Tartu Ülikool

Loodus- ja täppisteaduste valdkond

Tehnoloogiainstituut

Jürgen Kottise

Dockeri konteineritel põhinev haldustarkvara Robotont 3 õpperobotile

Bakalaureusetöö (12 EAP)
Arvutitehnika eriala

Juhendaja:

PhD Veiko Vunder

Tartu 2025

Resümee/Abstract

Dockeri konteineritel põhinev haldustarkvara Robotont 3 õpperobotile

Robotont on Tartu Ülikoolis välja töötatud avatud lähtekoodiga robotiplatvorm õppetööks ja teadustööks. Robootikarakenduste arendamisel on üks olulisemaid väljakutseid erinevate tarkvara-sõltuvuste haldamine ning keskkondade kiire vahetamine. Käesoleva bakalaureusetöö käigus töötati välja lahendus, mis kapseldab Robotont 3 õpperoboti ROS-põhised rakendused Dockeri konteineritesse ja võimaldab neid hallata nii roboti kasutajaliidese kui ka veebiliidese kaudu. Lahendus koosneb teenusest, mis vahendab andmesidet püsivara ja konteineri vahel, kasutades pseudoterminali, püsivara alammenüüst konteinerite käivitamiseks ja peatamiseks ning Flask-põhisest veebiliidesest. Lahendus vähendab süsteemi konfigureerimise aega ja kõrvaldab sõltuvuskonfliktid.

CERCS: T125 Automatiseerimine, robootika, juhtimistehnika

Märksõnad: Docker, ROS, robootika, haldustarkvara

Management software for Robotont 3 based on Docker containers

Robotont is an open-source robot platform developed at the University of Tartu for teaching and for research. One of the key challenges in the development of robotic applications is the management of different software dependencies and the rapid switching between environments. In this thesis, a solution was developed that encapsulates the ROS-based applications of the Robotont 3 into Docker containers and allows them to be managed via both the robot's on-board user interface and the web interface. The solution consists of a service that mediates data communication between the firmware and the container using a pseudo-terminal, a firmware submenu for starting and stopping containers, and a Flask-based web interface. The solution reduces system configuration time and eliminates dependency conflicts.

CERCS: T125 Automation, robotics, control engineering

Keywords: Docker, ROS, robotics, management software

Sisukord

Joonised

Tabelid

Lühendid, mõisted

ROS - *Robot Operating System*

PTY - Pseudoterminal

USB - *Universal Serial Bus*

LED - *Light Emitting Diode*

VNC - *Virtual Network Computing*

API - *Application Programming Interface*

1 Sissejuhatus

Robootika valdkond on viimastel aastatel jõudsalt arenenud, 2024. aastal jõudis kasutuses olevate robotite arv ligikaudu 3,9 miljonini [1]. Tööstuslike robotite ülemaailmne turuväärtus on 2025. aastal jõudnud kõigi aegade kõrgeimale tasemele, ligikaudu 16,5 miljardi USA dollarini [2]. Uued tehnoloogiad, näiteks tehisintellekt ja masinõpe, on seda kasvu kiirendanud [1]. Hariduses ja teadustöös kasutatakse üha enam avatud platvorme ja õpperoboteid, mis võimaldavad tudengitel ja inseneridel omandada praktilisi kogemusi robotisüsteemide arendamisel.

Üheks oluliseks robootikarakenduste aluseks on avatud lähtekoodiga platvorm ROS (Robot Operating System), mis pakub robustset raamistikku erinevate robootikasüsteemide arendamiseks, testimiseks ja juurutamiseks [3]. ROSi mitmekülgsus ja paindlikkus on teinud sellest standardi, mida kasutavad teadlased, insenerid ja arendajad üle maailma [4]. Selle platvormi laialdane kasutamine on võimaldanud luua kogukonnapõhise lähenemise, kus uued tööriistad ja teegid on pidevalt kättesaadavad ja arendatavad [3]. ROSi arhitektuur on üles ehitatud sõlmede (*nodes*) baasil, mis suhtlevad omavahel läbi kuulutamise ja tellimise (*publish/subscribe*) mehhanismi [5]. Sõlmed võimaldavad jagada protsesse ja funktsioone, hõlbustades seeläbi komplekssete robootikasüsteemide arendust [5].

Robotont on Tartu Ülikoolis välja töötatud avatud lähtekoodiga robotiplatvorm õppetööks ja teadustööks [6]. Roboti peamine eesmärk on pakkuda mitmekülgset ja paindlikku platvormi, mida saab hõlpsasti kohandada erinevateks uurimis- ja õpetamisprojektideks [6]. Sellel on modulaarne disain, mis võimaldab lihtsalt lisada erinevaid andureid ja seadmeid, et täita spetsiifilisi ülesandeid, nagu näiteks autonoomne navigeerimine, objektide tuvastamine ja liikumiste planeerimine [7]. Robotont toetab laialdaselt kasutatavat ROS platvormi, mis võimaldab robootikafunktsioonide tõhusat arendamist ja integreerimist [7].

Tihtipeale soovitakse ühes robotis käitada erinevaid rakendusi, näiteks kaugjuhtimine, kaardistamine või hoopis midagi muud. Igal rakendusel võivad olla eri sõltuvused, teekide versioonid või erinevad ROS versioonid (ROS 1 või ROS 2). Traditsiooniliselt paigaldatakse roboti pardaarvutisse ühe operatsioonisüsteemi alla rakendus ja kõik vajalik selle kasutamiseks. Seega võivad erinevate projektide teegid sattuda konfliktidesse ning erinevate versioonide paralleelne ülalhoidmine võib olla probleemne [8].

Üheks lahenduseks sellistele tarkvarahalduse probleemidele on konteinertehnoloogia kasutamine. Docker võimaldab rakenduse koos kõigi selle sõltuvustega pakendada standardiseeritud konteinerisse, mis pakub ühtlustatud ja isoleeritud keskkonda [9]. Erinevalt traditsioonilistest virtualiseerimismeetoditest, kus virtuaalmasinad vajavad oma operatsioonisüsteemi, jagavad konteinerid sama operatsioonisüsteemi tuuma (*kernel*), mis teeb nad palju kiiremaks ja ressursikasutuse efektiivsemaks [9]. Docker võimaldab rakendada tarkvarakomponente koos kõigi vajalike sõltuvustega ühte konteinerisse, tagades, et rakendus töötab ühtemoodi erinevates keskkondades [9]. See omadus on oluline robootikarakenduste puhul, kus süsteemi stabiilsus ja sõltuvuste täpne haldamine on kriitilise tähtsusega [8]. Erinevalt traditsioonilisest otse operatsioonisüsteemile paigaldamisest saab konteineri hõlpsalt teises masinas uuesti luua, mis lahendab "minu masinas töötab" probleemi suurel määral [8]. Konteinerid võimaldavad hariduslikus kontekstis luua eelhäälestatud turvalise katsetuskeskkonna, kus tudengid saavad robotit kasutada ilma riskita operatsioonisüsteemi tasemel midagi rikkuda.

Käesolev bakalaureusetöö keskendub konteinertehnoloogia rakendamisele Tartu Ülikoolis arendatud Robotont 3 õpperobotil. Töö eesmärgiks on luua süsteem, mis võimaldab roboti tarkvaralisi rakendusi kapseldada Dockeri konteineritesse ja kasutades loodud konteinereid lihtsustada Robotondi rakenduste vahel vahetamist läbi haldustarkvara. See tagab, et robotit saab paindlikult kasutada erinevatel eesmärkidel (õppetöö, demonstratsioonid, arendusprojektid) ilma aeganõudva ümberseadistamiseta.

2 Ülevaade kirjandusest

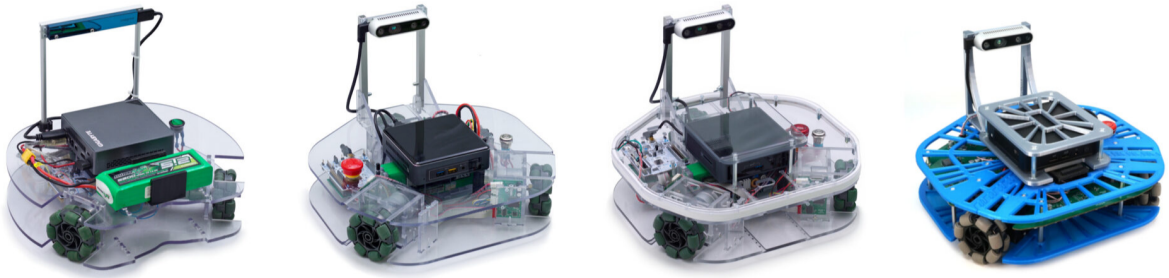
2.1 Robotont platvorm

Robotont on avatud lähtekoodiga omniliikuv robotiplatvorm, mis on loodud kasutamiseks hariduses ja teadustöös [6]. Platvorm adresseerib vajadust täita lõhe lihtsamate väiksema arvutusvõimsusega õpperobotite ja kallimate suurema arvutusjõudlusega tööstusrobotite vahel [10]. Robotont on edukalt kasutust leidnud ülikoolide kursustel, kutse- ja täiendõppes ning ka veebikursustel robootika ja ROS-i õpetamiseks [10], demonstreerides selle praktilist väärtust õppevahendina.

Robotonti on arendatud mitmes järjestikuses generatsioonis (vt joonis 2.1). Igaühes on võrreldes eelmisega toimunud tehniline areng. Antud töös kasutatakse Robotondi 3. generatsiooni, kujutatud joonisel 2.1 (d). Robotont 3 säilitab kõik eelkäijate tehnilised võimalused ja ROS-ökosüsteemi integratsiooni, kuid täiustab kere. Eesmärgiks on olnud lihtsam valmistamine ja kokkupanek ning suurem modulaarsus [7].

Robotont 3 platvorm on varustatud võimeka pardaarvutiga (Intel NUC) ja sügavuskaameraga (Intel RealSense), mis annavad süsteemile piisava arvutusvõimsuse ja sensoorse taju kaasaegsete autonoomsete robotülesannete jaoks [7]. Kere on 3D-prinditud ja koosneb viiest moodulist: raamimoodul, akumoodul, arvutimoodul, kaameramoodul ning mootorimoodul [7].

Robotont 3 elektroonika tugineb ühele terviklikule trükkplaadile. Plaadile on integreeritud kõik põhilised elektroonikakomponendid: STM32 mikrokontroller, mis vahendab suhtlust pardaarvuti, mootoriajamite ja andurite vahel, eraldi toitehalduse mikrokontroller, kolme mootori draiverid, laadimiskontroller ning kasutajaliidese komponendid (kvadtatuurkoodernupp, avariilüliti, OLED-ekraan, LEDid) [7].



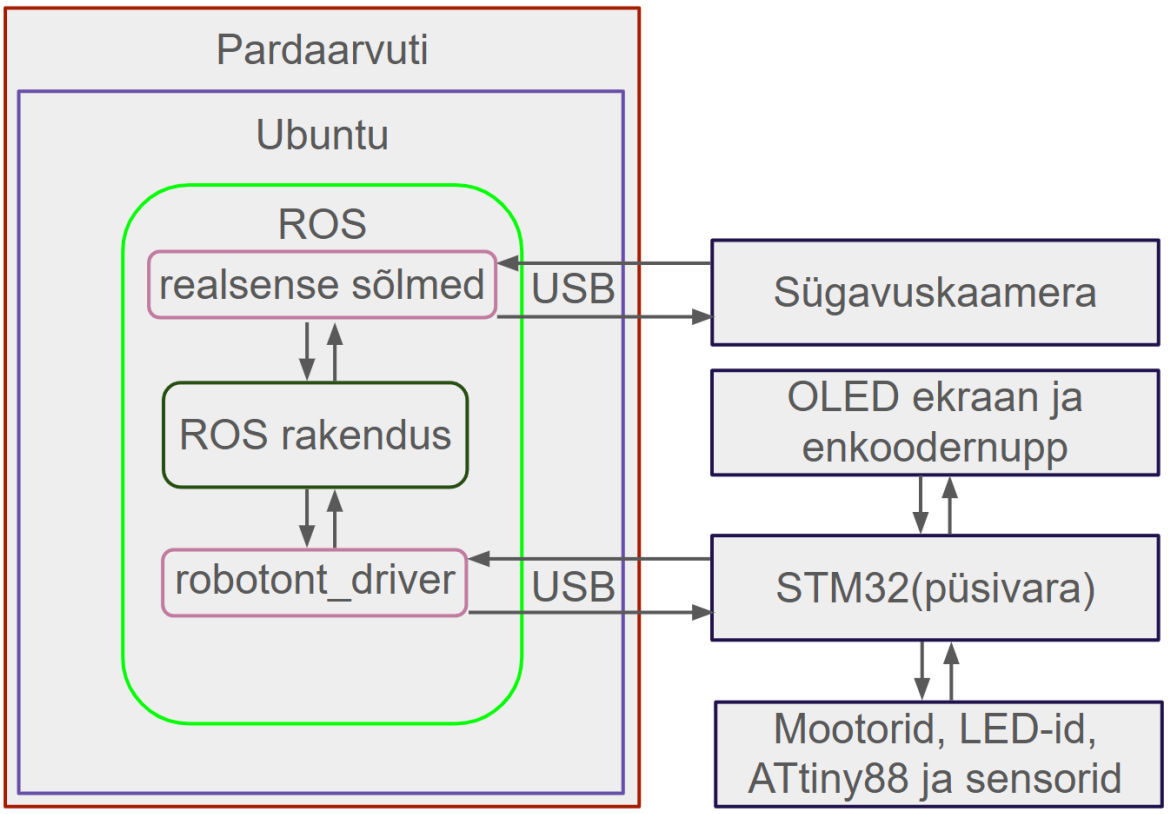
(a) (b) (c) (d)

Joonis 2.1: Robotondi generatsioonid. (a) 1. generatsioon, (b) 2. generatsioon, (c) 2.1 generatsioon, (d) 3. generatsioon

2.1.1 Robotondi tarkvaraline arhitektuur

Robotont 3 tarkvaralise arhitektuuri saab jagada kaheks peamiseks osaks: pardaarvuti (Intel NUC, kus töötab Ubuntu koos ROS-iga) ja mikrokontroller (STM32 mikrokontroller püsivaraga), vt joonist 2.2. Pardaarvutis töötab Ubuntu operatsioonisüsteem, mille peal jooksevad erinevad ROS sõlmed.

- **RealSense ROS sõlmed** haldab USB kaudu ühendatud Intel RealSense sügavuskaamerat. See sõlm loeb kaamerast vajalikud andmed ja avaldab neid ROS-i rubriikidesse, et neid saaks kasutada visualiseerimiseks, navigeerimiseks ja muudeks tegevusteks.
- **robotont_driver** on ROS-i kimp, mis suhtleb USB kaudu STM32 püsivaraga. Saadetakse liikumiskäske mikrokontrollerile ning mikrokontrollerist saab tagasisidet (nt odomeetriat). Suhtlus toimub jadaliidese põhise tekstiprotokolli abil [11].
- **ROS rakendus** võib olla näiteks kõrgema taseme ROS-sõlm või ROS-sõlmed, mis loovad liikumiskäsklusi ja töötlevad sensoriandmeid. See sõlm või need sõlmed suhtlevad *robotont_driver*-iga ROS-i rubriikide kaudu [11].



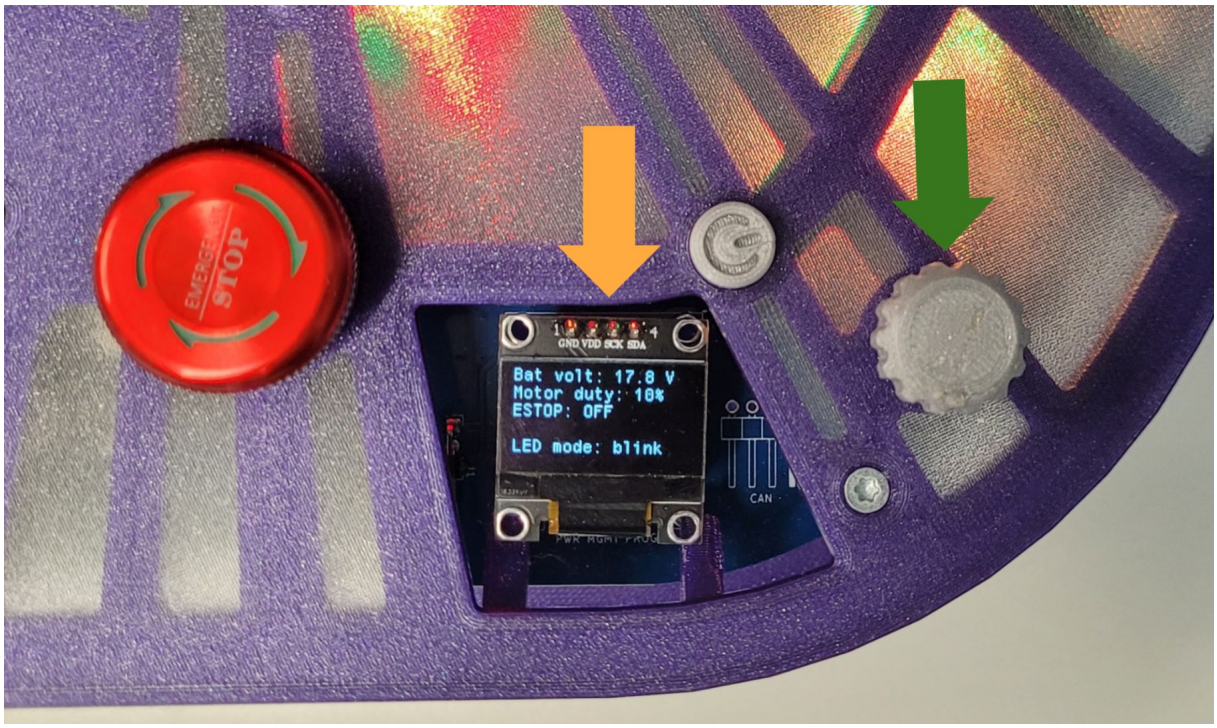
Joonis 2.2: Robotondi lihtsustatud tarkvaraline arhitektuur

Pardaarvuti suhtleb STM32 mikrokontrolleriga USB kaudu. Suhtlusprotokoll on tekstipõhine, paketid sisaldavad kahe tähega käsku ja neile järgnevaid parameetreid [12, 13]:

- `RS:x:y:z\r\n` – kiirused telgede suhtes
- `MS:M0:M1:M2\r\n` – mootorite individuaalsed kiirused
- `DC:M0:M1:M2\r\n` – mootori täitetegur
- `OD:a:b:c:x:y:z\r\n` – odomeetria
- `LD:Indeks:Punane:Roheline:Sinine\r\n` – üksiku valgusti muutmine
- `LS:Algus:Lõpp:Värv1:Värv2:...:Värv n\r\n` – valitud segmendi valgustite väärtuste muutmine
- `LM:Režiim:Parameeter 1:Parameeter 2:Parameeter 3:...:Parameeter n\r\n` – valgustuse režiimide muutmine

STM32 mikrokontrolleri püsivara juhib järgmisi komponente:

- **Mootorid** - Mootori draiver arvutab mootori juhtsignaali täiteteguri ja saadab igale mootorile õige PWM signaali [12]. PID kontroller reguleerib mootori täitetegurit [12].
- **LED-id** - Roboti trükkplaadil on 60 adresseeritavat LED valgustit, mis on ühendatud STM32 külge [13]. Valgusteid juhitakse läbi püsivara [13].
- **OLED-ekraan, koodernupp ja kasutajaliides** - STM32 mikrokontrolleriga on ühendatud SSD1306 ekraanimoodul ja kvadratuurkoodernupp [14]. Robotondile on loodud kasutades neid komponente kasutajaliides [14], vt joonis 2.3.
- **Sensorid ja toitehaldus** - STM32 mikrokontrolleriga on ühendatud ATtiny88 mikrokontroller, mis teostab vajalikud toitesüsteemi mõõtmised ja saadab tulemused põhiprotsessorisse (STM32) [15]. Edasi saadetakse tulemused pardaarvutisse(ROS-sõlme) ja kasutajaliidesesse [15].



Joonis 2.3: Robotondi kasutajaliides, kollase noolega on tähistatud OLED-ekraan, rohelise noolega koodernupp.

2.2 Robot Operating System (ROS) ja Robotont platvorm

ROS on saanud robootikatarkvara arenduses standardiks [4]. Tegemist on avatud lähtekoodiga tarkvararaamistiku ja tööriistakoguga, mis on laialdaselt kasutusel nii teaduslikus uurimistöös kui ka tööstuslikes rakendustes [4]. ROS-i eesmärk on pakkuda ühtset platvormi, mis hõlbustaks robootika tarkvara koostalitlust [3]. Selle modulaarne arhitektuur võimaldab jagada robotisüsteemi funktsionaalsused eraldi sõlmedeks, mis suhtlevad omavahel kindlate liideste kaudu (sõnumid rubriikidel ehk *topics*, teenused, jms) [3]. See võimaldab arendajatel luua taaskasutatavaid komponente ning kombineerida neid komplekssemate süsteemide ülesehitamiseks. ROS-is hoitakse funktsionaalsus modulaarsena, paigutades algoritmid ja teegid eraldi mugavalt paigaldatavatesse kimpudesse, mida kasutatakse erinevate ülesannete jaoks, alates roboti andurite draiveritest kuni kõrgemate algoritmideni nagu lokaliseerimine, navigeerimine ja masinnägemine. ROS-i ökosüsteem on aastatega kasvanud ulatuslikuks ja aktiivseks, tuhanded kasutajad panustavad uute pakettide loomisse ning olemasolevate hooldusesse, mis tähendab, et praktiliselt iga tavapärase robootikaprobleemi jaoks leidub juba aluslahendus või näidis ROS-i kujul. Robotont platvorm kasutab seda ökosüsteemi ära, pakkudes riistvara, mis toetab ROS-i. See tähendab, et Robotondil on olemas vajalikud liidesed ja konfiguratsioonid, et ROS-i sõlmed saaksid juhtida roboti mootoreid, lugeda andureid ja kõike muud vajalikku [7]. Robotont 3 puhul on arendamisel ka ROS 2 tugi, arvestades üldist suundumust robootikamaailmas liikuda ROS 2 suunas [7].

2.3 Docker ja konteinertehnoloogia robootikas

Docker on tänapäeval üks populaarsemaid konteinertarkvara platvorme. Konteiner on isoleeritud keskkond, mis sisaldab kõike vajalikku rakenduse juurutamiseks: käituskeskkonda, operatsioonisüsteemi tööriistu, vajalikke teeke, konfiguratsioone ja rakendust ennast [9]. Konteinerid jagavad operatsioonisüsteemiga kernelit, mis teeb nende ressursikasutuse virtuaalmasinatega võrreldes väiksemaks [9]. Docker pakub lihtsat vahendit konteinerite loomiseks läbi Dockerfile spetsifikatsiooni ja käivitamiseks. 2015. aastal loodi Open Source Robotic Foundation'i poolt ametlikud ROS Docker tõmmised Docker Hub keskkonda [16]. Need sisaldavad kõiki standardseid ROS-i distributsioone (melodic, noetic, foxy, humble, jazzy jne) ning võimaldavad arendajatel kiiresti saada puhas ROS keskkond, ilma et peaks ROS-i käsitsi paigaldama [8, 17, 18].

Robotonti kasutades on loodud 2022. aastal veebipõhine õppe- ja tarkvaraarenduse keskkond ROS-i õppimiseks füüsiliste robotitega üle veebi [19]. See keskkond kasutab Dockeri konteinereid, VNC-d ja brauseris töötava broneerimissüsteemi abil eelkonfigureeritud Ubuntu ja ROS keskkondi. [20]

Konteineripõhine robootika on muutumas üha aktuaalsemaks teemaks. 2017. aastal ilmus raamat "Robot Operating System (ROS): The Complete Reference (Volume 2)", mille peatükis "ROS and Docker" käsitleti põhjalikult konteinerite kasutamist ROS-i arenduse korduvuse ja juurutamise huvides [16]. Teaduskirjanduses on hakatud konteinerite kasutust analüüsima nii tarkvaraarenduse seisukohast (nt Malhotra ja kolleegide süstemaatiline ülevaade konteinerite hoolduse väljakutsetest [21]) kui ka konkreetsemalt robootika tarbeks. Melo jt tutvustasid 2022. aastal konteineripõhist raamistikku ROS rakenduste arenduse toetamiseks, mis hõlmab kogu arendustsüklit alates simulatsioonist kuni lõpliku juurutamiseni [22]. See viitab sellele, et konteinerite kasutamine robootikas pole vaid lahendus mingitele spetsiifilistele probleemidele, vaid liigub süstemaatilisema kasutuse suunas.

Praktilisel tasandil on robootikakogukonnas mitmeid tööriistu ja projekte, mis Dockerit ära kasutavad. Näiteks on olemas tööriistad nagu *rocker*, mis laiendavad Dockerit robootika tarbeks, nt lihtsustades GUI rakenduste käivitamist konteinerist, X11 edastust jms [23, 24]. Mitmed robootika platvormid on hakanud pakkuma oma tarkvara konteineritena. Näiteks Clearpath Robotics jagab mõnede oma robotite jaoks Dockeri tõmmiseid, et kasutajad saaksid kiirelt alustada [25, 26].

Google Cloud Robotics Core on Kubernetesel põhinev platvorm, mis toetab ROS-i ja mida kasutatakse robotifarmide haldamiseks ning neis rakenduste jagamiseks [27]. See võimaldab lihtsamaid tarkvarauuendusi ning turvalist kahepoolset suhtlust pilvega [27]. KubeROS pakub ühtset platvormi ROS 2 rakenduste automaatseks juurutamiseks, kasutades Kubernetes't [28].

Samuti võimaldab KubeROS ressurside hajutamist [28].

Lisaks Dockerile ja Kubernetesele kasutatakse ka teisi konteineritel põhinevaid lähenemisi. *Snap* on *Canonical*-i välja töötatud pakendussüsteem, mida kasutatakse ka ROS-i komponentide levitamisel [29]. Teaduskeskkondades kasutatakse ka Singularity konteinereid, näiteks mõnes akadeemilises robootikakursuses on eelistatud Dockerile Singularity, kuna seda on lihtsam kasutada, sest puuduvad erinevalt Dockerist eraldi konteinerid ja tõmmised ning see sobitub hästi ülikoolide arvutiklastritesse [30]. Siiski on Docker tänu oma laiale tööstuslikule kasutusele ja tööriistade ökosüsteemile haridusrobotite jaoks sobiv ja hästi toetatud valik.

3 Ülevaade probleemidest

Robotont 3 arhitektuuri kõrgeim tase on Intel NUC pardaarvuti Ubuntu 22.04 operatsioonisüsteemiga ning ROS tarkvaraga [7]. Pardaarvuti suhtleb üle jadaliidese emaplaadiga, mis juhib roboti riistvara (mootorid, andurid) [7]. Kasutajad saavad roboti riistvara ise valmistada (3D-prinditav kere, avalikud joonised) ning tarkvara osas kasutada laialt levinud ROS ökosüsteemi [7]. See tähendab, et Robotonti saab kasutada väga erinevate rakenduste käitamiseks, alates lihtsast kaugjuhitavast sõidukist kuni autonoomsete navigatsioonialgoritmideni. Praktikas on aga ühe roboti raames erinevate rakenduste vahel vahetamine ja haldamine osutunud keeruliseks. Näiteks tudengite praktikumides või avalikel demonstratsioonidel võib ühel päeval olla vaja Robotonti kasutada kaugjuhtimise harjutuseks, teisel päeval *SLAM*-algoritmi testiks ja kolmandal päeval hoopis uue andurisüsteemi katsetamiseks. Iga selline rakendus eeldab spetsiifilist tarkvaralist seadistust (erinevad ROS-i paketid, konfiguratsioonid, vahel isegi erinev ROS-i versioon). Nende kõikide hoidmine ja haldamine ühes süsteemis tekitab mitmeid probleeme, mida käsitleme järgnevalt.

3.1 Arenduskeskkondade eraldamine

Erinevate robootikaprojektide paralleelne arendamine ilma keskkondade eraldamiseta on keerukas, sest eri projektide sõltuvused võivad olla vastuolus ning ühe masinaga mitme projekti kallal töötamine muutub ebapraktiliseks. Eriti teravalt ilmneb probleem siis, kui on vaja korraga kasutada mitut erinevat ROS-i distributsiooni (nt *Jazzy*, *Humble* või *Melodic*) või koguni eri versioone nagu ROS 1 ja ROS 2 [8]. Iga ROS-i distributsioon on ametlikult toetatud vaid kindlatel Ubuntu versioonidel. *Noetic* on toetatud Ubuntu 20.04 operatsioonisüsteemil, *Jazzy* aga Ubuntu 22.04 peal. Seetõttu võib tähendada mitme distributsiooni samaaegne paigaldamine ühte masinasse kas eraldi virtuaalkeskkondade seadistamist või lausa *dualboot*-i kasutuselevõttu. Iga kord, kui algab uus projekt või on vaja Robotont ümber häälestada mõneks teiseks ülesandeks, kulub märkimisväärselt aega süsteemi seadistamisele. Tuleb paigaldada vajalikud tarkvarapaketid, seadistada konfiguratsioonifailid, lahendada sõltuvuste konfliktid jms. Arendajad kulutavad aega keskkonna ettevalmistamiseks selle asemel, et keskenduda sisulisele probleemile. Lisaks kaasneb risk, et vale seadistus või teegi versioon võib tekitada ettearvamatuid vigu, mille põhjust on keeruline diagnoosida. Praegune olukord, kus keskkonna taastamine tähendab *dualboot*i kasutamist või ROS-i uuesti paigaldamist, pole efektiivne.

3.2 Tarkvarakeskkonna jagamine ja taaskasutus

Haridusliku kontekstis soovitakse tihti, et mitu tudengit saaksid töötada sama robotiga eri aegadel või et üks tudeng saaks oma arendatud lahenduse lihtsalt teisele üle anda. Kui Robotondil on üht projekti toetav keskkond paigas, siis selle asendamine teise tudengi projektiga võib nõuda keerukat seadistamist. Samuti, kui midagi läheb valesti (nt tudeng muudab süsteemis mõnd olulist seadet või paigaldab vale versiooniga tarkvara), võib roboti taastamine töökorda olla aeganõudev. Töökeskkonna reprodutseerimine on keerukas ja aeganõudev isegi kui dokumentatsioon on olemas. Kellegi teise seadistuste käsitsi läbi tegemine on aeganõudev ja erinevate vigade tekkimise tõenäosus on suur. See vähendab Robotondi kui õppeplatvormi kasutusmugavust.

3.3 Kasutajaliideste puudujäägid

Tüüpilised Robotondi kasutajad on tudengid, gümnaasiumiõpilased ja ka demonstratsioonide publik. Nende eesmärk ei ole süveneda Linuxi pakihalduritesse või ROS-i versioonidesse, vaid panna robot tegema mingit kindlat tegevust. Seda on kõige mugavam teha kasutajaliidese abil, aga hetkel ükski Robotondi kasutajaliides seda ei võimalda.

3.4 Probleemi kokkuvõtteks

Robotont 3 praegune tarkvaraline seadistus ei toeta mugavalt mitme erineva ROS-põhise rakenduse haldamist. On keskkondade konflikti oht, ümberseadistamine võtab palju aega ning olemasolevatel kasutajaliidestel pole hetkel võimekust vahetada roboti funktsiooni. Dockeri konteineritel põhinev lahendus võimaldab potentsiaalselt need kitsaskohad kõrvaldada, pakkudes standardsel viisil pakendatud, kiiresti vahetatavaid ja taastatavaid töökeskkondi. Järgnevalt sõnastatakse täpsemalt käesoleva töö eesmärgid ja antakse lühiülevaade lahenduse ideest.

4 Töö eesmärgid ja nõuded

Selle bakalaureusetöö eesmärk on arendada välja Dockeri konteineritel põhinev haldustarkvara Robotont 3 õpperobotile, mis lahendaks eelpool kirjeldatud tarkvarakeskkonna haldamise probleemid. Eesmärgi saavutamiseks on püstitatud järgmised alamülesanded:

1. **Dockeri põhise süsteemi ülesseadmine Robotondil:** Robotont 3 pardaarvutile tuleb paigaldada Dockeri mootor ja luua erinevate roboti funktsioonide jaoks kõik vajalik Dockeri konteinerite loomiseks. Tuleb luua Pythoni teenus, mis saab sisendiks juhiseid(püsivarast või veebiliidesest), mille peale käivitatakse või peatatakse konteinereid. Samuti peab see teenus vahendama jadaliidese kaudu suhtlust püsivara ja konteineri vahel ning vahetama andmeid püsivaras implementeeritava menüüga.

2. **Püsivaras alammenüü loomine konteinerite haldamiseks:** Luua püsivaras alammenüü, kuhu dünaamiliselt lisatakse kõik saadavalolevad keskkonnad (nt *ros2_jazzy_teleop*, *ros1_melodic_teleop*). Alammenüü peaks näitama saadavalolevate keskkondade nimekirja ning andma võimaluse keskkonnale vastavat konteinerit käivitada või peatada.

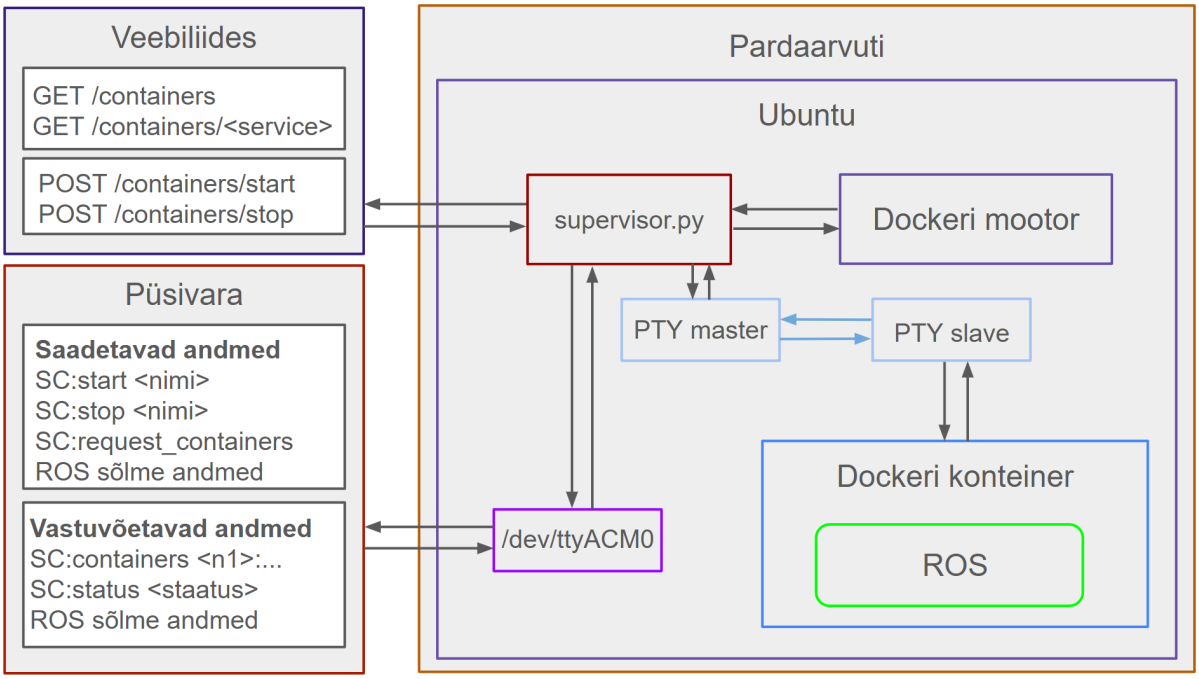
3. **Veebiliidese loomine konteinerite haldamiseks:** Luua vahendid üle võrgu konteinerite käivitamiseks ja peatamiseks. See tähendab, et lisaks robotil olevale kasutajaliidesele peaks olema võimalik eemalt (näiteks sülearvutist üle WiFi või kohtvõrgu) valida mis keskkonnale vastavat konteinerit käivitada või peatada.

Töö käigus realiseeritakse ülaltoodud eesmärgid Robotont 3 riistvaral.

5 Lahendus

5.1 Süsteemi arhitektuuri ülevaade

Loodud konteineripõhine haldustarkvara süsteem koosneb mitmest omavahel suhtlevast komponendist, mille üldine arhitektuur on näidatud alloleval joonisel 5.1.



Joonis 5.1: Arhitektuuri ülevaade

Käesolev peatükk annab ülevaate süsteemi tehnilisest arhitektuurist.

1. Käsitletakse Docker konteinerite loomist ehk kuidas Dockerfile'i põhjal luuakse tõmmis(*image*) ja sellest käivitatakse konteiner ning milline on arendustöövoog.

2. Selgitatakse püsivara muudatusi, millega loodi dünaamiline konteinerite valiku alammenüü roboti püsivaras.

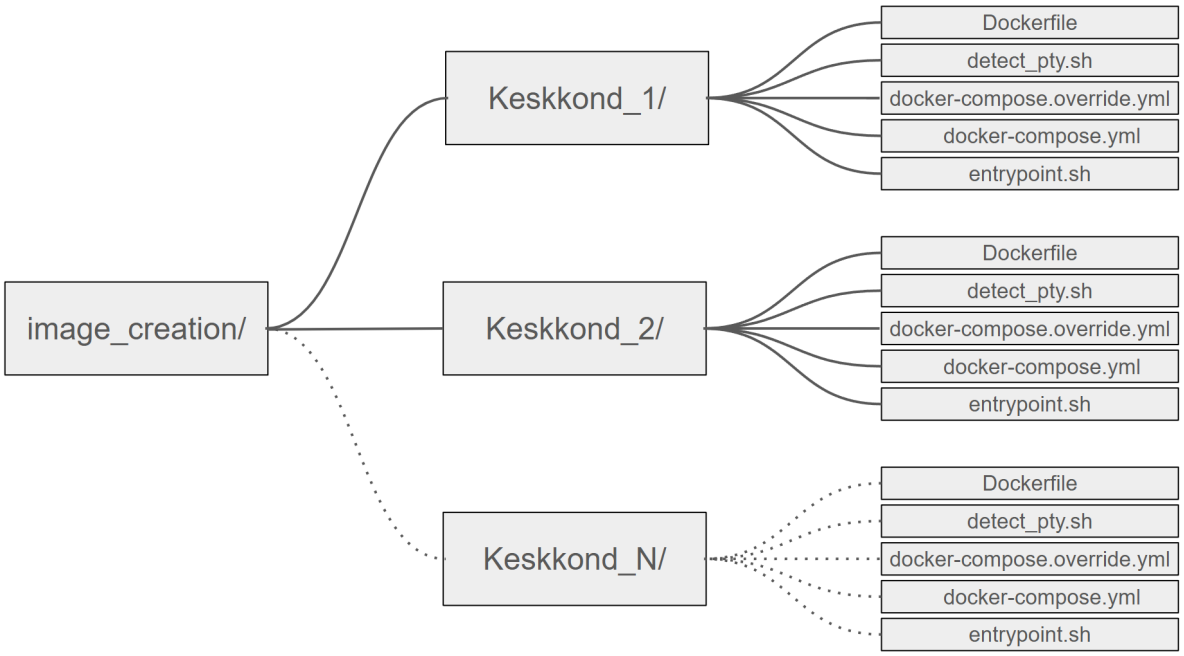
3. Kirjeldatakse jadaliidese (*serial*) realiseerimist roboti püsivara ja konteineris töötava tarkvara vahel pseudoterminalide (PTY) abil ning supervisor.py teenuse rolli andmeside vahendamisel.

4. Käsitletakse konteinerite käivitamist Docker Compose'i abil ning kuidas kasutatakse `docker-compose.override.yml` faili, et siduda õiged pseudoterminalid konteineritega, ning `detect_pty.sh` skripti toimimist koos superviisoriga.

5. Tutvustatakse Flask-raamistikul põhinevat veebiliidest konteinerite haldamiseks, kirjeldades vastavaid API lõpppunkte ning kliendipoolset rakendust, mis lubavad kasutajatel konteinerite käivitamist ja peatamist mugavas veebikeskkonnas.

5.2 Dockeri konteinerite loomine

Failistruktuuri mõistes on konteinerite seadistused hoitud korrastatult. Versioonihaldustarkvara Git vahendusel on iga konteinerikeskkond eraldi kaustas koos oma Dockerfile'i ja konfiguratsioonifailidega (vt joonis 5.2).



Joonis 5.2: Failistruktuur

Tüüpiline konteineri loomise ja käivitamise töövoog on järgmine:

1. **Dockerfile'i kirjutamine:** arendaja loob või uuendab Dockerfile'i, mis kirjeldab konteinerikeskkonna sõltuvused ja rakenduse paigaldamise sammud.

2. **Tõmmiste loomine:** pardaarvutis käivitatakse Docker *build*-protsess (näiteks käsuga `docker build` või Docker Compose abil) vastava Dockerfile'i alusel. Selle tulemusena luuakse lokaalsesse Dockeri süsteemi uus tõmmis. Tõmmis sisaldab kogu konteineris vajaminevat tarkvara.

3. **Konteineri käivitamine:** valmis tõmmisest käivitatakse konteiner. Kasutame selleks Docker Compose faili ja käsuga `docker compose up -d` antud konteineri kaustas käivitatakse vastav konteiner. Docker Compose fail määrab, millised ressursid (võrgud, kaustad või seadmed) konteineriga seotakse. Kuna tõmmist pole hoidlast vaja alla laadida, võetakse varasemalt loodud lokaalne tõmmis otse kasutusele. Konteiner käivitub ning alustab ROS-sõlme või muu rakenduse tööd.

4. **Töö ja peatamine:** konteiner töötab isoleeritult, pakkudes näiteks roboti juhtimise teenust. Vajadusel saab konteineri peatada kas käsitsi (`docker down`) või läbi loodud kasutajaliideste ning ka uuesti käivitada. Halduse teeb mugavaks see, et nii püsivara kui ka veebiliides saavad konteinerite olekut jälgida ja anda käske nende juhtimiseks (sellest täpsemalt peatükkides 5.4 ja 5.6).

Sedalaadi töövoog tagab, et iga tarkvarakomponent jookseb oma konteineris, vältides varasemalt kirjeldatud probleeme. Kuna tõmmiste loomine toimub pardaarvutis, on tagatud, et konteiner sobitub täpselt roboti keskkonda (näiteks õige riistvaraarhitektuur, seaded). Alternatiivina saaks tõmmiseid ette valmistada serveris ja hoida tõmmiste hoidlas, aga seda käesolev süsteem veel ei rakenda.

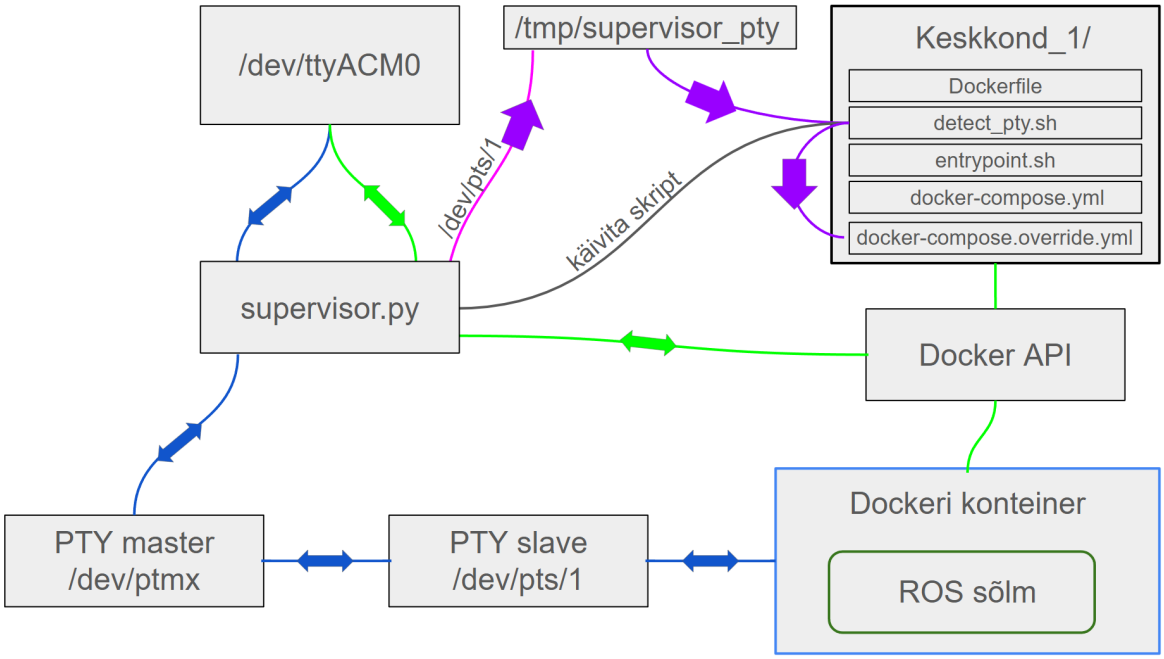
5.3 Jadaühendus roboti ja konteineri vahel

Roboti mikrokontroller ja pardaarvuti tarkvara suhtlevad omavahel jadaliidese kaudu. Mikrokontroller on ühendatud USB kaabli abil pardaarvutiga ja on nähtav pardaarvutile seadmena `/dev/ttyACM0`. Selle ühenduse kaudu liiguvad andmed kahes suunas: sensorite ja mootori andmed püsivarast pardaarvutisse ning roboti juhtkäsklused pardaarvutist tagasi mikrokontrollerisse. Tavaolukorras võiks üks programm avada selle seadme ja suhelda otse püsivaraga. Meie süsteemis on aga olukord keerukam. Jadaühendust ühe seadmega peavad jagama korraga kaks osapoolt: konteineris töötav rakendus (nt ROS sõlm, mis suhtleb roboti riistvaraga) ning konteinereid haldav supervisor.py.

Linuxi keskkonnas ei saa kaks protsessi korraga ühte seadet kirjutamiseks avada, seega vajasime lahendust, mis võimaldaks sõnumeid vahendada. Lahenduseks on pseudoterminali kasutamine. Pseudoterminali kasutamise idee on pärit Andres Saki tööst „Avatud robotplatvormi Robotont 3 kasutajaliidese väljatöötamine“ [14]. Pseudoterminal (PTY) on paar virtuaalseid seadmefaile (nt `/dev/ptmx` ja `/dev/pts/1`), mille abil saab tekitada asünkroonse kahesuunalise suhtluse [31]. *Master* pool (`/dev/ptmx`) on mõeldud kontrollivale protsessile, mis loeb ja kirjutab andmeid. *Slave* pool käitub kui tavaline jadaporti ühendatud seade teisele protsessile. Meie süsteemis töötab `supervisor.py` *master* poolega ning konteineriga ühendatakse *slave* pool, seejuures konteineri poolelt vaadates oleks justkui otseühendus püsivaraga.

Käime läbi `supervisor.py` peamised funktsioonid, vt joonist 5.3:

- `supervisor.py` loob käivitumisel uue PTY paari ning avab neist *master* poole enda jaoks. *Slave* poole failitee (nt `/dev/pts/5`) edastatakse Docker Compose faili.
- supervisor.py avab jadapordi(`/dev/ttyACM0`) ja kuulab sealt tulevaid andmeid. Kõik saabunud andmepaketid analüüsitakse funktsioonis `filter_and_process_data()`. Kui tuvastatakse, et tegu on SC(*supervisor command*) käsuga, mida püsivara on saatnud, siis filtreeritakse see teiste andmete hulgast välja ning seda ei saadeta konteinerile edasi. Selle asemel töötleb supervisor ise selle käsu. Näiteks võib püsivara saata käsu `SC:start ros2_jazzy_driver`, mis annab teada, et kasutaja valis konteineri `ros2_jazzy_driver` käivitamise.
- `supervisor.py` tõlgendab seda ning käivitab vastava konteineri kasutades Docker API-d.
- Samuti võib püsivara saata päringuid (nt `SC:request_containers`), mille peale supervisor vastab nimekirjaga konteineritest, saates tagasi üle seriali eriformaadis andmed, mida püsivaras `cmd.c` käsitleb ja menüüsse kuvab. Kõik sellised SC käsud on spetsiifilised haldussõnumid, mis on meie protokolli osa.
- Kui saabuv sõnum ei ole SC-käsk (näiteks odomeetria andmed, sensorite väärtused vms), siis `supervisor.py` edasi midagi ise ei tee peale andmete vahendamise. Need andmed kirjutatakse PTY *master* poolde, seejärel ilmuvad need konteineri jaoks PTY *slave* poolel (konteineri jaoks oma `/dev/ttyACM0` sisse).
- Samamoodi, kui konteineripoolsest otsast (PTY *slave*) tuleb andmeid (nt konteineris töötav ROS sõlm saadab mootori kiiruste käske püsivarale), loeb supervisor.py need oma *master* otsast ja edastab muutmata kujul välja mikrokontrolleriga ühendatud füüsilisse jadaporti.



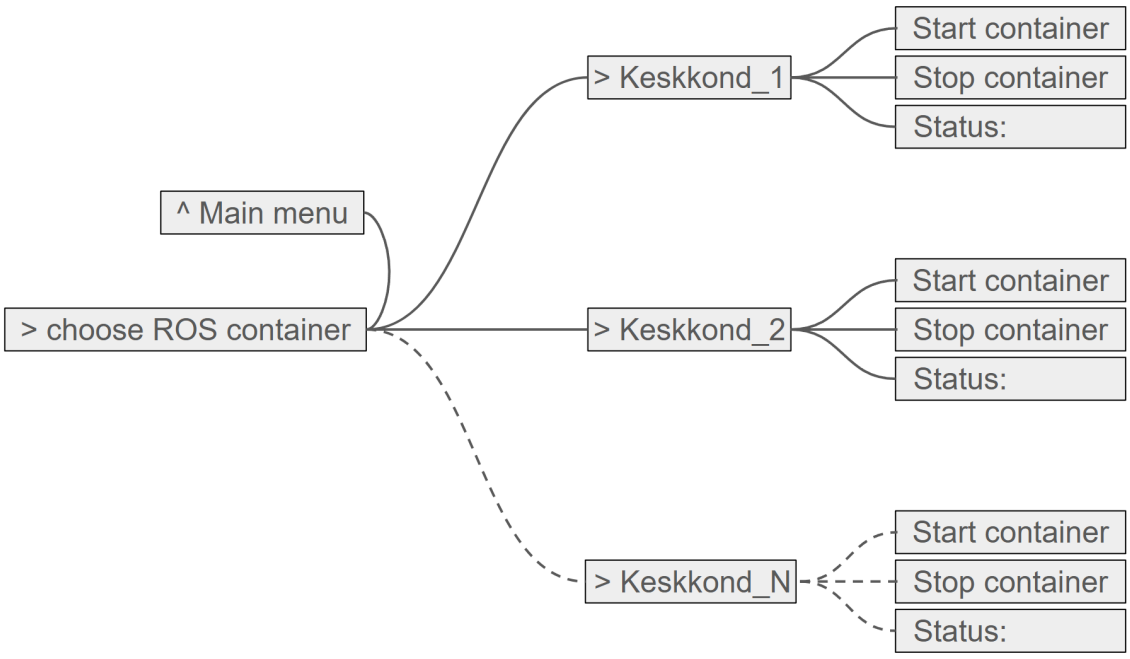
Joonis 5.3: Andmete liikumine läbi `supervisor.py`

Seega käitub `supervisor.py` vahendajana, mis vajadusel tõlgib või filtreerib, aga enamasti edastab andmeid. Selline arhitektuur võimaldab nii konteineril kui ka püsivaral näha justkui otsest jadaühendust, kuigi tegelikkuses on vahel vahendav tarkvarakiht. Pseudoterminali kasutus oli hädavajalik, et jagada üks füüsiline jadaliides mitme protsessi vahel. Ilma selleta poleks konteiner (ROS sõlm) saanud samaaegselt suhelda mikrokontrolleriga nii, et ka püsivara halduskäsud (nagu konteinerite käivitamine) kohale jõuaksid.

5.4 Alammenüü loomine püsivaras ja muudatused andmeedastuse loogikas

Roboti emaplaat on varustatud SSD1306 OLED-ekraani ja koodernupuga, mis võimaldavad kasutajale kuvada menüüsid ja võtta vastu käske. Konteinerite haldamise tarbeks lisati püsivarasse uus alammenüü, mis loetleb pardaarvutis saadaval olevad konteinerkeskkonnad ning näitab nende staatust (nt kas konteiner on hetkel käivitatud või peatatud), vt joonist 5.5. Kasutaja saab menüü kaudu valida konteineri ja anda käsu selle peatamiseks või käivitamiseks ilma pardaarvuti käsureale ligipääsuta.

Tehniliselt nõudis see püsivara koodi muutmist kahes kohas: menüü struktuuri haldavas failis `menu.c` ja käsuliidese failis `cmd.c`. Failis `menu.c` lisati juba olemasolevale alammenüü kirjele *Choose ROS container* sisu, mille alla kuuluvad dünaamilised alamkirjed iga konteinerikeskkonna jaoks, vt joonist 5.4. Erinevalt staatilistest menüüelementidest pidi konteinerite alamvalik olema loodav käigupealt, ehk selle sisu (konteinerite nimed) muutuvad sõltuvalt sellest, millised konteinerid on süsteemis olemas. `menu.c` koodis rakendati

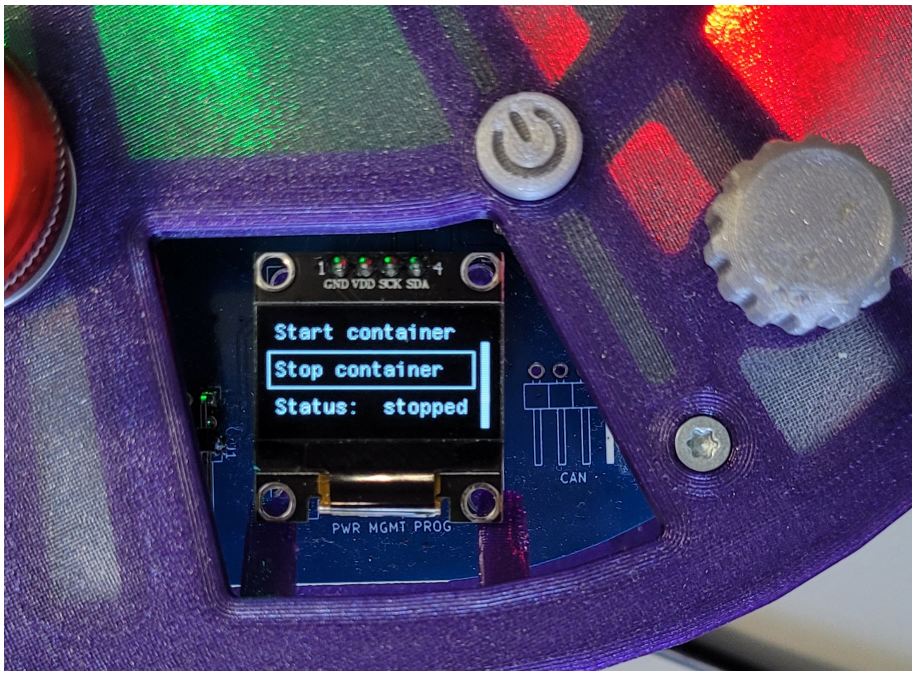


Joonis 5.4: Alammenüü struktuur

mehhanism, kus konteinerite menüü loetakse kokku andmestruktuurist, mida püsivara uuendab vastavalt pardaarvutist saadud infole, vt tabelit 5.1. OLED-ekraanil kuvatakse iga leitud konteinerikeskkonna nimi, millele vajutades sisenetakse omakorda alammenüüsse. Selles menüüs on kirjed "Start container", "Stop container"koos konteineri olekuga, mis vastavalt olukorrale on kas"running", "starting", "stopping", "stopped"või "error". Kui konteinerite loend on pikem kui ekraanile mahub, võimaldab menüüs liikumine kõiki kirjeid kerida.

Sõnumi vorm	**Kirjeldus**
Püsivara → Supervisor	
`SC:start <nimi>`	Käivita konteiner `<nimi>`.
`SC:stop <nimi>`	Peata konteiner `<nimi>`.
`SC:status <nimi>`	Päring: anna valitud konteineri staatus.
`SC:request_containers`	Päring: saada konteinerite loend.
Supervisor → Püsivara	
`SC:containers clear`	Alusta loendi saatmist.
`SC:containers add <nimi>`	Lisa üks konteiner loendisse.
`SC:containers done`	Loendi lõpp.
`SC:status <staatus>`	Vastus konkreetse konteineri staatuse päringule (*running*, *stopped*, *error*).
`SC:status <staatus>`	Üldine olekuteavitus (saadetakse ilma pärimata kui toimub mingi tegevus, nt *starting*, *stopping*).

Tabel 5.1: Püsivara ja `supervisor.py` vahelise suhtlusprotokolli põhisõnumid



Joonis 5.5: Loodud konteineri käivitamise ja peatamise alammenüü püsivaras.

5.5 Docker Compose kasutus

Konteinerite käivitamise hõlbustamiseks kasutatakse Docker Compose tööriista. Iga konteinerikeskkonna jaoks on koostatud Docker Compose fail (`docker-compose.yml`), mis kirjeldab konteineri käivitamise parameetreid: milline tõmmis käivitada, millised seadmed, pordid ja köited (*volume*) ühendada, keskkonnamuutujad jne. Meie juhtumil on oluline, et konteinerisse ühendataks korrektne seadmefail roboti mikrokontrolleri jaoks. Nagu eelnevalt kirjeldatud, peab iga konteiner kasutama `supervisor.py` poolt loodud pseudoterminali *slave* poolt.

Igal konteinerikeskkonnal on lisaks põhilisele `docker-compose.yml` failile ka `docker-compose.override.yml` fail. `docker-compose.override.yml` fail saab üle kirjutada või täiendada `docker-compose.yml` failis defineeritud seadeid [32]. `docker-compose.yml` faili kasutatakse kasutajapoolsete seadistuste tegemiseks.

Süsteemi ülesseadmise käigus seadistatakse *host*-süsteemi SSH port 22222-ks ja `docker-compose.yml` failis märgitakse `ports: - "22:22"`. Nii edastatakse konteineri port 22 kohalikku võrku ja luues *host*-masina IP-ga SSH ühenduse, ühendatakse hoopis konteinerisse. See on arendajale mugavam olukordades, kus tööd tehakse konteineri siseselt.

`supervisor.py` kirjutab õige pseudoterminali failitee `/tmp/supervisor_pty` faili. `detect_pty.sh` skripti käivitamisel lisatakse `docker-compose.override.yml`

faili `/tmp/supervisor_pty`-s määratud failitee. See skript käivitatakse vahetult enne konteineri käivitamist `supervisor.py` poolt. See mehhanism on oluline, sest pseudoterminali nimi võib süsteemis muutuda igal käivitusel või sõltuvalt teiste pseudoterminalide olemasolust.

Kui `docker-compose.override.yml` on nõuetekohaselt uuendatud, käivitatakse konteiner Docker Compose abil. Nüüd saab konteiner mikrokontrolleriga suhelda läbi pseudoterminali silla, nagu oleks tegemist otseühendusega. Uue keskkonna lisamisel piisab tema kaustas Dockerfile'i ja Docker Compose failide loomisest ning `supervisor.py` teab, kuidas seda käivitada ja siduda õige pseudoterminaliga.

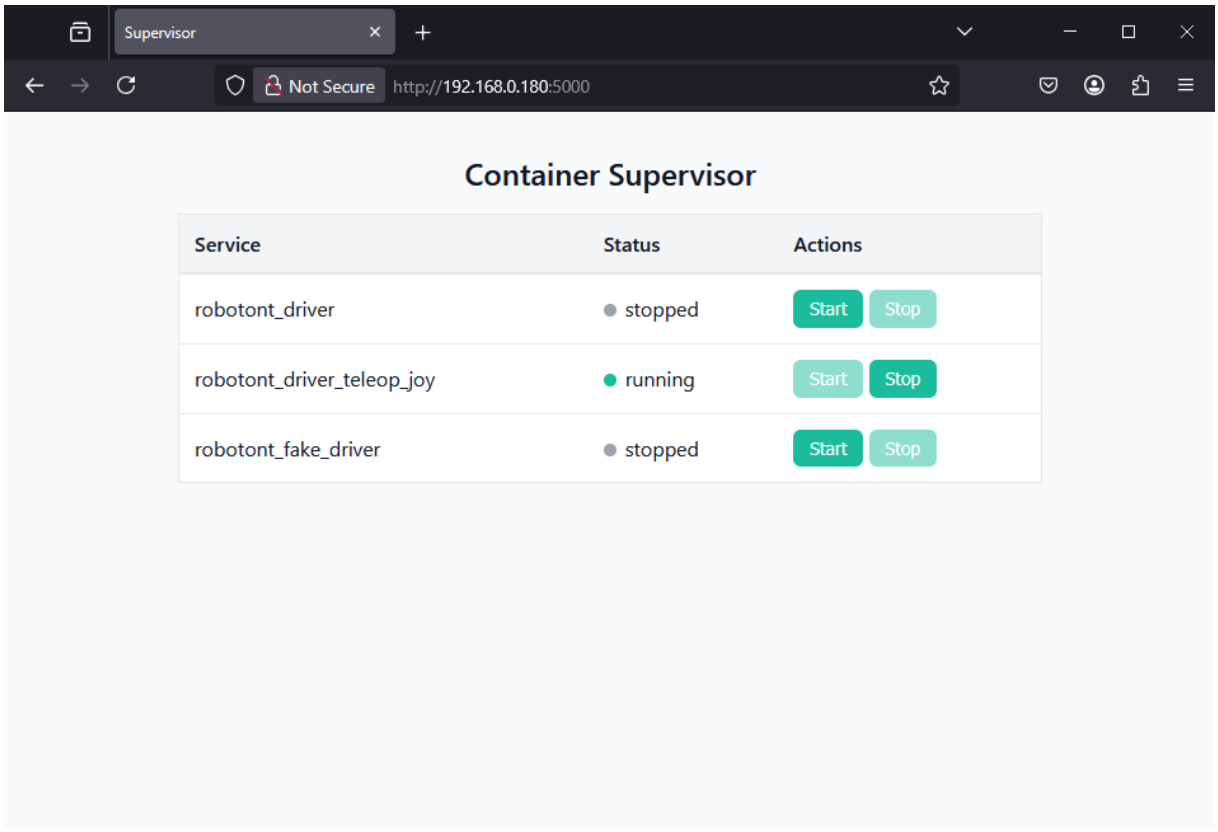
5.6 Veebiliides konteinerite haldamiseks

Lisaks püsivaras tehtud menüüle on konteinerite haldamiseks loodud ka lihtne veebiliides, vt jooniseid 5.6 ja 5.7. `supervisor.py` sisaldab sisseehitatud *Flask* raamistiku põhjal töötavat veebiserverit, mis pakub REST-stiilis API lõpp-punkte konteinerite info pärimiseks ja juhtkäskude edastamiseks. Nii saab kasutaja ühendada tavalise veebibrauseriga roboti pardaarvutis oleva veebiserveri külge, vaadata konteinerite olekuid ning käivitada või peatada neid mugava graafilise liidese kaudu. Flaski veebiserveri REST API lõpp-punktid on defineeritud allolevas tabelis 5.2.

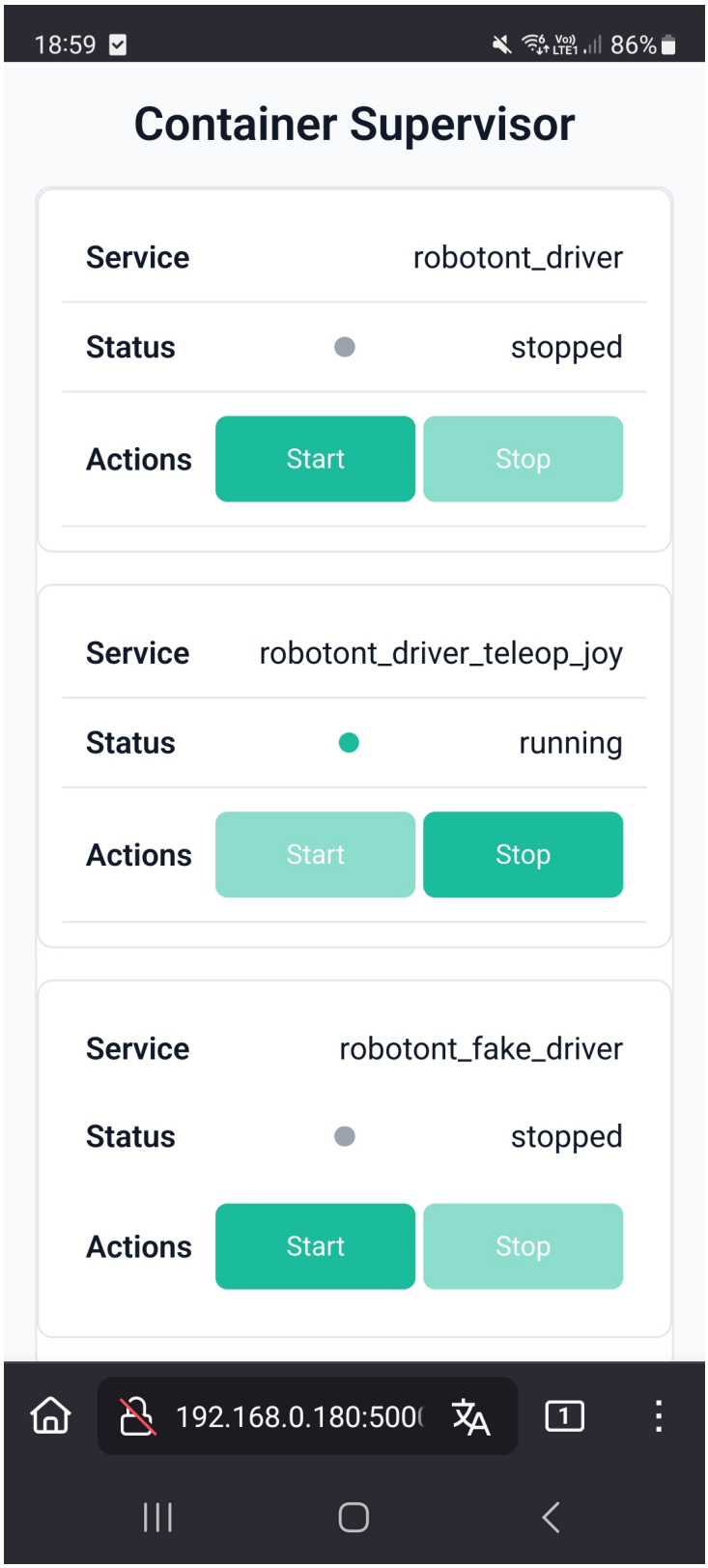
HTTP-meetod	Lõpp-punkt	Kirjeldus
GET	`/containers`	Tagastab JSON-formaadis nimekirja kõigist konteineritest koos nende hetkestaatusedega (nt *running*, *stopped*).
POST	`/containers/start`	Käivitab JSON-päringu väljal `name` määratud teenuse konteineri.
POST	`/containers/stop`	Peatab JSON-päringu väljal `name` määratud teenuse konteineri.

Tabel 5.2: `supervisor.py` pakutavad REST-API lõpp-punktid konteinerite haldamiseks

`supervisor.py` seob tabelis 5.2 olevad HTTP-päringud sama loogikaga, mida kasutatakse püsivara SC käskude töötlemisel



Joonis 5.6: Veebiliides arvuti brauseris



Joonis 5.7: Veebiliides nutitelefoni brauseris

6 Lahenduse testimine

6.1 Roboti kasutajaliidese alammenüü

Roboti kasutajaliidese alammenüü struktuur (vt joonist 5.4) on intuitiivne ja seda on mugav kasutada. Konteineriga mingit tegevust tehes saab kohest visuaalset tagasisidet, seega ei teki olukorda, kus ollakse ebakindel, kas nupu vajutamine toimis või mitte, vt joonist 5.5. Samuti ilmuvad uued konteinerid menüüsse automaatselt, kui need lisatakse süsteemi.

6.2 Veebiliides

Veebiliidese ülesehitus on intuitiivne ja ülevaatlik, mistõttu on seda mugav kasutada. Olukordades, kus tegutsetakse arvutis (vt joonist 5.6) või nutitelefonis (vt joonist 5.7), võib olla veebiliidese kasutamine mugavam kui roboti kasutajaliidese kasutamine. Sarnaselt roboti kasutajaliidesele, antakse ka veebiliideses mingi tegevuse korral koheselt visuaalset tagasisidet kasutajale ja uued konteinerid ilmuvad loendisse automaatselt.

6.3 Mitme ROS keskkonna paralleelne tugi

Robotont 3 suudab nüüd käitada erinevaid ROS-põhiseid rakendusi eraldatud konteinerites.

Testimise käigus loodi kolm näidiskonteinerit:

1. `ros2_jazzy_driver` võimaldas siluda ja testida suhtlust roboti ja konteineri vahel. Konteiner põhineb Ubuntu 24.04 tõmmisel, millele lisatakse kihtidena vajalikud konfiguratsioonid, tarkvarapaketid, sõltuvused ja repositooriumid. Seejärel ehitatakse lähtekoodist draiver (`robotont_driver`), seadistatakse SSH server ja määratakse `docker-compose.yml` failis teenus, mida antud konteiner jooksutab oma eluea jooksul.

2. `ros2_jazzy_teleop` lisas draiverile kõrvale sellega suhtleva ROS 2 Jazzy kaugjuhtimise sõlme (`demo_teleop`), mis saab kasutajalt sisendeid ja juhib sisendite põhjal robotit.

3. `ros1_melodic_teleop` on eelneva näidiskonteineriga sama funktsionaalsusega, aga on ROS 1 Melodic põhine. Kasutati Ubuntu 18.04 tõmmist, sest seda toetab ROS 1 Melodic. Samuti olid kõik lisatud konfiguratsioonid, tarkvarapaketid, sõltuvused ja repositooriumid ROS 1 Melodic põhised.

Katsetused näitasid, et kõiki neid keskkondi sai ühe ja sama Robotont 3 riistvara peal jooksutada, lülitudes nende vahel mõne käsuga või kasutades loodud kasutajaliideseid. Kui üks konteiner peatati ja teine käivitati, siis keskkondade vahetamiseks kuluv aeg oli ligikaudu 1-20 sekundit kui tõmmis oli juba vahemälus. Kui seda polnud vahemälus, siis tõmmise ehitamine võis võtta kuni 10 minutit. Olgugi, et 10 minutit on suhteliselt pikk aeg, on see oluliselt kiirem kui vastava tarkvarakomplekti käsitsi ehitamine ja tõmmist ehitatakse vaid üks kord.

Haldustarkvara stabiilsuse hindamiseks jäeti `ros2_jazzy_teleop` kaugjuhtimise funktsionaalsusega näidiskonteiner tööle üheks tunniks, andes ligikaudu iga paari minuti tagant sisendeid roboti juhtimiseks. Probleeme ei tuvastatud. Selleks, et saada põhjalikumat ülevaadet stabiilsusest ja jõudlusest, tuleks viia läbi põhjalikumad testid.

7 Kokkuvõte

Käesolevas bakalaureusetöös loodi Dockeri konteineritel põhinev haldustarkvara Robotont 3 õpperobotile. See hõlmab roboti kasutajaliidesele alammenüü lisamist, veebiliidese loomist, Dockeri mootori paigaldamist, sobiva failisüsteemi loomist ja kõike eelnevat kokku siduva Pythoni teenuse loomist, vt lisa 7.

Töö eesmärgiks oli lahendada mitme erineva ROS-põhise rakenduse haldamisega kaasnevaid probleeme, selleks püstitati alamülesanded: Dockeri põhise süsteemi ülesseadmine, püsivaras alammenüü loomine konteinerite haldamiseks ja veebiliidese loomine konteinerite haldamiseks. Kõik alamülesanded täideti ja leiti seeläbi lahendused ROS-põhiste rakenduste haldamisega kaasnevatele probleemidele.

Töö tulemusena sai loodud süsteem, mis võimaldab ühel riistvaral kasutada mitut erinevat ROS-i rakendust selliselt, et need ei sõltu üksteisest. Süsteem võimaldab läbi roboti kasutajaliidese ja veebiliidese mugavalt vahetada ROS-i rakendusi ilma käsurida kasutamata.

Kirjandus

[1] "International Federation of Robotics, Top 5 Robot Trends 2024," https://ifr.org/ifr-press-releases/news/top-5-robot-trends-2024, International Federation of Robotics, (Vaadatud: 20. mai 2025).

[2] "International Federation of Robotics, Top 5 Robot Trends 2025," https://ifr.org/ifr-press-releases/news/top-5-global-robotics-trends-2025, International Federation of Robotics, (Vaadatud: 20. mai 2025).

[3] "Open Source Robotics Foundation, ROS Introduction," https://wiki.ros.org/ROS/Introduction, Open Source Robotics Foundation, (Vaadatud: 20. mai 2025).

[4] "Open Source Robotics Foundation, Why ROS?" https://www.ros.org/blog/why-ros/, Open Source Robotics Foundation.

[5] "Open Source Robotics Foundation, Nodes," https://wiki.ros.org/Nodes, Open Source Robotics Foundation, (Vaadatud: 20. mai 2025).

[6] R. Raudmäe, "Avatud robotplatvorm robotont," Tartu, Estonia, 2019.

[7] E. Mõtshärg and V. Veiko and R. Raudmäe and M. Muro and I. Drikkit and L. Tšigrinski and R. Köidam and A. Aabloo and K. Kruusamäe, "Robotont 3–an accessible 3d-printable ros-supported open-source mobile robot for education and research," *Frontiers in Robotics and AI*, vol. 11, 2024.

[8] "Robotic Sea Bass, A Guide to Docker and ROS," https://roboticseabass.com/2021/04/21/docker-and-ros/, Robotic Sea Bass, (Vaadatud: 20. mai 2025).

[9] "Docker Inc., What is a Container?" https://www.docker.com/resources/what-container/, Docker Inc., (Vaadatud: 20. mai 2025).

[10] R. Raudmäe and S. Schumann and V. Vunder and M. Oidekivi and M. Nigol and R. Valner and H. Masnavi and A. K. Singh and A. Aabloo and K. Kruusamäe, "ROBOTONT – Open-source and ROS-supported omnidirectional mobile robot for education and research," *HardwareX*, vol. 14, p. e00436, 2023.

[11] S.-E. Paap, “Ros2 draiver õpperobotile robotont,” https://hdl.handle.net/10062/107710, 2024.

[12] L. Tšigrinski, “Õpperoboti robotont püsivara arhitektuuri uuendamine,” https://hdl.handle.net/10062/107713, 2024.

[13] R. Köidam, “Valguslahenduse tarkvara väljatöötamine õpperobotile robotont,” https://hdl.handle.net/10062/107707, 2024.

[14] A. Sakk, “Avatud robotplatvormi robotont 3 kasutajaliidese väljatöötamine,” https://hdl.handle.net/10062/107712, 2024.

[15] R. Valge, “Toitepinge ja tarbevoolu monitoorimine ning toitehalduse püsivara loomine õpperobotil robotont,” https://hdl.handle.net/10062/107714, 2024.

[16] R. White and H. Christensen, “Ros and docker,” in *Robot Operating System (ROS): The Complete Reference (Volume 2)*, A. Koubaa, Ed. Springer, Cham, 2017, pp. 285–308.

[17] “Open Source Robotics Foundation, OSRF Docker Images,” https://github.com/osrf/docker_images, Open Source Robotics Foundation.

[18] “Open Source Robotics Foundation, osrf/ros,” https://hub.docker.com/r/osrf/ros, Open Source Robotics Foundation.

[19] D. Krūmiņš, “Web-based learning and software development environment for remote access of ros robots,” http://hdl.handle.net/10062/83040, 2022.

[20] D. Krūmiņš and S. Schumann and V. Vunder and R. Põlluäär and K. Laht and R. Raudmäe and A. Aabloo and K. Kruusamäe, “Open remote web lab for learning robotics and ros with physical and simulated robots in an authentic developer environment,” *IEEE Transactions on Learning Technologies*, vol. 17, pp. 1313–1326, 2024.

[21] R. Malhotra and A. Bansal and M. Kessentini, “A systematic literature review on maintenance of software containers,” *ACM Comput. Surv.*, vol. 56, no. 8, pp. Article 193, 38 pages, 2024.

[22] P. Melo and R. Arrais and S. Teixeira and G. Veiga, “A container-based framework for developing ros applications,” in *Proc. 2022 IEEE 20th Int. Conf. on Industrial Informatics (INDIN)*, 2022, pp. 280–285.

[23] “RStudio, Inc., The Rocker Project,” https://rocker-project.org/, RStudio, Inc.

[24] “RStudio, Inc., rocker,” https://github.com/osrf/rocker, RStudio, Inc.

[25] “j3soon, Docker image for Clearpath’s Husky A200 on ROS 1 Melodic,” https://github.com/j3soon/docker-ros-husky, j3soon.

[26] “j3soon, ROS 2 Essentials,” https://j3soon.github.io/ros2-essentials/, j3soon.

[27] “Google Cloud Robotics, Overview of Cloud Robotics Core,” https://googlecloudrobotics.github.io/core/overview.html, Google Cloud Robotics.

[28] Y. Zhang and C. Wurll and B. Hein, “KubeROS: A Unified Platform for Automated and Scalable Deployment of ROS2-based Multi-Robot Applications,” in *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA 2023)*. IEEE, 2023, pp. 9097–9103.

[29] “Canonical Ltd., Tutorial 1: Packaging our first ROS application as a snap,” https://canonical-robotics.readthedocs-hosted.com/en/latest/tutorials/snaps-core/packaging-ros-application-as-snap/, Canonical Ltd., (Vaadatud: 20. mai 2025).

[30] “ROS Discourse, What environments do you use for training and courses?” https://discourse.ros.org/t/what-environments-do-you-use-for-training-and-courses/26473, ROS Discourse.

[31] “Michael Kerrisk, pty(7) — Linux manual page,” https://man7.org/linux/man-pages/man7/pty.7.html, Michael Kerrisk, (Vaadatud: 20. mai 2025).

[32] “Docker Inc., Merge Compose files,” https://docs.docker.com/compose/how-tos/multiple-compose-files/merge/, Docker Inc., (Vaadatud: 20. mai 2025).

Lisa 1

Töö tulemusena arendatud tarkvara — haldustarkvara ja veebiliides, muudatused püsivaras `menu.c` ja `cmd.c` failides

`https://github.com/ut-ims-robotics/kottise-thesis-2025-robotont-docker`

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Jürgen Kottise

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

   “Dockeri konteineritel põhinev haldustarkvara Robotont 3 õpperobotile”

   mille juhendaja on Veiko Vunder

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

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

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

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

Jürgen Kottise
05.05.2025