TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Informaatikainstituut

ITI40LT

Erki Hindo 135063IAPB

ROS-IL PÕHINEVATE ROBOTITE TELEMEETRIA JA KAUGJUHTIMINE ANDROIDI RAKENDUSEGA

Bakalaurusetöö

Juhendaja: Gert Kanter

Magister

Lektor

Tallinn

2014

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Erki Hindo
23.05.2016

Annotatsioon

Minu töö eesmärk on ehitada mobiilirakendus, mis võimaldab Android telefonil paindliku infovahetust robotiga üle Wi-Fi, kasutades ROS ühendust.

Selle töö jaoks on rakendus mõeldud TTÜ Robotiklubi jalgpalliroboti juhtimiseks, kuid rakendus on nii üles ehitatud, et väikeste muudatustega on võimalik teistel arendajatel kasutada enda andmestruktuuridega. Keerulistemate andmestruktuuride jaoks on olemas vastav dokumentatsioon ja juhis, mis aitavad arendajatel muuta rakendust täiesti enda soovi järgi.

Töö võimaldab seadistada lihtsalt robotitele kaugjuhtimist võimaldavat mobiilrakendust, mida robotiga või Android platvormiga mitte tuttavad kliendid saavad kasutada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 8 peatükki, 7 joonist ja 8 tabelit.

Abstract

The main aim of this research is to develop a mobile app, which enables an easily changeable connection between an Android device and a robot over a Wi-Fi connection using ROS.

This app is meant to controll a football robot in TTÜ Robotiklubi, but it is built so that it is easy for other developers to use with their data structures. For more complex data structures there exist the necessary documentation and guides to allow developers to change the application as they see fit.

This application provides an easy way for people, who are not familiar with either the robot or the Android platform, to set up an application which enables remote control of a robot.

The thesis is in Estonian and contains 32 pages of text, 8 chapters, 7 figures and 8 tables.

Lühendite ja mõistete sõnastik

ROS	Robot Operating System
JSON	Javascript Object Notation
API	Application Program Interface
P2P	Peer to peer
JAR	Java Archive
SD	Secure Digital
MD5	Message-digest algorithm
HTML	Hyper Text Markup Language
Boolean	Boole`i avaldis
IP	Internet Protocol

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

1.1 Taust ja probleem

Igal aastal TTÜ Spordihoones peetaval robotivõistlusel Robotexil võetakse mõõtu seitsmeteistkümnel erineval robotivõistlusel. Üheks võistluseks on jalgpall, mille jaoks ehitatakse neljarattalised kuni kaheksakilosed robotid.

Iga aastaga muutuvad võistluse reeglid ning robotid muutuvad järjest keerukamateks. See vähendab jätkusuutlikust, kuna on raskem leida uusi inimesi, kes tahaksid jätkata ning kunagi üle võtta jalgpallirobotite tarkvara arendamise.

MTÜ TTÜ Robotiklubis alustati 2013. aastal jalgpalliroboti Sangpomm ehitamist, mis osales 2014 ning 2015 Robotex üritusel, ning nendest viimasel saavutas võistlusel kolmanda koha.
Seni oli võimalik näha, kuidas robot töötab ainult läbi keerulise algoritmi tööle panemise, mis pani roboti jalgpalli mängima, kuid ei andnud mingit tagasisidet kasutajale.

Soovisin pakkuda võimalust tutvuda robotiga sellega ise mängides ning teistele arendajatele teha lihtsaks rakenduse kasutamine enda masinatel. Selleks pidin looma dokumenteeritud ning hästi struktureeritud rakenduse.

1.2 Ülesande püstitus

Lõputöö eesmärgid on:
1. Luua rakendus, millega on võimalik juhtida ROS-i robotit kasutades Wi-Fi juhtmeta võrguühendust.
2. Kujundada rakendus sellisena, et arendajatel oleks võimalik võimalikult vähese ajakuluga ja tööga kasutada rakendust erinevatel robotitel.

1.3 Metoodika

1. Luua mobiilrakendus Linux virtuaalmasinas Android Studio arenduskeskkonnas, Java programmeerimiskeelt kasutades.
2. Kujundada rakendus, järgides tarkvara objektorienteeritud tarkvara arhitektuuri häid tavasid järgides.
3. Panna rakendusse sisse erinevaid liideseid: liugur, tekstiväljad, nupud; mida teised arendajad saavad vajadusel kasutada.
4. Luua rakendusele konfiguratsioonifaili tugi.
5. Vormistada põhjalik dokumentatsioon rakendusest.

2. Olemasolevad lahendused

2.1 Parrot

2.1.1 Võimalused

Hetkel on valdav enamus kontrollerrakendusi mõeldud droonidele. Üks avaliku koodiga rakenduse pakkuja on Parrot, kelle ametliku rakenduse nimi on AR.FreeFlight. Selle rakenduse kujundus sarnaneb natuke minu tööga - on olemas kaks juhtkangi, näidatakse robotilt pilti nutitelefoni ekraanil. On olemas ka Bluetooth mängupuldi toetus.

2.1.2 Tehnilised piirangud

Parrot on raskelt spetsialiseerinud rakenduse just enda droonide kontrollimiseks. Dokumentatsioonis on välja toodud, et rakendus saab kontrollida just nende kaheksat drooni. Neil ei ole välja toodud enda unikaalsete sõnumite rakendamine, mis muudab rakenduse kasutamise enda robotil väga keeruliseks. Parrot droonid ei kasuta ka ROS liidestust, mis muudab rakenduse käesolevale Sangpommile kasutuks.

2.2 Oddwerx

2.2.1 Võimalused

2011 Kickstarteris alustatud projekt nimega Oddwerx on kõige sarnasem rakendus minu omale. Rakendus võimaldab kasutada ROS-liidestust, kontrollida robotit läbi kahe juhtkangi ning näidata roboti kaameralt saadud videot. Oddwerxi põhieesmärk ei olnud aga roboti kaugjuhtimine, vaid autonoomsed tegevused, nagu labürindi läbimine või objektide tuvastamine kasutades selleks mobiiltelefoni kaamerat.

2.2.2 Tehnilised piirangud

Kuna Oddwerx ei saanud Kickstarteris enda eesmärki kätte - 66 000 dollarit, maeti projekt aastal 2012 täielikult maha. Projektist on jäänud alles ainult Kickstarter leht ning nende enda koduleht paari videoga. Ei ole olemas ei rakendust, koodi ega dokumentatsiooni.

2.3 Rosbridge

2.3.1 Võimalused

Rosbridge võimaldab läbi JSON API anda ROS funktsionaalsust mitte-ROS programmidele. See võimaldaks teha rakenduse asemel HTML leht ning kontrollida robotit läbi selle.

Rosbridge`i on ka võimalik kasutada Java programmeerimiskeeles.

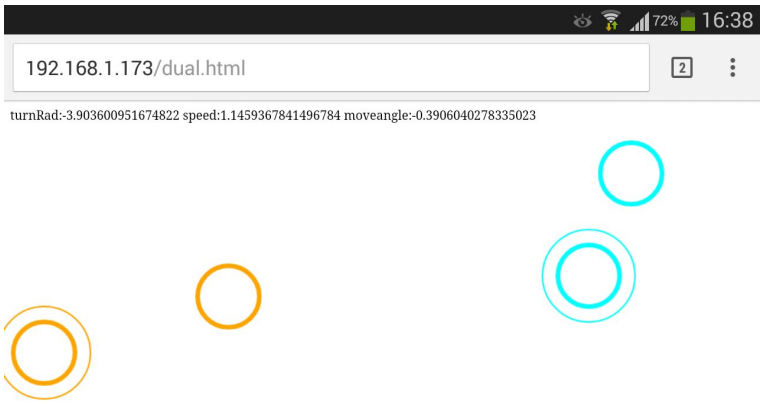
Suureks plussiks võrreldes Rosjavaga on ülesseadmise ning koodi lihtsus.

2.3.2 Tehnilised piirangud

Probleeme tekitab internetibrauseri kasutamine nutiseadme peal. Brauserid võtavad rohkem akut ning mälu. Kontroller näiteks seiskub tihti ning kaob kontroll roboti üle.

Brauseris on olemas liides, mis ei sobi kontrollerile. Puudutades ekraanile, tekib URL riba, puudutades tagasi-nuppu, minnakse HTML lehelt minema. Puududates telefonil tagasinuppu viib brauser koheselt lehelt minema, muutes kasutamise ebameeldivaks. Seadete nupp toob esile brauseri seaded ning ekraani keeramine muudab liidese orientatsiooni.

Näidisrakendus:



Joonis 1. NÄIDE: Rosbridge näidisrakendus.

Antud joonisel on Rosbridge`ga tehtud näidisrakendus, mis kasutab Rosbridge Javascript versiooni ning kahte juhtkangi sisendi saamiseks. Näidis on avatud Chrome brauseris nutitelefoniga. Rohkemat funktsionaalsust sellel rakendusel pole.

Rosbridge Java versioon, millega üritasin Android rakendust teha, töötab ainult arvutis, Androidi peal ei ole see toetatud. Rosbridge`i on hetkel võimalik telefonis ainult läbi brauseri kasutada.

3. ROS

3.1 Sissejuhatus

ROS ehk Robot Operating System on avatud lähtekoodiga süsteem robotite juhtimiseks. ROS-i peamiseks eesmärgiks on toetada koodi jagatavust ning koostööd robootikavaldkonna teadustöös ning arenduses. Üks ROS-i süsteem koosneb mitmest iseseisvast sõlmest, mis suhtlevad teiste sõlmedega kasutades avaldamise/tellimise(subscribe/publish) mudelit. Näiteks kindel andur võib olla teostatud kui sõlm ehk node, mis tähendab, et see hakkab avaldama enda andmeid sõnumite jadana. Neid sõnumeid võib lugeda loendamatu arv teisi node.

3.2 Eelised

- Keelte variatsioon: ROS on implementeeritud mitmes modernses programmeerimiskeeles. Pikemat aega töötab ROS juba Pythonis, C++ ja Lispis ning hiljuti on hakatud toetama Javat ja Lunat.
- P2P suhtlus: Selleks, et vältida keeruliste süsteemide kasutamisel sõnumite ummistumist, kasutab ROS Peer to peer suhtlust koos puhverdamise ja üles otsimise süsteemi. See võimaldab erinevatel komponentidel vabalt teineteisega otse suhelda.
- Õhuke: ROS on arendatud nii, et seda teostavad algoritmid hoitakse üksikutes käivitatavates failides. See aitab suurendada8 jagatavust ning hoida süsteemi väiksena. Samuti teeb see ROS-i lihtsasti kasutatavaks ning testitavaks, hoides keerulisi algoritme teekides.

3.3 Puudused

- Operatsioonisüsteem: Hetkel töötab ROS ainult Unix-baasil süsteemidel. Microsoft Windowsi peale on üle toodud ainult minimaalne versioon, mis on praktiliselt kasutamatu.

  See on põhjuseks, miks pidin arendamiseks kasutama Oracle Virtual Machine VirtualBoxi ning Ubuntu virtuaalmasinat, kuhu oli eelnevalt juba ROS installitud.

- Keeruline alustada: Uutel inimestel võtab palju aega tutvuda selle süsteemiga ning samuti võtab palju aega ROS-i paigaldamine, kuna ROS-il on mitmeid erinevaid versioone ning erinevatel arvutitel on erinevad konfiguratsioonid. Lõputöö kirjutamise ajal toetab ROS Indigo versioon Androidi, kuid ROS-il endal on juba 2 uuemat versiooni. Erinevuste pärast on väljas hästi palju erinevaid versioone installatsiooni juhistest, mis võib segadust tekitada.

  Veel võtab aega süsteemi korralikult tööle panemine ning algoritmidest aru saamine. Igal ROS-i versioonil on näiteks unikaalne viis, kuidas tekitada .msg failidest .jar failid.

4. Robotist

4.1 Andmete tüübid

Sangpomm kasutab infovahetuseks enamasti unikaalseid sõnumitüüpe. Kokku on neid neli. Üks, mida väljastab, kaks, mis peituvad väljastatava sõnumi sees ning üks, mida loeb. Ainsaks mitteunikaalseks sõnumiks on kaamerapilt, mis kasutab CompressedImage tüüpi.

4.1.1 Cmd.msg - sissetulev sõnum

Tabel 1. Cmd.msg kirjeldus

absSpeed	Kiirus, millega robot liigub.
radAngle	Suund, kuhu robot liikuma hakkab.
turnRate	Nurk, kuhu poole robot pöörlema hakkab.
kick	Predikaat, kas robot kasutab löögimehhanismi.
kickStrength	Löögi tugevus.
text	Sõnumiga kaasa antud täpsustav tekst.

4.1.2. ImageData.msg - väljastatav sõnum

Tabel 2. ImageData.msg kirjeldus

balls	Pallide objektid.
yellowGoal	Kollase värava objekt.
blueGoal	Sinise värava objekt.
source	Täpsustav tekst, kas pilt tuleb esi- või tagakaameralt.

4.1.3. Ball.msg - sissetulev sõnum

Tabel 3. Ball.msg kirjeldus

angle	Nurk roboti keskpunkti ja palli vahel.
distance	Kaugus pallini.
forbidden	Boolean, mis täpsustab, kas pall on väljaku

	joontest väljaspool.

4.1.4. Goal.msg - sissetulev sõnum

Tabel 4. Goal.msg kirjeldus

angle	Nurk roboti keskpunkti ja värava vahel.
distance	Kaugus väravani.
visible	Boolean, kas antud värav on nähtav.

4.2. Java liides

Kuna .msg faile ei saa otse Android projekti importida, pidin genereerima sõnumifailidest .jar failid.

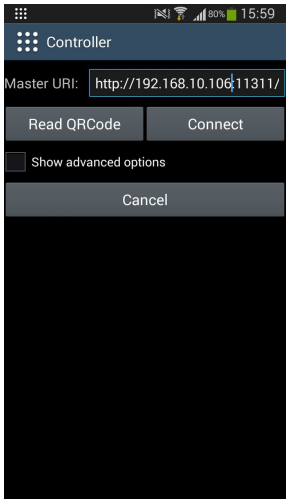
.jar failide genereerimine ei ole mahukas tegevus, kuid vajab korrektset protseduuri, et genereeritud failid sobiksid kokku algsete sõnumifailidega.

Üks viis Java failide genereerimiseks, mida mina kasutasin, on selline, kus peab tegema tühja projekti ning nimetama selle vastavalt sõnumite emaklassile. Pärast seda peab sisse kopeerima sõnumitüübid ja konfiguratsioonifailid ning siis failidesse kirjeldama õiged sisendid ning väljundid. Lõpuks piisab ainult *catkin_make* käsust ja tekkinud faili importimisest projekti.

Lõputööd tehes tuli välja, kui oluline on .jar failidel silma alati peal hoida ning rutiinseid kontrolle teha . Selle projekti arenduskäigus muudeti robotil ühte .msg faili nime ning robot ei aksepteteerinud järsku enam rakendust, kuna sõnumitüüpide MD5 summa ei olnud osapooltelt võrdne.

5. Disain

5.1 Ühenduse valimine

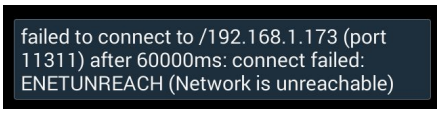


Joonis 2. Ühenduse valimise vaade.

Rakenduse avades on esimeseks vaateks IP-aadressi sisestamise aken. See on Rosjavaga kaasa antud vaade. Antud vaate funktsionaalsuseks on robotiga ühenduse loomine ning rakenduse kontrolleri avaldamise. Pärast kontrolleri avamist antaksesõnumitellijatele ning avaldajatele sisestatud IP-aadress ning pannakse nad tööle. Vajutades nupule “Cancel” on võimalik minna kontrolleri vaatesse infovahetust alustamata. Nii on võimalik vaadata kontrolleri disaini nendel, kellel puudub robotiga ühenduse võimalus.

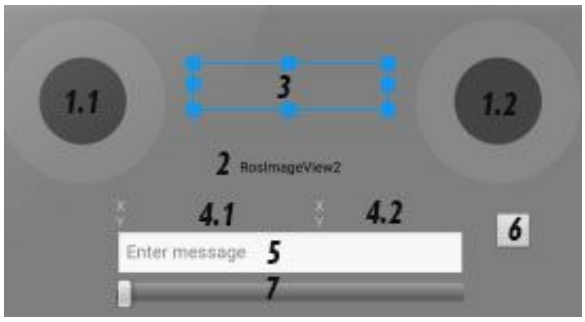
Mina lisasin rakendusele ning sellest tulenevalt sellele vaatele veateadete näitamise. Kasutasin Android.widget.Toast-i.

Näiteks, kui telefonil pole internetiühendust tekib teade:



Joonis 3. Veateate kuvamine.

5.2 Kontroller



Joonis 4. Kontrolleri objektid.

Antud joonisel on toodud välja kontrolleri põhivaates asuvad objektid. Kokku on neid 7 erinevat. Sangpommi kontrollimiseks kasutan hetkel ainult nelja. Ülejäänud kolm on mõeldud rakenduse paindlikumaks tegemiseks.

1. Kaks juhtkangi - kõige olulisemad objektid vaates. Kasutasin programcreek.com/java-api-examples/ lehel üles laetud DualJoystickView klasse. Need klassid näitavad juhtkange ekraanil ning jälgivad nende liikumist. Ise lisasin juhtkangide positsiooni põhjal tehted, mis arvutavad välja kiiruse, liikumissuuna ning pöörlemisnurga.
   1.1. Vasakpoolne juhtkang kontrollib roboti kiirust ja liikumissuunda. Mida kaugemal juhtkang on keskpunktist, seda suurem on kiirus.
   1.2. Parempoolne juhtkang kontrollib roboti pöörlemisnurka ning löögimehhanismi. Liigutades kangi horisontaalselt hakkab robot pöörlema. Lükates kangi täiesti üles, antakse robotile löögikäsk. Ma ei pannud löögimehhanismi kontrolli nupule(6.), sest nii ei pea enda sõrmi ekraanilt ära võtma ning üritama nupule pihta saada. Antud robotil ei piisa ainult ühest ainsast “Kick” käsust. Robotil võtab aega löögimehhanismi laadimine ning lahti laskmine, sellepärast peab saatma käsku seni, kuni see on täidetud, mis võib võtta aega kuni paarsada millisekundit.

2. Roboti kaamerapilt. Kasutasin parandatud versiooni RosImageView`st ning CompressedImage failitüüpi, et oleks võimalik näidata ekraanil kaamerast tulevat pilti. Mina enamasti ei kasuta aga seda objekti, kuna see võtab rohkem telefoni mälu ja akut. Sobilik kasutada, siis kui on hea ruuter, hea telefon ning soov kellelgi demonstreerida roboti kaamerat.

3. Teksti kast, mis on seotud roboti infokuulajaga. Iga uue sõnumiga uuendatakse teksti. Hetkel näitab kast sõnumit "KICK", kui robot näeb palli enda löögimehhanismi ees. See aitab kasutajal kaamerapilti kasutamata ning roboti ees seismata saada aru, kunas on võimalik robotil palli lüüa. Kasutasin Rosjava RosTextView`d ja roboti väljastatavat sõnumitüüpi ImageData. Tegin ka funktsiooni, mis säilitab "KICK" sõnumit kastis kauem, kuna uusi sõnumeid saadetakse paarkümmend korda sekundis võib muidu kastis olev tekst hakkata vilgutama.

4. Juhtkangide asukohtade koordinaadid. Kasutasin Androidi tavalisi TextView`sid, mida uuendatakse iga kord, kui juhtkange liigutatakse. Need annavad kasutajale rohkem tagasisidet selle kohta, kui palju ta juhtkange liigutab. Muutuvad suuruses -10 kuni 10.

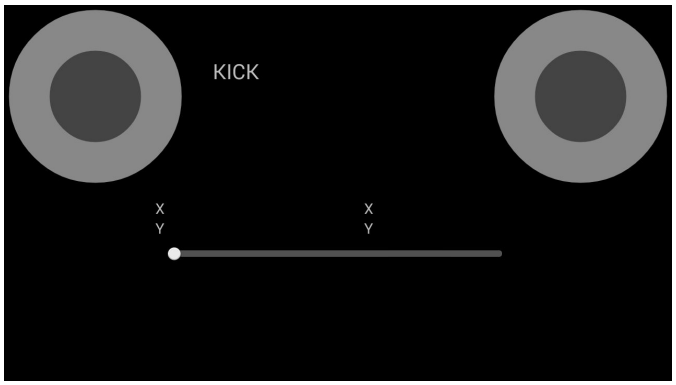
5. Sisestatav tekst. Kasutasin Androidi tavalist EditTexti. See objekt võimaldab võtta sisendiks mingi teksti ning panna selle saadetavasse sõnumisse sisse . Antud hetkel ei kasuta robot selle funktsionaalsust ning selle kasutamine on valikuline. Mõeldud teistele kasutamiseks.

6. Tavaline nupp. Kasutasin Androidi Buttonit. Antud rakenduse versiooni funktsionaalsus puudub, kuna otsustasin juhtkangile anda löögimehhanismi kontrolli. Mõeldud teistele kasutamiseks.

7. Liugur. Kasutasin Androidi Seekbari. Antud robotil töötab liugur kui liikumiskiiruse kordaja - vasakul pool on kõige madalam kordaja ning robot sõidab kõige aeglasemalt. Mida paremale liugurit liigutada, seda kiiremaks muutub robot.

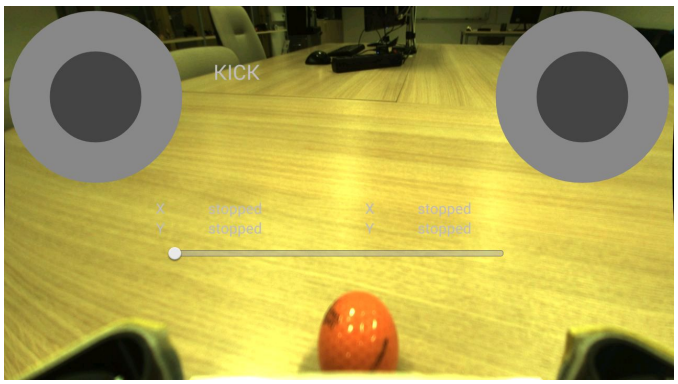
5.3 Valitud disain

Sangpommi kontrollimiseks ning enda telefoni mälu arvestades sobib kõige paremini selline liides, kus pole kaamerapilti ning ebavajalikud objektid välja jäetud. Olemas on juhtkangid, tekstiväljad ning liugur.



Joonis 5. Valitud disan.

Kuid kui on olemas hea internetiühendus ning soov robotit demonstreerida siis sobib liides, kus on olemas kaamerapilt.



Joonis 6. Disain kaamerapildiga.

Antud joonisel on kõik samad objektid, mis eelmisel, kuid juures on kaamerapilt. Kaamerapildi kuvamine võib põhjustada juhtkangide aeglustumist.

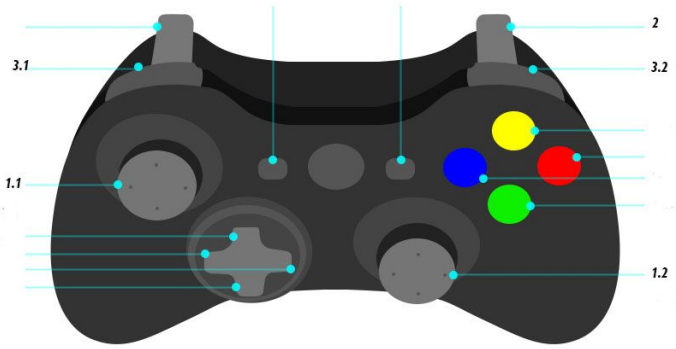
5.4 Bluetooth kontroller

Läbi rakenduse on võimalik saada sisendit ning sellest tulenevalt kontrollida robotit kasutades Bluetooth ühendusega pulti. Puldiga kontrollimiseks on vaja üles seadistada üldine ühendus puldi ning nutitelefoni vahel ning siis ainult robotiga ühendust.

Rakenduses kasutasin Androidi funktsiooni:

```
@Override

    public boolean onKeyDown(int keyCode, KeyEvent event)
```



Joonis 7. Bluetooth kontrolleri funktsionaalsus

Antud joonisel on välja toodud, mis nuppudel võtab rakendus sisendit. Ülejäänud nuppe on võimalik väga lihtsasti rakendusega kooskõlastada. On vaja teada vaid Bluetooth kontrolleri nupu koodi.

Hetkel kirjutab Bluetooth kontroller andmeid samadesse muutujatesse nagu kontrolleri vaates asuvad objekti. Soovi korral on võimalik defineerida iga nupu jaoks eraldi muutuja. See viiks lahku muutujad, mille tõttu oleks siis võimalik saata robotile veel rohkem erinevaid käske, kuid hetkel selle vajadus puudub.

Puldi nuppude funktsionaalsus:

1. Juhtkangide funktsionaalsus on enamasti sama, välja arvatud parem juhtkang, millel puudub löögimehhanismi kontroll.

2. Päästikule on üle antud löögimehhanismi kontroll.

3. Nendele nuppudele on üle antud liuguri funktsionaalsus.
    3.1. Vähendab kiirust.
    3.2. Suurendab kiirust.

6. Seadistus

Rakenduse dünaamiliseks tegemiseks lõin konfiguratsioonifaili nimega gui.txt ning klassi GuiReader, mis rakenduse tööle panemisel otsib konfiguratsioonifaili esiteks telefoni SD-kaartilt ning siis rakenduse seest.

Konfiguratsioonifaili lõin oma süntaksi, millega seadistatakse nii kontrolleri objektid, sõnumitellijad kui ka sõnumiavaldajad.

Valitud disaini (kaamerapildiga) konfiguratsioon:

```
inputs:
        -type: slider
        xy: 0 500
        range: 1 5
        -type: joysticks
        size: 1250 400
        -type: joysticksTable
        xy: 400 300

Listener:
        -msg_type: msgs/ImageData
        topic: /vision_data

image_listener:
        -topic: /front_cam/image/compressed

publishers:
        -topic: /com_cmd
        msg_type: msgs/Cmd
        wait: 100
        data: leftJSdistance leftJSangle rightJSxangle rightJSup
```

6.1 Konfiguratsioonifaili ülesehitus

Konfiguratsioonifail on kokku jaotatud nelja sektsiooni.

Esimeseks haruks on kontrolleri vaates asuvad objektid ehk inputs. Rakendust avades on kõik objektid peidetud, kuid tuues selles harus nad välja, muutuvad nad nähtavaks.

Tabel 5. Inputs haru käsud

Käsutüüp	**Omadus**
-type	Defineerib, millist objekti näidata. Ainult selle rea olemasolul näidatakse objekti asukohaga 0;0 ning suurusega, mis võtab ematüüpi objekti suuruse.
xy	Liigutab eelnevalt defineeritud objekti.
range	Muudab eelnevalt defineeritud objekti andmete ulatust. Võimalik kasutada juhtkangidel ning liugur objektil
size	Muudab eelnevalt defineeritud objekti suurust.

Teiseks on andmekuulajate konfigureerimisharu ehk listener. Kuulaja defineeritakse omapäraselt. Msg_type on esimene omadus, mida peab defineerima, sest kuulaja kasutab RosTextView`d, mis nõuab kindlat konstruktorit sõnumitüübiga.

Tabel 6. Listener haru käsud

Käsutüüp	**Omadus**
-msg_type:	Defineerib andmetüübi, mida kanalist jälgima hakatakse.
topic	Defineerib kanali, millest hakata andmeid kuulama.

Kolmandaks haruks on roboti kaamerapildi näitaja ehk image_listener. Teoreetiliselt võib see ning eelmine haru olla koos, kuid ma eraldasin nad, kuna eelmises harus on vaja defineerida andmetüüp ning siin on sunniviisiliselt see CompressedImage tüüpi.

Tabel 7. Image_listener haru käsud

Käsutüüp	**Omadus**
-topic	Defineerib kanali, millest hakata kaamerapilti jälgima.

Ning viimaseks haruks on andmeteavaldajad ehk publishers.

Tabel 8. Publishers haru käsud

Käsutüüp	Omadus
-topic	Defineerib kanali, kuhu hakatakse sõnumeid saatma.
msg_type	Sõnumitüüp, mille formaadis hakatakse sõnumeid saatma
wait	Ooteaeg enne järgmise sõnumi saatmist
data	Kirjeldab, mis andmetega sõnumitüüp täidetakse. Lähemalt seletatud järgmises punktis.

6.2 Telemeetria

Käsuga `data:` defineeritakse andmed, millega sõnum genereeritakse. Andmed sisestatakse väljadesse vastavalt nii, mis järjekorras on dokumentatsioonis andmevälju kirjutatud ning mis järjekorras on käsu `data:` taha objektide andmed kirjutatud. See võimaldab seadistada kontrolleri vaate objektide funktsionaalsust vastavalt vajadusele, mis suurendab rakenduse kasutusvõimalusi.

Eeltoodud konfiguratsioonifaili näites määratakse Cmd sõnumi absSpeedile vasaku juhtkangi kaugus keskpunktist, radAnglelile määratakse vasaku juhtkangi asukohast välja arvutud nurgaga ja nii edasi kuniks muutujani kick. Kuna `data:` taha on kirjutatud ainult neli muutujat, kuid Cmd sõnumis on kuus välja lisatakse automaatselt viimastele väljadele väärtused 0 ja “”. Kui kirjutada muutujanime asemel midagi muud, näiteks number või kirje, saadetakse igas sõnumis kirjutatud number või kirje.

Muutes konfiguratsioonifailis `data:` käsu taga olevat muutujaid vastavalt vasaku või paremajuhtkangi andmete vastu on võimalik vahetada juhtkangide funktsionaalsus - siis kontrollib vasak juhtkang roboti pöörlemist ning löögimehhanismi ja parem kontrollib roboti liikumissuunda ning kiirust.

Kasutades rakenduses SD-kaardi pealt loetud konfiguratsioonifaili on võimalik ka kirjutada ühele sõnumiväljale parool, mida kaasatakse kõikidesse sõnumitesse. Nii on võimalik jagada rakendust mitmele telefonile, kartmata, et robotit hakatakse segadusse ajama mitmelt telefonilt tulevate sõnumitega. See võimaldab demonstreerida uudishimulikele rakendust nende enda telefonidest, juhul, kui nad lubavad või saavad panna enda telefoni SD-kaardi.

Sisendite põhjal olen defineerinud 15 erinevat muutujat. Juhtkangidel on mõlemal 6 muutujat, mis näitavad näiteks juhtkangide X, Y asukohta või kaugust keskpunktist. Ma olen kirjutanud ka muutujatele andmetüüpide konverteerimis funktsioonid. Näiteks, kui tahetakse saata kindlat numbrit võidakse kirjutama editText väljale mingi numbritest koosnev text ning siis InputManageris asuv funktsioon getFloat() muudab sisestatud teksti ujukomaga numbriks enne saatmist.
Rakenduse andmesaatja toetab hetkel seitset erinevat andmetüüpi. Rakenduse vastuvõtja toetab hetkel kuute.

Võimalikud väljastatavad sõnumitüübid: msgs/Cmd, std_msgs/Bool, std_msgs/Int32, std_msgs/String, std_msgs/Float32, std_msgs/Byte, std_msgs/Char.

Võimalikud vastuvõetavad sõnumitüübid: msgs/ImageData, std_msgs/String, std_msgs/Bool, std_msgs/Byte, std_msgs/Int32, std_msgs/Float32.

6.3 Näide

Järgmisena toon ma välja ühe seadistuse, mida on võimalik kellelgi kasutada enda robotil või süsteemil. Selle näite eesmärk on demonstreerida seadistuse paindlikust ning võimekust.

Konfiguratsioonifail:

```
inputs:
        -type: slider
        xy: 0 600
        size: 300 100
        range: 0 100
        -type: button
        xy: 200 200
        -type: enterText
        xy: 0 500

listener:
        -msg_type: std_msgs/Bool
        topic: /someone_behind_door
```

publishers:
    -topic: /ava_uks
    msg_type: std_msgs/Bool
    wait: 200
    data: buttonPress
    -topic: /displei
    msg_type: std_msgs/String
    wait: 5000
    data: enteredText
    -topic: /valgustus
    msg_type: std_msgs/Int32
    data: slider

Selle näite konfiguratsioon võimaldab kontrollida ühte firmas asuvat ust ning sellega seotud vidinaid.

Sisenditest on valitud liugur, nupp ning teksti sisestamise vaade. Kõigile on määratud mingi asukoht ekraanil ning liugurile on määratud tema väikseim, suurim väärtus ja suurus ekraanil.

Kuulaja on määratud kuulama kanalit nimega /someone_behind_door, mis näitab sõnumit, kui keegi on ukse taga.

Andmete avaldajaid on siin kokku 3. Esimene määratatakse topicule /ava_uks sõnumitüübiga boolean. Ooteajaks iga sõnumi vahel on 200 ms ning sõnumi sisuks määrati boolean, mis jälgib, kas ekraanil asuvat nuppu on alla vajutatud. Näiteks, kui nuppu vajutatakse, saadetakse boolean väärtusega true, seejärel uks avaneb.

Teiseks saatjaks on String tüüpi sõnum, mida võidakse kasutatada ukse kõrval oleval kuvaril, et näidata mingit sõnumit. Sõnumiks võib olla: “Suletud”. Sõnum võetakse teksti sisestamise kastist ning saadetakse iga 5 sekundi tagant.

Kolmandaks saatjaks on defineeritud Int32 tüüpi saatja, mis võib kontrollida ukse ees olevat valgustust. Andmed võetakse slider muutujast, mis sõltub liugur objektist. Kui on halb nähtavus, võidakse valgustust tõsta ukse ees. Kuna aega pole määratud saadetakse sõnum iga 100 ms tagant.

Selles konfiguratsioonifailis puudub image_listener haru, seega ei kuvata ekraanile ühtegi pilti.

7. Avalik dokumentatsioon

Rakendusest paremini arusaamiseks olen loonud sellele Githubi Wiki. Kokku lõin üle viieteistkümne lehe. Wiki põhimõte on peamiselt anda juhiseid nendele, kes tahavad arendada rakendust edasi või modifitseerida seda enda robotile sobilikuks.

Kõige pikemalt ja põhjalikumalt kirjutasin juhise, mis seletab kuidas endale arvutisse alla laadida kõik vajalik arendamiseks - virtuaalmasinast controller projektini.

Olen kirjeldanud ka konfiguratsioonifaili süntaksi ning funktsionaalsust. Konfiguratsioonifaili kohta olen kõikide harude kohta kirjutanud eraldi seletuse. Samuti olen kirjeldanud enamusi klasse, mida rakendus kasutab. Iga klassi kohta olen seletanud klassi üldist eesmärki ning funktsioonide funktsionaalsusi.

Olen ka kirjutanud juhise, kuidas luua enda .msg failidest .jar fail ning kuidas seda enda projekti importida.

8. Kokkuvõte

Bakalaureuse töö tulemusena sai loodud Android rakendus, millega on võimalik kontrollida ROS-i kasutavat robotit nimega Sangpomm. Lisaks nimetatule on võimalik rakendust kasutada teiste robotitega.

Töö esimeses osas on välja toodud mitmed sarnased rakendused, nende plussid ning miks neid Sangpommiga ühel või teisel põhjusel kasutada ei saa. Teises osas kirjeldatakse kasutatavat süsteemi ja robotit koos enda andmetüüpidega.
Töö keskel on antud ülevaade Sangpommi kontrollivast rakenduse ülesehitusest. Veel on kirjeldatud lisafunktsionaalsusest kui Bluetooth kontrolleri toest.
Viimases osas on toodud välja töö aspektid, mis muudavad antud rakenduse paindlikuks ning teistele arendajatele arusaadavamaks.

Bakalaureuse töö alguses kirjutatud eesmärkidest sai esimene edukalt täidetud. Teise eesmärgi, milleks on vähene ajakulu teistel arendajatel, lahendamiseks sai tehtud konfiguratsioonifaili tugi, millega saab seadistada liidest ning valida andmetüüpe. Veel sai tehtud dokumentatsioon, mis aitab arendajatel viia muudatusi projekti sisse kiiremini. Ainsaks suureks ajakuluks teistel arendajatel on projekti alla laadimine ning üles seadmine.
Viimasest eesmärgist tulenevalt on vaja seda projekti kindlasti veel edasi arendada. Vastavalt tagasisidele võivad välja tulla kohad, mida võiks teisiti teha või edasi arendada.

Viidatud allikad

1. Kontrollitav robot Sangpomm. [WWW] https://www.robotiklubi.ee/projektid/robotex/2014/sangpomm (30.04.2015)
2. Virtualbox. [WWW] https://www.virtualbox.org/ (12.08.2015)
3. Arendamiseks vajalik virtaalmasin. [WWW] http://nootrix.com/software/ros-indigo-virtual-machine/ (12.08.2015)
4. Parrot rakendus. [WWW] http://developer.parrot.com/ (11.04.2016)
5. Oddwerx projekt. [WWW] http://www.oddwerx.com/ (11.04.2016)
6. Oddwerx projekt leheküljel Kickstarter. [WWW] https://www.kickstarter.com/projects/ologic/oddwerx-autonomous-smartphone-robots
7. Rosbridge. [WWW] http://wiki.ros.org/rosbridge_suite (20.05.2015)
8. ROS. [WWW] http://wiki.ros.org/ (30.04.2015)
9. Android vidinad. [WWW] http://developer.android.com/index.html (17.02.2016)
10. Juhtkangid. [WWW] http://www.programcreek.com/java-api-examples/index.php?source_dir=AndroidRover-master/RoverRemote/src/com/jimmyblaze/roverremote/DualJoystickView.java# (23.11.2015)
11. Kasutatud RosImageView. [WWW] https://github.com/damonkohler/rosjava/issues/141 (8.2.2016)
12. Projekti lähtekood. [WWW] https://github.com/erkihindo/controller (08.05.2016)
13. Projekti Wiki. [WWW] https://github.com/erkihindo/controller/wiki (08.05.2016)