokontrolleri poolt saadetavaid andmepakette ja suunab need edasi sõnede lugemine ja nende põhjal odomeetria sõnumite koostamine.
- Draiver arvutab, väärtustab ja kuulutab *nav_msgs::Odometry* sõnumeid rubriigis *odom*.
- Draiveri arhitektuur võimaldab läbi modulaarsuse teiste pistikprogrammide arendamist.
- Draiveril on matkeraiveri funktsionaalsus käivitada draiverit ja sellega töötada ilma füüsilise robotita.
- Draiver avaldatakse pakihaldusüssteemis (nt. *apt*, *yum*).
- Draiveri arhitektuur verifitseeritakse teiste haridusrobootikaplatvormide draiverite näidetel.

6. Robotondi ROS2 draiver

6.1 Disain ja arhitektuur

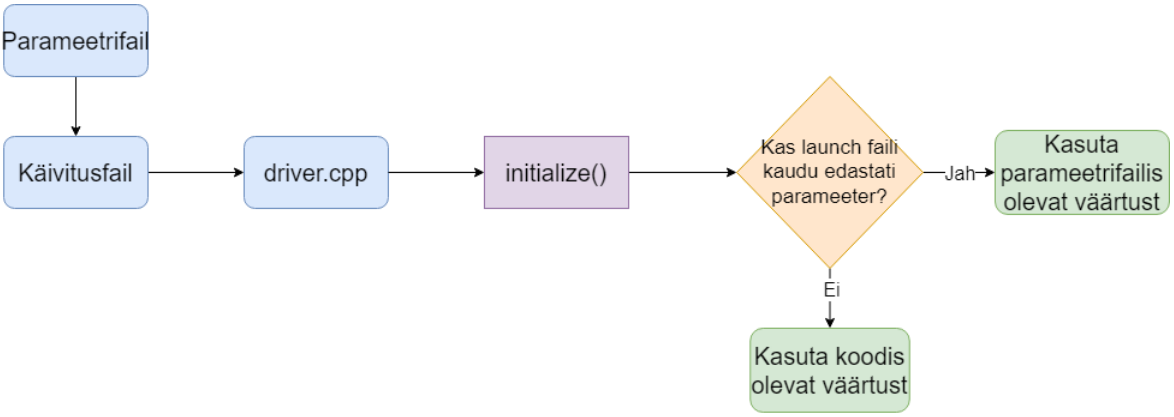
Robotondi ROS2 draiver disainiti antud töö raames modulaarsena ning on jagatud mitmeks komponendiks: ROS2 sõlm, riistvara suhtluse haldur ja pistikprogrammid spetsiifiliste funktsioonide jaoks (joonis 6) [43]. Implementeeritud on draiver koos mootorite ja odomeetria pistikprogrammidega ning matkedraiver. Lähtekood asub Lisa 1. Matkedraiver võimaldab koodi käivitada ja testida ilma füüsilise Robotondita, seda simuleerides. [43]



Joonis 6. Robotont ROS2 draiver arhitetktuur

Võrreldes ROS1 draiveriga lisati ROS2 draiveris võimalus draiveri käivitamisel spetsifitseerida, milliseid pistikprogramme käivitatakse (joonis 6) [43]. ROS1 draiveri käivitamisel initsialiseeritakse pistikprogrammid parameetrite järgi, mis on otse koodi

kirjutatud, mitte konfiguratsioonifailist. See tähendab, et ka juhul kui pistikprogrammi ei ole spetsiifilises konfiguratsioonis vaja, kulutatakse selle käskude täitmiseks protsessori arvutustsükleid. ROS2 pistikprogramme käivitatakse koos draiveri käivitamisega, seega puudub võimalus pistikprogramme dünaamiliselt sõlme töösoleku ajal lisada või eemaldada. Kontroll, milliseid pistikprogramme käivitada, toimub failis *driver.cpp* funktsioonis *initialize()*, kus võetakse pistikprogrammide kohta käivad parameetrid (algusega *plugin_*) ja käivitatakse vaid need, mille puhul on kahendväärtus tõene (joonis 7) [43]. Mootori ja odomeetria pistikprogrammide puhul tuleb arvestada, et kuigi on võimalik neid mitte käivitada, on need vajalikud tagamaks draiveri korrektset toimimist vastavalt töö nõuetele. [43]



Joonis 7. Pistikprogrammide käivitamise loogika

6.2 Lahenduse kirjeldus ja võrdlus ROS1 draiveriga

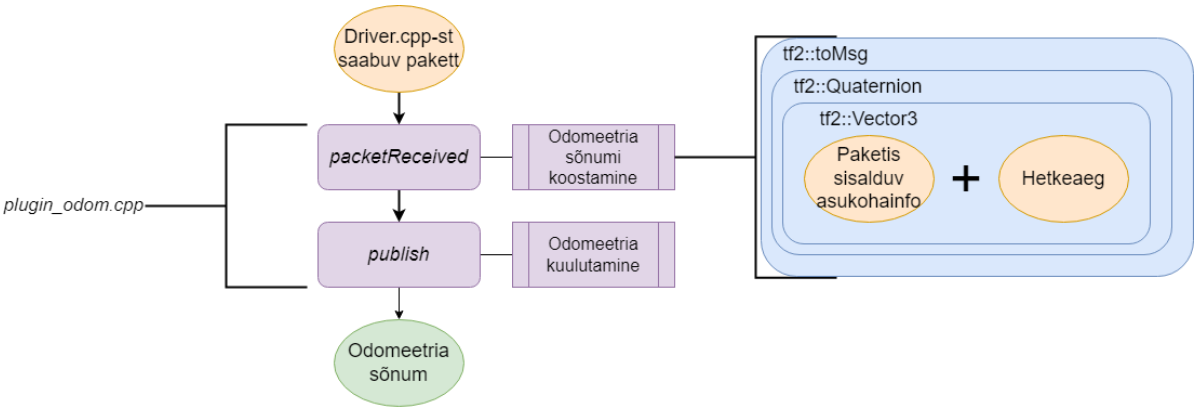
Draiver (*driver.cpp*) toob esile kahe draiveri kõige suuremad erinevused: pistikprogrammide laadimine ja süsteemi initsialiseerimine [43].

- Draiveri klass tuletatakse *Node* klassist. See peegeldab ROS2 sõlmepõhist disaini.
- Initsialiseerimine on eraldatud klassi konstruktorist ja võimaldab dünaamilist parameetrite põhist konfigureerimist.
- Kasutusel on parameetrid, mida kasutatakse otsustamaks, milliseid pistikprogramme laadida.
- Iga käivitatud pistikprogrammi uuendatakse andmepaketiga riistvara klassist.

Riistvara klass (*hardware.cpp)* kasutab jadaühenduse loomiseks ja üle selle suhtlemiseks *serial_driver* teeki, mis toetub *ASIO* ja *io_context* teekidele [44]. See võimaldab asünkroonset suhtlust erinevate riistvaraliste komponentidega, haldab portide avamist,

sulgemist ja konfigureerimist ning andmete lugemist ja kirjutamist. Draiveri kontekstis annab see lihtsa ja robustse meetodi riistvaraga suhtlemiseks ilma, et andmete saatmine või vastuvõtmine blokeeriks põhilõimes toimuvat. Juhul kui ühendus peaks hanguma või katkema, taasluuakse see, et roboti töö saaks jätkuda. Luuakse puhver, kuhu kogutakse jadaliidesest tulevaid sõnesid. Samuti edastab riistvara klass andmepakette mikrokontrollerile mootorite kiiruste seadmiseks. Informatsiooni liikumine visualiseeritud joonisel 6 riistvara komponendist väljuvate ja sinna sisenevate nooltega. Sobiva jadaühenduse teegi leidmine oli üks suurimaid väljakutseid draiveri kaasajastamisel. [43]

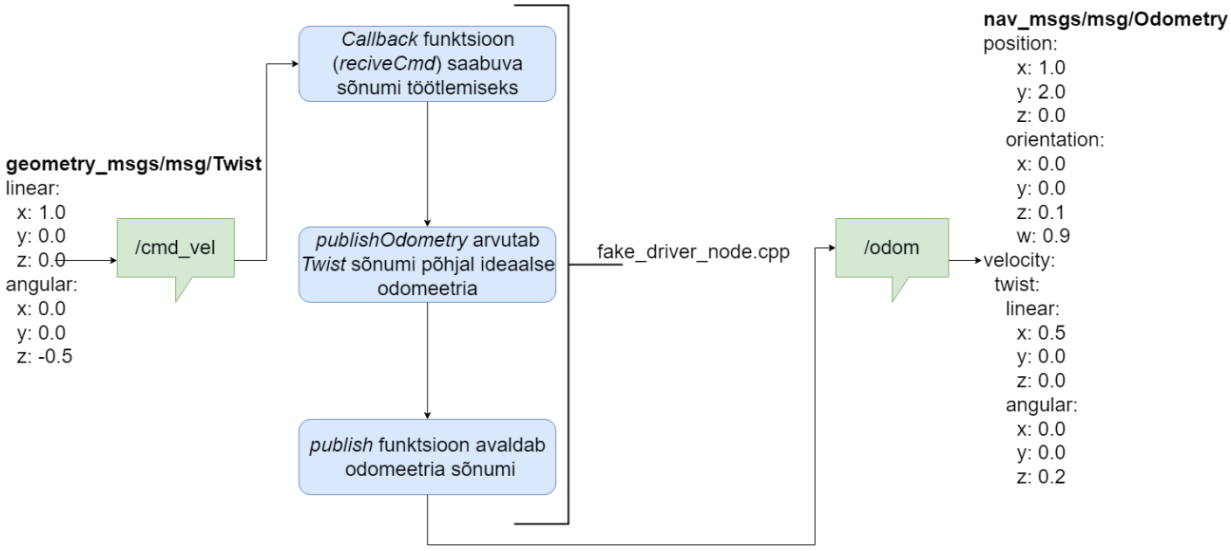
Odomeetria pistikprogramm (*plugin_odom.cpp*) loeb riistvara klassi poolt loodud andmepakettide puhvrit ja kalkuleeritakse riistvarast saabunud sõnede põhjal standardse ROS odomeetria sõnumi väärtused, millega väärtustatakse */odom* rubriiki saadetavad sõnumid. Kvaterniooni arvutamine ilmestab seda, kuidas ROS1 funktsioonid asendatakse ROS2-s uutega. ROS1 kasutab *tf* teeki ja funktsiooni *tf::createQuaternionMsgFromYaw*, et luua *geometry_msgs::Quaternion* objekt. ROS2 kasutab *tf2* teeki ja funktsioone *tf2::toMsg*, *tf2::Quaternion* ja *tf2::Vector3* et luua *geometry_msgs::msg::Quaternion* objekt. Andmete liikumist läbi draiveri komponentide ja funktsioonide näeb jooniselt 8. [43, 33]



Joonis 8. Odomeetria sõnumite koostamine ja avaldamine

Mootorite pistikprogramm (*plugin_motors.cpp*) loob rubriigi */cmd_vel* ja jääb ootama sinna saabuvaid käske. Rubriigis */cmd_vel* avaldatakse *geometry_msgs/msg/Twist* tüüpi sõnumeid. Uue sõnumi avaldumisel loeb *plugin_motors.cpp* sõnumi sisu, teisendab selle sõneks, loob paketi ja kutsub välja *hardware.cpp* funktsiooni selle paketi saatmiseks robotile. Rubriigist sõnumi lugemise ja mikrokontrollerile sõne edastamise töövoog on kujutatud joonisel 4 [33]. ROS2 versiooni jaoks kaasajastati faili ülesehitus, rubriikide tellimised ja konstruktorid. [43]

Matkedraiveri (*fake_driver_node.cpp*) puhul töödeldakse andmeid *fake_driver_node.cpp* failis asuvates funktsioonides, sest andmeid ei ole reaalselt vaja edastada ega lugeda. Sõnumeid avaldatakse */odom* rubriigis ja loetakse */cmd_vel* rubriigist identselt tavalise draiveri sõlmele. Sarnaselt ROS1 draiverile integreeritakse saabunud kiiruse sõnumeid ja väljastatakse nende põhjal ideaalne odomeetria, sest ei arvestata roboti massiga ega seetõttu ka kiirendusega. Andmete liikumist näeb jooniselt 9 [43]. ROS2 matkedraiver kasutab odomeetria pistikprogrammis kirjeldatud uusi *tf2* funktsioone. [43]



Joonis 9. Matkedraiveri rubriigid ja sõnumite liikumine

Konfiguratsioonifail (*robotont_params.yaml*) sisaldab muutujaid, mis draiveris on kasutusel käivituse ajal süsteemi konfigureerimiseks. Parameetritega seadistatakse jadaühenduse seaded (*device name, baud rate, flow control, parity, stop bits*) ja käivitatavad pistikprogrammid (*plugin_odom, plugin_motor, plugin_led_module,* jne). Konfiguratsioonifaili loetakse käivitusfaili käivitamise ajal. Konfiguratsioonifail kaasajastati vastavalt ROS2 nõuetele. [43]

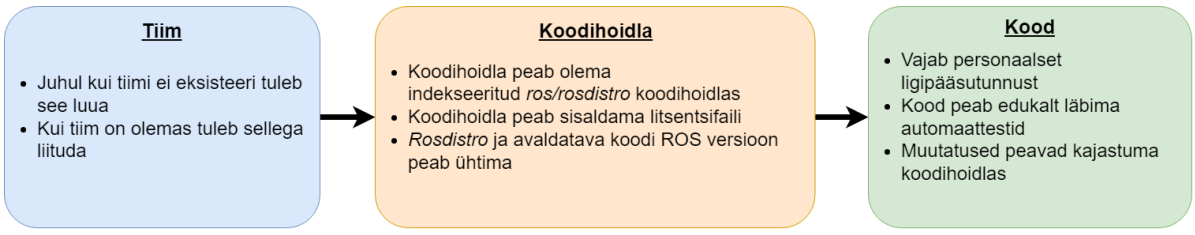
Käivitusfailid (*driver_basic.launch, face_driver.launch*) on loodud draiveri lihtsamaks käivitamiseks. Need käivitavad vastavalt draiveri või matkedraiveri. Käivitamisel on oluline koht parameetritel, mida on võimalik seadistada kolmes erinevas kohas: *launch* faili käivitamisel käsurealt käsu parameetrina, konfiguratsioonifailis või draiveri koodis. Matkedraiveril parameetrid puuduvad. Käivitusfaili on ROS2 versioonis kirjutatud *Python* keeles, ROS1 kasutas XML formaati. [33, 43]

Uue pistikprogrammi loomiseks tuleb kirjutada vastavad lähtekoodi ja päise failid, mis sisaldavad pistikprogrammi funktsionaalsust. Selleks, et pistikprogramm ROS2 draiverisse integreerida, on vaja *driver.cpp* failis initsialiseerimise funktsioonis pistikprogramm vastavalt parameetrile initsialiseerida. See tagab, et riistvaralt loetavad paketid edastatakse ka loodavale pistikprogrammile ja et pistikprogramm käivitatakse korrektselt. Draiveri arhitektuur ei piira pistikprogrammi funktsionaalsust. Töö esitamise hetkeks on Robotondi draiveri baasfunktsionaalsusele tuginedes juba lisatud teiste tudengite poolt erinevaid pistikprogramme näiteks akuinfo edastamiseks või valguslahenduse juhtimiseks. [45, 46]

6.3 Draiveri avaldamine

Selleks, et draiver oleks lihtsasti kättesaadav kõikidele kasutajatele, avaldati ROS2 pakk *buildfarm*is. Paki avaldamine teeb selle saadavaks pakihaldussüsteemides (näiteks *apt* Ubuntu operatsioonisüsteemis), võimaldab automaatselt genereerida rakendusliidese dokumentatsiooni ja indekseerib paki *ROS indexisse*, mis on veebipõhine platvorm, mis kuvab infot erinevate saada olevate ROS-i kimpude ja pakkide kohta. See protseduur on erinev esmakordsel ja iteratsioonid avaldamisel. [47]

Enne kui on võimalik pakki avaldada, peab olemas olema avaldamise meeskond (*release team*). Kuna tegu on esimese Robotondiga seonduva ROS2 pakiga, siis selline meeskond puudus ja esmalt oli vaja see luua (joonis 10) [47]. Meeskonna loomiseks on vaja täita taotlus, mis on *GitHub*is *ros2-gbp-github-org* koodihoidlas probleemi raporteerimise malli kujul. Malli täitmisel on vaja sisestada meeskonna nimi, koodihoidlad, kus avaldatav pakk asub, lähtekoodi hoidla ja meeskonna liikmete nimed. Peale malli täitmist vaatab administraator avalduse üle ja kinnitab selle või palub teha muudatusi.



Joonis 10. ROS2 paki avaldamise lihtsustatud sammud ja tingimused

Avaldatava paki koodihoidla (*release repository*) on hoidla, kus hoitakse faile, mida avaldamise protsessi käigus genereeritakse ja *ROS buildfarm* hiljem kasutab, ning

puhverdatakse avaldamise protseduuri konfiguratsioonid, et järgnevaid avaldamisi lihtsustada.

Töö käigus läbitud avaldamisprotsessi tulemusena on: Robotondi ROS2 draiver indekseeritud *rosdistro/humble/distributions.yaml* failis, millega indekseeritakse installeerimiseks saadaolevad pakid [52], Robotondi tiim registreeritud organisatsioonina *ros2-gbp-github-org* nimekirjas vastava avaldamise koodihoidlaga [53] ning *ROS 2 release repositories* organisatsiooni alla kuuluvas hoidlas on avaldatud draiveri failid [54].

Nende tulemuste põhjal on edaspidine pakkide avaldamine Robotondi tiimi alt lihtsustatud, sest organisatsioon on juba registreeritud ja on olemas näide, mille põhjal saab järgmiseid pakke avaldada. Ka draiveri uue iteratsiooni avaldamine ei vaja enam mitmesuguste protseduuride läbimist, piisab lähtekoodi muudatuste hoidlasse salvestamisest ja avaldamise protsessi käskude käivitamisest.

Järgnevalt on kirjeldatud töö käigus läbitud paki avaldamise protsessi. Peale seda kui meeskond ja koodihoidla on loodud, tuleb järgida esmakordse avaldamise juhendit ROS2 dokumentatsioonis:

1. Loodi personaalne ligipääsutunnus (*personal access token*) *GitHub*is ja lisati see *~/.config/bloom* konfiguratsioonifaili seadmes, kus lähtekood asub.
2. Veenduti, et seadmes asuv lähtekood on uuendatud ja vastab *GitHub*is olevale.
3. Loodi muudatuste logi käsuga *catkin_generate_changelog –all*. See loob faili *CHANGELOG.rst*, kuhu kirjeldatakse kõik muudatused, mis enne avaldamist tehtud on. Esmakordsel avaldamisel genereeritakse faili kõik *GitHub*i muudadustega tehtud kommentaarid. Seda faili saab tekstiredaktoriga muuta.
4. Muudeti paketi versiooni käsuga *catkin_prepare_release*. See suurendab versiooni numbrit *package.xml* failis ja kajastab need muudatused ka koodihoidlas.
5. Kasutati käsku *bloom-release –new-track –rosdistro <distributsiooni nimi> –track <distributsiooni nimi> <hoidla nimi>*, et alustada avaldamise protsessi. Peale käsu sisestamist küsitakse erinevat informatsiooni, et pakki õigesti avalikustada.
    a. Olemasoleva paki uue versiooni avaldamiseks tuleb kasutada käsku *bloom-release -rosdistro <distributsiooni nimi> <hoidla nimi>*
6. Kui kõik info on õigesti sisestatud ja veateateid ei tekkinud, loodi ka tõmbetaotlus (*pull request*), et tehtud muudatused *rosdistro* hoidlas avalikustuksid

Eeldusel, et paki ehitamine õnnestub ja tõmbetaotlus kinnitatakse, saab pakki testida *ros-testing* hoidlast. Selleks tuleb muuta pakihalduri installatsiooniallikaid ja meeles tuleb ka pidada, et need hiljem tagasi muuta [55]. Iga kahe kuni nelja nädala tagant sünkroniseeritakse *ros-testing* hoidla põhilisse ROS hoidlasse, kust saavad kasutajad pakke installeerida.

Paki installeerimiseks veenduti et ROS2 hoidlad on seadmesse konfigureeritud ja seejärel saab pakke installeerida näiteks käsuga *sudo apt install ros-<ROS versioon>-paki-nimi*. Robotont ROS2 draiveri puhul on käsuks *sudo apt install ros-humble-robotont-driver*. [56]

7. Kokkuvõte

Käesoleva töö eesmärgiks oli töötada välja Robotondi ROS2 draiver ning parendada selle käigus pistikprogrammide kasutamist, valideerida draiveri arhitektuuriline ülesehitus teiste haridusrobotplatvormide ROS2 draiverite põhjal ja avaldada draiver pakihaldusüsteemis.

Loodi ROS2 versiooni draiver Robotondile, mis sisaldab endas baasfunktsionaalsust roboti liigutamiseks ja odomeetria sõnumite kuulutamiseks. Draiver jälgib ROS2 ülesehitust ja kasutab uut funktsionaalsust, mis on saadaval vaid ROS2 versioonis.

Pistikprogrammide kasutamine on modulariseeritum ja nende käivitamine on sõltuv parameetrite seadmisest. Uute pistikprogrammide integreerimine draiverisse on loogiline ja seda ilmestavad ka uuele draiverile teiste kasutajate loodud pistikprogrammid.

Arhitektuuriline ülesehitus on valideeritud 4.3 Clearpath Robotics TurtleBot 4 ja Husarion Panther robootikaplatvormide põhjal. Arhitektuur vastab standardsele ülesehitusele.

Robotont ROS2 draiver on avaldatud pakihaldussüsteemides, seda on võimalik lihtsasti paigaldada ja uute iteratsioonide avaldamine on lihtsustatud. Avaldamise protseduuri läbimine lihtsustab tulevikus ka teiste Robotondiga seotud pakkide avaldamist.

Töö käigus loodud ROS2 draiver kaasajastas Robotondi platvormi teadus- ja õppetöövahendina ning suurendas töökindlust, sest võimaldab platvormi kasutamist ka peale ROS1 ametliku toe lõppu aastal 2025.

Viited

[1] „ Heilo Altin, haridusrobootika juhtivspetsialist“. Vaadatud: 19. mai 2024. [Võrgumaterjal]. Saadaval: https://tuit.ut.ee/et/sisu/heilo-altin-haridusrobootika-juhtivspetsialist

[2] „robotont“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: http://robotont.ut.ee/avaleht

[3] „ DSpace Repository :: Search “. Vaadatud: 19. mai 2024. [Võrgumaterjal]. Saadaval: https://dspace.ut.ee/search?query=robotont

[4] Y. Pyo, H. Cho, R. J. Jung, ja T. Lim, *ROS Robot Programming*. ROBOTIS Co.,Ltd., 2017.

[5] „Wizards of ROS: Willow Garage and the Making of the Robot Operating System“, IEEE Spectrum. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://spectrum.ieee.org/wizards-of-ros-willow-garage-and-the-making-of-the-robot-operating-system

[6] „The Origin Story of ROS, the Linux of Robotics“, IEEE Spectrum. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://spectrum.ieee.org/the-origin-story-of-ros-the-linux-of-robotics

[7] „Robot Operating System / Code Commit Log“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://sourceforge.net/p/ros/code/10/log/?path=

[8] „Repositories - ROS Wiki“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://web.archive.org/web/20140205072158/http://wiki.ros.org/Repositories

[9] „ROS Tutorials and Turtles - ROS robotics news“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://www.ros.org/news/2009/12/ros-tutorials-and-turtles.html

[10] „ROS 1.0 - ROS robotics news“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://www.ros.org/news/2010/01/ros-10.html

[11] „Interns and Visiting Scholars | Willow Garage“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://web.archive.org/web/20180622215146/http://www.willowgarage.com/pages/community/interns-and-visiting-scholars

[12] „The Results Are In: PR2 Beta Program Recipients! | Willow Garage“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://web.archive.org/web/20200128060749/http://www.willowgarage.com:80/blog/2010/05/04/pr2-beta-program-recipients

[13] „Announcing ROS Answers - ROS robotics news“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://www.ros.org/news/2011/02/announcing-ros-answers.html

[14] „ROS on the Move: TurtleBots available for preorder | Willow Garage“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://web.archive.org/web/20210423184600/http://www.willowgarage.com/blog/2011/04/18/turtlebots-available-preorder

[15] „100 Repositories - ROS robotics news“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://www.ros.org/news/2011/05/100-repositories.html

[16] „Open Source Robotics Foundation | Willow Garage“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://web.archive.org/web/20180616215943/http://www.willowgarage.com/blog/2012/04/16/open-source-robotics-foundation

[17] „Distributions - ROS Wiki“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: http://wiki.ros.org/Distributions

[18] „Why ROS 2?“ Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: http://design.ros2.org/articles/why_ros2.html

[19] „ROS 2 Documentation — ROS 2 Documentation: Rolling documentation“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://docs.ros.org/en/rolling/

[20] „Distributions — ROS 2 Documentation: Rolling documentation“. Vaadatud: 25. oktoober 2023. [Võrgumaterjal]. Saadaval: https://docs.ros.org/en/rolling/Releases.html

[21] „Understanding topics — ROS 2 Documentation: Rolling documentation“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.ros.org/en/rolling/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html

[22] „roscore - ROS Wiki“. Vaadatud: 23. aprill 2024. [Võrgumaterjal]. Saadaval: https://wiki.ros.org/roscore

[23] „ROS 1 vs ROS 2 What are the Biggest Differences?“ Vaadatud: 17. märts 2024. [Võrgumaterjal]. Saadaval: https://www.model-prime.com/blog/ros-1-vs-ros-2-what-are-the-biggest-differences

[24] „ROS/Tutorials/UnderstandingNodes - ROS Wiki“. Vaadatud: 25. oktoober 2021. [Võrgumaterjal]. Saadaval: http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

[25] „ROS/Tutorials/UnderstandingTopics - ROS Wiki“. Vaadatud: 25. oktoober 2021. [Võrgumaterjal]. Saadaval: http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

[26] „ROS/Tutorials/UnderstandingServicesParams - ROS Wiki“. Vaadatud: 25. oktoober 2021. [Võrgumaterjal]. Saadaval: http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams

[27] „ROS/Concepts - ROS Wiki“. Vaadatud: 15. mai 2024. [Võrgumaterjal]. Saadaval: http://wiki.ros.org/ROS/Concepts

[28] ed, „ROS1 vs ROS2, Practical Overview For ROS Developers“, The Robotics Back-End. Vaadatud: 17. märts 2024. [Võrgumaterjal]. Saadaval: https://roboticsbackend.com/ros1-vs-ros2-practical-overview/

[29] „Changes between ROS 1 and ROS 2“. Vaadatud: 17. märts 2024. [Võrgumaterjal]. Saadaval: https://design.ros2.org/articles/changes.html

[30] „multimaster_fkie - ROS Wiki“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://wiki.ros.org/multimaster_fkie

[31] „robotont/robotont-mechanics“. robotont, 20. märts 2024. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/robotont/robotont-mechanics

[32] „robotont/robotont-electronics-mainboard“. robotont, 23. veebruar 2024. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/robotont/robotont-electronics-mainboard

[33] „robotont/robotont_driver“. robotont, 12. november 2021. Vaadatud: 22. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/robotont/robotont_driver

[34] „EU project: OFERA - micro-ROS“. Vaadatud: 24. aprill 2024. [Võrgumaterjal]. Saadaval: https://www.eprosima.com/index.php/products-all/r-d-projects/eu-project-ofera-micro-ros

[35] „Features and Architecture“, micro-ROS. Vaadatud: 24. aprill 2024. [Võrgumaterjal]. Saadaval: https://micro.ros.org/docs/overview/features/

[36] „About“, Husarion Store. Vaadatud: 30. aprill 2024. [Võrgumaterjal]. Saadaval: https://store.husarion.com/pages/about-us

[37] „Robots“, Husarion Store. Vaadatud: 30. aprill 2024. [Võrgumaterjal]. Saadaval: https://store.husarion.com/collections/robots

[38] „husarion/rosbot_xl_ros: ROS 2 packages for ROSbot XL“. Vaadatud: 30. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/husarion/rosbot_xl_ros/tree/master

[39] „husarion/rosbot_ros: ROS packages for ROSbot 2, 2R and 2 PRO“. Vaadatud: 30. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/husarion/rosbot_ros

[40] „panther_ros/panther_hardware_interfaces/src at release-2.0.1 · husarion/panther_ros“. Vaadatud: 30. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/husarion/panther_ros/tree/release-2.0.1/panther_hardware_interfaces/src

[41] turtlebot, „Overview · User Manual“. Vaadatud: 1. mai 2024. [Võrgumaterjal]. Saadaval: https://github.com/turtlebot4-user-manual/software/overview.html

[42] „turtlebot4/turtlebot4_node at humble · turtlebot/turtlebot4“, GitHub. Vaadatud: 1. mai 2024. [Võrgumaterjal]. Saadaval: https://github.com/turtlebot/turtlebot4/tree/humble/turtlebot4_node

[43] „ut-ims-robotics/paap-thesis-2024-robotont-driver “. Vaadatud: 22. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/ut-ims-robotics/paap-thesis-2024-robotont-driver

[44] „GitHub - ros-drivers/transport_drivers: A set of ROS2 drivers for transport-layer protocols.“ Vaadatud: 15. aprill 2024. [Online]. Available at: https://github.com/ros-drivers/transport_drivers

[45] „robotont_driver/src/plugin_bat_state.cpp at humble-devel-robert · robotont/robotont_driver“. Vaadatud: 1. mai 2024. [Võrgumaterjal]. Saadaval: https://github.com/robotont/robotont_driver/blob/humble-devel-robert/src/plugin_bat_state.cpp

[46] „robotont_driver/src/plugin_led_module.cpp at humble-devel- · robotont/robotont_driver“. Vaadatud: 1. mai 2024. [Võrgumaterjal]. Saadaval: https://github.com/robotont/robotont_driver/blob/humble-devel-raimo/src/plugin_led_module.cpp

[47] ROS Index“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://index.ros.org/

[48] „Releasing a Package — ROS 2 Documentation: Rolling documentation“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.ros.org/en/rolling/How-To-Guides/Releasing/Releasing-a-Package.html

[49] „Release Team / Repository — ROS 2 Documentation: Rolling documentation“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.ros.org/en/rolling/How-To-Guides/Releasing/Release-Team-Repository.html#join-a-release-team

[50] „First Time Release — ROS 2 Documentation: Rolling documentation“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.ros.org/en/rolling/How-To-Guides/Releasing/First-Time-Release.html

[51] „New Issue · ros2-gbp/ros2-gbp-github-org“, GitHub. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://github.com/ros2-gbp/ros2-gbp-github-org

[52] „ ros/rosdistro/humble/distribution.yaml “. Vaadatud: 19. mai 2024. [Võrgumaterjal]. Saadaval: https://github.com/ros/rosdistro/blob/master/humble/distribution.yaml

[53] „ ros2-gpp/ros2-gbp-github-org/robotont.tf “. Vaadatud: 19. mai 2024. [Võrgumaterjal]. Saadaval: https://github.com/ros2-gbp/ros2-gbp-github-org/blob/latest/robotont.tf

[54] „ ros2-gpp/robotont_driver-release “. Vaadatud: 19. mai 2024. [Võrgumaterjal]. Saadaval: https://github.com/ros2-gbp/robotont_driver-release

[55] „Testing with pre-release binaries — ROS 2 Documentation: Rolling documentation“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.ros.org/en/rolling/Installation/Testing.html

[56] „Installing Packages — Industrial Training documentation“. Vaadatud: 15. aprill 2024. [Võrgumaterjal]. Saadaval: https://industrial-training-dev.readthedocs.io/en/latest/_source/session1/ros2/2-Installing-Existing-Packages.html

Lisad

Lisa 1

Töö käigus välja töötatud draiver asub aadressil:
https://github.com/ut-ims-robotics/paap-thesis-2024-robotont-driver/tree/main

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, _____Sven-Ervin Paap_____,

(autori nimi)

annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

_____ROS2 DRAIVER ÕPPEROBOTILE ROBOTONT_____,

(lõputöö pealkiri)

mille juhendaja on _____Veiko Vunder_____,

(juhendaja nimi)

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

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 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

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

Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Sven-Ervin Paap

21.05.2024