TARTU ÜLIKOOL
Arvutiteaduse instituut
Infotehnoloogia õppekava

Arvi Kaasik

Mobiilirakendus roboti juhtimiseks

Bakalaureusetöö (6 EAP)

Juhendajad: Taavi Duvin, MA
Alo Peets, MSc
Anne Villems, MSc

Tartu 2016

Mobiilirakendus roboti juhtimiseks

Lühikokkuvõte:

Eestis õpetatakse kooliõpilastele programmeerimist selleks, et panustada Eesti kui E-riigi tulevikku. Üks viis selleks on viia läbi erinevaid robootika alaseid huvilaagreid. Robotite reaalajas juhtimiseks on vaja arendada juurdekäiv juhtimispult. Tänapäeval on väga laialdaselt levinud nutiseadmete kasutamine ning käesoleva bakalaureusetöö tulemusena luuakse materjalid, mille abil on võimalik robootikahuvilistel programmeerida enda roboti juhtimiseks mobiilirakendus ning paigaldada rakenduse enda isiklikule nutiseadmele. Töös arendatakse edasi Taavi Karelsoni bakalaureusetöö Raspberry Pi robotist Alo Peetsi poolt edasi arendatud NUTIROBOTi projekti. NUTIROBOTi edasiarendusele luuakse kokkusobiv Android mobiilirakendus. Roboti ja nutiseadme vaheline suhtlus toimub üle WiFi võrgu võrgupäringute abil. Raspberry Pi robotile lisatakse video edastamise tugi ning andurilt andmete lugemise tugi. Näidisrakendusele arendatakse Video voogedastuse näitamise tugi, juhtkang roboti liigutamiseks, nupud roboti komponentide sisse-välja lülitamiseks.

Võtmesõnad:

Android, Raspberry Pi, Mobiilirakendus

CERCS: P175 Informaatika, süsteemiteooria

Mobile application to control a robot

Abstract:

In order to secure Estonia’s future as an E-country, pupils are taught programming. One way to teach programming is by organizing different robotics themed lessons. In order to control robot’s behaviour a remote control device is needed. Smart phones and tablets are very commonly used and with this paper results in materials that enable robotics enthusiasts to develop a mobile application that can be used as enthusiast’s robot’s control remote. Raspberry Pi robot developed in Taavi Karelson’s bachelor thesis was further developed by thesis supervisor Alo Peets into a project called NUTIROBOT. With this paper NUTIROBOT project is further developed. For resulting robot Android mobile application is developed. Robot and smartphone communicate using WiFi network using HTTP requests. Raspberry Pi robot is further developed by adding video streaming support and support for reading data from distance sensor. For the mobile application

video streaming capacity support, joystick to move the robot and buttons to turn on and of different robot components are added.

Keywords:
Android, Raspberry Pi, Mobile Applications
CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus

2015. aastal lõi Taavi Karelson enda bakalaureusetöö „Nutiseadmest kaugjuhitava roboti ehitamine Raspberry Pi näitel“ käigus juhendi nutiseadmest juhitava ratastega roboti loomiseks. T.Karelsoni bakalaureusetööd arendas edasi töö juhendaja Alo Peets. Nutiseadmest roboti juhtimiseks kasutati mobiilirakenduse asemel HTML5 abil veebilehelt nutiseadme akkelerомeetrilt andmete lugemist, mistõttu on oluliselt piiratud roboti juhtimise viisid. Seetõttu otsustas autor selle töö kirjutamisel laiendada roboti juhtimisvõimalusi kirjutades lihtsama mobiilirakenduste arendamise eestikeelse juhendi ja näidisrakenduse. Autor valis Androidi platvormi, sest Androidi nutiseadmed on laialt kasutuses ja odavamad kui iOS operatsioonisüsteemiga nutitelefonid ning tahvelarvutid. Kuna mobiilirakendusele on võimalik lisada palju erinevaid juhtimismehhanisme ning isegi videot vaadata, otsustas autor ka robotit edasi arendada luues lihtsa pommiroboti.

Käesoleva bakalaureusetöö eesmärgiks on edasi arendada NUTIROBOTi projekti, lisades seni olemasolevale funktsionaalsusele video jäädvustamise ning edastamise ja kaugusanduriga kauguse mõõtmise funktsionaalsust. Teiseks eesmärgiks on arendada projekti roboti juhtimiseks sobiv Androidi operatsioonisüsteemil mobiilirakendus. Kolmandaks eesmärgiks on jäädvustada töö käik ning selle abil luua kooliõpilastele kasutatav juhend.

Käesoleva bakalaureusetöö esimeses peatükis selgitab autor juhendi abil, kuidas luua lihtsamat mobiilirakendust, millised on peamised rakenduse komponendid ja kuidas mobiilirakendus nutiseadmele paigaldada. Teises peatükis kirjutab autor, kuidas näidisrobotit ühendada ja tööle panna. Kirjeldatakse ka lisatava kaugusanduri tööpõhimõtet. Autor kirjeldab, kuidas näidisrakendus ja selle komponendid töötavad. Autor toob välja teises peatükis ka tekkinud probleemid ja toob välja autori jaoks sobinud lahendused.

2. Android mobiilirakenduste arendamine

Android on Linuxil baseeruv operatsioonisüsteem, mida kasutatakse väga paljudes seadmetes sealhulgas tahvelarvutid, nutitelefonid, -kellad, -televiisorid. Androidi operatsioonisüsteemiga või operatsioonisüsteemi modifikatsioonidega nutiseadmeid on maailmas enimmüüdud [1]. Kuna Android nutiseadmed on nii levinud, on need väga mugavaks võimaluseks erinevate robotite juhtimisel. Lihtsama mobiilirakenduse arendamine on jõukohane igale arendajale, kellel on programmeerimisega kokkupuuteid.

2.1 Töökeskkonna loomine

Käesoleva bakalaureusetöö alguses kaalus autor näidisrakenduse arendamiseks kahte keskkonda. Esimene kaalutud variant oli Xamarini platvorm, mille abil kirjutatakse koodi programmeerimisekeeles C#. Xamarini platvormi abil on võimalik kirjutada mobiilirakendusi peaaegu kõigile nutiseadmetele, sealhulgas Android, iOS ja Windows Phone. Teine kaalutud variant oli Android Studio, kus kood kirjutatakse keeles Java, ning millel on tugi ainult Android nutiseadmetele. Android Studio valiti näidisrakenduse arendamiseks, kuna Java keelt kasutatakse tänapäeval rohkem ning autor arvas, et vähemlevinud programmeerimisekeele õppimine võib mõningaid huvilisi heidutada. Töö käigus kasutati Android Studio 1.5.1 versiooni.

2.2 Android Studio nõuded

Selleks, et Android Studio arenduskeskkonnas mobiilirakendusi luua, peab arvuti vastama Android Studio miinimumnõuetele. Järgnevad nõuded arvuti operatsioonisüsteemi järgi [2].

Windows

- Microsoft® Windows® 10/8/7/Vista (32 või 64-bit)
- Vähemalt 2 GB operatiivmälu, soovitatavalt 4 GB
- 400 MB kõvaketta ruumi

- Vähemalt 1 GB ruumi Android SDK (*Software Development Kit*), emulaatori piltide jaoks
- Resolutsioon vähemalt 1280 x 800
- *Java Development Kit (*JDK*)* 7

Mac OS X

- Mac® OS X® 10.8.5 kuni 10.9 (Mavericks)
- Vähemalt 2 GB operatiivmälu, soovitatavalt 4 GB
- 400 MB kõvaketta ruumi
- Vähemalt 1 GB ruumi Android SDK, emulaatori piltide jaoks
- Resolutsioon vähemalt 1280 x 800
- *Java Development Kit (*JDK*)* 7

Linux

- GNOME või KDE töölaud
- GNU C Teek (*glibc*) 2.15 või uuem
- Vähemalt 2 GB operatiivmälu, soovitatavalt 4 GB
- 400 MB kõvaketta ruumi
- Vähemalt 1 GB ruumi Android SDK, emulaatori piltide jaoks
- Resolutsioon vähemalt 1280 x 800
- *Java Development Kit (*JDK*)* 7

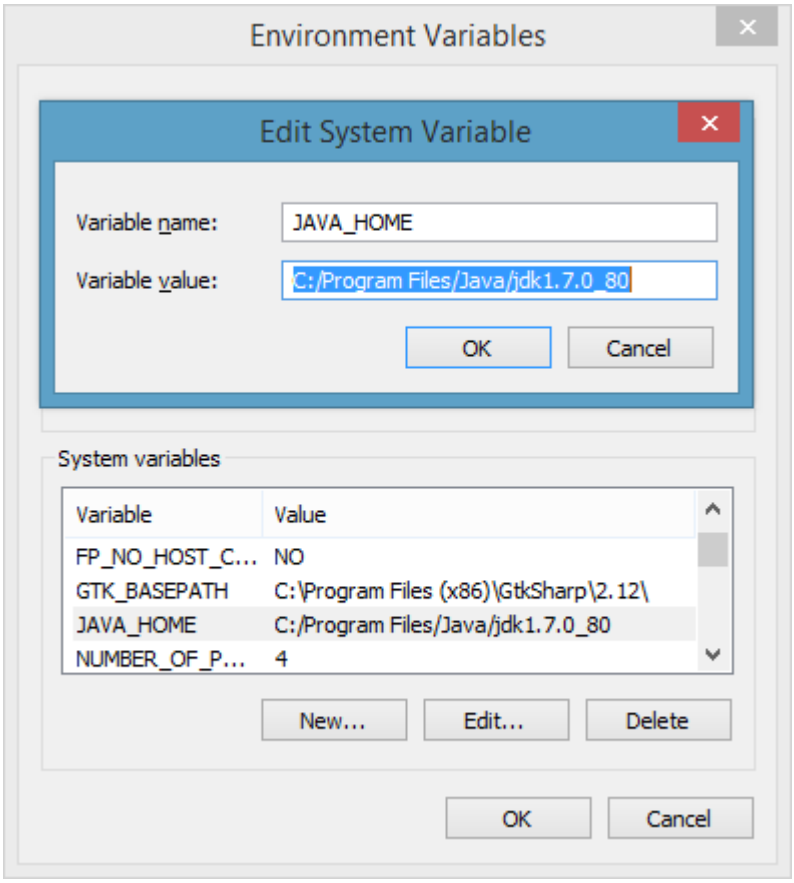
Järgnevalt kirjeldab autor, kuidas on võimalik Android Studio allalaadida ja paigaldada.

2.3 Android Studio allalaadimine ja paigaldamine

Android Studio on võimalik allalaadida veebilehelt http://developer.android.com/sdk/index.html. Programmi allalaadimine võib võtta mitu tundi, sest programmi paigaldusfaili maht on üle 1 GB. Järgmisena tuleb käivitada allalaetud .exe fail ja järgida viisardi juhendeid. Arvestama peab sellega, et programmi käivitades võib programm paigaldada veel erinevaid komponente, mis võtab aega. Esmasel käivitamisel soovitab autor *Windows* operatsioonisüsteemi kasutajatel lülitada välja viirusetõrje programmid, kuna Android Studio paigaldab SDK *Manager* tööriista, mis ei tööta, kui mõni programm kasutab paigaldamiseks vajalikke kaustu.

Android Studio kasutamiseks on vajalik JDK *(Java Development Kit)* versiooni 6 või kõrgem. Android 5.0 ja kõrgemate Androidi operatsioonisüsteemidele rakenduste arendamisel peab olema arvutis paigaldatud vähemalt JDK 7. JDK versiooni kontrollimiseks tuleb avada käsurida ja trükkida sisse "javac -version". Kui JDK ei ole paigaldatud, või on madalam, kui versioon 6, on vajaminevat versiooni võimalik laadida alla veebilehelt http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.

Peale JDK installeerimist on vajalik defineerida JAVA_HOME keskkonnamuutuja. Windows operatsioonisüsteemil on seda võimalik teha minnes „Computer" → „System Properties" → „System Settings" → „Environment Variables". Edasi tuleb lisada uus muutuja mille nimeks on „JAVA_HOME" ja väärtuseks JDK kaust (Joonis 1). Juhul kui Android Studio ei leia JDK platvormi, võib proovida kontrollida, kas muutuja väärtus on õige ja kaldkriipsud õiges suunas (Joonis 1). Kontrollima peab, et tegemist on õige JDK versiooniga.



Joonis 1: Keskkonnamuutuja näidis

JDK sisaldab endas suurt osa keeles Java arendamiseks vajalikke teeke ning JDK olemasolu korral võib arendaja kirjutada lisaks mobiilirakendustele ka muud tarkvara keeles Java. Töökeskkonna seadmisest mobiilirakenduste kirjutamiseks aga ei piisa. Nimelt toimub rakenduse arendus eraldiseisvas projektis. Projekti loomist või olemasoleva projekti kasutuselevõtmist kirjeldab autor järgnevas peatükis.

2.3.1 Mobiilirakenduse projekti allalaadimine

Mobiilirakenduse arendamisel on võimalik alustada uue projektiga või kasutada juba olemasolevat projekti. Tarkvara arenduses laetakse projekt tavaliselt üles internetti repositooriumisse. Repositooriumi keskkonnast lähtuvalt võib projekti allalaadimine olla erinev. Antud juhul on repositoorium GitHub keskkonnas, https://github.com/ArviKaasik/RoboticsApp veebiaadressil. GitHub keskkkonnast saab projekti allalaadida klõpsates projekti lehel “Download ZIP” nupul. Järgnevalt tuleb .zip fail allalaadida ning lahti pakkida sinna, kus projekti soovitakse hoida.

Android Studio programmi esmakordsel käivitamisel näidatakse ekraani, kus on võimalik valida, kuidas projekti avada. Esmakordsel käivitamisel soovitab autor kasutada uue projekti loomise asemel eelnevalt mainitud repositooriumist kättesaadavat projekti. Olemasoleva projekti kasutamiseks tuleb klõpsata nupul "Open an existing Android Studio project". Avanevas aknas tuleb leida kaust, kuhu pakiti lahti repositooriumist allalaetud projekt, valida projekti kaust ja klõpsata "OK" nupul. Android Studio võtab järgnevalt projekti kasutusele. Juhul, kui projekti kasutusele võtmisel näidatakse veateateid, uuenda tarkvara komponente vajutades veateadete linkidel või taaskäivita programm. Kui projekti avamise aken ei avane, on võimalik projekt avada tööriistaribalt. Selleks tuleb vajutada „Fail" → „Open" ja edasi valida repositooriumi projekti kaust. Uue projekti loomiseks tuleb klõpsata „Fail" → „New" → „New Project", järgnevalt tuleb järgida viisardi juhendeid ja valida toetatavad seadmed ning esimene avatav tegevus (*Activity*). Tegevustest kirjutatakse täpsemalt peatükis 2.4.3 Tegevused.

Töökeskkonna ja projekti loomise või olemasoleva kasutusele võtmisega on kõik eeldused mobiilirakenduse arendamiseks täidetud ning saab alustada rakenduse arendamisega.

2.4 Mobiilirakendus

Enne Androidi platvormil mobiilirakenduste kirjutamist on oluline teada mõningaid platvormi spetsiifilisi reegleid, mida tutvustatakse järgnevas peatükis. Android Studios kasutatakse mobiilirakenduse kirjutamisel kahte keelt: XML ja Java.

XML-i kasutatakse rakenduses peamiselt kujunduse kirjeldamiseks, kujundite joonistamiseks ja erinevate projekti siseste väärtuste kirjeldamiseks. Lisaks on XML-is kirjutatud ka projekti manifest fail, milles kirjeldatakse projektiks oluline info (Täpsemalt kirjutatakse manifesti failist 2.4.1 Manifest peatükis).

Java keelt kasutatakse Android rakenduse selliste komponentide programmeerimiseks, mida XML ei kata. Kujundus on võimalik kirjutada Java keeles XML-i asemel, aga see võib osutuda algaja programmeerija jaoks keerukamaks. Teine põhjus, miks Javas ei ole

kujunduse kirjutamine nii hea mõte, kui XML-is on see, et siis on kõik kujundus ülejäänud koodist rohkem lahus ning projekti on lihtsam hallata.

2.4.1 Manifest

Projekti lähtekaustas asub projekti jaoks väga oluline "AndroidManifest.xml" fail. Selles failis kirjeldatakse kõik mobiilirakenduse jaoks oluline, sealhulgas:

- *Java* projekti spetsiifiline identifikaator.
- Rakenduse komponendid, mille hulgas on tegevused (tegevustest on kirjutatud täpsemalt peatükis 2.4.3).
- Rakendusele antavad õigused.
- Rakenduse nimi ja ikoon.
- Rakenduse stiili fail.
- Kõige madalam Androidi operatsioonisüsteemi versioon, millel on võimalik mobiilirakendust käivitada. [3]

2.4.2 Ressursid

Ressursside all käsitletakse Androidi mobiilirakenduste puhul erinevate ekraanikuvade kujundust, pilte, stiile. Lisaks defineeritakse ressursside kaustas erinevad projektis kasutatavad väärtused, näiteks erinevad suurused, sõned ja kasutatavad värvid. Järgnevates alampeatükkides kirjeldatakse ressursi tüübi järgi erinevaid ressurrse. Androidi mobiilirakenduste projektides hoitakse ressursse */res* alamkaustades.

2.4.2.1 Pildid ja kujundid

Erinevad pildifailid või XML-is kirjeldatud kujundid on *Drawable/* kaustas. Android toetab *bitmap* tüüpi failiformaatidest *.png*, *.jpg* ja *.gif* faililaiendeid, kusjuures *.png* failiformaat on eelistatuim ning *.gif* faililaiendit ei soovitata kasutada [4]. *Bitmap* tüüpi failid on sellised failid, kus salvestatakse pilt mällu bittide ehk arvude 1 ja 0 jadana. Lisaks on võimalik Androidi puhul kasutada .9.png faile, mille puhul saab määrata venitatav

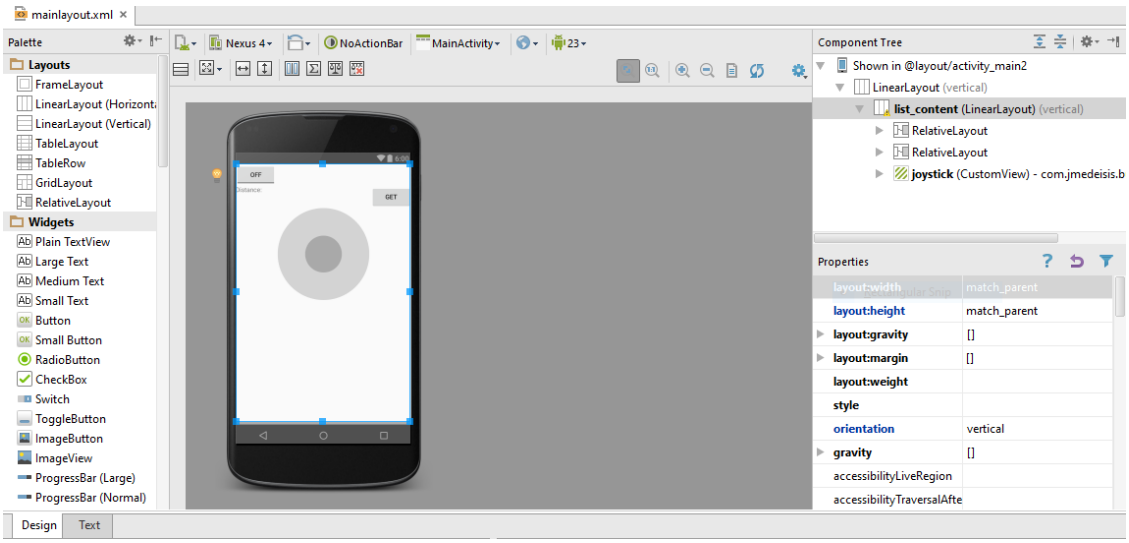
piirkond ja tänu sellele on võimalik väikesi pilte osaliselt venitada [5]. Nii on võimalik luua jutumulle, mis mahutavad täpselt enda sees paikneva teksti.

Drawable/ kaustasid võib olla ka mitu. Nimelt on Androidi nutiseadmetele erinev pikslitihendus, mis tähendab, et kahel seadmel võib olla ühe ruutsentimeetri peal erinev arv piksleid. Seetõttu venitab Androidi operatsioonisüsteem vaikimisi pildi õigesse suurusesse, mis võib aga tekitada graafilisi vigu. Selleks, et pildid oleksid teravad iga pikslitiheduse puhul, on võimalik määrata iga pikslitihedusega eraldi *Drawable/* kaust lisades kausta nimele "-" ja pikslitiheduse tähis, näiteks *Drawable-xxxhdpi/*. [6]

2.4.2.2 Kujundus

Nagu käesolevas töös on üleval pool mainitud, kasutatakse XML-i ka kujunduse (*layout*) kirjeldamiseks. Lisaks kujunduse kirjeldamisele XML keeles, on võimalik kujundust kirjeldada kasutades graafilist liidest. Kujundust kirjeldavaid XML-faile hoitakse */layout* kaustades. Kujundust on väga mugav Android Studios arendada, sest *preview* aknas kuvatakse reaalajas ligikaudne pilt lehe väljanägemisest.

Graafilise liidese (Joonis 2) abil on algajatel programmeerijatel oluliselt lihtsam mobiilirakenduse kujunduse ja ka seal olevate komponentide lisamine ning paigutuse muutmine. Selle avamiseks ava kujunduse fail klõpsates sellel 2 korda Android Studios ja kontrolli, et all vasakus ääres oleks "Design" nupp valitud. Elementide lisamiseks lohista vasakul olevast aknast pealkirjaga "Palette" elemente telefonikujutisele. Vastavalt asukohale, kuhu elemendi liigutasid genereeritakse automaatselt XML kood. Elementi saab valida klõpsates otse elemendil telefoni kujutisel või vajutades vastaval elemendil *Component Tree* aknas (asub paremal üleval nurgas). Elementi valides avaneb *Component Tree* akna all *Properties* aken. Seal on võimalik kõike natukene täpsemalt kirjeldada.



Joonis 2: Kujunduse *designer* vaade

Kuigi graafilise liidese abil on võimalik luua erinevaid vaateid ja seal sees liigutada erinevaid elemente, võib selle abil täpse kujunduse tegemine osutuda väga raskeks või võimatuks. Detailseks kujunduseks soovitab autor seetõttu XML koodis kujunduse tegemist. Koodis kujunduse tegemiseks tuleb klõpsata “Design” kõrval olevat “Text” nuppu. Lihtsaim kujunduse XML fail (Joonis 3) sisaldab endas *Layout* tüüpi, mis omakorda sisaldab teisi elemente. Väline Layout sisaldab endas alati rida „xmlns:android=”http://schemas.android.com/apk/res/android””.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Joonis 3: Näidis *layout* XML koodist [7]

Detailsema ja täpsemalt defineeritud kujunduse struktuur võib minna oluliselt keerulisemaks, kui joonisel 3 toodud näites. Elemendid saab alati asendada kujunduse kirjeldusega ning sinna omakorda paigutada kujundusi või elemente. Seda võtet kasutatakse ka selle bakalaureusetöö raames kirjutatud näidisrakenduses.

Näidisrakenduses realiseeritud kujunduse tüübid on *LinearLayout* ja *RelativeLayout*. *LinearLayout* järjestab kõik enda sees deklareeritud kujunduse elemendid vastavalt *android:orientation* väärtuse suunale vertikaalselt või horisontaalselt. *RelativeLayout* võimaldab paigutada elemente üksteisest sõltumatult. Näiteks võimaldab see tüüp paigutada elemendi ekraani kõige alumisse serva. Selle tüübi puhul tasub arvesse võtta, et kujundusel, mis ühel ekraanil näeb hea välja, võib mõnel kitsamal ekraanil elemendid üksteist katta.

2.4.2.3 Menüüd

Menüü on üks tavalisemaid kasutajaliidese elemente mobiilirakenduste puhul. Menüü kirjutatakse XML-is ja see paikneb */menu* kaustas.

Kuigi Android seadmetelt, mille operatsioonisüsteem 3.0 või uuem ei nõuta enam menüü riistvara nupu olemasolu ja traditsionaalsest menüü avamisest igal pool on loobutud, ei tähenda see, et Androidi menüüd ei oleks enam päevakohased [8]. Praegune Androidi mobiilirakenduste arenduse suund on selline, et igale tegevusele, mis ei ole üle terve ekraani (nagu näiteks mängud või sellised rakenduse tegevused, kus näidatakse üle ekraani pilti) lisatakse ekraani ülemisele äärele tööriistariba. Tööriistaribale lisatakse vajalikud nupud ja antud kontekstis ka menüü nupp. Tööriistaribast tuleb täpsemalt juttu peatükis 2.4.3.2.

2.4.2.4 Menüü loomine

Menüü loomiseks peab programmeerima menüü kujunduse (Joonis 4). Hiljem kasutatakse seda kujundust menüüsse elementide lisamiseks. Menüü kujunduse sisuks on menüü enda kirjeldus ning menüü sisuks olevate valikuvariantide kirjeldus ja kujundus. Menüü

elemendid tähistatakse märksõnaga *item* ja nende puhul peaks defineerima pealkirja kasutades *android:title* märksõna, id kasutades *android:id* märksõna. Elementide järjekorra määramiseks kasutatakse *android:orderInCategory* märksõna. Järjestus toimub nii, et kõige väiksema arvuga element on kõige üleval ja alla tulevad järjest kasvavate numbritega elemendid.

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:app="http://schemas.android.com/apk/res-auto"
3      xmlns:tools="http://schemas.android.com/tools"
4      tools:context="com.app.robotics.roboticsapp.MainActivity">
5      <item
6          android:id="@+id/action_choose_connection"
7          android:orderInCategory="100"
8          android:title="Choose Connection"
9          app:showAsAction="never" />
10     <item
11         android:id="@+id/action_choose_sensor_mode"
12         android:orderInCategory="101"
13         android:title="Choose Sensor Mode"
14         app:showAsAction="never" />
15 </menu>
16
```

Joonis 4: Menüü kujunduse näidis

Lisaks peab kirjutama üle tegevuse klassis "**public boolean** onCreateOptionsMenu(Menu menu)" meetodi kasutades Java puhul *@Override* käsku. Ülekirjutuses peame *MenuInflater* klassi abil menüü kujunduse kasutusele võtma ja tagastama *true*. Näidiseks vaata joonist 5.

```
183     @Override
184     public boolean onCreateOptionsMenu(Menu menu) {
185         // Inflate the menu; this adds items to the action bar if it is present.
186         getMenuInflater().inflate(R.menu.menu_main, menu);
187         return true;
188     }
```

Joonis 5: onCreateOptionsMenu (Menu menu) ülekirjutamise näidis

Android 3.0 või uuema operatsioonisüsteemi kasutatakse onCreateOptionsMenu (Menu menu) meetodit tegevuse loomisel, et näidata eelnevalt kirjeldatud kujundust. *True* väljastame selleks, et meie eest ära kirjutatud tegevuses kirjutatud kood saaks aru, et me

oleme andnud menüü kujunduse ning et on tarvis menüü luua. Ülejäänud menüü ehitamise teeb süsteem juba ise ning rohkem midagi implementeerima ei pea.

2.4.2.5 Menüü valikutele reageerimine

Sarnaselt menüü loomisele reageeritakse menüüs valikute tegemisel juba olemas oleva meetodiga. Menüü valikutele reageerimise puhul on selleks meetod "**public boolean** onOptionsItemSelected (MenuItem item)" (Joonis 6).

```
197     @Override
198     public boolean onOptionsItemSelected(MenuItem item) {
199         //...
202         int id = item.getItemId();
203
204         //noinspection SimplifiableIfStatement
205         if (id == R.id.action_choose_connection) {
206             if (!this.isFinishing()) {
207                 urlAlert.show();
208             }
209             return true;
210         } else if (id == R.id.action_choose_sensor_mode) {
211             if (!this.isFinishing()) {
212                 CheckCorrectBullet();
213                 sensorAlert.show();
214             }
215             return true;
216         }
217         return super.onOptionsItemSelected(item);
218     }
```

Joonis 6: onOptionsItemSelected (MenuItem item) ülekirjutamise näidis

Meetodisse antakse kaasa *MenuItem* tüüpi *item*. Selle *item*'i järgi saab kasutaja meetodis otsustada, kuidas reageerida. Näiteks saab võrrelda *item*'i *id* järgi, millise menüü valikuvariandi kasutaja valis ning vastavalt sellele navigeerida mõnda teise tegevusse või teha midagi muud, mida programmeerija soovib, et see valik teeks. Sarnaselt onCreateOptionsMenu (Menu menu) meetodile peab väljastama *True*, kui me valiku valimisele reageerisime ning *False*, kui me mingil põhjusel midagi ei teinud. Selline olukord võib tekkida näiteks arenduse käigus, kui ununeb lisada

2.4.2.6 Väärtused

Eelnevalt defineeritud väärtusi, näiteks sõnesid, värve ja suurusi hoitakse */values* kaustas. Väärtused kirjutatakse XML-is ja nende poole on võimalik pöörduda täpselt samamoodi nagu ülejäänud ressursside puhul.

2.4.2.7 Ressursside kasutamine

Rakenduse kompilleerimise käigus loob *aapt* tööriist iga ressursi kohta eraldi ID, millele viidatakse koodis, kui ressurssi kasutatakse. Koodis ressursside kasutamiseks tuleb meetodi argumendiks väärtustada vastava ressursi ID ja Androidi meetodid kasutavad neid ID-sid ressursside leidmiseks. Ressursside ID-d salvestab *aapt* "R" klassis tüübi sees. ID kätte saamise valem on R.tüüp.ressursi_id.

Näiteks Android Studios Java koodis Strings.xml (Joonis 7) sees deklareeritud "Choose sensor send type" sõne kasutamiseks on võimalik kasutada koodirida: "String titleString = getString(R.string.***sensor_alert_title***);".

```
1   <resources>
2       <string name="app_name">RoboticsApp</string>
3       <string name="large_text"...>
92      <string name="action_settings">Settings</string>
93      <string name="sensor_alert_title">Choose sensor send type</string>
94  </resources>
```

Joonis 7: Ressursifaili näide Strings.xml

Sellega on peamised ressursside tüübid ja nende kasutusviisid ülevaatlikult kirjeldatud. Järgnevas peatükis kirjeldab autor, mis on tegevused, miks need on olulised, milline on tegevuse elutsükkel ja kuidas on võimalik tegevuse elutsüklit ressursside kokkuhoidmisel ära kasutada.

2.4.3 Tegevused

Androidi mobiilirakendustel on väga olulisteks komponentideks tegevused. Tegevuse abil on võimalik kasutajal teha kindlaid operatsioone näiteks video vaatamine ja kaamera abil pildi jäädvustamine. Kuna suurem osa tegevusi suhtlevad kasutajaga, loob tegevuse klass ka kasutajaliidese akna. Enamasti katab tegevus terve nutiseadme ekraani. Kõik tegevused, mida kasutaja soovib enda rakenduses kasutada, peavad olema manifesti failis deklareeritud kujul: <**activity**> tegevuse kirjeldus </**activity**>. [9]

Dünaamiliselt tegevuse sisu vahetamisel kasutatakse tavaliselt kilde (*Fragment*), mille abil on võimalik teha keerukamat ja rohkem arenenud kasutajaliidest ning lisada tegevusse rohkem funktsionaalsust. Selles bakalaureusetöös kilde ei kasutata, kuna roboti juhtimine nutiseadme abil ei nõua eriti dünaamilist kasutajaliidest ning kildude implementeerimine võib olla algaja programmeerija jaoks liialt keerukas. [10]

Igal tegevusel on 2 peamist meetodit:

- protected void onCreate (Bundle savedInstanceState), mis kutsutakse välja, kui tegevus pannakse tööle. Selles meetodis peaks programmeerija setContentView(int LayoutResId) abil kujunduse XML faili deklareerima ning findViewById(int id) abil võtma kasutusele kujunduses olevad elemendid, et tegevuse koodis oleks võimalik nende elementidega midagi teha. [11]
- protected void onPause (), mida kutsutakse välja, kui kasutaja navigeerib tegevusest minema. Kui on palju tegevusi, oleks mõistlik siin salvestada kasutaja poolt tehtud muudatused. Hiljem selle tegevuse uuesti alustades saab need muudatused kasutusele võtta.

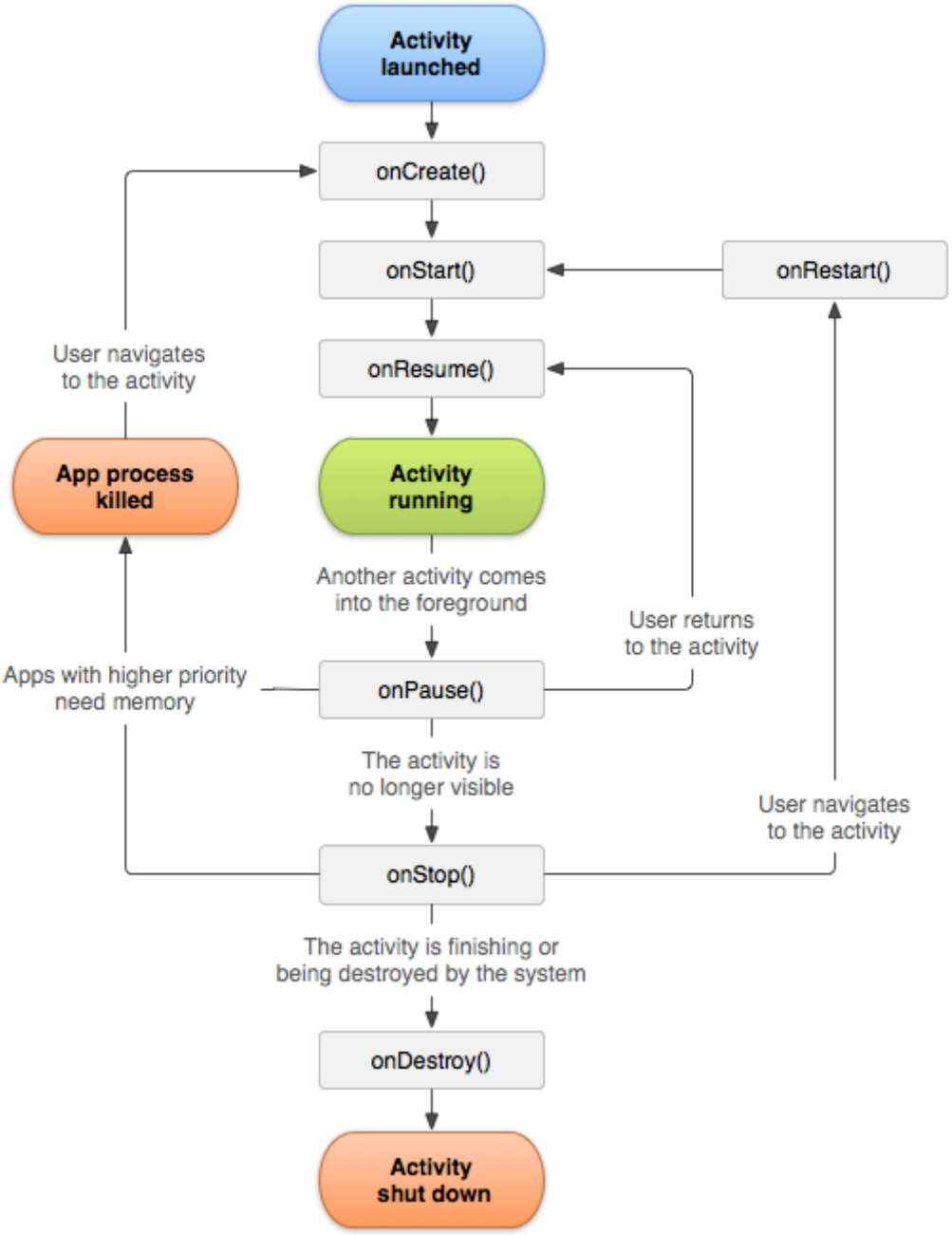
2.4.3.1 Tegevuse elutsükkel

Üks mobiilirakendus võib kasutada mitmeid tegevusi ja neid on vaja hallata. Android mobiilirakendustes on selleks süsteemi pinu (*system stack*). Kui käivitatakse üks tegevus, liigub see süsteemi pinu kõige pealmiseks elemendiks ja saab töötavaks tegevuseks. Eelmine tegevus jääb pinus seniks sellest tegevusest alumiseks ja ei ole ekraanil, kuni viimati lisatud tegevust kinni ei panda. [12]

Tegevusel on neli erinevat seisundit:

- Tegevus on ekraani peal ja järelikult süsteemi pinu kõige pealmine tegevus. See tegevus on töötav või aktiivne tegevus.
- Tegevus ei ole enam fookuses, kuid on jätkuvalt nähtav (selline olukord esineb, kui pealmine fookuse saanud tegevus on läbipaistev või ei kata veel täielikult eelnevat tegevust). Selline tegevus on pausil (paused) ning kõik selle tegevuse andmed hoitakse mälus. Oluline on arvesse võtta, et juhul, kui rakendusel on kriitiliselt vähe vaba mälu, võib süsteem selle tegevuse hävitada ja mälu kasutuseks vabastada.
- Tegevus on täielikult kaetud mõne teise tegevusega ning on peatatud (stopped). Selline seisund on sarnane pausi staatusega ning ainuke erinevus on, et süsteem hävitab sellise tegevuse vabamalt, kui pausi staatuses tegevuse.
- Kui tegevus on peatatud või pausil staatusel, võib süsteem selle mälu vabastada ning kõik jooksvad protsessid lõpetada. Kui tulla tagasi sellises staatuses tegevusse, peab tegevuse taaskäivitama. [12]

Tegevuse käivitamisest kuni tegevuse sulgemiseni läbib tegevus kõiki neid staatuseid ning tegevuse eluea jooksul lülitatakse erinevasse staatusesse ümber lülitamisel erinevaid meetodeid (Joonis 8).



Joonis 8: Tegevuse elutsükkel

Vastavalt tegevuse elutsüklile ja programmeeritud funktsionaalsusele võib programmeerija joonise 8 meetodites kindlaid operatsioone teostada. Näiteks võib programmeerija soovida tegutseda nii:

1. Videoedastamise nupule vajutades meelde jätta, et videot edastatakse väärtustades tegevuse boolean tüüpi väärtuseks *True*.
2. onPause () meetodis peatada video edastamine.
3. onResume () meetodis kontrollida, kas punktis 1 mainitud boolean tüüpi väärtus on *True* või *False* ning *True* puhul taaskäivitada video edastamine.

Videoedastuse ajutine peatamine vähendab nutiseadme ressursside kulutamist säästes seadme akut.

2.4.3.2 Tegevuse tööriistariba

Tegevuse tööriistariba (*action bar* või *App bar*) on tegevuse külge lisatud tööriistariba. Seal hoitakse selliseid nuppe, millega Android mobiilirakenduste kasutajad on tuttavad teistest rakendustest. Peamised tööriistariba elemendid on ligipääs kindlale funktsionaalsusele (näiteks otsing), navigatsiooni võimaldamine ja vaadete vahetamine. Selline lähenemine lisab rakendusele järjekindlust. Antud bakalaureusetöö näidisrakenduses kasutatakse tööriistariba menüü avamiseks. [13]

Tööriistariba on võimalik lisada mitmel võimalusel. Autor kasutas näidisrakenduses Androidi *v7 appcompat* teegist *Toolbar* klassi, millele on lihtne tulevikus uut funktsionaalsust juurde lisada [13].

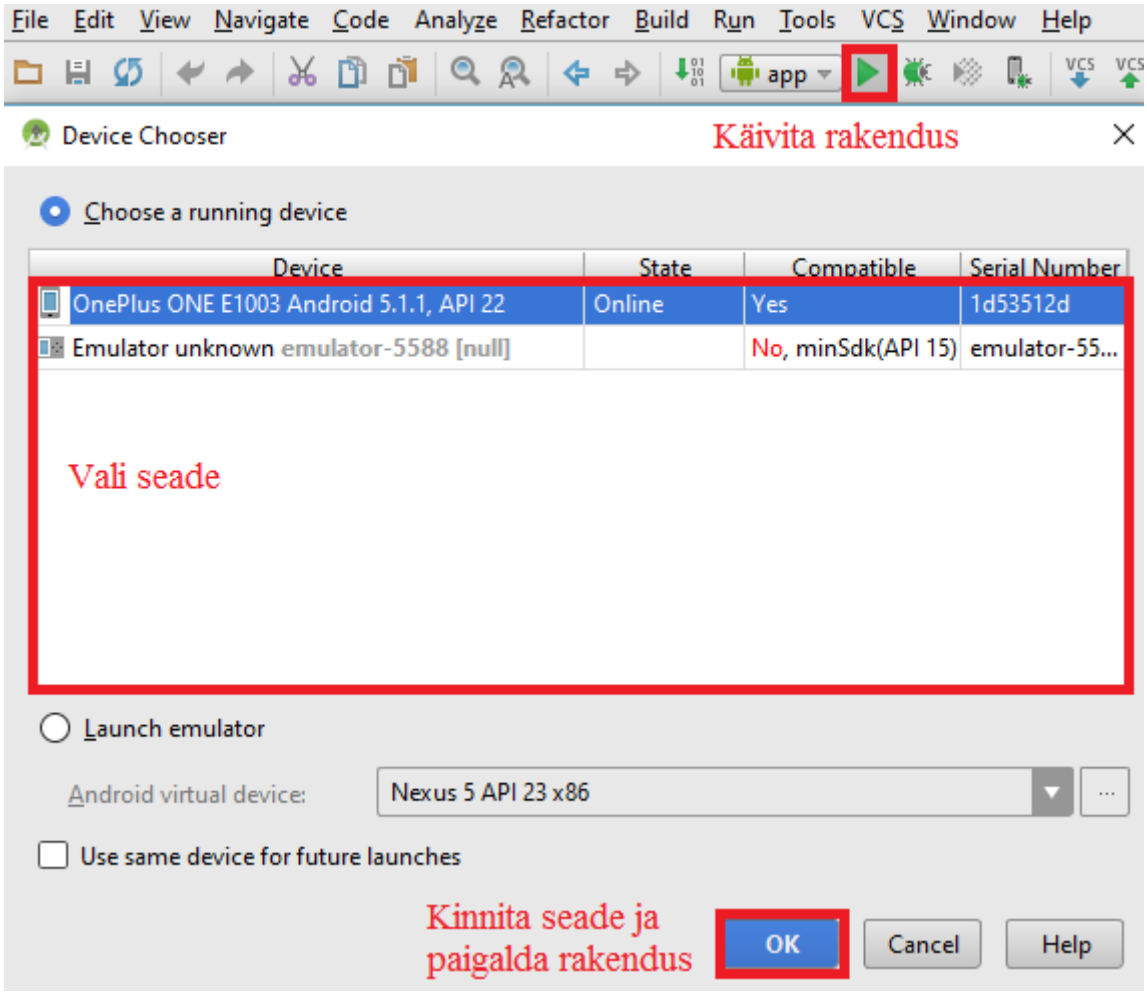
2.5 Mobiilirakenduse käivitamine

Rakenduse arendamine võib olla küll huvitav kogemus, kuid lõpuks peab kuidagi kasutaja mobiilirakenduse enda nutiseadmesse paigaldama. Arenduse käigus peab tihti tegema palju muudatusi ning seetõttu ei ole mõistlik rakendust alati kokku pakkida ning *Google Play Store* teenusesse üles laadida. Sel põhjusel on võimalik lähtekoodi olemasolul ehitada rakendus lokaalselt valmis ja salvestada nutiseadmele.

Selleks on vajalik kõigepealt seadmel lubada arendusrežiim. Kahjuks erineb see tegevus igal tootjal ja igal Androidi versiooni. Android 5.0 ja 5.1 puhul on paljudel seadmetel võimalik seda teha minnes „Settings“, „About Phone (Tablet)“ ja vajutada 7 korda nuppu

„Build number“. Kui ekraanile on tulnud tekst „You are now a developer!“, on kõik hästi. Järgmiseks on vaja lubada USB silumisrežiim, selleks vajuta „Settings“, värskelt tekkinud „Developer options“ ja edasi lülita „USB debugging“ nupp sisse asendisse. Järgnevalt luba USB silumine vajutades „OK“ nuppu. Nüüd peaks iga kord, kui ühendada nutiseade arvuti külge üle USB kaabli, tekkima dialoog-aken, kus küsitakse kas lubada selle arvuti puhul USB silumine. Lisaks näidatakse seal ka arvuti RSA võtit. Selleks, et Android Studio nutiseadme ära tunneks peab lubama arvutiga USB silumine.

Kui silumine on arvutiga lubatud, peaks saama rakendust paigaldada seadmele. Selleks vajuta rakenduse käivitamise nuppu (Joonis 9), vali seade ja kinnita valik. Järgnevalt on vaja oodata kannatlikult, kuni Android Studio pakib rakenduse kokku vastavalt nutiseadmel olevale riistvarale ja tarkvarale ning paigaldab rakenduse nutiseadme peale. Rakenduse paigaldamise aeg võib võtta seadmest olenevalt kuni 10 minutit ning tasub varuda kannatlikust. Kui rakendus on nutiseadmele paigaldatud avaneb rakendus automaatselt ja peaks olema kasutamiskõlbulik.



Joonis 9: Rakenduse käivitamine

Selle peatüki lõpuks peaks lugeja suutma rakendust käivitada. Järgmises peatükis kirjeldatakse täpsemalt demorobotit ja juurde käivat näidisrakendust. Autor soovitab esimese mobiilirakendusena käivitada koodist kas selles bakalaureusetöö käigus arendatud näidisrakendust või mõnda Android Developer kodulehel näidisena tehtud rakendust.

3. Näidisrakendus ja Demorobot

Käesoleva bakalaureusetöö käigus arendas autor näidisrakenduse ja demoroboti, mida on võimalik mobiilirakenduse abil juhtida. Nii näidisrakenduse kui ka demoroboti kood on täielikult kätte saadavad GitHubi repositooriumis lingil https://github.com/ArviKaasik/RoboticsApp. Juhend kuidas repositooriumist projekt allalaadida ja seda kasutada on peatükis 2.1.3 Mobiilirakenduse projekti allalaadimine.

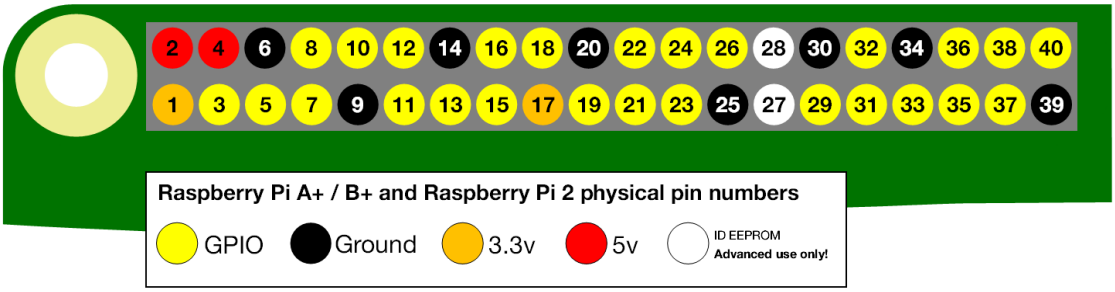
Järgnevates alampeatükkides kirjeldab autor, kuidas demorobot kokku panna ning programmeerida ning kuidas demoroboti juhtimiseks tehtud näidisrakendust kasutada.

3.1 Demorobot

Demoroboti aluseks võttis autor NUTIROBOTi projekti (kättesaadav veebilehel "http://nutirobot.ut.ee "), mis on omakorda Taavi Karelsoni 2015. aasta bakalaureusetöö „Nutiseadmest kaugjuhitava roboti ehitamine Raspberry Pi näitel" (kättesaadav veebilehel „http://dspace.ut.ee/handle/10062/50449") edasiarendus. NUTIROBOT on robot, mis suudab sõita kahe servo abil ning lülitab põlema vastavalt liikumissuunale punase või sinise LED tule. NUTIROBOTi tööd kontrollib Raspberry Pi (juhendi järgi Raspberry Pi 2 mudel B+) arvuti. Suhtlus toimub üle WiFi võrgu ning juhtimiseks kasutatakse HTML5 keeles kirjutatud veebilehte. Veebilehel loetakse NUTIROBOTi juhi nutiseadme güroskoobi andmeid, need edastatakse Raspberry Pil töötavale serverile, mis töötleb neid andmeid ning muudab andmed servode jaoks sobivaks suuruseks.

Raspberry Pi on odav ja väike arvuti, mille on omaks võtnud paljud robootikahuvilised. Peamised põhjused, miks Raspberry Pi arvutid on nii populaarsed on suuruse ja hinna kohta suur võimekus ning suur hulk viike, mida saab kasutada erinevate andurite ja mootoritega suhtlemiseks. Raspberry Pi 2 siinid on nummerdatud kindlal printsiibil. Kui asetada Raspberry Pi lauale nii, et siinid jäävad plaadi peale ja kõik arvuti peal olevad kirjad on õiget pidi loetavad, on siini numbrid järjest kasvavad vasakult paremale. Siine on

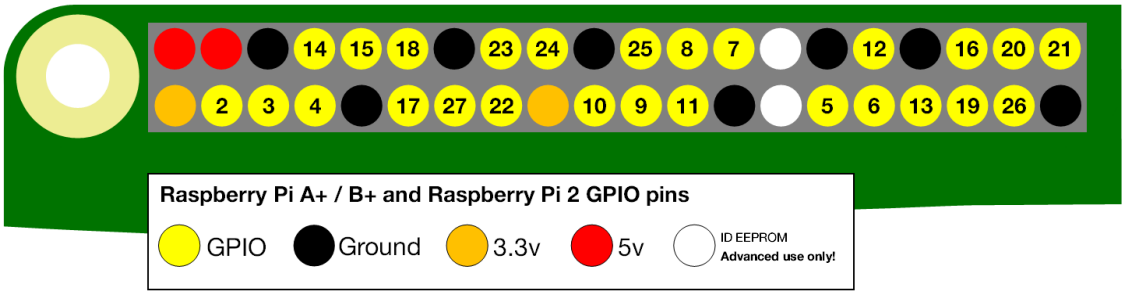
kaks rida ja paaritu numbriga siinid on eelnevalt mainitud asetuse puhul alumisel real (Joonis 10).



Joonis 10: Raspberry Pi 2 siinide numeratsioon [14]

Viike on kokkuvõttes nelja tüüpi (Joonis 11):

- Maandussiinid, mis on tavaliselt joonistel märgitud musta värviga.
- 3.3V väljundsiinid, kust saab 3.3V pingega voolu.
- 5V väljundsiinid, kust saab 5V pingega voolu.
- GPIO siinid, mida kasutatakse ühendatud seadmete töö reguleerimiseks pinge muutmise abil ja ka seadmetelt andmete lugemiseks.



Joonis 11: Raspberry Pi GPIO siinide numeratsioon [14]

Järgnevas peatükis toob autor välja demorobotis autori poolt realiseeritud funktsionaalsusesed ja seadmed, mis neid funktsionaalsusi täidavad.

3.1.1 Funktsionaalsus

NUTIROBOTi projektis lisatakse robotile baasfunktsionaalsus (liikumine ja tuled), aga on võimalik lisada oluliselt rohkem andureid ja seadmeid. Selle töö käigus lisas autor demorobotile juurde USB ühendusega veebikaamera video edastamiseks ja ultraheli kauguse mõõtja roboti tagant kauguse mõõtmiseks. Kuna kaamera on demoroboti külge fikseeritud ning sellega ei ole näha roboti taha, aitab kaugusandur mõõta objekti kaugust demoroboti taga. Lisaks seob autor näidisrakenduses LED tule sisse- ja väljalülitamise nupuga.

Raspberry Pi-l on küll olemas eraldi kaameramoodul, kuid autor eelistas kasutada selle asemel USB veebikaamerat. Veebikaamera suurim eelis selle bakalaureuse töö suhtes on kasutatavus ja levik. Nimelt on võimalik veebikaamerast pilti edastada kõigi arvutite abil, millel on USB port ning millel on veebikaamera jaoks sobivad draiverid paigaldatud. Nimekiri veebikaamerate mudelitest, mida Raspberry Pi toetab asub veebilehel http://elinux.org/RPi_USB_Webcams. Selle nimekirja puhul peaks tähele panema, et punasega märgitud mudelid töötavad, aga võib esineda veateateid. Käesolevas töös kasutatakse Logitech C270 mudeliga veebikaamerat.

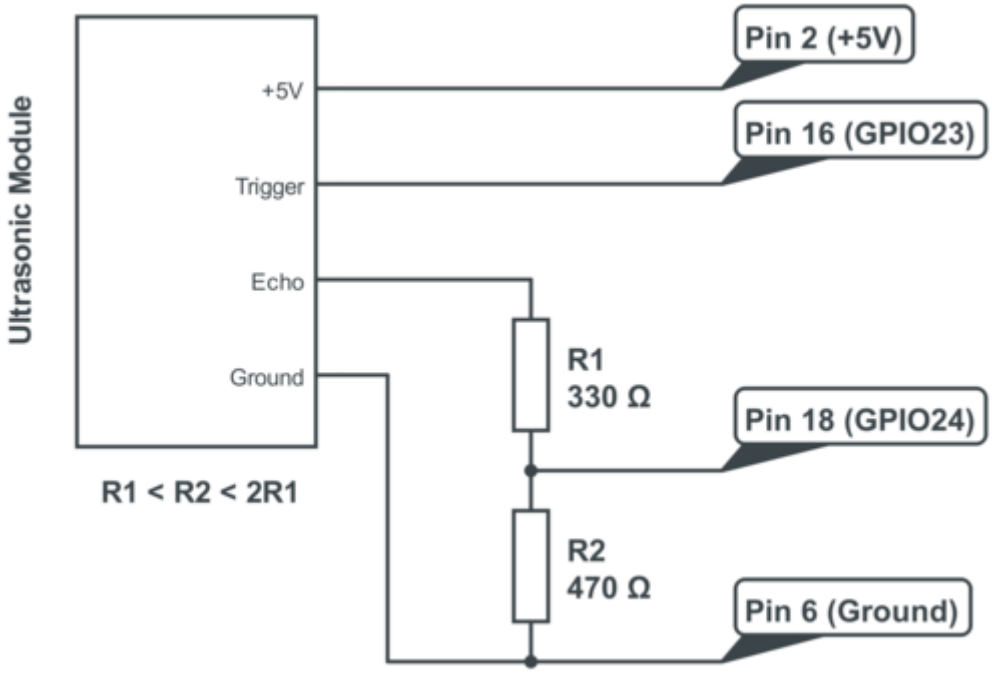
Kaugusandurina kasutas autor demoroboti kokku panemisel HC-SR04 andurit. Selle anduri minimaalne mõõtmiskaugus on 2 sentimeetrit ja maksimaalne mõõtmiskaugus on 4 meetrit. Kaugusandur töötab põhimõttel:

- *Trigger* siinile antakse lühike, 10 mikrosekundit kestev signaal.
- Moodul saadab 8 tsüklilise ultraheli sagedusel 40 kHz.
- *Echo* võtab vastu peegelduva heli.
- Mõõdetakse helikiiruse ja möödunud aja abil objekti kaugus andurist.

Üldiselt on vahemaa arvutamise valem aeg * kiirus. Arvesse tuleb võtta, et saadetud ultraheli läbib objekti kaugust mõõtes mooduli ja objekti vahemaa kaks korda (esimene kord, kui liigub objektini ja teine kord, kui liigub tagasi moodulisse). Seega uus valem on:

(aeg * kiirus) / 2. Antud olukorras on kiirus helikiirus (umbes 340 m/s) ja aeg mooduli poolt mõõdetav aeg. Järelikult vaja minev valem on: (mõõdetud aeg * 340) / 2. Demorobot arvutab kaugust sentimeetrites: (mõõdetud aeg * 34000) / 2. Anduri kõige täpsemaks mõõtmiseks soovitab tootja, et mõõdetav objekti külg, mis on anduri poole, pindala oleks vähemalt 0,5 m^2. [15]

Eelnevalt mainitud kaugusanduri mudeli juhtmete ühendamisel tuleb arvestada, et seade vajab 5V pingega vooluallikast voolu. Lisaks on seadme ühendamisel tarvis ühendada *ECHO* ja *Trigger* juhtmed ning maanduse. Saadetav *trigger* signaal võib olla 3.3V pingega ja neid on viike on mitmeid. Demorobotis ühendas autor *trigger* juhtme viigu numbriga 38 (GPIO20) külge. *Echo* ühendamine on keerulisem, nimelt on sellel kaks väärtust: madal (0V) ja kõrge (5V). Siini pinget hoitakse madralal kuni moodul ootab kauguse arvutamist ning siis tõstetakse pinge kõrgeks. Kõrgena hoitakse pinget nii pikalt, kui kaua kulus aega kauguse mõõtmiseks. Järelikult peame me suutma mõõta Raspberry Pi siini pinget 5V, et leida mõõtmiseks kulunud aeg. Peamine probleem on, et Raspberry Pi siinid, mida on võimalik kasutada pinge mõõtmiseks saab tõsta maksimaalselt umbes 3.3V suuruseks. Seetõttu peame mooduli poolt saadetava signaali pinge väiksemaks tegema. Üks viis, kuidas seda teha on jagada pinge takistite abil. Ühe takisti (R1) peaks lisama *Echo* ja loetava siini vahele ning teise takisti (R2) maanduse ja loetava siini vahele (Joonis 12). Kui kasutada võrdseid takisteid, jagatakse pinge kaheks ja Raspberry Pi siini jõuaks 2.5V. Kui R2 on 2 korda suurema takistusega, kui R1, jõuab Raspberry Pi siini 3.33V, mida Raspberry Pi siin juba suudab lugeda. Järelikult peaksime valima takistid nii, et R2 on suurema takistusega, kui R1 ja väiksema takistusega, kui 2*R1. Demorobotis võttis autor kasutusele 330 ja 470 Ω takistusega takistid. [16]



Joonis 12: Ultraheli mooduli ühendamise Raspberry Pi-ga näidis [16]

Selle peatüki sees antud ülevaade aitab mõista robotile lisatud seadmete tööpõhimõtet. Järgnevas peatükis saab näpunäiteid, kuidas see funktsionaalsus realiseerida ning teha robot, mida oleks võimalik näidisrakenduse abil juhtida.

3.1.2 Demoroboti ühendamine

Roboti kõigi komponentide töötamiseks on vajalik, et robotit juhtivas koodis kasutusel olevad siinid kattuksid roboti elektriskeemiga. Seetõttu ei ole Demorobotis tehtud ühendused kohustuslikud ning iga roboti kokkupanija võib kasutada Demorobotist erinevat ühendamisskeemi (Lisa 1). Teistsugusel ühendamisel tuleb tähele panna, mis siinid täidavad näidiskoodis mingit otstarvet ja viia ka koodis sobivad muudatused sisse.

Demoroboti ühendamisel tekkis nii mõneski kohas olukord, kus on vaja kolm juhet kokku ühendada. Üks võimalus on seda teha on lisada robotile maketeerimislaud ning ühendada vajalikud kaablid selle abil kokku. Alternatiiv on lõigata antud juhetmete isoleerkate lahti,

sees olev juhtme sisu kokku keerata, joota ühendus üla ning katta elektrit isoleeriva teibiga.

Järgnev peatükk kirjeldab, kuidas demoroboti juhtimine toimub. Sinna on ka lisatud kaamera kontrollimiseks oluline informatsioon.

3.1.3 Demoroboti komponentide kontrollimine

Sarnaselt NUTIROBOTile juhitakse demorobotit HTTP päringute abil. Autor modifiseeris NUTIROBOTil juba olemasolevat serveri koodi ning lisas servode liigutamise loogikale kaugusandurilt andmete lugemise ja päringule vastamise. Varem lülitati LED tuli sisse või välja vastavalt roboti liikumissuunale. Töö käigus muudeti LED tule loogikat nii, et LED tuli lülitatakse sisse või välja päringuga.

Video edastamiseks üritas autor esialgu kasutada Linux operatsioonisüsteemile arendatud *motion* teenust. Kahjuks video edastamine ei toiminud tõrgeteta, video edastati mõned sekundid ja peale seda ühendus katkes. Lisaks oli video edastamise ajal video pilt väga aeglane. Peale mõningast silumist otsustas autor kaaluda muid variante.

Järgmine lahendus, mida autor proovis oli kasutada OpenCV python teeki veebikaamerast pildi salvestamiseks (Joonis 13) ning flask teeki teise videoserveri (Joonis 14) haldamiseks. Selle serveri eesmärk on edastada MJPEG vormingus videot. MJPEG tähendab, et server saadab kliendile järjest JPEG pilte, kliendi seade vahetab neid pilte järjest luues liikuva pildi illusiooni. Selline lahendus tähendab, et heli ei edastata. Autor otsustas kasutada kaamerast pildi salvestamiseks just OpenCV teeki, kuna tegemist on vabavaralise teegiga ning sellel on toetus väga paljudes erinevates programmeerimiskeeltes (C, C++, Python). Teek on hästi optimiseeritud ning selle kasutamist võib kaaluda ka arvutite puhul, millel on väiksem jõudlus, kui Raspberry Pi-l. [17]

```python
import cv2

class VideoCamera(object):
    def __init__(self):
        self.video = cv2.VideoCapture (0)
        self.video.set(3,320)
            self.video.set(4,240)
        self.video.set(5,30)


    def __del__(self):
        self.video.release()

    def get_frame(self):
        success, image = self.video.read()
        ret, jpeg = cv2.imencode('.jpg',image)
        return jpeg.tobytes()
```

Joonis 13: camera.py

```python
from flask import Flask, render_template, Response
from camera import VideoCamera

app = Flask(__name__)

@app.route("/")
def index ():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
            b'Content-Type:image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
        mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Joonis 14: main.py

Autor määras videopildi resolutsiooniks 240x320 pikslit ning video edastuse sageduseks 30 pilti sekundis (Joonis 13 read 6 - 8). Suuremat resolutsiooni autor vajalikuks ei pidanud, kuna pildikvaliteet ei paranenud oluliselt. Lisaks ei ole vajalik suurema

sagedusega pilti vahetada, kuna 240x320 piksline resolutsioon on juba piisavalt sujuv kiirus ühtlase ühenduse puhul. Autor määras video edastamiseks võrgupordi 5000 ja muude käskude serveri pordiks 8080.

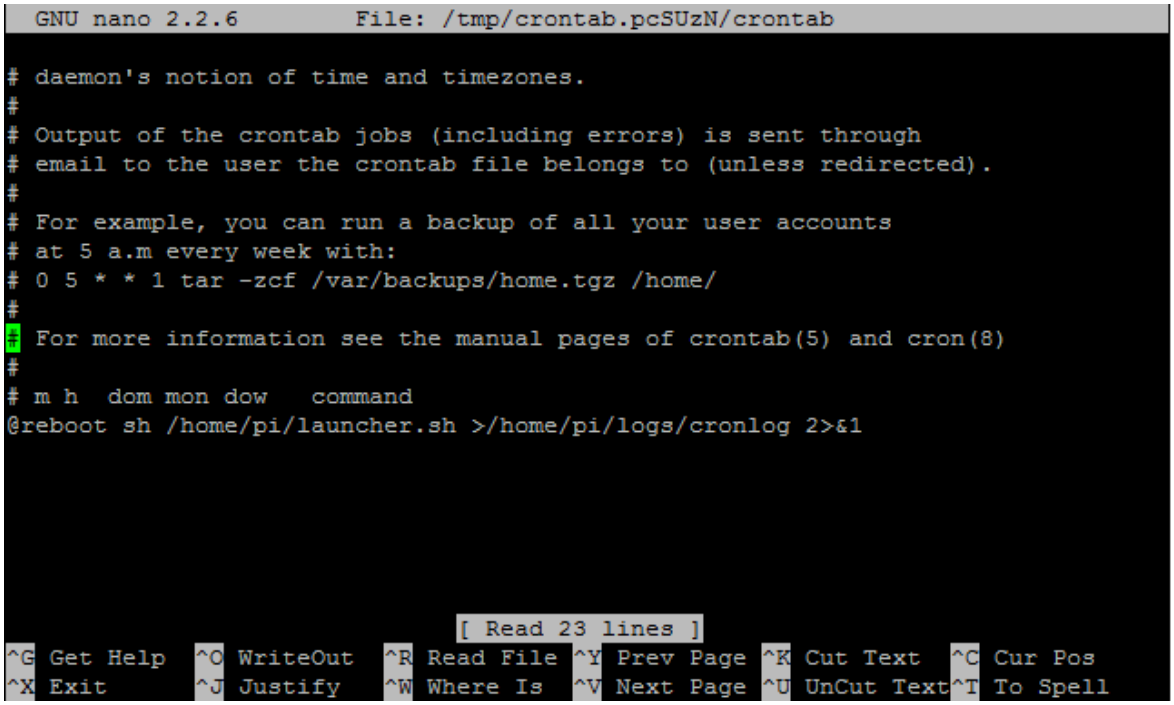
Demoroboti töötamiseks on vaja kolme python faili. Lisaks joonistel 13 ja 14 näidatud main.py ja camera.py failidele on vaja kasutada ka NUTIROBOTi projektist võetud ja autori poolt modifitseeritud NutiRobot.py faili. Kõik need failid on kättesaadavad GitHubi keskkonnas, https://github.com/ArviKaasik/RoboticsApp veebilehel . Kõige lihtsam viis need failid alla laadida on need failid saata enda meilile, lisada Raspberry Pi-le kuvar, hiir ja klaviatuur. Järgnevalt käivitada Raspberry Pi, avada veebibrauseris enda meil ning laadida sealt alla vajalikud failid. Autor soovitab need paigutada enda kodukausta, kasutajanime „pi“ puhul /home/pi.

Ainult main.py, camera.py ja NutiRobot.py failidest ei pruugi aga piisata, sest veebikaamerast pildi saatmise server kasutab pythoni teeke, mis ei ole Raspberry Pil esialgselt kaasas. Kõik selles lõigus kirjeldatud käsud peab käivitama Raspberry Pi käsureal. Flask teeki kasutame selleks, et teha MJPEG formaadis videot edastav server. Flaski teegi allalaadimiseks ja paigaldamiseks kasuta käsku „**sudo pip install flask**“. OpenCV jaoks on kaks erinevat teeki vaja paigaldada. Esmalt kompileeritud baasteek cv2 käsuga „**sudo pip install cv2**“ ja pythoni spetsiifiline teek käsuga „**sudo apt-get install python-opencv**“. Opencv salvestab pildi JPEG formaadis ning selle peab ümber tegema baitide massiiviks ning edastama selle päringus. JPEG faili teeb baidimassiiviks ümber käsk jpeg.tobytes (Joonis 14, rida 17) ning selle käsu toimimiseks on tarvis ka numpy teeki, mis lisab erinevad hästi optimiseeritud ühest andmetüübist teise andmetüüpi ümber kirjutamiseks vajalikud meetodid. Numpy lisamiseks kasuta käske: „**sudo apt-get install python2.7-dev**“ ja „**sudo pip install numpy --upgrade**“. Peale nende teekide paigaldamist saab kasutaja mõlemad serverifailid käivitada ilma vigadeta.

NutiRobot.py on kirjutatud Python3 süntaksiga ja veebikaamera failid main.py ja camera.py on kirjutatud Python2 süntaksiga. NutiRobot.py käivitamiseks töötab kui demoroboti käsureal kirjutada käsk „**sudo python3 NutiRobot.py**“ ja veebikaamera puhul „**sudo python2 main.py**“. Need käsud käivitavad mõlemad serverid ning serverid on siis

valmis päringutele vastama ja robotit juhtima. Camera.py faili ei pea eraldi käivitama, kuna main.py kasutab ise camera.py meetodeid saadetava JPEG pildi tegemiseks.

Mõistlik oleks need serverid automaatselt käima panna iga kord, kui demorobot käivitatakse. NUTIROBOTi projektis käivitatakse NutiRobot.py „/etc/profiles“ failis koodireaga „**sudo python3 NutiRobot.py**“, mis käivitab selle automaatselt. Tekib probleem - kuna programm jääb seda käsku täitma ning mõningaid operatsioone ei jätkata. Näiteks ei olnud autoril seetõttu enam võimalik monitori abil Raspberry Pi tööd juhtida, vaid oli võimalik ainult SSH protokollil ühenduda. Selle vea parandamiseks lisas autor projektile launcher.sh skripti, mis käivitab taustal mõlemad serverid. Launcher.sh skripti käivitamiseks kasutas autor Linuxi *crontab* programmi. Esmalt peab kasutajal olema launcher.sh skript, mille võib kirjutada ise valmis või allalaadida repositooriumis PI kaustast. Näidisskript navigeerib kasutaja pi kodukausta ning käivitab seal olevad serverifailid käskudega „**sudo python3 NutiRobot.py &**“ ja „**sudo python2 main.py &**“. Oluline on ära märkida, et käskude lõppu on lisatud „&“ märgid, mis käsevad skriptil käivitada mõlemad koodiread taustal ilma ühe programmi töö lõppu ootamata. Ilma „&“ märgita käivitatakse esimene server ning teist üritatakse käivita alles siis, kui esimene server oma töö lõpetab. Kui skript on valmis, peab kasutaja käivitama selle skripti *crontab* programmiga. *Crontab* programmi saab avada käsuga „**sudo crontab -e**“. Esimesel korral võib Raspberry Pi küsida, millise tekstiredaktoriga *crontab* avada, autor kasutas *nano* programmi. Järgnevalt avaneva akna lõppu (Joonis 15) peab lisama koodirea „**@reboot sh /home/pi/launcher.sh**“ (eeldusel, et launcher skript asub kasutaja pi kodukaustas).



Joonis 15: *crontab* avanev aken koos skripti käivitamise näidisega

Selles peatükis kirjeldatud tegevuse tulemusel peaks demoroboti tööle lülitamisel automaatselt vajalikud serverid tööle lülituma ning peale võrku ühendumist reageerima mobiilirakendusest tulevatele käskudele. Näidisrakendusest täpsemalt kirjutab autor järgmises peatükis.

3.2 Näidisrakendus

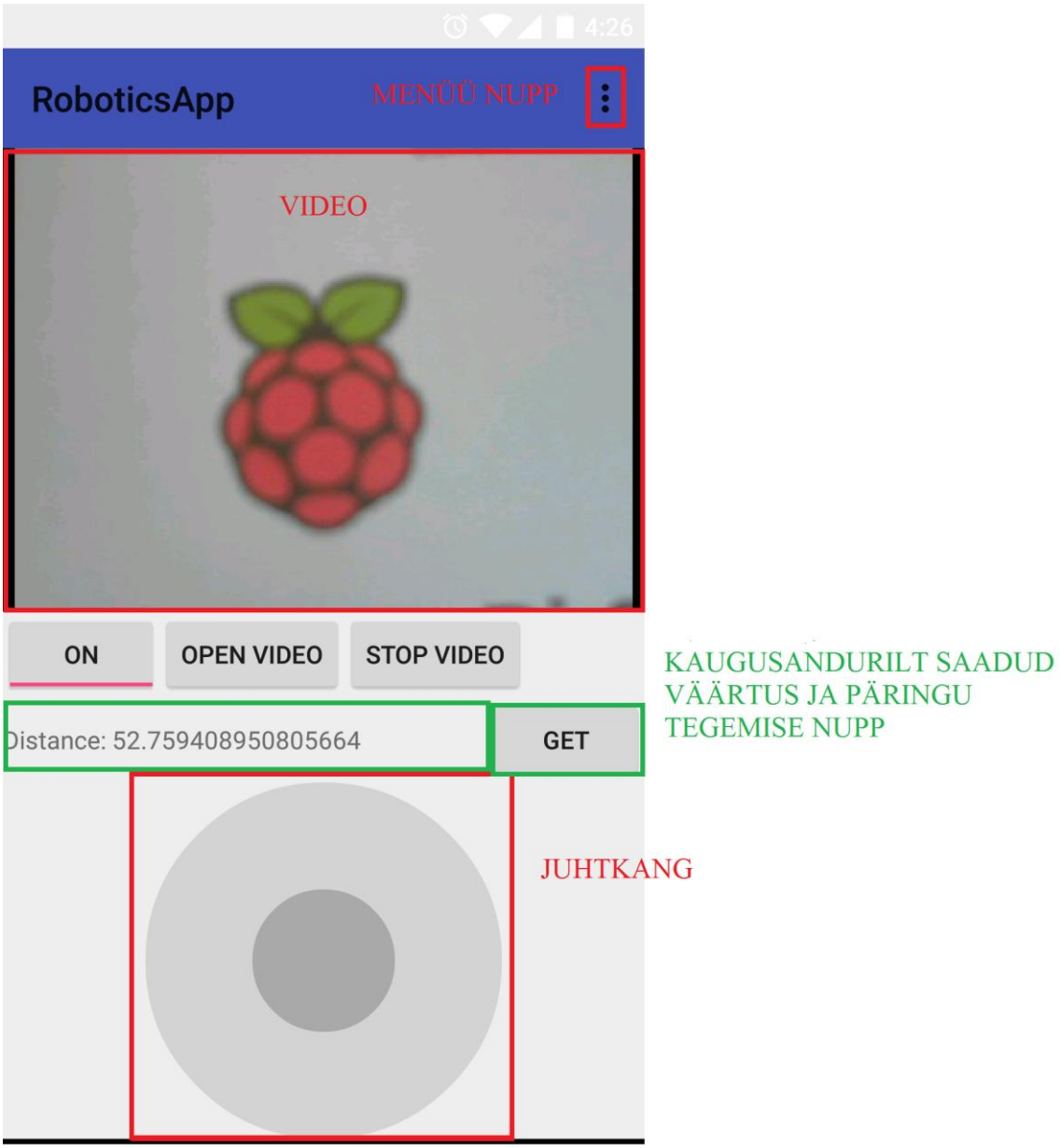
Selles peatükis kirjeldab autor, kuidas näidisrakendust kasutada ja mis kasutajaliidese komponendid on näidisrakenduses. Autor annab ka mõningaid näpunäiteid, kuidas lihtsamaid vigu likvideerida.

3.2.1 Näidisrakenduse kasutamine

Näidisrakendusel on kõigest üks tegevus. See tegevus täidab kõik rakenduse funktsionaalsuse. Autor arendas selle tegevuse põhimõttel, et videopilt, juhtkang, nupud ja andurilt andmete näitamise väli mahuksid kõik ühele ekraanile. Kui kogu sisu ei mahuks

ekraanile, peaks terve sisu paigutama konteinervaatesse ning selle konteinervaate paigutama omakorda *ScrollView* sisse. *ScrollView* võimaldab enda sees olevat vaadet kerida vertikaalselt ja horisontaalselt. Näidisrakenduse arendamise käigus eelnevalt mainitud variant aga ei toiminud, sest juhtkangi liigutamine ja terve ekraani kerimine sattusid konflikti ning juhtkangi munaku liigutusi enam ei registreeritud. Seetõttu soovitab autor piirata sisu ühele ekraanile.

Rakenduse avamisel suunatakse kasutaja ainsale tegevusele. Selle tegevuse komponendid (joonis 16) on menüü (asub tegevuse ribal vasakul nurgas), LED tule sisse/välja lülitamise nupp(OFF/ON), video edastamise alustamise (OPEN VIDEO) ja lõpetamise (STOP VIDEO) nupud, kaugusanduri andmete ala (Distance), kaugusandurilt andmete küsimise nupp (GET) ja roboti juhtimiseks vajalik juhtkang. Menüüd avades näidatakse kahte valikuvarianti: „Choose Connection“ ja „Choose Sensor Mode“.



Joonis 16: Näidisrakenduse kasutajaliidese ekraanitõmmis komponentide kirjeldusega

„Choose Connection“ avab akna, kus kasutaja sisestada Raspberry Pi IP aadressi. Sisestuse kinnitamiseks peab kasutaja vajutama „CHOOSE URL“ nuppu. Oluline on märkida, et autor kirjutas näidisrakenduse nii, et kasutaja sisestab IP aadressi ja lisab sinna lõppu koolon, mis tähistab, et järgneb serveri pordi number. Pordi numbrid lisas autor HttpRequest faili sisse kahe muutujana, mille rakendus lisab automaatselt võrgupäringutele vastavalt päringu eesmärgile. Video jaoks on muutuja „videoport“ ja muude käskude jaoks „commandport“.

„Choose Sensor Mode“ nupp avab akna, milles saab valida, kuidas küsitakse Raspberry Pi serverilt kaugusanduri andmeid. Valikuvariandid on:

- „Don’t Send“ on vaikimise valitud variant. „Get“ nupp on nähtamatu ja seepärast kaugusandurilt andmete lugemise päringut teha ei saa.
- „Send After Button Press“ variandi puhul on nupp nähtaval ning nupule vajutades tehakse kaugusandurilt andmete lugemise päring.
- „Send Periodically“ variandi valides tehakse päringuid regulaarselt 1 sekundise intervalliga. „Get“ nupp on nähtamatu.

Valikuvariant kinnitatakse valides valikuvariantide aknas „SET“ nuppu.

Näidisrakenduse kasutamiseks peab leidma lokaalses võrgus oleva demoroboti. Demoroboti võib leida mitmel viisil. Juhul, kui demorobot on ühendatud nutiseadme poolt loodud kuumkoha (*hotspot*) võrku, on nutiseadmes kuumkoha sätetes nähtaval kõik ühendatud seadmed koos IP aadressiga. Alternatiiv on kasutada kohalikku WiFi võrku. Selleks, et leida kohalikus võrgus ühendatud seadmeid on võimalik kasutada erinevaid arvutitele mõeldud rakendusi ja ka nutiseadmetele mõeldud rakendusi. Autor kasutas demorakenduse IP aadressi leidmiseks Fing rakendust, millega saab otsida kohalikus võrgus kõiki seadmeid. Fing rakendus näitab lisaks kohaliku võrgu seadmetele lisaks IP-le ka seadme tüüpi.

WiFi või kuumkoha valimisel on nii plusse kui ka miinuseid. Praeguse lahenduse puhul peab võrgu valimisel roboti monitoriga ühendama ja graafilises liideses seadistama WiFi võrguga liitumise. Kuna alati ei pruugi ekraaniga ühendamine võimalik olla ja kuumkohaga ühendamiseks peab ühenduse seadistama ainult ühe korra, võib mõnedes olukordades olla kuumkohaga ühendamine lihtsuse tõttu parem valik. Teisalt teeb praeguse laheduse juures demorobot võrgupäringuid ja edastab videopilti, mis võib pikema aja juures tarbida andmesidemahtu. Juhul, kui WiFi võrku väga tihti ei vahetata, on mõistlikum kasutada kohalikku võrku, sest siis on andmeside kulud väiksemad.

Peale rakenduse demorobotiga ühendamist peaksid kõik nupud töötama ja juhtkangi liigutamisel peaks robot liikuma. Kui ükski komponent ei tööta, peaks kontrollima, kas

roboti akupank on täis ning kas robot on lokaalses võrgus nähtaval. Lisaks peaks kontrollima, kas on lisatud õige IP aadress. Kui eelnevad variandid on tehtud, peaks nutiseadme ühendama arvuti külge ja kontrollima Android Studio logi ning siluma edasi selle abil. Näidisrakenduse ja demoroboti kasutamisel ei tohiks silumine olla vajalik. Juhul, kui roboti juhtimine toimib ja videot ei edastada, võib proovida veebikaamera USB kaabli taasühendamist. Autor täheldas, et pikema aja jooksul kadus ühendus veebikaameraga ning peale taasühendamist töötas kõik hästi edasi.

Selle peatüki lõpuks peaks lugeja suutma kasutada näidisrakendust (Joonis 16) demoroboti juhtimiseks. Järgnevas peatükis kirjeldab autor peamisi rakenduse programmi komponente.

3.2.2 Näidisrakenduse koodi peamised komponendid

Kuna näidisrakendus on väga mahukas, ei hakka autor iga meetodit lahti kirjutama. Siiski annab autor selles peatükis ülevaate olulisematest meetoditest.

Näidisrakenduses on neli Java klassi – HttpRequest, MainActivity, MjpegInputStream ja MjpegView. MjpegView ja MjpegInputStream klassid on MJPEG ühenduse jaoks vajalikud klassid, mille autoriks on Independent JPEG Group. Mõlema klassi koodi algusesse on lisaks autoritele lisatud ka kaasaskäiv litsents. Nende klasside kasutamisel peab nõustuma litsensiga ning käesoleva bakalaureusetöö raames võib neid klasse sellisel kujul kasutada. HttpRequest klassis on võrgupäringud, mida kasutatakse demorobotiga suhtlemiseks. MainActivity on näidisrakenduse ainukese tegevuse implementatsioon. Selles klassis loetakse sisse kujundus ja selle komponendid. Selles klassis reageeritakse ka kõigile nupuvajutustele ja juhtkangi liigutamisele.

HttpRequest klassis on iga päringu jaoks eraldi meetod, mis paneb kokku saadetava päringu teekonna ja väärtused. Teekond oleneb, millist komponenti soovitakse juhtida, näiteks servo jaoks lisatakse teekond „/update_servo?x=x*&y=y*“, kus x* ja y* on servo

väärtused. Vastavalt sellele teekonnale oskab demorobot päringuid üksteisest eristada. PostRequest meetod paneb päringu kokku, saadab serverile ning võtab vastu vastuse.

MainActivity peamine meetod on onCreate meetod. Selles meetodis luuakse kõik menüüd ja dialoogid ning kujunduses kirjeldatud elementide kohta Java objektid. Neile Java objektidele lisatakse ka kuulamismeetodid, mis käivitavad HttpRequesti päringud. Kaugusandurilt andmeid kuulav meetod uuendab kasutajaliideses kaugusanduri andmeid. MainActivity muudab ka vastavalt kasutajaliidest vastavalt menüüs tehtud valikutele. Selles klassis on ka „VIDEO_RESOLUTION_WIDTH“ ja „VIDEO_RESOLUTION_HEIGHT“ muutujad. On väga oluline, et video edastamisel oleksid nende väärtused samad, mis Raspberry Pi videoserveril pildi resolutsiooni väärtused.

Android platvormil käib töö mitmel lõimel. Kõige kiirem lõim haldab tavaliselt kasutajaliidest ning teisi lõimsid saab kasutada muudeks tegevusteks, näiteks bluetoothi suhtluseks või võrgupäringuteks. Androidi rakendustel ei ole soovitatud teha võrgupäringud kasutajaliidese lõimel ja selleks peab võrgupäringud arendaja suunama teistele lõimedele seniks, kuni sealt tuleb vastus ja vastuse saades muutma kasutajaliidest uuesti kasutajaliidese lõimel. Teistele lõimedele delegeeritakse ülesanded AyncTask.execute() meetodiga (Joonis 15) ja kasutajaliidesele delegeeritakse ülesanded runOnUiThread() meetodiga (Joonis 15).

```
262        AsyncTask.execute(() → {
265                final String sensorValue = request.SendSensorRequest();
266                runOnUiThread(() → {
269                        if (sensorValue != null) {
270                            sensorValueView.setText(sensorValue);
271                        }
272                });
274        });
```

Joonis 17: Kaugusanduri päringu saatmine ja vastuse abil kasutajaliidese uuendamine

Suurema huvi korral aitavad repositooriumi koodi lisatud kommentaarid mõista täpsemalt, erinevate meetodite eesmärke ja tööpõhimõtteid.

4. Kokkuvõte

Selle bakalaureusetöö lõpptulemusena lõi autor eestikeelse juhendi, mille abil on robootikahuvilistel võimalik teha oma roboti juhtimiseks rakendus. Selles töös on kajastatud näidisrakenduse ja -roboti lõppkood koos juhistega, kuidas mõlemat iseseisvalt tööle saada.

Töö käigus arendatud näidisrobotile oli planeeritud lisada video edastamise võimekus ning vähemalt ühe anduriga suhtlus, mis said teostatud. Esialgselt oli autoril plaanis lisada bluetooth tehnoloogia abil suhtlemine ning uurida video edastamise võimalusi üle bluetooth suhtluse. Kuna video edastamine ning rakenduses vastu võtmine osutusid oodatust keerukamaks ja ajamahukamaks, otsustas autor bluetooth toe kõrvale jätta. Robotiga suhtlemiseks kasutab rakendus WiFi päringuid. Näidisrobot ja –rakendus on arendatud eeldusel, et mõlemad seadmed on kohalikus võrgus.

Käesolevat tööd võib edukalt kasutada järgnevate bakalaureusetööde aluseks. Kuna bluetooth tugi jäi lisamata, on see üks võimalik funktsionaalsus, mida implementeerida. On võimalik ka uurida, kas nutiseadmetel ning bluetooth adapteritel on piisav andmeedastuse võimekus, et videot edastada bluetooth tehnoloogiat kasutades. Keskenduti Android mobiilirakendustele ning jättis esialgu kõrvale teised platvormid, millele võiks samuti teha näidisrakenduse. Üle WiFi võrgu suhtluse tehti väga lihtsalt tasandil, näidisrobotile võib lisada juhendi, kuidas teha roboti peal töötavaid serverid, mis on kättesaadavad väljaspool kohalikku võrku. Näidisrakendusel pole vahet, kas server, kuhu päringuid tehakse on kohalikus võrgus või mitte, on vajalik ainult URL aadress. Võrgupäringuid arvesse võttes võib lisada ka autentimise ning sellega kõrvaldada roboti kaaperdamise võimaluse.

5. Kasutatud materjalid

[1] IDC Research, Inc., „IDC müüdud nutiseadmete statistika,“ IDC Research, Inc., [Võrgumaterjal]. Kättesaadav: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Kasutatud 08 05 2016].

[2] „Android Studio ametlik allalaadimise veebileht,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/sdk/index.html#Requirements. [Kasutatud 2 2 2016].

[3] „Android Developer ametlik manifest juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/guide/topics/manifest/manifest-intro.html. [Kasutatud 02 04 2016].

[4] „Android Developer ametlik drawable juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/guide/topics/resources/drawable-resource.html#Bitmap. [Kasutatud 10 04 2016].

[5] „Android developer ametlik drawable juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/guide/topics/resources/drawable-resource.html#NinePatch. [Kasutatud 10 04 2016].

[6] „Android developer ametlik erinevate ekraanisuuruste toetamise juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/guide/practices/screens_support.html. [Kasutatud 10 04 2016].

[7] „Android developer ametlik kujunduse juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/guide/topics/ui/declaring-layout.html#write. [Kasutatud 17 04 2016].

[8] „Android developer ametlik menüüde juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/guide/topics/ui/menus.html. [Kasutatud 12 04 2016].

[9] „Android tegevuste ametlik juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/reference/android/app/Activity.html. [Kasutatud 01 05 2016].

[10] „Android ametlik kildude juhend,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/reference/android/app/Activity.html#Fragments. [Kasutatud 01 05 2016].

[11] „Android ametlik onCreate (Bundle SavedInstanceState) dokumentatsioon,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle). [Kasutatud 2016 05 01].

[12] „Android ametlik tegevuse elutsükli dokumentatsioon,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle. [Kasutatud 01 05 2016].

[13] „Android tööriistariba ametlik dokumentatsioon,“ [Võrgumaterjal]. Kättesaadav: http://developer.android.com/training/appbar/index.html. [Kasutatud 01 05 2016].

[14] „Raspberry Pi GPIO dokumentatsioon,“ Raspberry Pi Foundation, [Võrgumaterjal]. Kättesaadav: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/. [Kasutatud 05 05 2016].

[15] „HC-SR04 ametliku tootja toote dokumentatsioon,“ ElecFreaks, [Võrgumaterjal].

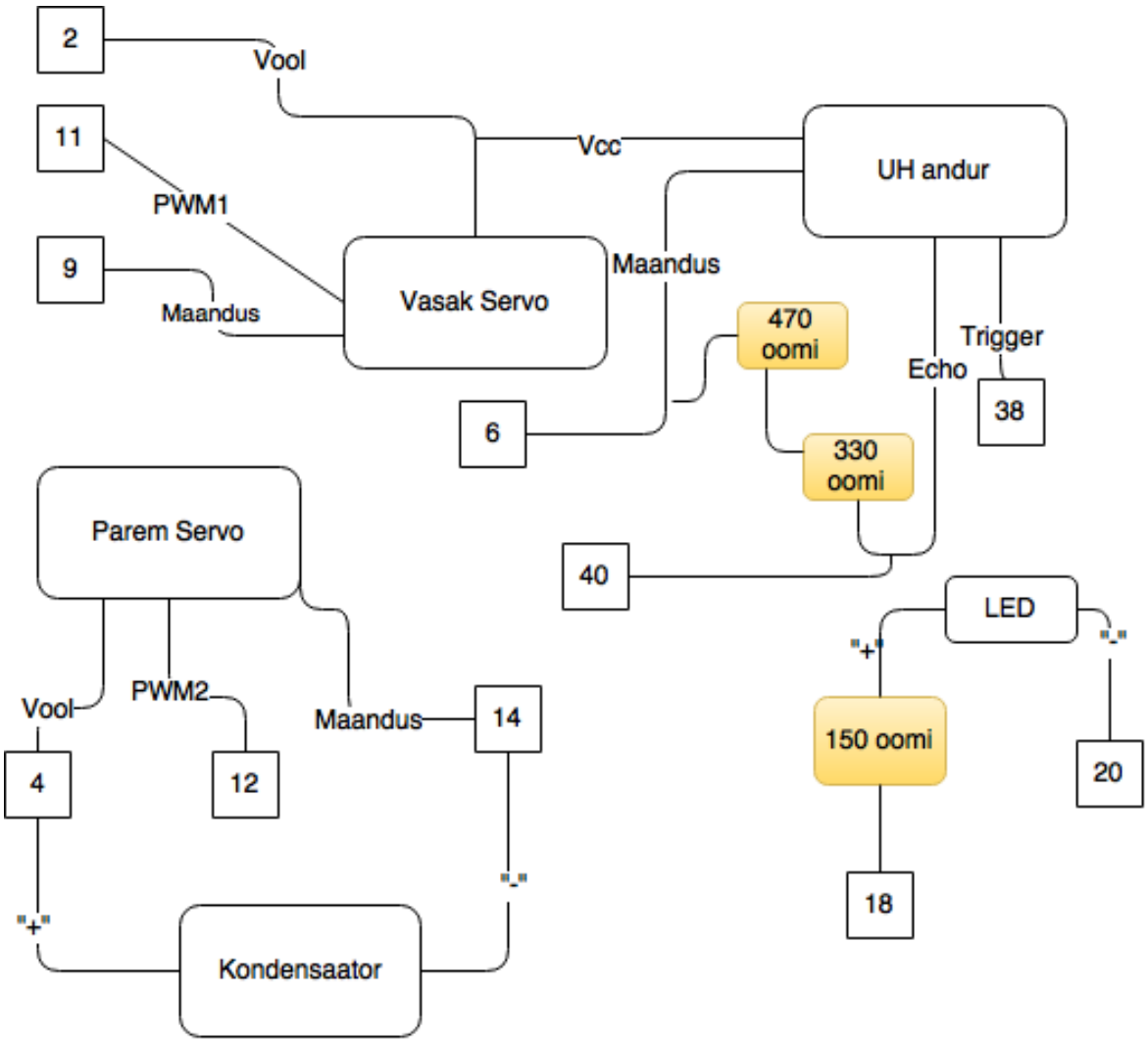
Kättesaadav: http://www.micropik.com/PDF/HCSR04.pdf. [Kasutatud 05 05 2016].

[16] M. Hawkins, „Ultrasonic Distance Measurement Using Python – Part 1,“ [Võrgumaterjal]. Kättesaadav: http://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python-part-1/. [Kasutatud 05 05 2016].

[17] Itseez, „OpenCV ametlik koduleht,“ Itseez, [Võrgumaterjal]. Kättesaadav: http://opencv.org/. [Kasutatud 07 05 2016].

Lisad

Lisa 1



Lisa 1: Demoroboti ühendamise skeem

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Arvi Kaasik**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Mobiilirakendus roboti juhtimiseks**,
(*lõputöö pealkiri*)

mille juhendajad on Taavi Duvin, Alo Peets, Anne Villems,
(*juhendajate nimed*)

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, **12.05.2016**