Tartu Ülikool
loodus- ja täppisteaduste valdkond
tehnoloogiainstituut

Martin Maidla

Avatud robotiarendusplatvormi Robotont omniliikumise ja odomeetria arendamine

Bakalaurseusetöö (12 EAP)
Arvutitehnika eriala

Juhendaja:
Robootika dotsent Karl Kruusamäe

Tartu 2018

Resümee/Abstract

Avatud robotiarendusplatvormi Robotont omniliikumise ja odomeetria arendamine

2017. aastast on Tartu Ülikooli tehnoloogiainstituudis arendusel avatud robotiplatvorm nimega Robotont. Robotondi eesmärgiks on leida kasutust nii teadus- kui ka haridusvaldkonnas. 2017. aastal valmis Robotondile tarkvaraline lahendus, mis andis platvormile ühilduvuse ROSi raamistikuga ja liikumis- ja juhtimisvõimekuse.

Käesoleva bakalaureusetöö eesmärk on parandada Robotont platvormi põhivara, lisada Robotont draiverile odomeetria arvutamine ja universaalne omniliikumine.

CERCS: T125 Automatiseerimine, robootika, control engineering

Märksõnad: robootika, ROS, Robotont, draiver, odomeetria, omniliikumine

Omnimotion and odometry development for open robot development platform Robotont

Since 2017, an open robot platform named Robotont has been in development in the Institue of Technology of University of Tartu. The purpose of Robotont is to find use in fields such as scientific research and as well as education. In the year 2017, a software solution was developed for Robotont, which provided the platform with compatibility for ROS framework and added movement and control functionality.

The purpose of this thesis is to fix the firmware of Robotont platform and add odometry calculation and universaal omnimotion to the current Robotont driver.

CERCS: T125 Automation, robotics, control engineering

Keywords: robotics, ROS, Robotont, driver, odometry, omnimotion

Sisukord

Mõisted ja lühendid

omniliikumine – *omnimotion* – liikumine, mis võimaldab robotil liikuda tasapinnal mistahes suunas, mistahes rotatsiooniga;

odomeetria – *odometry* – mootorite tagasiside põhjal arvutatud roboti asukoht ja asend;

ROS – *Robot Operating System* – raamistik, mis hõlbustab robotite tarkvara arendamist.

1. Sissejuhatus

Tartu Ülikool tehnoloogiainstituudis on 2017. aastast arendusel avatud robotiplatvorm nimega Robotont. Robotondi eesmärk on leida kasutuse teadus- ja haridusvaldkonnas. Robotondi komponentide valik ja ehitus on valitud sellised, et see oleks nii keerukas, et seda oleks võimalik kasutada teadusvaldkonnas, kuid piisavalt lihtne, et seda oleks võimalik kasutada ka hariduslikel eesmärkidel – robootika ja teaduse populariseerimine ja robootika õpetamine alates gümnaasiumi astmest, olles järguks levinud LEGO Mindstorms EV3 platvormile.

2017. aastal lisati Robotondile ROSi tugi ning arendati välja draiver, mis lisas juhtimis- ja liikumisfunktsionaalsuse, kuid millel ei olnud universaalset omniliikumist ega odomeetriat.

Käesolev töö on järg eelnevale tehtud tööle, mille põhjal saab sõnastada käesoleva töö eesmärgid – lisada olemasolevale Robotont draiverile odomeetria ja universaalse omniliikumise võimekus.

Käesoleva lõputöö esimeses pooles (pt 2) antakse ülevaade robotplatvormi Robotont üldstruktuurist, elektroonikast ja tarkvarast. Antakse ka ülevaade eelnevast tehtud tööst ning sõnastatakse käesoleva lõputöö eesmärgid. Töö teises pooles (pt 3 ja 4) kirjeldatakse töö käigus draiverile lisatud funktsionaalsusi ning nende testimismeetodeid. Bakalaureusetöö lõpus (pt 5) võetakse kokku peamised saavutused ning üles kerkinud probleemid nende võimalike lahendustega.

2. Robotplatvorm Robotont

2.1. Sissejuhatus

Robotont (Joonis 1) on Tartu Ülikooli tehnoloogiainstituudis 2017. aastast arendamisel olev avatud robotiarendusplatvorm ROSi võimekusega. ROS (*Robot Operating* System) on robotiarendustarkvara, mis võimaldab luua modulaarseid süsteeme.

Robotondi riistvara on valitud selline, et sellega oleks võimalik sooritada mitmeid erinevaid tegevusi. Kuna Robotont kasutab omnirattaid, on võimalik Robotondiga teostada omniliikumist. Mootorite küljes on koodrid, mis võimaldavad arvutada odomeetriat ehk roboti asukohta lähtuvalt mootoritelt saadud infole. Robotondi küljes on RGB-D sügavuskaamera, mille abil on võimalik sooritada ruumi kaardistamist. Kaardistamine ja odomeetria on üks osa SLAMist (ingl k s*imultaneous localization and mapping*), mis on tegevus, kus robot kaardistab tundmatut keskkonda, ise samal ajal sooritades rajaplaneerimist selles keskkonnas [1].

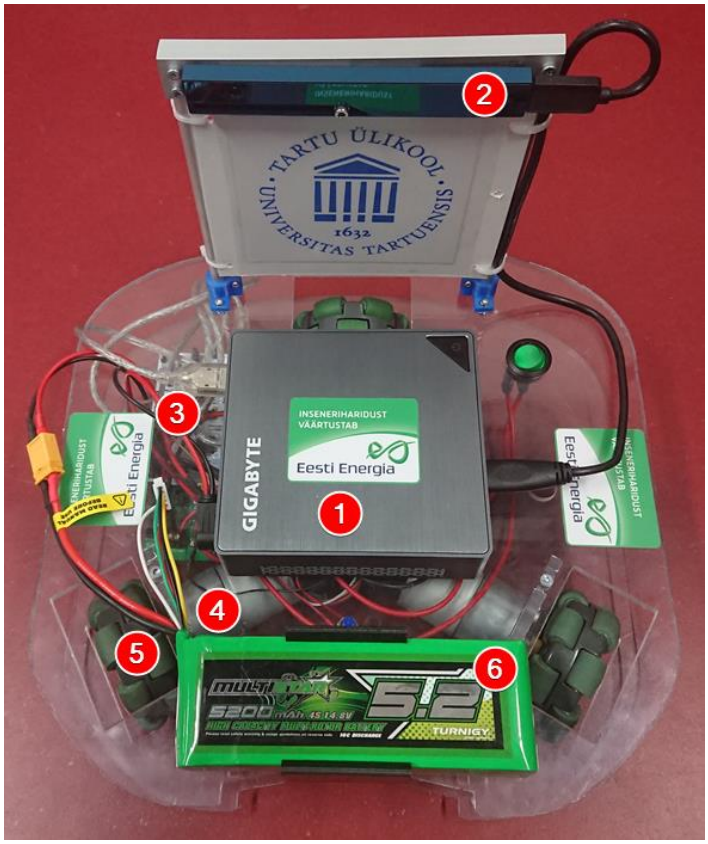
Robotont platvormi on sobilik kasutada teaduslikes uuringutes, kus tegeletakse nt rajaplaneerimise, kaardistamise ja parverobootikaga. Robotont platvormil on potentsiaali leida kasutust ka haridusvaldkonnas: huviharidus, tehnoloogiaõpe ning inseneriõppe populariseerimine.[2]

Robotondi aluseks on Tartu Ülikooli robotiklubis arendatud mobiilne robot, mida kasutati rahvusvahelisel Robotexi jalgpallivõistlusel. [2, 3]

2.2. Robotondi üldstruktuur

Joonis 1 on kujutatud Robotont ja selle peamised komponendid. Robotondi kere koosneb kahest polükarbonaadist valmistatud alusplaadist, millest üks moodustab roboti alumise kihi ja teine pealmise kihi. Alusplaadid on omavahel ühendatud risti asetsevate plaatidega, mille külge kinnitatakse ka mootorid. Robotondi liikumisvõime annavad 3 mootorit (Pololu 19:1 alalisvoolu mootorid [4]), mille külge on kinnitatud VEX Robotics 2,75″ topeltrullikutega omnirattad [5]. Alumise kihi külge on puksidega ühendatud nii juhtkontroller kui ka iga mootori kontroller (kokku 4 kontrollerplaati).

Robotondi pealmise kihi külge on kinnitatud juhtarvuti, RGB-D kaamera ja akuhoidja.



Joonis 1. Robotont ja selle põhikomponendid. 1 – juhtarvuti; 2 – RealSense™ sügavuskaamera; 3 – mbed juhtkontroller; 4 – alalisvoolumootor; 5 – omniratas; 6 – aku.

2.3. Robotondi elektroonika

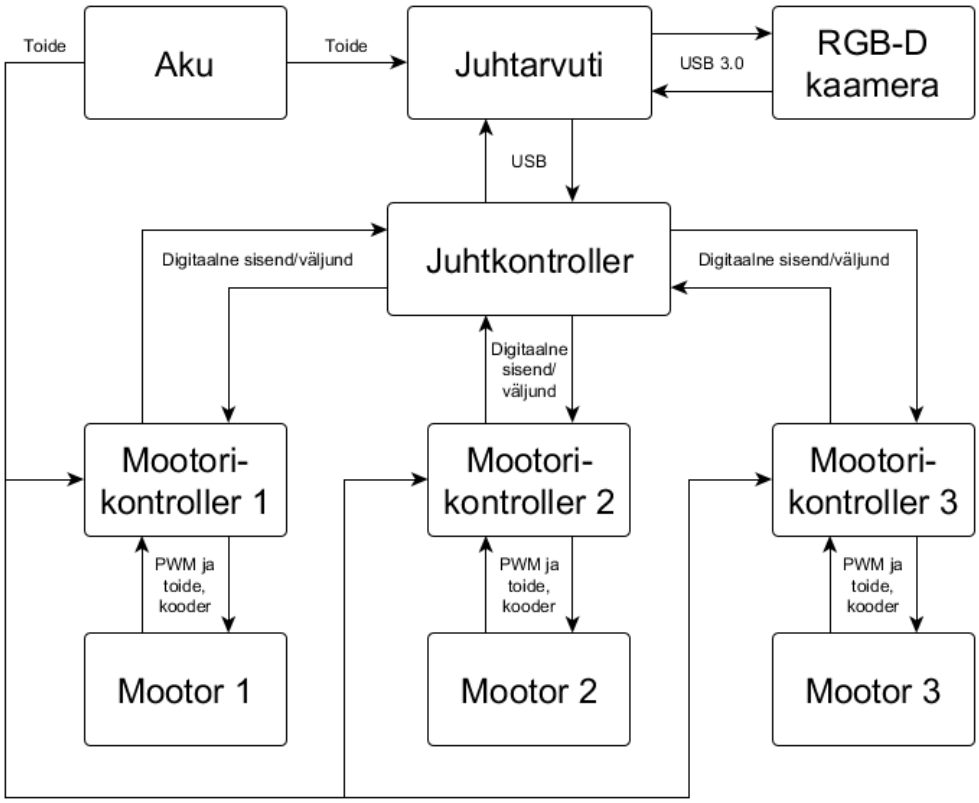
Robotondi elektroonikasüsteemi ülesandeks on luua vajalikud ühendused kõikide elektroonikakomponentide vahel ning see koosneb järgnevatest komponentides: juhtarvuti, juhtkontroller, 3 mootorikontrollerit, sügavuskaamera ning aku. Kogu elektroonikasüsteemi funktsionaalne ülevaade on toodud Joonis 2.

Robotondi juhtarvuti (Gigabyte BRIX GB-BSi5-6200 [6]) ülesanne on edastada juhtkontrollerile mootorite kiiruseid ja võtta vastu juhtkontrollerilt mootorite koodrite väärtusi.. Juhtarvuti eelis on selle väike suurus ja mitmete levinud ühendusliideste olemasolu, andes võimaluse juhtarvutiga ühendada mitmeid lisatarvikuid. Juhtarvutiga on üle USB liidese ühendatud juhtkontroller ja üle USB 3.0 liidese Intel® RealSense™ Camera R200

sügavuskaamera [7]. Sügavuskaamera võimaldab kuvada RGB pilti ja sellele vastavat sügavuspilti ehk 3D punktipilve.

Juhtkontrollerina on kasutusel STMicroelectronics NUCLEO-L476RG arendusplaat [8]. Juhtkontrolleriga on ühendatud kolm mootorikontrollerit [9]. Ühendusi mootorikontrollerite ja juhtkontrolleri vahel loovad ribakaablid. Mootorikontrollerid on ühendatud mootoritega, ning nende ülesanne on edastada juhtkontrollerisse saadetud kiirused mootoritele, edastada mootorite koodritelt saadud väärtused juhtkontrollerile ning juhtida voolu akust mootoritele.

Kogu elektroonika saab toidet ühelt 4-elemendiliselt liitiumpolümeer (LiPo) akult.



Joonis 2. Robotondi elektroonikat ja selle ühendusi kirjeldav skeem

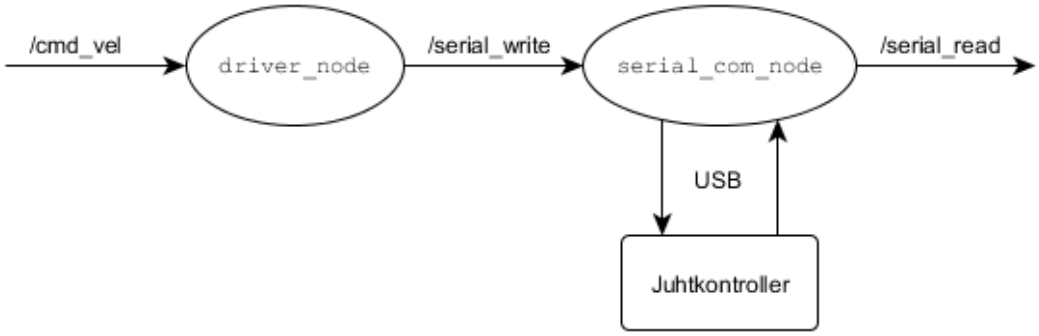
2.4. Robotondi tarkvara

Robotondi juhtarvuti peal käitatakse operatsioonisüsteemi Ubuntu 16.04 ja robootika arendusplatvormi ROS Kinetic Kame. ROSi tööpõhimõte on jooksutada programme, ehk sõlmesid (ingl k *node*) [10], mis suhtlevad omavahel standardiseeritud sõnumitega (ingl k *message*) [11]. Sõlmedel on võimalik avaldada sõnumeid kindlatel teemadel (ingl k *topic*) [12], kust on võimalik teistel sõlmedel lugeda nendel teemadel avaldatud sõnumeid.

Juhtkontrolleri peal käitatakse mbed operatsioonisüsteemil põhivara, mille ülesandeks on vastu võtta üle jadaliidese edastavaid käske mootorite kiirustega, saata mootorikontrolleritele mootorite kiirused ja tagastada juhtarvutile mootorikontrolleritelt saadud enkoodrite väärtused.

2.5. Eelnev töö

2017. aastal valmis Raid Vellerind bakalaureusetöö käigus Linuxi draiver Robotont platvormi jaoks, mis ühildub ROSi ökosüsteemiga. Loodud draiveri eesmärgiks oli anda Robotont platvormile juhtimis- ja liikumisvõimekus, mis ühildub ROS Navigation teegi nõuetega. Töö jooksul valmisid kaks ROSi sõlme: `driver_node` ja `serial_com_node` (Joonis 3).



Joonis 3. Raid Vellerind bakalaurseusetöö käigus valminud draiveri sõlmede ülevaade.

Sõlme `driver_node` ülesandeks jälgida sõnumeid teemal /cmd_vel (tegu on ROS Navigation teegi põhinõudega [13]), teisendada sealt saadud kiirused juhtkontrollerile sobivaks ning edastada teisendatud kiirused teemal /serial_write.

Sõnumitüüp, mida edastatakse teemal /cmd_vel on geometry_msgs/Twist, mis sisaldab välju roboti joon- ja nurkkiiruse kirjeldamiseks.

Sõlme `serial_com_node` ülesandeks on jälgida sõnumeid teemal /serial_write, saata mootorite kiiruseandmed läbi jadaühenduse juhtkontrollerile ja võtta juhtkontrollerilt vastu mootorite tagasiside, mida kuulutatakse teemal /serial_read.

Sõnumid teemadel /serial_write ja /serial_read on std_msgs/String-tüüpi.

Raid Vellerindi bakalaureusetöö tulemusi demonstreeriti roboti juhtimisel nii klaviatuuri kui ka Android nutitelefoni abil.

Loodud lahenduses esines siiski ka mõningaid puudusi:

1. Problemaatiliseks osutus juhtarvuti ja -kontrolleri vahelise jadaühenduse stabiilsus, mis avaldus aegajalt roboti suuna ja kiiruse muutusena. Pakuti välja hüpotees, et ebastabiilsuse põhjuseks on riistvara häired.
2. Draiveris teostatud liikumine on püsiprogrammeeritud, mis võimaldab valminud draiverit kasutada Robotondile sarnaste robotitega (ainult kolm omniratast).
3. Draiveris ei implementeeritud odomeetria abil roboti positsioneerimist.

2.6. Käesoleva töö eesmärk

Käesoleva töö eesmärgiks on analüüsida ja parandada juhtkontrolleri jadaühendust, luua Robotont platvormi juhtimisdraiverile universaalne omniliikumine, mida oleks võimalik kasutada mistahes rataste arvu ja asendiga, ning teostada mootorite koodrite põhjal odomeetriat.

3. Robotondi ROSi draiver

3.1. Draiveri edasiarendus

3.1.1. Probleem olemasoleva juhtkontrolleri põhivaraga

Robotondi üheks võtmetähtsusega komponendiks on mbed juhtkontroller, mille ülesandeks on võtta vastu juhtarvuti poolt edastatud kiirused, teisendada need mootorikiirusteks (impulsilaiusmodulatsiooni täitetegur) ning tagastada juhtarvutile pidevalt mootorite koodrite väärtused, mida saab ROSi draiver kasutada roboti asukoha hindamiseks odomeetria abil.

Töö alguses esines mbed juhtkontrolleril mitmeid seletamata tõrkeid. Kontroller sai kätte kiirused, tagastas kiirused ning suutis mootoreid juhtida, kuid enamvähem iga 20 sekundi tagant esines tõrge, mis väljendus roboti ootamatus liikumises. Tõrge kestis 5 sekundit, ning seejärel käitus robot nii, nagu oodatud, kuni uuesti möödus 20 sekundit, mille järel kordus protsess uuesti.

Vea võimalike põhjustena sõnastati järgnevad hüpoteesid:

1. Puudused PID-kontrolleri implementatsioonis
2. Jadaühenduse seadistamisviga
3. Tarkvara algoritmika/arhitektuuri viga juhtkontrolleri põhivaras
4. Tootjapoolne viga

Toodud järjekorras alustati ka vea põhjuse uurimist. PID algoritmi ülesanne juhtkontrolleris on piirata pinget, mis mootorite liigutamiseks neile rakendatakse, et robot kiirendaks sujuvamalt ja teeks vähem äkilisemaid liigutusi. Et tuvastada, kas viga PID-kontrolleri teostuses võib olla kirjeldatud tõrgete põhjustajaks, eemaldati põhivarast PID-kontrolleri implementatsioon. Selgus, et tõrgete põhjus ei olnud PID algoritmis.

Järgmiseks uuriti jadaühendust juhtarvuti ja kontrolleri vahel. Testimise käigus muudeti boodikiirust ja sagedust, millega ROS saadab sõnumeid üle jadaühenduse, ning vastavalt ka põhivaras. Ka nende muutmine ei andnud tulemusi.

Juhtkontrolleri põhivara sai täiendatud - asendati algselt kasutatud BufferedSerial [14] jadaühenduse teek mbed'i enda siseeehitatud jadaühenduse teegiga [15], asendati isetehtud andmete sisselugemis-funktsioon mbed'i enda jadaühenduse teeki sisseehitatud sisselugemis-funktsiooniga ning lisati kontrolle, kas jadaühenduse puhvris on andmeid, mida saab sisse lugeda. Ka need tegevused ei toonud lahendust.

Katsetuste käigus selgus, et andmed, mida juhtkontrollerile arvutist saadeti „nihkusid", st ühe andmehulga lõpust kadus sümbol ära, mis ilmus seejärel uue andmehulga algusesse. Protsess kestis nii kaua, kuni ühe sõnumi jagu sümboleid oli nihkunud. See andmete nihkumine põhjustas roboti ootamatud liikumised.

Andmete nihkumine esines nii mbed'i sisseehitatud jadaühenduse teeki kui ka BufferedSerial teeki kasutades. Mbed kommuuni foorumites on arutelu [16] MODSERIAL [17] jadaühendusteegi käitumise üle. MODSERIAL teek sisaldab endas funktsioone „getc" [18] ja „putc" [19], mille funktsioonid on vastavalt sisse lugeda ja välja saata üks sümbol (ingl k *char*). Mõlemad funktsioonid muudavad pointeri väärtuseid („buffer_count[RxIrq]" lugemise puhul, „bufferCount[TxIrq]" saatmise puhul), mis viitavad arvudele, mis näitavad kui palju sümboleid on veel sisend- või väljundpuhvris. Näiteks, kui kasutada funktsiooni „getc", loetakse sisse

sümbol, mille järel suurendatakse pointeri „buffer_count[RxIrq]“ väärtust, mille järel tagastatakse sümbol. Kuna pointeri muutmine mõlemas funktsioonis ei ole atomaarne operatsioon (ingl k *atomic operation*), võib pointeri muutmise hetkel toimuda katkestus, mille käigus võidakse selle pointeri väärtust muuta. Lahendus sellele probleemile oli pointeri muutmise ajaks peatada katkestuste toimumine.

Põhivaras sai jadaühenduse teegiks MODSERIAL, millel on eelnev mainitud puudus eemaldatud. Tehtud asendus eemaldas algselt esinenud tõrke – robot liigun nii, nagu oodatud, ilma probleemideta.

Jadaühenduse teegi asendamisega kerkis aga esile uus probleem – juhtkontrolleri poolt tagasi saadetud mootorite koodrite väärtused ei jõua terviklikuna arvutisse. Katse käigus selgus, et iga kindla arvu saadetud sõnumite järel andmed „nihkusid“. „Nihe“, mis esines algselt andmete lugemisel arvutist juhtkontrollerisse, esineb nüüd vastupidises olukorras.

3.1.2. Parameetriserver

ROSi parameetriserver on sõnastik, mille ülesanne on pakkuda võimalust sõlmedel lugeda ja kirjutada andmeid käitusfaasis. Kuna parameetriserver ei ole loodud suurt jõudlust nõudvate arvutuste jaoks, on sobilik hoida seal püsivaid, mitte-binaarseid, andmeid. Parameetriserver suudab hoida endas XML-RPC poolt toetatud andmetüüpe.[20]

Parameetriserver võimaldab muuta roboti programme modulaarsemaks – sõlmed saavad andmete väärtused samast kohast. Töö käigus lisati Robotondi draiverile käsud, talletada konfiguratsioonifailist „wheel_params.yaml“ (Joonis 4) (Tabel 1) andmed parameetriserverisse, kust on võimalik sõlmedel lugeda andmeid omniliikumise ja odomeetria arvutamiseks.

```yaml
wheelAmount: 3
wheelRads: [5.235988, 3.141593, 1.047198]
wheelDistanceFromCenter: 0.125
wheelRadius: 0.035
gearboxReductionRatio: 18.75
encoderEdgesPerMotorRevolution: 64
pidControlFrequency: 50
```

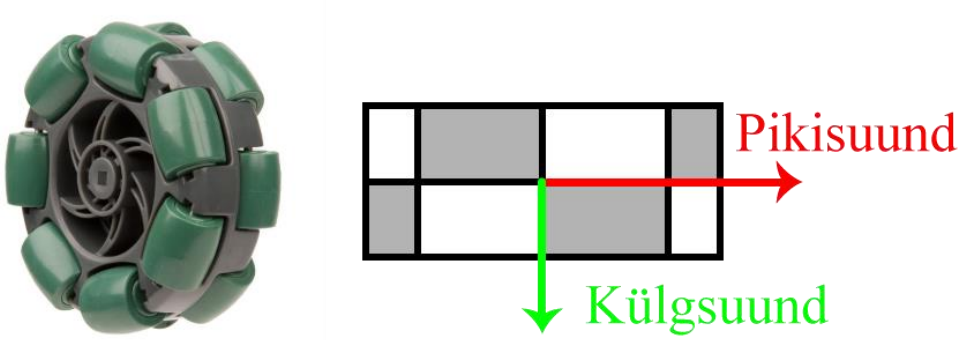
Joonis 4. Robotont „wheel_params.yaml" faili sisu.

Parameetri võti	**Andmetüüp**	**Kirjeldus**
wheelAmount	Täisarv	Omnirataste arv
wheelRads	Kahend-ujukomaarvud (ingl k double), loendis	Nurgad rataste positiivsete suundade ja võrdlussuuna vahel
wheelDistanceFromCenter	Kahend-ujukomaarv	Rataste kaugused roboti keskkohast
wheelRadius	Kahend-ujukomaarv	Ratta raadius
gearboxReductionRatio	Kahend-ujukomaarv	Mootori ülekandetegur
encoderEdgesPer-MotorRevolution	Täisarv	Mootori koodri sammude arv ühes täispöördes
pidControlFrequency	Täisarv	Sagedus, millega PID-kontroller arvutusti teostab

Tabel 1. Faili „wheel_params.yaml" faili sees hoitavate parameetrite võtmed ja neile vastavad andmetüübid ja kirjeldused

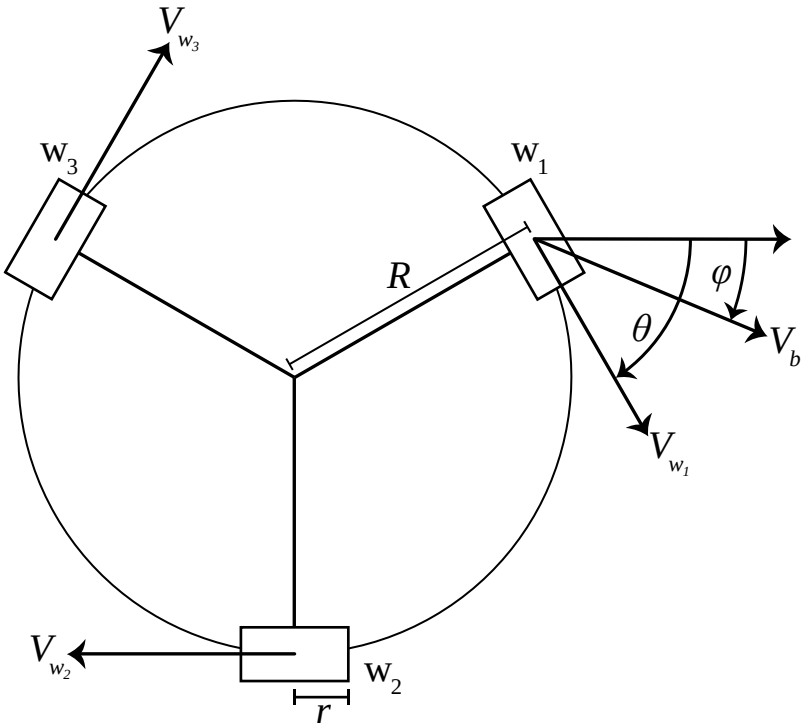
3.1.3. Omniliikumine

Omniliikumine võimaldab robotil liikuda tasapinnal mistahes suunas, mistahes rotatsiooniga. Omniliikumise teostamiseks robotil on eelduseks omniratаste olemasolu. Omniratastel on rullikud, mis paiknevad ratta välisäärel, mille tõttu käitub omniratas pikisuunas liikumisel nagu tavaline ratas, kuid külgsuunas liikumisel võimaldavad rullikud rattal passiivselt külgsuunas kaasa veereda (joonis 3).



Joonis 5. Omniratas [5]*, mis on kasutusel Robotondis, ja omniratta liikumissuunad*

Draiveri sõlme `driver_node` täiustati, mille käigus lisati sõlmele omniliikumise teostamiseks vajalikud arvutused. Omniliikumise implementeerimiseks kasutati Reiko Randoja poolt loodud juhendit [21].



Joonis 6. Robotont ratta 1 omniliikumise ja odomeetria jaoks vajalikud suurused, vektorid ja kõigi rataste positiivsed suunad

Roboti ühe ratta joonkiirust $wheelLinearVelocity$ (meetrit sekundis) on võimalik leida järgneva valemi (1) abil.

$$wheelLinearVelocity = R * robotAngularVelocity + robotSpeed * cos(\theta - \phi) \quad (1)$$

$$robotSpeed = \sqrt{(robotSpeedX^2 + robotSpeedY^2)} \quad (2)$$

$$\phi = arctan2(robotSpeedY, robotSpeedX) \quad (3)$$

Omnirataste kaugust keskpunktist (meetrit) tähistab R. Nurka roboti omniratta positiivse suuna ja võrdlussuuna vahel (radiaani) tähistab θ. Nurka roboti suuna ja võrdlussuuna vahel (radiaani) tähistab ϕ (Valem 3). (Joonis 6) R ja θ väärtused loeb sõlm sisse parameetriserverist, vastavalt võtmetega „wheelDistanceFromCenter“ ja „wheelRads“.

Teemalt /cmd_vel loeb `driver_node` roboti joonkiiruse (meetrit sekundis) x- ja y-telgede komponendid – $robotSpeedX$ ja $robotSpeedY$, ning roboti nurkkiiruse (radiaani sekundis) ümber z-telje $robotAngularVelocity$.

Roboti kiirust $robotSpeed$ (meetrit sekundis) on võimalik arvutada valemi (2) abil.

Roboti rataste nurkkiirused, mis on teisendatud mootorite juhtimiseks sobivateks väärtusteks on võimalik leida kasutades valemit (4).

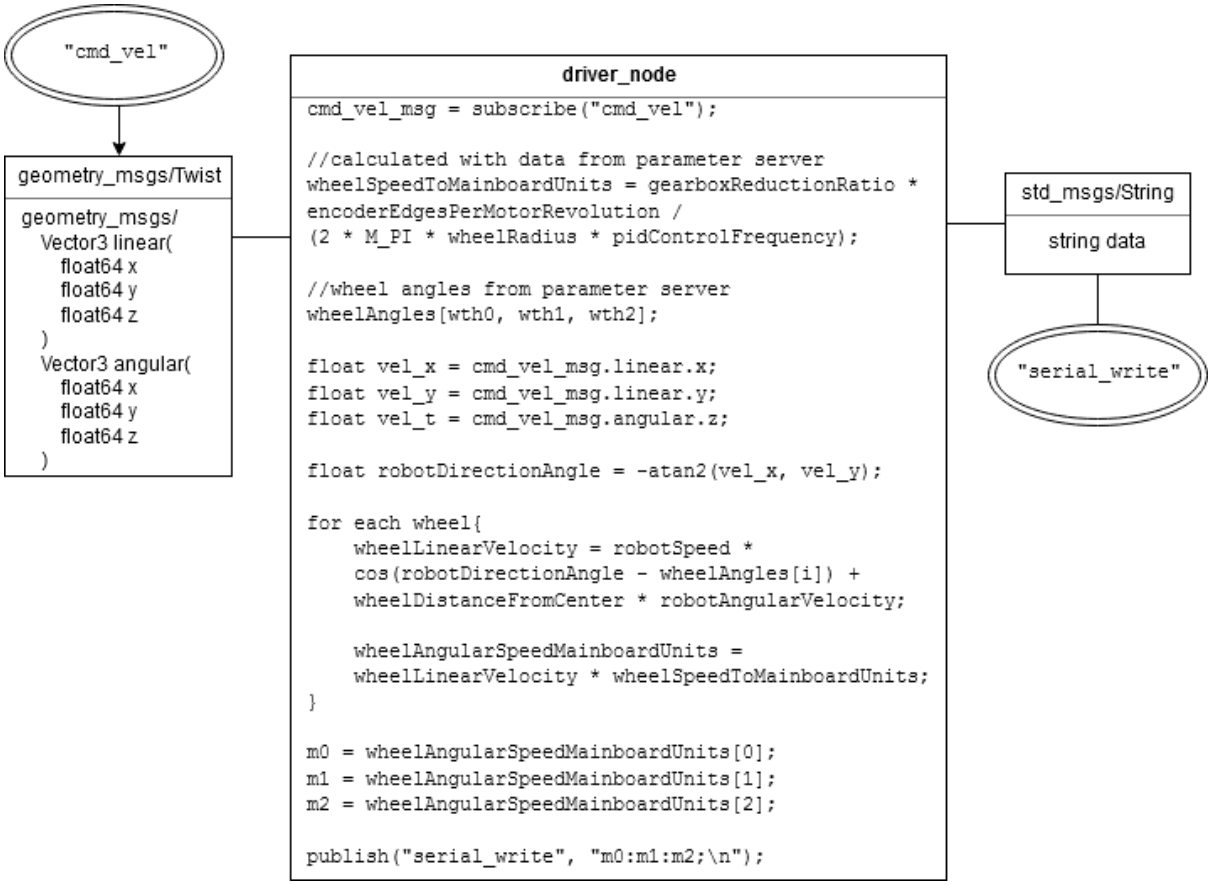
$$wheelAngularSpeedMainboardUnits = wheelSpeedToMainboardUnits * wheelLinearVelocity \quad (4)$$

$$wheelSpeedToMainboardUnits = \frac{gearboxReductionRatio * encoderEdgesPerMotorRevolution}{2 * \pi * \text{wheelRadius} * \text{pidControlFrequency}} \quad (5)$$

Valem 5 abil saab leida $wheelSpeedToMainboardUnits$, mis on arv, millega saab ratta kiiruse teisendada mootorite juhtimiseks sobivateks väärtusteks.

Kõik väärtused $wheelSpeedToMainboardUnits$ arvutamiseks on konstantsed ning need laaditakse parameetriserverist sõlme käitamisajal. Mootorite ülekandetegurit tähistab $gearboxReductionRatio$. Mootori koodri samme ühes täispöördes tähistab $encoderEdgesPerMotorRevolution$. Omniratta raadiust (meetrit) tähistab wheelRadius. Sagedust (hertsides), millega PID-kontroller juhtkontrolleris arvutab, tähistab pidControlFrequency.

Iga ratta jaoks arvutatakse eraldi nurkkiirused, mis seejärel avaldatakse teemal /serial_write, kujul „m0:m1:m2;\n“, kus m0, m1 ja m2 on rataste nurkkiirused.

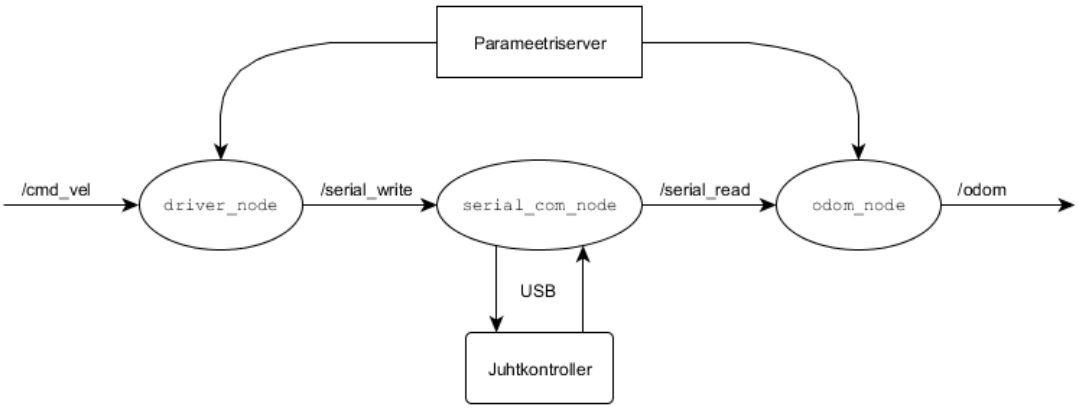


Joonis 7. `driver_node` *sõlme programm väljendatud pseudokoodina koos vastavate sõnumitüüpidega*

3.1.4. Odomeetria

Odomeetria on informatsioon roboti asukoha ja asendi kohta, mis on arvutatav mootorite tagasiside põhjal. Odomeetria abil roboti positsiooni hindamiseks loodi sõlm `odom_node`. Antud sõlme ülesandeks on lugeda mootorite hetkkiirused ja nende abil arvutada roboti asukoht ja asend.

`odom_node` sõlme ülesandeks on võtta vastu sõnumeid teemalt /serial_read, lugeda sealt välja mootorite koodrite väärtused, ehk mootorite kiirused, mille põhjal arvutatakse roboti hetkasukoht, hetkasend ja hetkjoon- ja nurkkiirused, mis edastatakse edasi teemal /odom.



Joonis 8. Käesoleva töö jooksul draiverile lisandunud sõlm `odom_node` ja selle loogiline koht draiveris.

Mootorite hetkkiirused avaldab `serial_com_node` sõlm teemal /serial_read, sõnumitüübiga std_msgs/String. Sõnumi sisu kuju on „m0:m1:m2;\n“, kus m0, m1 ja m2 tähistavad vastava mootori hetkkiirust, mis saadakse mootorite koodritelt.

Sõnumid teemal /odom on nav_msgs/Odometry tüüpi, ning sisaldavad endas hetkaega, hetkasukohta ja -asendit ning hetkjoon- ja nurkkiiruseid. Sõlm edastab sõnumid teemal /odom, kuna see on üks osa ROSi nagivatsiooni teegist (*Navigation Stack*) [13], mis on mitmetest sõlmedest koosnev teek, mis tegeleb roboti liikumisega. Teek võtab sisendiks odomeetria, sensorite väärtused ning lõppasukoha ja -asendi, mille põhjal teek arvutab kiiruse käsud, mille robot peaks liikuma. Navigatsiooni teegi nõudeks on odomeetria avaldamine teemal /odom, sõnumitüübiga nav_msgs/Odometry.

`odom_node` arvutab roboti joon- ja nurkkiirused kasutades maatriksarvutusi [22].

Omnirataste kaugust keskpunktist (meetrit) tähistab R. Nurka roboti omniratta positiivse suuna ja võrdlussuuna vahel (radiaani) tähistab θ. Nende andmete põhjal saab koostada maatriksi (6).

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} cos\theta_1 & sin\theta_1 & R \\ cos\theta_2 & sin\theta_2 & R \\ cos\theta_3 & sin\theta_3 & R \end{bmatrix} \begin{bmatrix} V_{bx} \\ V_{by} \\ \Psi \end{bmatrix} \quad (6)$$

$$\begin{bmatrix} V_{bx} \\ V_{by} \\ \Psi \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \quad (7)$$

$$p_i = \frac{enc_i}{wheelSpeedToMainboardUnits} \quad (8)$$

Olgu maatriks (7) maatriksi (6) inverteerimisel saadud maatriks.

Maatriksis (6) tähistab p_i mootori koodritelt saadud väärtusi (enc_i), mis on teisendatud päriselulisteks väärtusteks $wheelSpeedToMainboardUnits$ abil (valem (5)) valemiga (8).

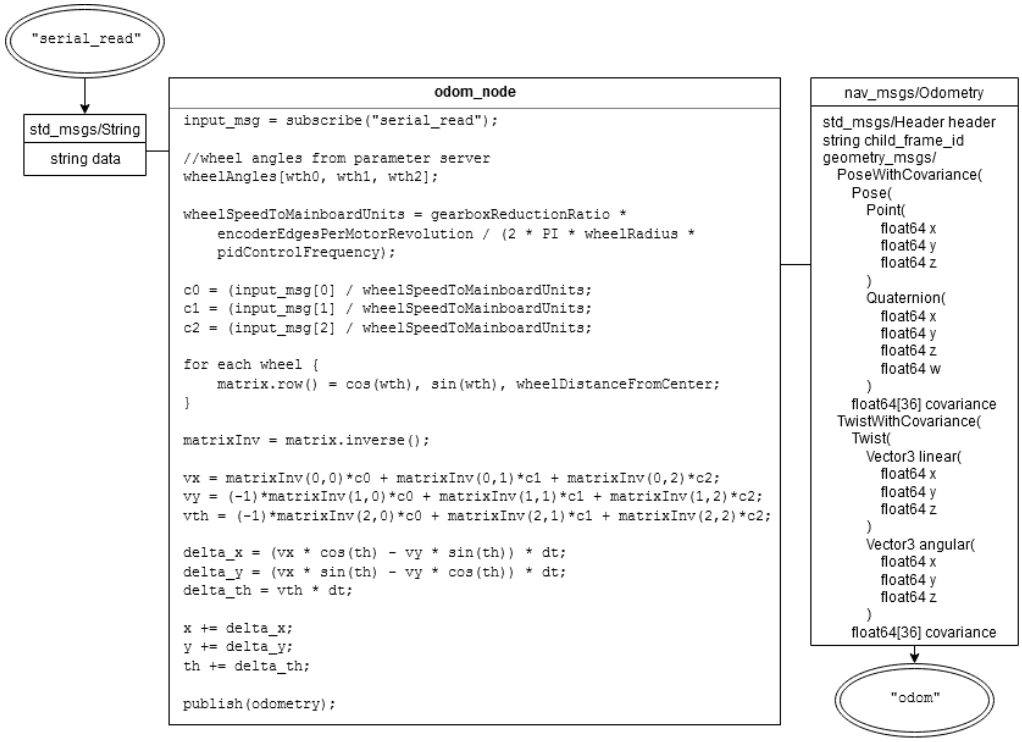
Roboti hetkkiiruse (meetrit sekundis) x- ja y-telgede komponendid (V_{bx} ja V_{by}) ning roboti hetknurkkiirus (radiaani sekundis) Ψ on võimalik leida valemite (9), (10) ja (11) abil, mis kasutavad maatriksi (7) liikmeid.

$$V_{bx} = a_1 * p_1 + b_1 * p_2 + c_1 * p_3 \quad (9)$$

$$V_{by} = a_2 * p_1 + b_2 * p_2 + c_2 * p_3 \quad (10)$$

$$\Psi = a_3 * p_1 + b_3 * p_2 + c_3 * p_3 \quad (11)$$

Valemites (9), (10, (11) leitud kiiruseid kasutades leitakse roboti hetkasukoht, hetkasend ja hetkjoonkiirused ja -nurkkiirused, mis avaldatakse teemal /odom.



Joonis 9. `odom_node` *sõlme programm väljendatud pseudokoodina koos vastavate sõnumitüüpidega*

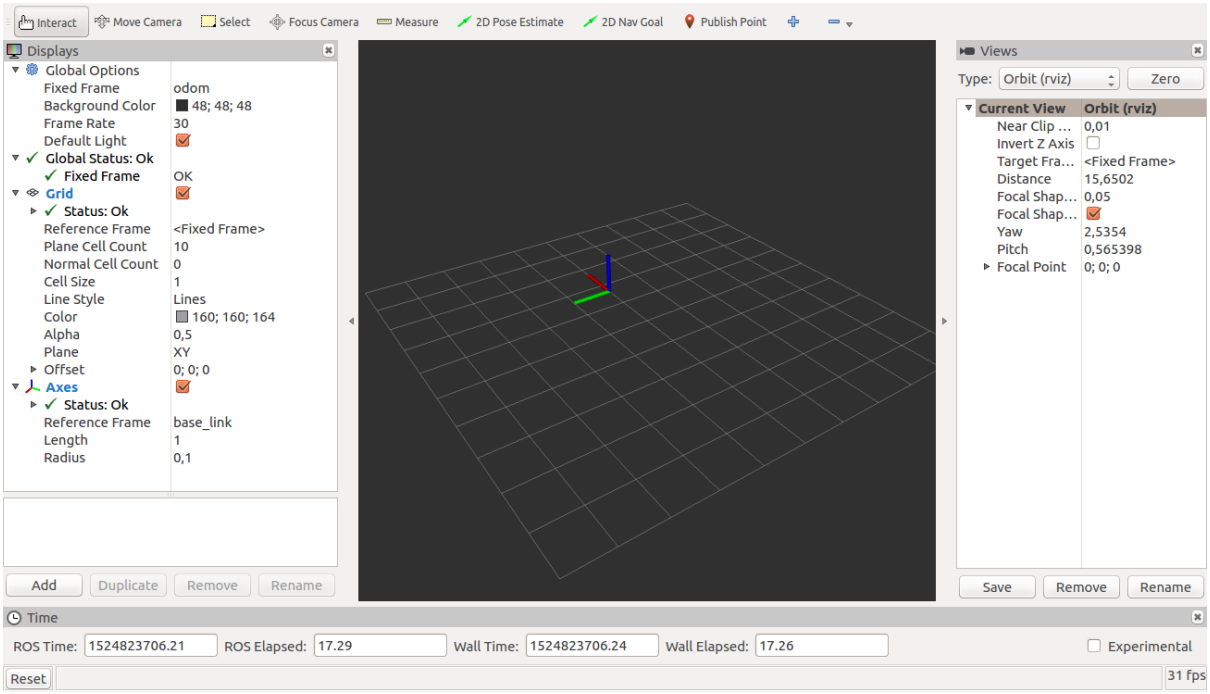
4. Tulemused

4.1 Esmane odomeetria testimine

Töö tulemusena valmis Robotont robotiplatvormile odomeetria sõlm `odom_node`, mis kuulutab sõnumeid teemal /odom, ning on selle põhjal võimeline roboti liikumist kujutama arvutiprogrammis.

Kuna ROS on loodud modulaarsena, lubas see testida odomeetria töötavust, ilma, et robot peaks ise liikuma. Odomeetria sõlm `odom_node` jälgib teemat /serial_read, mis on mootorite hetkkiirused kodeeritud kujul, st arvud näitavad, mitu koodri sammu läbib mootor sekundis. Kuna mootoritelt saadud kiirused teemal /serial_read on võrreldavad kiirustega, mida saadetakse teemalt /serial_write juhtkontrollerisse, oli võimalik testida odomeetriat, kasutades odomeetria arvutamiseks kiiruseid /serial_write teemalt.

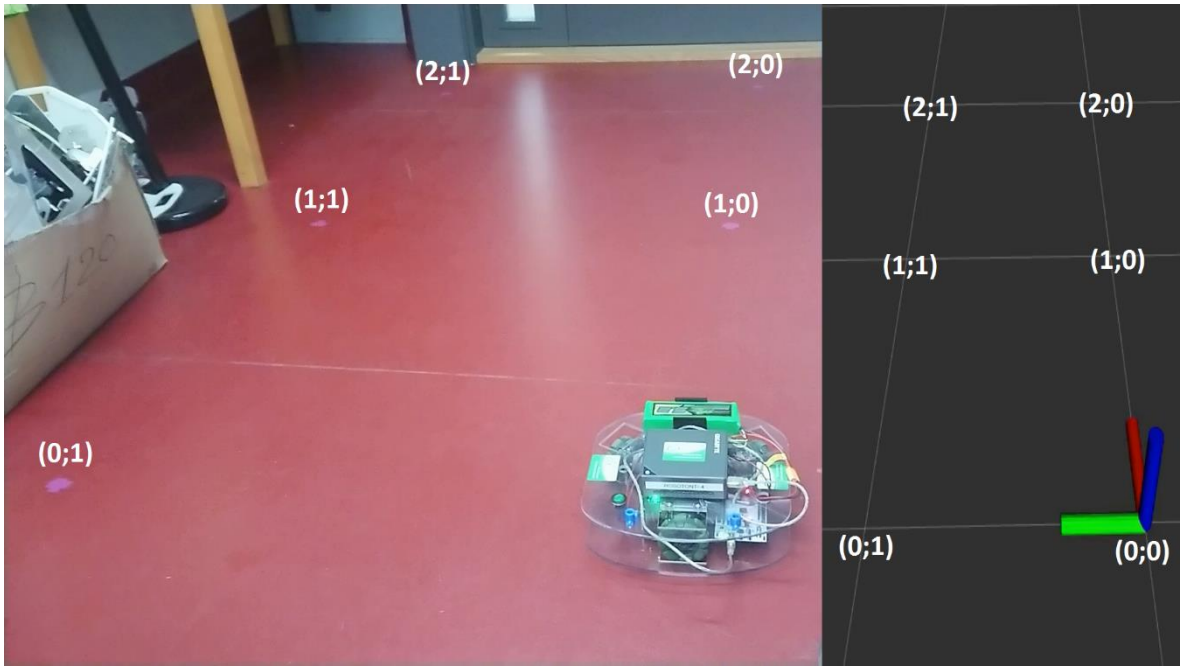
ROSiga tuleb kaasa kolmemõõtmeline visualiseerimise tarkvara RViz (*ROS Vizualization*) [23], mis võimaldab kolmemõõtmelises ruumis kujutada roboti liikumist ja olekut. Kuna roboti hetkkiiruseid ja asukohta avaldatakse teemal /odom, saab liikumist RVizi abil visualiseerida jälgides selle teema sõnumeid.



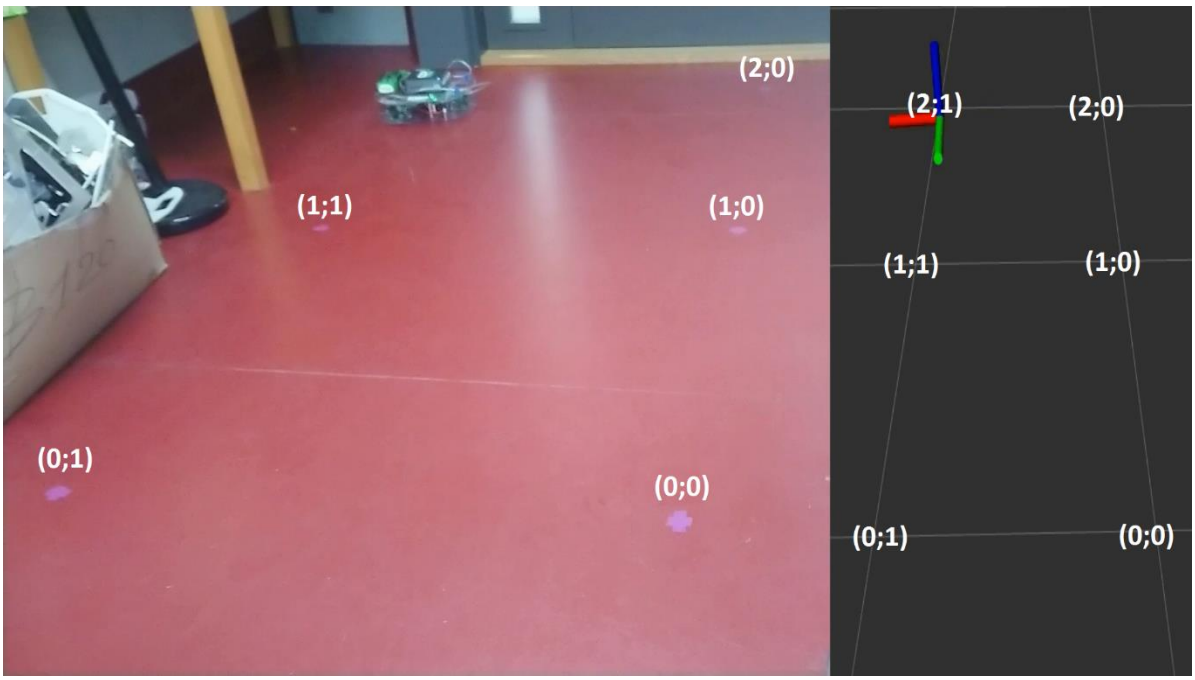
Joonis 10. Ekraanitõmmis RViz programmist

4.2 Funktsionaalne odomeetria analüüs

Loodud odomeetria sõlmega tekkis võimalus Robotondi liikumist saadetud väärtuste põhjal võrrelda Robotondi tegeliku asukohaga. RViz kujutab odomeetriat, mis on arvutatud /serial_write sõnumi väärtuste põhjal (Joonis 11). Joonis 12 on näha roboti asukoha ja asendi muutu. Kuna odomeetria arvutati /serial_write teema väärtuste põhjal, siis kui robotile saata käsk seisma jäämiseks, jääb RViz programmis robotina kujutatud teljestik koheselt seisma, kuid robot liigub veel väikese maa edasi, kuna robotil on veel inertsi.



Joonis 11. Pilt Robotondist asukohas (0;0) (vasakul), kuvatõmmis RViz programmist, teljestik kujutab Robotondi keskkohta.



Joonis 12. Pilt Robotondist asukohas umbes (2;1) (vasakul), kuvatõmmis RViz programmist, kus on näha teljsetiku asukoha muutu vastavalt roboti asukohale

5. Tulemuste analüüs ja järeldused

5.1 Peamised tulemused

Käesoleva töö tulemusena sai parandatud Robotont juhtkontrolleri põhivara, millega paranes Robotondi liikumine. Töö tulemusena arendati Robotondi juhtimisdraiveri funktsionaalsust, andes võimekuse juhtimisdraiverit kasutada robotiga, millel vähemalt 3 omniratast, mis asetsevad ringjooneliselt ümber roboti keskkoha ning roboti mootoritevaheline nurk võib olla mistahes suurusega. Töö tulemusena valmistati draiver jaoks uus sõlm, mis on võimeline 3 omniratta odomeetriat arvutama.

5.2 Järgmised arendustegevused

5.2.1 Jadaliides

Töö üheks eesmärgiks oli jadaühenduse parandamine. Esmatähtis oli parandada roboti juhtimine jadaliidese kaudu, mis sai ka teostatud. Nimetatud paranduse läbiviimisel kerkis üles probleem mistahes andmete saatmisega juhtkontrollerilt juhtarvutile, mis piiras ka tegeliku odomeetria arvutamist. Töö käigus tehtud katsetuste põhjal tehti kindaks, et probleem jadaühendusega ei ole tingitud riistvaralistest häiretest. Võimalik probleemi põhjus võib olla tingitud nii Robotondi jadaühenduse draiverist kui ka juhtkontrolleri põhivarast. Jadaühenduse draiver loeb praegu sisse nii palju andmeid kui on jadaühenduse sisend-puhvris, kuid ei kontrolli reavahetuse märgi ’\n’ olemasolu, mistõttu loetakse sisse suurem hulk andmeid kui vaja ning puhvri täitumisel kaovad sümbolid ära. Juhtkontrolleri põhivara saadab hetkel andmeid *main* funktsioonis *while* silmuses.

Üks võimalikest lahendustest oleks Robotondi jadaühenduse draiverisse lisada vajaliku sümboli kontroll ja juhtkontrolleri põhivaras kasutada ära katkestusi, et andmeid edastada. Teine võimalik lahendus oleks kasutada rosserial protokolli, mis võimaldab arendusplaatidel käituda ROSi sõlmedena [24].

5.2.2 Odomeetria

Töö üheks eesmärgiks oli odomeetriat teostav draiver. Töö käigus valmis draiver, mis suudab teostada odomeetriat ainult kolmel rattal, piirates selle kasutust. Kuigi odomeetria jaoks piisab kolmest rattast, on rohkemate rataste puhul võimalik kasutada 3 ratta kombinatsioone n rattast. Sellisel juhul tuleb arvutada iga kombinatsiooni jaoks odomeetria, ning lõpuks kõikide vahel leida keskmine tulemus [25].

6. Kokkuvõte

2017. aastal alustati Tartu Ülikooli tehnoloogiainstituudis arendust avatud robotiplatvormiga Robotont, mille eesmärgiks on leida kasutus nii teadus- kui ka haridusvaldkonnas. 2017. aastal arendati Robotondile ka tarkvaraline lahendus, millega realiseeriti ühilduvus ROSi raamistikuga ning liikumis- ja juhtimisvõimekus.

Selle töö eesmärgiks oli parandada Robotont platvormi juhtkontrolleri põhivara, lisada Robotondi draiverile universaalne omniliikumine ja odomeetria.

Töö tulemusena parandati põhivara, millega paranes Robotondi liikumine. Töö tulemusena lisati draiverile universaalne omniliikumine, võimaldades teostada omniliikumist vähemalt 3 omnirattaga. Draiverile sai lisatud sõlm, mis suudab arvutada kolme omniratta põhjal odomeetria.

Draiverile lisatud sõlm edastab odomeetria teemal /odom, sõnumitüübiga nav_msgs/Odometry, mis on nõutud ROSi navigatsiooni teegi kasutamiseks.

Viited

[1] A. R. Khairuddin, M. S. Talib, and H. Haron, "Review on simultaneous localization and mapping (SLAM)," in *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2015, pp. 85–90, DOI: 10.1109/ICCSCE.2015.7482163.

[2] R. Vellerind, "Avatud robotiarendusplatvormi ROS võimekuse loomine Tartu Ülikooli Robotexi robootikaplatvormile," Tartu Ülikool, 2017.

[3] K.-G. Kruus, "Jalgpalliroboti löögimehhanismi elektroonikalahendus," Tartu Ülikool, 2013.

[4] Pololu, "19:1 Metal Gearmotor 37Dx68L mm with 64 CPR Encoder." [Võrgumaterjal]. Saadaval: https://www.pololu.com/product/2822. [Kasutatud: 26. aprill 2018].

[5] VEX Robotics, "Wheels." [Võrgumaterjal]. Saadaval: https://www.vexrobotics.com/edr-wheels.html. [Kasutatud: 13. mai 2018].

[6] Gigabyte, "GB-BSi5-6200 (rev. 1.0)." [Võrgumaterjal]. Saadaval: https://www.gigabyte.com/Mini-PcBarebone/GB-BSi5-6200-rev-10#ov. [Kasutatud: 10. mai 2018].

[7] Intel, "Intel® RealSense™ Camera R200." [Võrgumaterjal]. Saadaval: https://ark.intel.com/products/92256/Intel-RealSense-Camera-R200. [Kasutatud: 02. mai 2018].

[8] STMicroelectronics, "NUCLEO-L476RG." [Võrgumaterjal]. Saadaval: http://www.st.com/en/evaluation-tools/nucleo-l476rg.html. [Kasutatud: 26. aprill 2018].

[9] R. Randoja, "Liikumismoodul." [Võrgumaterjal]. Saadaval: http://digilabor.ut.ee/index.php/Liikumismoodul2012. [Kasutatud: 13. mai 2018].

[10] Open Source Robotics Foundation, “ROS Nodes.” [Võrgumaterjal]. Saadaval: http://wiki.ros.org/ROS/Concepts. [Kasutatud: 10. mai 2018].

[11] Open Source Robotics Foundation, “ROS Messages.” [Võrgumaterjal]. Saadaval: http://wiki.ros.org/Messages. [Kasutatud: 10. mai 2018].

[12] Open Source Robotics Foundation, “ROS Topics.” [Võrgumaterjal]. Saadaval: http://wiki.ros.org/Topics. [Kasutatud: 10. mai 2018].

[13] Open Source Robotics Foundation, “ROS navigation.” [Võrgumaterjal]. Saadaval: http://wiki.ros.org/navigation. [Kasutatud: 20. mai 2018].

[14] S. Grove, “BufferedSerial,” 2013. [Võrgumaterjal]. Saadaval: https://os.mbed.com/users/sam_grove/code/BufferedSerial/. [Kasutatud: 13. mai 2018].

[15] Arm Limited, “mbed Serial.” [Võrgumaterjal]. Saadaval: https://os.mbed.com/docs/latest/reference/serial.html. [Kasutatud: 13. mai 2018].

[16] “MODSERIAL race condition in receive?” [Võrgumaterjal]. Saadaval: https://os.mbed.com/forum/bugs-suggestions/topic/2936/. [Kasutatud: 18. aprill 2018].

[17] “mbed Cookbook MODSERIAL.” [Võrgumaterjal]. Saadaval: https://os.mbed.com/cookbook/MODSERIAL. [Kasutatud: 18-Apr-2018].

[18] A. Kirkham, “MODSERIAL GETC.cpp.” [Võrgumaterjal]. Saadaval: https://os.mbed.com/users/AjK/code/MODSERIAL/docs/ae0408ebdd68/GETC_8cpp_source.html. [Kasutatud: 20. mai 2018].

[19] A. Kirkham, “MODSERIAL PUTC.cpp.” [Võrgumaterjal]. Saadaval: https://os.mbed.com/users/AjK/code/MODSERIAL/docs/ae0408ebdd68/PUTC_8cpp_source.html. [Kasutatud: 20. mai 2018].

[20] “ROS Parameter Server.” [Võrgumaterjal]. Saadaval: http://wiki.ros.org/Parameter Server. [Kasutatud: 18. aprill 2018].

[21] R. Randoja, “Omni motion.” [Võrgumaterjal]. Saadaval: http://digilabor.ut.ee/index.php?title=Omni_motion. [Kasutatud: 18. aprill 2018].

[22] M. Ashmore and N. Barnes, “Omni-drive robot motion on curved paths: The fastest path between two points is not a straight-line,” in *AI 2002: Advances in Artificial Intelligence*, London, UK, UK: Springer-Verlag, 2002, pp. 225–236, DOI: 10.1007/3-540-36187-1_20.

[23] Open Source Robotics Foundation, “rviz.” [Võrgumaterjal]. Saadaval: http://wiki.ros.org/rviz. [Kasutatud: 26. aprill 2018].

[24] Open Source Robotics Foundation, “rosserial.” [Võrgumaterjal]. Saadaval: http://wiki.ros.org/rosserial. [Kasutatud: 20. mai 2018].

[25] P. Kallas, “ Probabilistic Localization of a Soccer Robot,” University of Tartu, 2013.

Lisad

Lisa 1

Robotondi ROSi draiver arendamise haldamiseks kasutati GitHub platvormi. Käesoleva töö raames loodud kood on leitav GitHubi repositooriumist. Käesoleva töö raames kirjutatud kood asub kaustas /robotont_driver.

https://github.com/ut-ims-robotics/robotont/tree/kinetic-devel-maidla/robotont_driver

Lisa 2

Robotondi mbed arendusplaadi põhivara haldamiseks kasutati mbed veebipõhist kompilaatorit. Käesoleva töö raames loodud põhivara on leidav mbed keskkonnast. Käesoleva töö raames kirjutatud kood asub kaustas RobotontNucleoMainboard.

https://os.mbed.com/users/maidlam/code/RobotontNucleoMainboard/

Lihtlitsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Martin Maidla, annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Avatud robotiarendusplatvormi Robotont omniliikumise ja odomeetria arendamine“, mille juhendaja on 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 20.05.2018