Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

Henry Maalinn

LEGO MINDSTORMS EV3-e programmeerimine Pythonis

Bakalaureusetöö (9 EAP)

Juhendajad: Anne Villems
Taavi Duvin
Alo Peets

Tartu 2017

LEGO MINDSTORMS EV3-e programmeerimine Pythonis

Lühikokkuvõte:

Käesoleva bakalaureusetöö põhiline eesmärk on tutvustada kuidas programmeerida LEGO MINDSTORMS EV3 robotit Pythonis. Lahendusena paigaldatakse LEGO MINDSTORMS EV3 programmeeritavale ajule ev3dev operatsioonisüsteem. Töö käigus valmis terviklik juhend ev3dev operatsioonisüsteemi paigaldamisest mälukaardile kuni roboti programmeerimiseni Pythonis kasutades JetBrains'i PyCharmi. Töös on esitatud ka 8 näidisülesannet, milles kasutatakse mootoreid, programmeeritava aju ekraani, heli, nuppe ja tulesid ning kaugus-, puute- ja värviandurit.

Võtmesõnad:

LEGO MINDSTORMS EV3, ev3dev, Python

CERCS: P175 Informaatika, süsteemiteooria

LEGO MINDSTORMS EV3 programming in Python

Abstract:

The aim of this thesis work is to introduce how to program LEGO MINDSTORMS EV3 robot in Python. In order to do so an ev3dev operating system was installed on LEGO MINDSTORMS EV3 brick. As a result of the thesis work a complete manual from installing ev3dev operating system to memory card to programming robot in Python using JetBrains PyCharm was composed. The thesis work outlines in addition eight different application examples, where motor, bricks sound, buttons and lights, as well as Ultrasonic, Touch and Colour sensors were used.

Keywords:

LEGO MINDSTORMS EV3, ev3dev, Python

CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus

Töö eesmärk on selgitada kuidas programmeerida LEGO MINDSTORMS EV3 robotit Pythonis. LEGO MINDSTORMS on LEGO robootika liin, mida on lihtne ja turvaline kasutada robootika õpetamiseks, kuna roboteid on lihtne kokku ja lahti monteerida ning ei vaja selleks eraldi tööriistu. Samas on LEGO ka võrdlemisi ohutu ning tavapärase kasutamise juures pole ohtu, et mõni komponentidest hävineks. Eestis on palju noori, kes on tutvunud programmeerimisega just tänu LEGO'le. LEGO MINDSTORMS on hea platvorm, mida kasutada programmeerimise tutvustamiseks, sest see on graafiline programmeerimine ning õpilastel on nii programmist lihtsam aru saada. LEGO robootika muutub koolides aina populaarsemaks, kuid mõne kursusega on õpilastel läbi proovitud EV3 komplektiga kaasas käivad juhendid. Graafilisel programmeerimisel on omad miinused. Programmi keerukuse kasvades muutub programm raskesti hoomatavaks. Lahenduseks on kasutada programmeerimiseks mõnda teist programmeerimiskeelt, näiteks Pythonit. Ka varem on saanud EV3-e programmeerida mõnes teises programmeerimiskeeles, ent selleks on vaja olnud kallist litsentsi, mis ei sobilik koolidele ja huviringidele. Tänu ev3dev operatsioonisüsteemile on võimalik LEGO MINDSTORMS EV3 roboteid programmeerida mitmetes erinevates programmeerimiskeeltes seal hulgas ka Pythonis. Töös kirjeldatud juhend peaks aitama robootikaringi juhendajal seadistada kõik vajalikud komponendid, et õpetada õpilastele, kuidas programmeerida LEGO MINDSTORMS EV3 roboteid Pythonis. Töös on detailselt kirjeldatud ev3dev operatsioonisüsteemi paigaldus LEGO MINDSTORMS EV3 programmeeritavale ajule ning Pythoni programmeerimiskeskkonna paigaldus koos seadistamisega. Töös kirjeldatud juhend on sedavõrd detailne just see tõttu, et töö on suunatud robootikaringi juhendajatele.

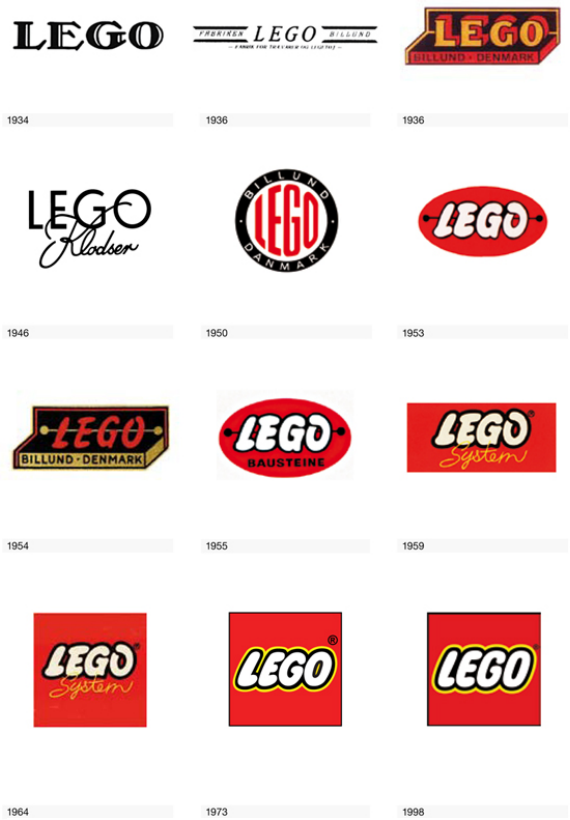
Lahenduse saavutamiseks tuleb EV3-e programmeeritavale ajule paigaldada Linux'i põhine ev3dev operatsioonisüsteem. Stabiilne versioon ev3dev'ist avalikustati 21. detsember 2016. Ev3dev kodulehelt leiab operatsioonisüsteemi paigaldamiseks inglise keelse juhendi, ent eesti keeles juhendit varem pole kirjutatud. Ev3dev operatsioonisüsteem on saadaval tasuta ning seda on võrdlemisi lihtne paigaldada. Peale operatsioonisüsteemi paigaldamise on töösesse lisatud ka JetBrains'i PyCharm paigaldus ning sünkroniseerimiseks ühenduse seadistamise. PyCharmi kasutamine muudab EV3 programmeerimise Pythonis oluliselt mugavamaks ning lihtsustab õppematerjalide õpetamise lihtsamaks. Töös räägitakse esmalt LEGO ja LEGO robotite ajaloost. Peatükis „Ev3dev paigaldamine ja PyCharmi ülesseadmine" kirjeldatakse täpselt, mis on vajalik ev3dev paigalduseks ning kuidas protsess täpselt toimub. Teises pooles kirjeldatakse kuidas seadistada JetBrains'i PyCharm nii, sellega loodav projekt saaks automaatselt robotisse. Viimases sisupeatükis on esitatud kaheksa ülesannet, mille lahendamiseks tuleb kasutada LEGO roboti mootoreid, kaugus-, puute- ja värviandurit ning EV3 programmeeritava aju ekraani, heli, nuppe ja tulesid. Ülesannetele on kirjutatud ka võimalikud hästi kommenteeritud lahendused, mis selgitavad python-ev3dev teegi erinevate meetodite tööd.

2. LEGO ja robotid

Käesolevas peatükis räägitakse LEGO ajaloost ja FIRST LEGO League'st. Ajaloost rääkides keskendutakse rohkem LEGO Education harule. Tänapäeval on LEGO Education haru üheks tähtsamaks osaks robotid. Roboteid on võimalik ise kokku monteerida ning programmeerida. Peatükis räägitakse LEGO Education WeDo-st, LEGO MINDSTORMS RCX-ist, LEGO MINDSTORMS NXT-st ja LEGO MINDSTORMS EV3-st. LEGO robotite populaarsus Eesti koolides ja huviringides kasvab ning seda näitab osavõtjate arvu kasv FIRST LEGO League Eesti üritustel.

2.1 LEGO ajalugu

LEGO Group on perefirma, mis asutati Billund'i linnas Taanis ja on tuntud LEGO brändi mänguasjade poolest. Firma lõi Ole Kirk Christiansen aastal 1932. Nimi LEGO on kokku pandud kahest taani keelsest sõnast *"leg"* - mängi ja *"godt"* - hästi, kuid samas *"lego"* tähendab ladina keeles "mina panen kokku" [1]. Algselt toodeti puidust mänguasju ja klotse, kuid aastal 1947 hakati neid tootma plastikust. 1949 aastal hakati muude toodete seas tootma ka tänapäeval tuntud omavahel seonduvatest klotsidest, mida hakati nimetama "automaatselt seonduvad klotsid"



Joonis 1. LEGO logo muutuste ajalugu [2].

(“*Automatic Binding Bricks*”) [3]. Aastal 2015 esimeses pooles sai LEGO Group’ist suurima tuluga mänguasja tootja maailmas. LEGO tõukas troonilt mänguasjade hiiglase Mattel’i, kes muuhulgas toodab Barbie nukke. LEGO edule aitas suuresti kaasa suure edu pälvinud animatsioon film „LEGO film“ [4]. Nagu jooniselt 1 näha võib, ainuüksi LEGO logo teinud läbi palju muutusi oma pika ajaloo jooksul ning logo, mis ilutseb klotsi karpidel tänapäeval on pärit aastast 1998.

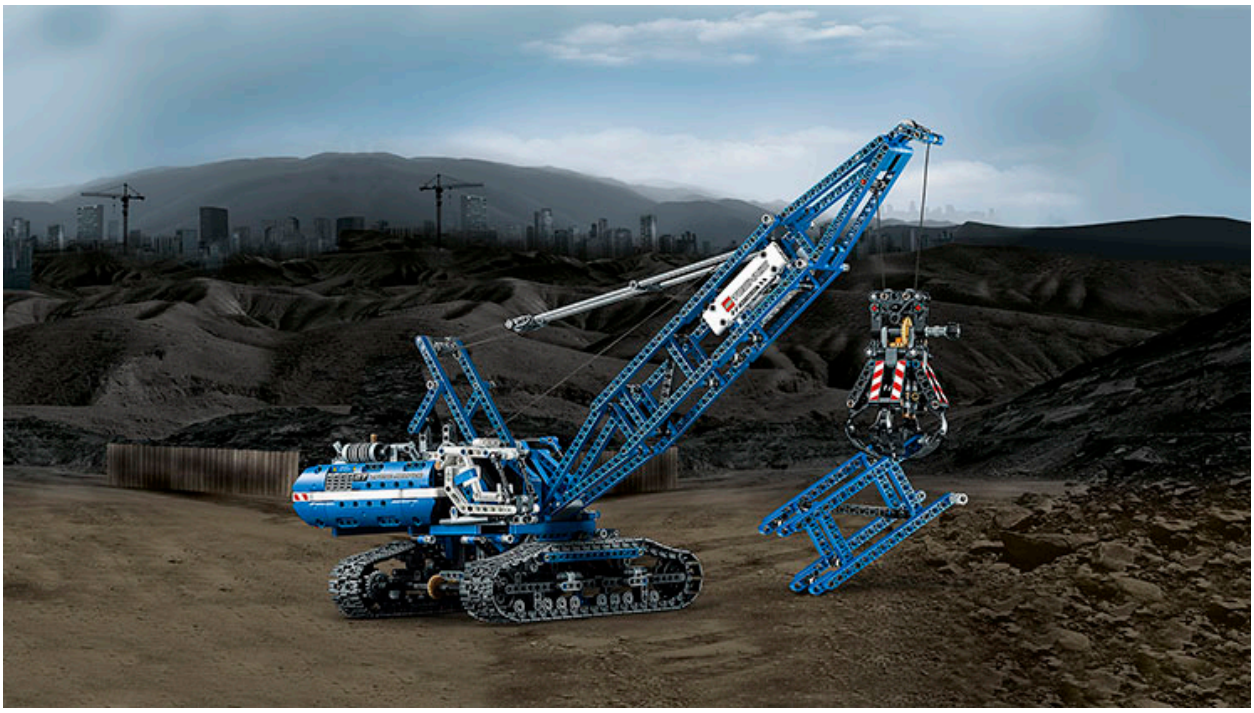
2.2 LEGO Education

LEGO Education (endise nimega Dacta) loodi aastal 1972 ning sihtgrupiks olid lasteaiad ja algklassid. Selle brändi komplektid olid mõeldud laste koostöö harjutamiseks. Põhiliselt erines LEGO Education tavapärastest LEGO komplektidest selle võrra, et koosnes klotsidest, mis olid suuremad ning kuulusid eelmise generatsiooni komplektidesse. Nii suudeti hoida komplektide hinnad madalad ning taskukohased koolidele ja lasteaedadele. [5]

Tänapäeval kuuluvad LEGO Education harusse sellised brändid nagu LEGO DUPLO ja LEGO TECHNIC [5]. LEGO DUPLO on tooteklass, mis on mõeldud 1,5 kuni 5 aastastele lastele. Selle

Joonis 2. LEGO DUPLO komplekt „*My First Caterpillar*“ ehk „Minu esimene röövik“ [6]

teeb eriliseks just see, et need klotsid on täpselt 2 korda suuremad kui tavalised LEGO klotsid, ent siiski on neid kahte mõõtmetelt erinevaid detaile võimalik kokku sobitada [7]. Joonisel 2 on toodud ka näide LEGO DUPLO komplektist „Minu esimene röövik“. LEGO TECHNIC on bränd, mille eesmärk on luua keerulisemaid ning liikuvaid mänguasju. Detailid TECHNIC komplektis on hoopis erinevad võrreldes tavapäraste klotsidega ning kokku sobitamine on keeruline kuid mitte võimatu. Komplektide sisu moodustavad detailid, mis enamasti mõeldud liikuma ning nendest tehtud ehitised on mobiilsed ühel või teisel moel. Selle brändi komplektidesse kuuluvad tihti ka



Joonis 3. LEGO TECHNIC sarjast pärit kraana komplekt nimega „*Crawler crane*“ [8].

erinevad mootorid, näiteks joonisel 3 kujutatud kraanat, mida on võimalik kahe kangi abil seadistada nii, et kraana liiguks mootorite jõul edasi või tagasi, pööraks vasakule või paremale. Samuti saab seadistada kui kaugele käpp ulatub ning muidugi ka käpa kõrgust [9].

Populaarsuse kasvades arendati Education haru ning praegu aitab LEGO lapsi õpetada alates lasteaiast kuni keskkoolini. LEGO Education programmi abil on võimalik õppida näiteks matemaatikat, loodusteadusi, arvutiteadust ning tehnikat ja masinaehitust. Selle jaoks on LEGO loonud erinevatel teemadel õppematerjale ja ülesandeid ning lahendades omandavad lapsed uusi teadmisi [10].

2.3 LEGO MINDSTORMS

LEGO MINDSTORMS on 1998 loodud LEGO robotite tooteklass. Esimene arvuti teel juhitav LEGO toode anti välja 1986, kuid MINDSTORMS alguseks loetaks LEGO MINDSTORMS RCX *Intelligent Brick* ja *Robotics Invention System* turule tulekut. LEGO MINDSTORMS *RCX Intelligent Brick*. valmis koostöös Massachusetts'i Tehnoloogia Instituudiga. Põhikomplektile oli

Joonis 4. LEGO RCX 2.0[11].

võimalik ka soetada kahte erinevat lisa: *RoboSports* ja *Extreme Creatures*. Juba aastal 1999 paisati turule täiendatud versioon LEGO MINDSTORMS *Robotics Invention System* 1.5 ning aasta hiljem täiustati komplekti veelgi ja välja lasti LEGO MINDSTORMS *Robotics Invention System* 2.0, mida on ka näha joonisel 4. [12]

2.4 LEGO Education WeDo

Lifelong Kindergarten on ühendus, mis otsib lahendusi, et muuta õppimine noorte jaoks põnevaks ja loovaks. Mitch Resnick ja tema *Lifelong Kindergarten* grupp võtsid 2006-l aastal ühendust Erik Hansen'iga, kes oli sellel ajal LEGO elektroonika uurimise ja arendusse osakonna juhataja, et arutada uut projekti. Kolm aastat hiljem oli projekt valmis ning uus LEGO komplekt valmis müümiseks. See oli LEGO WeDo, mida on näha ka joonisel 5. LEGO robotite populaarsus ning positiivne tagasiside oli neid ajendanud looma uue komplekti, mis oli suunatud alates 7-aastastele. Komplekt koosnes 150 osast ning sisaldas mootorit, tasakaaluandurit, liikumisandurit, LEGO

Joonis 5. LEGO WeDo komplekti erinevad võimalused [13].

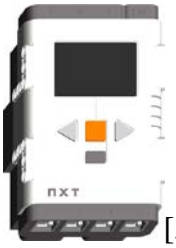
klotse, hammasrataste süsteemi ja lihtsasti kasutatavat tarkvara. Lisaks saavad lapsed kasutada *Media Lab*'i tasuta tarkvara *Scratch*'i, et programmeerida animatsioon, mis läheks kokku nende loodud robotiga, et kombineerida virtuaal- ja reaalmaailm. See komplekt annab lastele võimaluse ise midagi ehitada arendades see juures loovat mõtlemist, meeskonnatööd ja probleemi lahendamise oskust [14].

Aasta 2016 alguses toodi turule WeDo 2.0[15]. Uus generatsioon tõi kaasa palju uuendusi. Üks tähtsamad uuendused olid: Bluetoothi ja AA patareide kasutamine. Enam ei pea robot olema arvutiga ühenduses. Uus komplekt sisaldab uuendatud tasakaalu-ja liikumisandurit, ühte mootorit ning kokku 280 detaili. WeDo 2.0 komplekt on mõeldud korraga kasutamiseks kahele õpilasele [16]. Komplektiga koos käivat tarkvara saab alla laadida LEGO Education kodulehelt ning teotatud platvormide hulka kuuluvad: Windows 7 – Windows 10, Mac OS, Chromebook, iPad ja Android. WeDo 2.0 kasutamine huviringides teeb lihtsaks asjaolu, et seda on võimalik lihtsasti programmeerida ka tahvelarvutiga [17].

2.5 LEGO MINDSTORMS NXT

LEGO robotid olid võrdlemisi edukad ning otsustati luua uus generatsioon. Aastal 2006 toodi turule robot, mis kandis nime LEGO MINDSTORMS NXT. Viimase eeliseid eelkäija ees oli palju: parem ja suurem ekraan, võime mängida lihtsamaid helifaile, Bluetooth ühenduse võimalus ning lisapesa sensoritele [18]. Kolm aastat hiljem lasti välja uuem mudel - LEGO MINDSTORMS NXT 2.0. Komplektid erinevad üksteisest põhiliselt tavaliste klotside poolest, kuid uuemast pakist olid puudu ka näiteks valguse ja heli sensorid, ent selle asemel olid sinna lisatud värviandur ja lisa puuteandur. LEGO MINDSTORMS NXT 2.0 tõi endaga kaasa ka uue tarkvara [19]. Uus tarkvara tõi endaga kaasa umbes kaks korda parema töökindluse, ujukoma arvude toe ning andmete salvestamise võimaluse. Samuti lisati ka valgusanduri tugi [20].

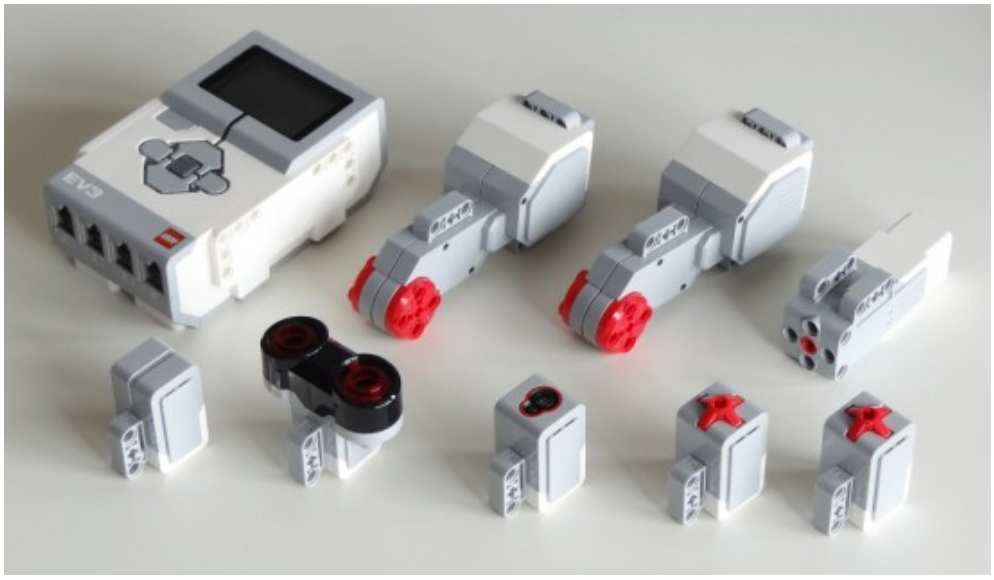
Tabel 1. LEGO MINDSTORMS NXT Education baas komplekti tähtsamad komponendid.

Nimetus	Kogus	Joonis
Programmeeritav aju	1	 [21]
Servomootor	3	 [21]
Ultraheliandur (kaugusandur)	1	 [22]
Heliandur	1	 [23]
Puuteandur	2	 [21]
Valgusandur	1	 [24]

Koolidele mõeldud, LEGO MINDSTORMS NXT Education, komplekt oli mõne võrra odavam kui tavaline komplekt ning sisaldas esimese versiooni andureid. Tabelis 1 on näha Education komplekti tähtsamad komponendid koos piltidega. Samuti ei käinud ka komplektiga kaasa tarkvara, vaid see tuli eraldi soetada, sest koolidel polnud vaja iga roboti jaoks eraldi litsentsi. Kui muidu töötas robot AA patareidega, siis Education komplektis asendati roboti toide laetava aku vastu. See lubas robotiga edasi töötada ka aku laadimise ajal. Teine suur eelis oli see, et enam ei pidanud aju roboti küljest lahti ühendama, et patareisid vahetada [19].

2.6 LEGO MINDSTORMS EV3

LEGO MINDSTROMS EV3 on LEGO MINDSTROMS liini kolmanda generatsiooni robot. Selle nimi EV3 tähendab: *evolution* ehk evolutsioon ja 3 on generatsiooni number. Koolidele mõeldud komplekt LEGO MINDSTORMS Education EV3 paisati müüki 2013 aasta augustis ning personaalseks kasutamiseks mõeldud komplekt LEGO MINDSTORMS EV3 Home Edition lasti välja juba kuu aega hiljem septembris [25]. Kaks komplekti olid võrdlemisi erinevad. Tähtsamad riistvaralised erinevused olid, et EV3 Home Edition sisaldas infrapuna andurit, infrapuna kaugjuhtimispulti ning aju vajab töötamiseks 6-te AA patareid, siis LEGO MINDSTORMS Education EV3 komplektis olid selle asemel güroskoop, ultraheliandur ning lisaks veel üks



Joonis 6. LEGO MINDSTORMS Education EV3 komplekti tähtsamad komponendid [26].

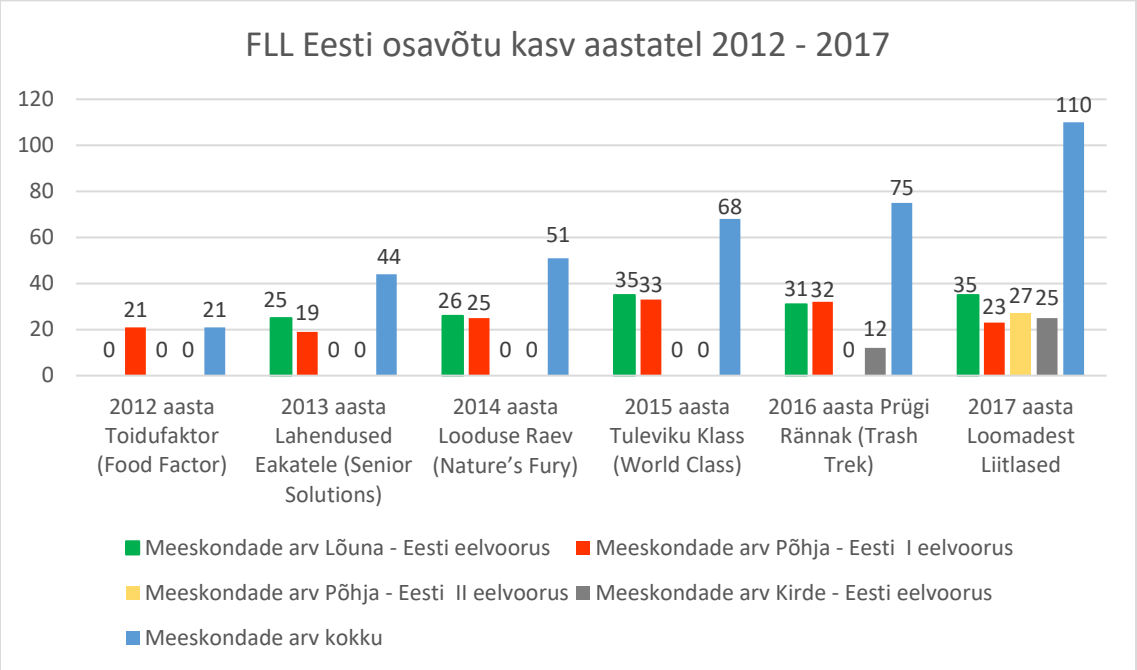
puuteandur. Programmeeritava aju toiteks oli komplektis aku. Tänu akule polnud vaja aju roboti küljest lahti monteerida, et patareisid vahetada ning see võimaldas robotit kasutada ka aku laadimise ajal [26]. Roboti ehitamiseks vajalikud elemendid kuuluvad LEGO Technic sarja. Komplektid erinesid ka ehituselementide aspektist mõnevõrra. EV3 Home Edition komplektis oli kaasas juhendid 17 robotile, mis olid pigem mängulisemad ja meelt lahutavamad nagu kahel jalal liikuv ning palle tulistav ning kahel jalal liikuv robot, madu ja skorpion, ent Education EV3 komplektile mõeldud juhendid sisaldasid rohkem funktsionaalsemaid roboteid nagu güroskoobi abil tasakaalu hoidev güropoiss, sorteermismasin ning robotkäsi [27]. Hiljem loodi veel üks komplekt LEGO MINDSTORMS Education EV3 Expansion Set ehk täiendus komplekt koolidele mõeldud komplektile. See sisaldas riistvara poolest ainult LEGO Technic ehitus elemente ning juhendeid uute robotite ehitamiseks. Lisa komplekti Expansion Set võis ka ühildada kodukasutaja komplektiga, ent kõiki juhendites kirjeldatud roboteid ehitada polnud võimalik, kuna LEGO MINDSTORMS Education EV3 komplekt ja LEGO MINDSTORMS EV3 Home Edition komplekt olid niivõrd erinevad. Koduseks kasutamiseks mõeldud komplektis oli juba kaasa nii litsents kui ka juhendid robotite ehitamiseks, siis koolidele mõeldud komplektides neid ei olnud. Tarkvara oli komplektidel üsna sama, kuigi Education EV3 komplekti juurde sobiv tarkvara sisaldas andmete talletamist, et teha erinevaid mõõtmiskatseid ning ka güroskoop ja ultraheli

andurite tuge, siis oli võimalik need käsitsi lisada ka EV3 Home Edition komplekti tarkvarale [26].

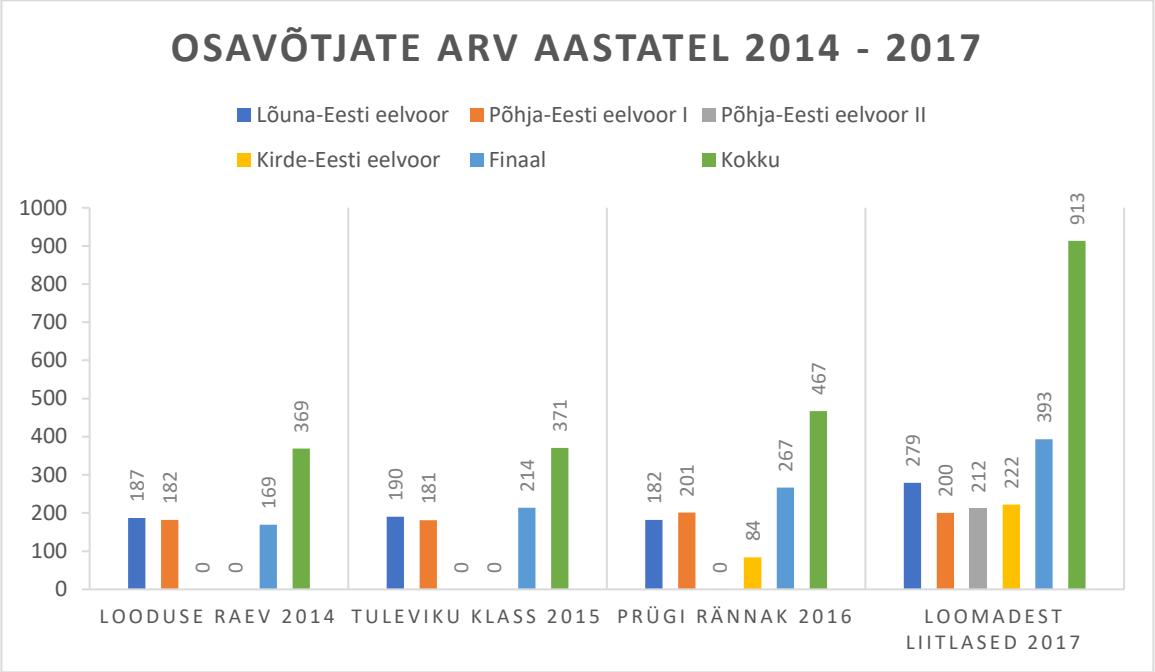
2.7 FIRST LEGO League ja FIRST LEGO League Eesti

FIRST ehk "*For Inspiration and Recognition of Science and Technology* " on ühendus, mis loodi aastal 1989 ning mille eesmärgiks on innustada noori õppima masinaehituse, teadus ja tehnoloogia alasid [28]. Koostöös LEGO'ga loodi programm FIRST LEGO League: võistlussari, kus noored 9-16 aastased peavad lahendama LEGO MINDSTORMS roboti abil ülesandeid, ent peavad ka tegema teemakohalise uurimistöö. FIRST LEGO League kogub iga aastaga aina rohkem populaarsust.

LEGO propageerib Education lehel oma toodete kasutamist ainetundide õpetamisel. Selleks ei ole vaja ehitada robotit. Komplektis oleva aju külge saab ühendada erinevaid andureid ja mõõteseadmeid ning selle abil teha erinevaid katseid ja mõõtmisi. See aitab lastele selgitada erinevaid loodusnähtuseid, füüsikat ja keemiat [29]. Peale LEGO enda andurite on võimalik kasutada ka kolmandate osapoolte, nagu Vernier, The Dexter Industries ja HiTechnic andureid. Nii öelda kolmandate osapoolte andureid on oluliselt rohkem kui LEGO andureid ning nendega on võimalik tuvastada ja mõõta rohkem erinevaid nähtusi ja väärtuseid



Joonis 7. FLL Eesti võistlustel osavõtnud meeskondade arv aastatel 2012-2017 [30].



Joonis 8. FLL Eesti võistlustel osalenud osavõtjate arv aastatel 2014 – 2017 [30].

LEGO robootika populaarsus Eesti koolides kasvab. Seda näitab osavõtvate meeskondade kasv FIRST LEGO League võistlustel (vaata joonis 7 ja 8). Võrreldes aastaga 2016 kasvas 2017 aastal osavõtjate arv peaaegu kahe kordselt. Samuti kasvab vajadus ka uudsete robootika programmide järele. Paljud õpilased, kes on osalenud FIRST LEGO League võistlusel on läbi katsetanud kõik roboti komplektiga kaasas olevad juhendid. Need õpilased vajavad uusi õppematerjale. Võimalus oleks proovida uusi ning keerulisemaid LEGO MINDSTORMS projekte. Kahjuks on LEGO MINDSTORMS graafilise programmeerimiskeelega on ebamugav keerulisi programme vähegi programmeerida. Suured programmid on kohmakad ja raskesti hoomatavad ning mõni nädal hiljem võtab koodist uuesti arusaamine kaua aega. LEGO iseenesest on suurepärane platvorm robotite ehitamiseks, sest detailid on lihtsast kokku pandavad ning vastupidavad ja samas saab roboti kerge vaevaga algosadeks ning ehitada midagi muud. LEGO roboti ehitamine pole kuidagi ohtlik ning tavapärase kasutamise juures ei ole vaja karta, et mõni komponentidest võiks kasutaja teadmatuse tõttu hävineda.

LEGO on kasvanud puuklotside tootmiselt robotite tootmisele. LEGO on säilitanud robotite juures ise-ehitamise-rõõmu ning samas pakub ka lastele programmeerimise tutvustamiseks ja õpetamiseks vägagi arvestatavat platvormi. LEGO on võimasalt ühendanud mängimise ja õppimise ning see on seda tähtsam, et programmeerimise oskus muutub aina hädavajalikumaks oskuseks. Tänu operatsioonisüsteemile ev3dev on võimalik roboteid programmeerida ka teistes keeltes, näiteks Pythonis. Programmeerimiskeele Python kasutamine EV3-e programmeerimiseks annab LEGO MINDSTORMS EV3-ele kindlasti lisa väärtuse. Järgmises peatükis kirjeldatakse kuidas paigaldada EV3 programmeeritavale ajule ev3dev operatsioonisüsteemi. Samuti

selgitatakse Pythoni programmeerimiskeskkonna PyCharmi kasulikkust roboti programmeerimisel ning kuidas paigaldada PyCharmi ja ühendust roboti ja PyCharmi vahel.

3. Ev3dev paigaldamine ja PyCharmi ülesseadmine

2016 aasta 21. detsembris lasti välja ev3dev[31]. Ev3dev on operatsioonisüsteem, mis on mõeldud kasutamiseks LEGO MINDSTORMS EV3 robotite peal. See võimaldab LEGO detailidest ehitatud robotit programmeerida erinevates programmeerimiskeeltes. Ev3dev toetab mitmeid levinud programmeerimiskeeli nagu Python, Javascript, Go, C++, C ning spetsiifilisemaid Vala, Genie ja teisi GObject'i baasiga programmeerimiskeeli koos Ev3devKit'iga. See ei tähenda, et teisi programmeerimiskeeli pole võimalik kasutada, vaid pigem seda, et teiste programmeerimiskeelte jaoks pole loodud veel juhendeid, neile veel pole loodud kõiki vajalike teeke ning nende keeltega esinevatele probleemidele pole veel hetkel lahendust leitud. Kasutaja võib ise proovida mõnda eelnevalt mitte nimetatud programmeerimiskeeltest tööle saada ning kas või ise selle keele jaoks vajalikud teegid kirjutada ja hiljem neid ka ev3dev veebikeskkonnas jagada [32]. Ka varem on olnud võimalik LEGO roboteid teistes programmeerimiskeeltes programmeerida, ent alternatiividel on üsna kallid litsentsi hinnad või väga puudulik dokumentatsioon. Ev3dev'i üheks autoriks on Ralph Hempel, kes töötab LEGO Group'is toote tehnoloogia osakonnas püsivara ja testimise meeskondade juhina. Ralph Hempel on ka python-ev3dev teegi autor. Tänu sellele teegile ongi võimalik EV3 robotit programmeerida ka Pythonis [33] [34].

Käes olevas peatükis kirjeldatakse ev3dev paigaldamist LEGO MINDSTORMS EV3 programmeeritavale ajule ning kolmandas peatükis kirjeldatakse EV3 programmeerimist Pythonis. Ev3dev paigaldusjuhendi aluseks on võetud inglise keelne paigaldus juhend ev3dev kodulehelt[1]. Töös tuuakse välja ka mõningad näidisülesanded, mis on lahendatud Pythonis. Esmalt paigaldatakse LEGO MINDSTORMS EV3 robotile operatsioonisüsteem ev3dev ja luuakse ühendus roboti ja arvuti vahel. Peatüki teises pooles kirjeldatakse JetBrains'i PyCharmi paigaldamist ja seadistamist kasutajale mugavamaks kasutamiseks. Juhend on üpris detailne eelkõige see tõttu, et juhendi sihtgrupiks on robootikaringi õpetajad. Peale juhendi läbimist on kasutajal operatsioonisüsteemi ev3dev peal töötav LEGO MINDSTORMS EV3 programmeeritavale aju, robot on ühenduses arvutiga üle Wi-Fi võrgu ning üles on seatud Pythoni programmeerimiskeskkond JetBrains'i PyCharm nii, et koodifailid PyCharmi projektis ning robotis on sünkroonselt uuendatud. Programmifailide sünkroonne uuendamine muudab EV3 programmeerimise Pythonis õpetamise juhendaja jaoks kui õppimise õpilaste jaoks oluliselt lihtsamaks.

[1] http://www.ev3dev.org/docs/getting-started/

3.1 Ev3Dev paigaldusjuhend

Ettevalmistus

LEGO MINDSTORMS EV3 programmeerimiseks Pythonis on vaja paigaldada EV3 peale Linuxi põhine operatsioonisüsteem ev3dev. Soovitatav oleks ka uuendada python-ev3dev teeki ning seejärel ongi robot valmis Pythonis programmeerimiseks. Esmalt tuleb teha mõningad ettevalmistused, et kogu see protsess oleks võimalikult sujuv.

Selleks, et programmeerida LEGO MINDSTORMS EV3 robotit Pythonis on vaja:

- LEGO MINDSTORMS EV3 programmeeritav aju ning muud detailid, millest robot ehitatakse.
- Mikro SD või mikro SDHC kaarti, mis oleks vähemalt 2GB suur, kuid mitte suurem kui 32GB. Suuremad kaardid on juba mikro SDXC ning need pole LEGO MINDSTORMS EV3 poolt toetatud. Nende kaartidega paigaldamine võib hanguda või ebaõnnestuda.
- Mikro SD kaarti adapterit arvutiga ühendamiseks. Töö käigus kasutati Kingstoni *micro SD reader* adapterit, mis ühendus arvuti USB pesasse.
- Ühendus arvuti ja EV3 programmeeritava aju vahel. Ühendust saab tekitada järgnevalt: USB kaabli abil, mis tuli kaasa EV3'ga, USB-na ühilduv Wi-Fi kaart, USB-na ühilduv Interneti kaabli pesa, Bluetooth Antud töös on kasutatud USB-na ühilduvat Edimax Wi-Fi kaarti.
- Wi-Fi võrku, üle mille ühendatakse arvuti ja robot.
- Arvutit administraatori õigustega, mida on tarvis, et SD kaardile paigaldada operatsioonisüsteem ning hiljem ka robotiga suhtlemiseks ja programmeerimiseks.
- Ev3dev kujutisfaili, see on paigaldusfail ev3dev operatsioonisüsteemile, mis tuleb paigaldada mälukaardile. Ev3dev kujutisfaili leiab ev3dev kodulehelt[2]. Alla laadimiseks tuleb valida *„Download for LEGO MINDSTORMS EV3 (211 MiB)"* ning seejärel alustatakse ev3dev operatsioonisüsteemi kujutisfaili laadimisega, mille suurus on 211 MiB.
- Programmi Etcher, mis vabavaraline ja avatud lähtekoodiga programm. Etcher'i abil saab väga lihtsalt paigaldada mälukaardile ev3dev operatsioonisüsteemi.
- Programmi PuTTY, mis on vajalik, et suhelda LEGO MINDSTORMS EV3-ga üle Wi-Fi.

Sellega piirdub ka minimaalne vajadus, kuid on veel paar tööriista, mis oluliselt lihtsustavad protsessi programmeerimise poole pealt. Python on selles mõttes väga vähe nõudlik keel, et seda on võimalik täitsa edukalt kirjutada igasuguse tekstiredaktoriga. Ev3dev on Linux'i põhine operatsioonisüsteem ning suhtlus EV3-ga toimub PuTTY abil, mis on käsurealiides. Kui kasutaja ei tunne end mugavalt käsureal, siis on võimalik minimaliseerida käsurea kasutamise vajadust ning enamus tööd siiski ära teha oma arvutis. Kogu programmeerimine on võimalik ära teha suvalises

[2] http://www.ev3dev.org/docs/getting-started/#step-1-download-the-latest-ev3dev-image-file

tekstiredaktoris, ent mugavam kasutada mõnda redaktorit, mis mõnel määral tunneb Pythonit. Pythonit tundva redaktori kasutamise teeb mugavaks see, et kasutajat teavitatakse kui ridade taanded on valed ning värvib ära süntaksi. Selleks sobib näiteks Notepad++, Atom või Pythoni enda Python idea GUI. Isiklikult leidsin, et kõige mugavam on kasutada JetBrains'i PyCharmi, mis on õppe eesmärkidel täiesti tasuta kasutatav. Sinna on võimalik lisada ev3dev teek, mis tõttu PyCharm hakkab prognoosima meetodeid, mida kasutaja parasjagu sisestab ning samuti aitab programm juba ise koodi paremaks teha soovitades aeg-ajalt ilusamaid koodi stiile, parandades taandeid ning redigeerides puuduvaid ja üleliigseid tühikuid.
Et käsurea kasutamist veel minimaliseerida võib kasutada PyCharm Professional versiooni võimalust ühenduda eemalasuva süsteemiga. Teine võimalus on kasutada WinSCP programmi, mis võimaldab lihtsasti faile arvutist EV3 peale laadida neid lihtsalt lohistades oma arvutist EV3-e sobivasse kausta. WinSCP programmiga saab üles seada ka automaatse sünkroniseerimise robotiga nii, et koodifaili uue versiooni salvestamisel arvutis laetakse uus versioon kohe robotisse. Mõlemast võimalusest räägitakse täpsemalt punktis 3.2.

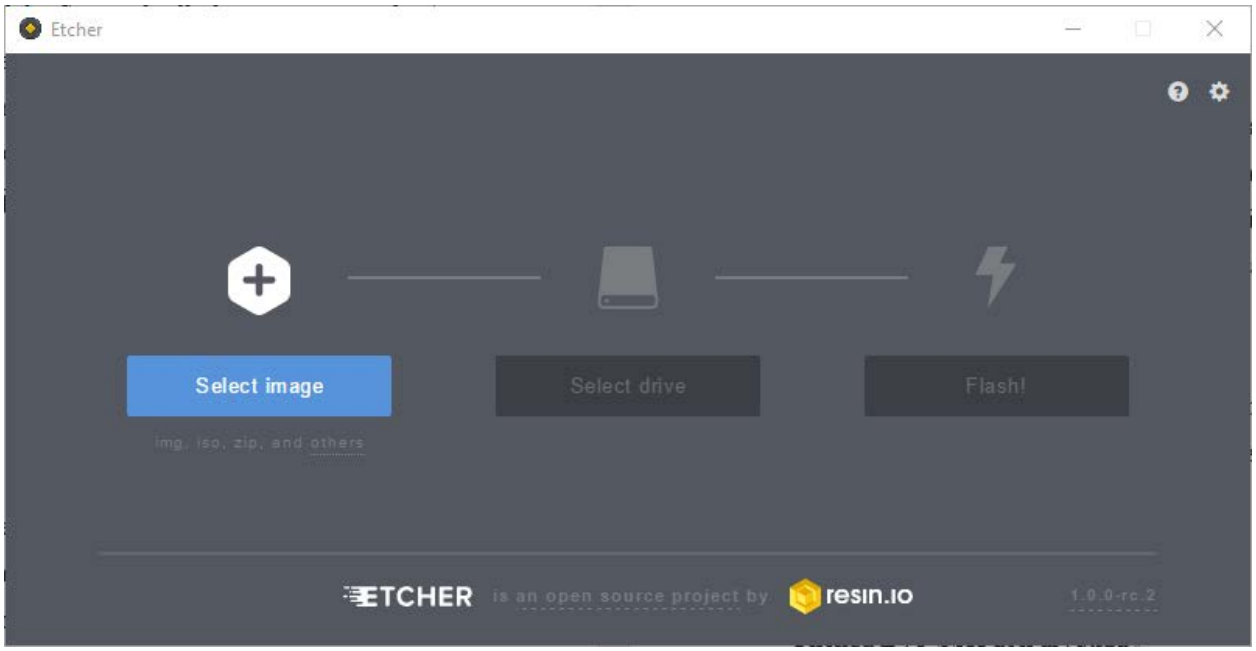
Etcher

Etcher on programm, mille abil saab lihtsasti *image file* ehk kujutisfaili[3] formaati kirjutada SD kaartidele ja mälupulkadele, et seda saaks hiljem kasutada näiteks operatsioonisüsteemi paigaldamiseks. Kujutisfail on ketta kloonimisega loodud fail, milles on mingi füüsilise salvestuskandja sisu täielik koopia koos struktuuriteabega [35]. See protsess erineb tavapärasest andmete kirjutamisest kuna operatsioonisüsteem on vaja paigaldada ja käivitada enne kui mõni teine operatsioonisüsteem seda teeb. Antud juhul on vaja, et ev3dev käivituks enne kui EV3 operatsioonisüsteem ning selleks ongi vaja ev3dev kirjutada SD kaardile Etcheri abil. Etcher programmi paigaldusfaili leiab Etcheri kodulehelt[4]. Lehe keskel on suur roheline nupp *„Download for Windows"*, mis käivitab alla laadimise. Teiste operatsioonisüsteemide jaoks leiab paigaldusfaili vajutades rohelise paremal ääres asuvale noolele. See avab rippmenüü, kus saab valida paigaldusfaili oma operatsioonisüsteemile. Peale allalaadimist saab programmi paigaldada käivitades paigaldusfaili. Paigaldamise lõppedes on Etcher valmis kasutamiseks ning nüüd vaja mikro SD või mikro SDHC kaardile paigaldada ev3dev kodulehelt allalaetud kujutisfail. Ev3dev operatsioonisüsteemi paigaldamiseks mälukaardile on vaja läbida järgnevad sammud. Joonistel 9-13 on kujutatud programmi Etcher kasutamist ning selle abil SD kaardile ev3dev kujutusfaili paigaldamist.

[3] http://www.vallaste.ee/
[4] https://etcher.io/

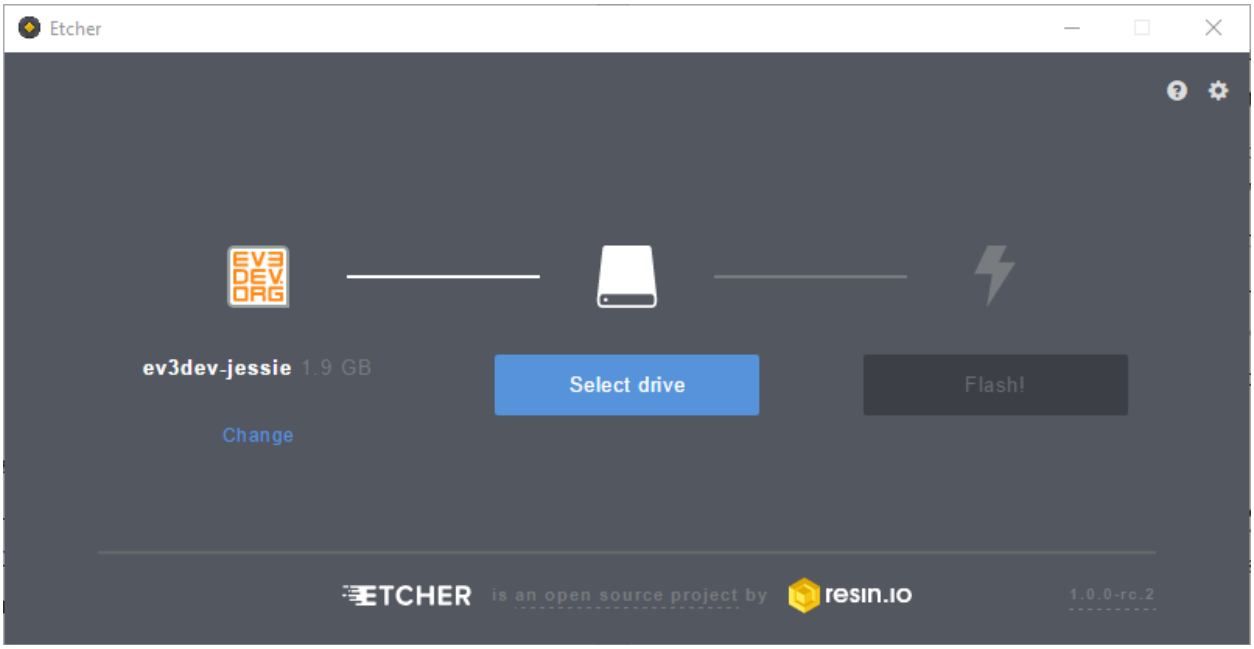
1. Käivita programm Etcher.

2. Vali allalaetud kujutisfail.



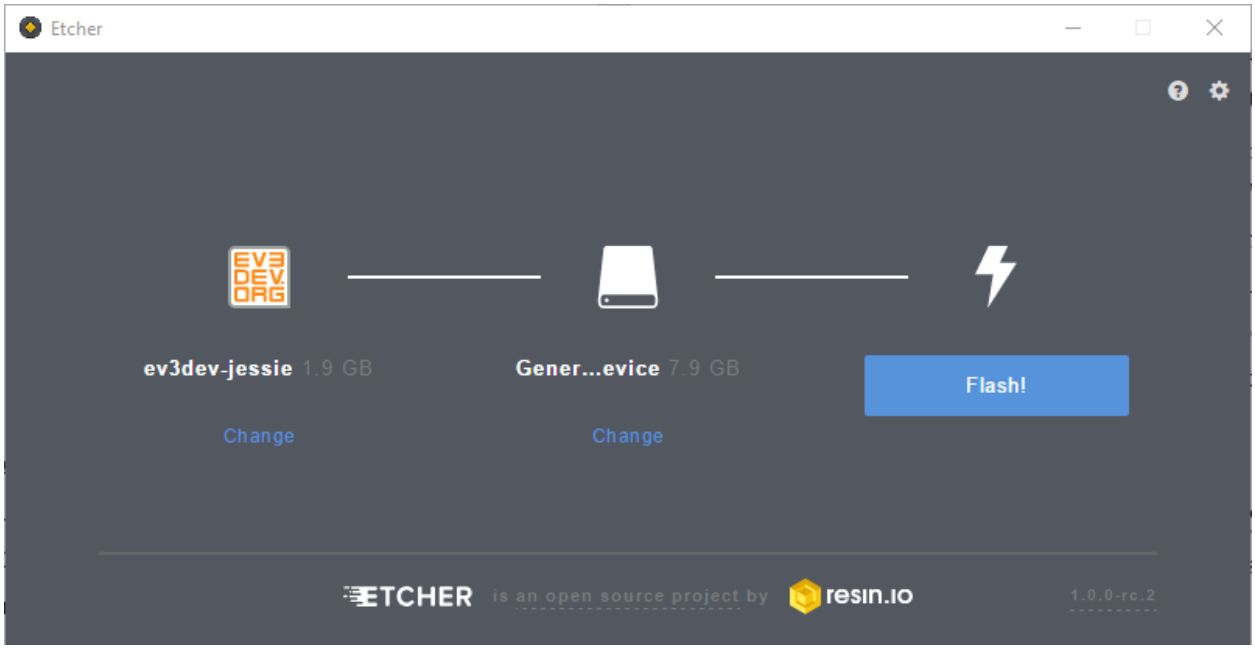
Joonis 9. Programm Etcher peale käivitamist.

3. Vali arvutisse ühendatud mikro SD või mikro SDHC kaart, millele operatsioonisüsteem paigaldatakse.

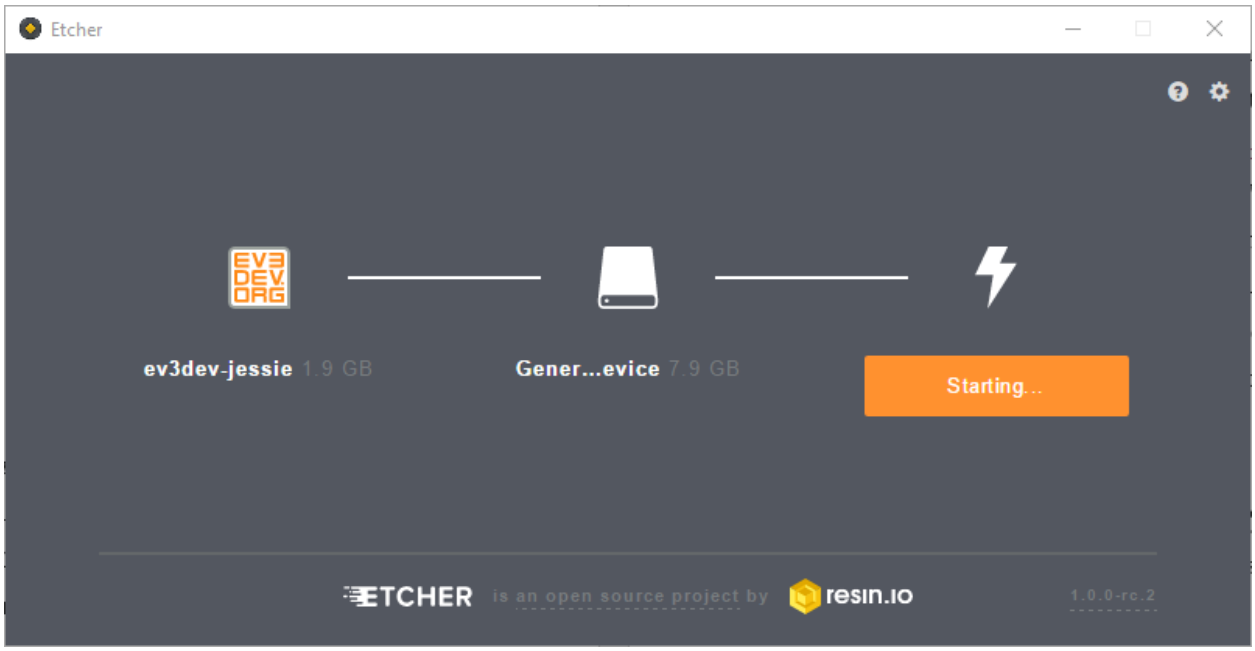


Joonis 10. Programm Etcher - paigaldamiseks on valitud ev3dev operatsioonisüsteemi kujutisfail.

4. Nüüd on piisav ettevalmistus tehtud ning võib käivitada kirjutamise nupuga „*Flash*“.

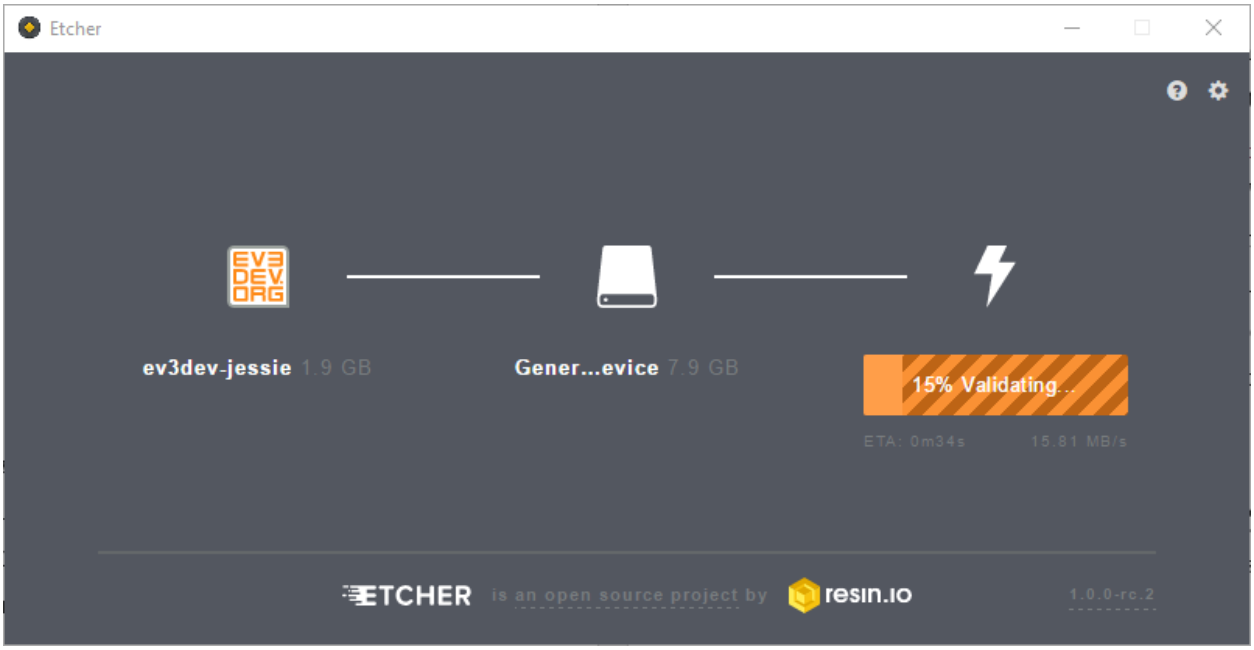


Joonis 11. Programm Etcher - valitud kujutisfail ning mälukaart, millele operatsioonisüsteem paigaldatakse.



Joonis 12. Programm Etcher -operatsioonisüsteemi kujutusfaili kirjutamine kaardile on käivitatud.

5. Programm kirjutab kaardile operatsioonisüsteemi kujutusfaili ning valideerib, et fail oleks terve.



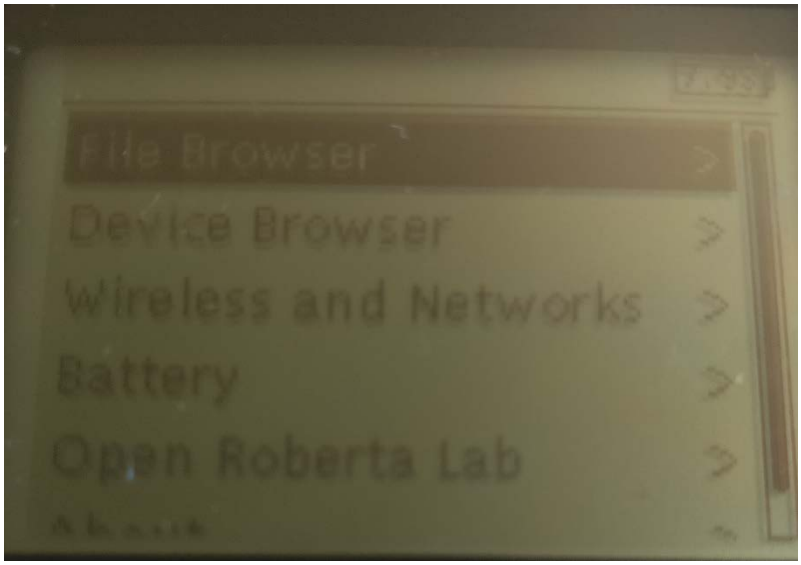
Joonis 13. Programm Etcher - peale kujutusfaili paigaldus kontrollitakse, et protsess oli edukas.

Kui kontrollimine on lõpetatud on operatsioonisüsteemi kujutusfail edukalt paigaldatud kettale ning mälukaarti võib arvutist eemaldada. Nüüd on võib mälukaarti sisestada EV3 programmeeritava aju SD kaari pesasse. Järgnevalt paigaldatakse ev3dev operatsioonisüsteem SD kaardile EV3 programmeeritava aju abiga, siis ühendatakse programmeeritav aju Wi-Fi võrku ning luuakse ühendus arvutiga.

Ev3dev paigaldamine LEGO MINDSTORMS EV3 programmeritavale ajule

Mälukaardile on paigaldatud ev3dev operatsioonisüsteemi kujutusfail. Operatsioonisüsteem hakkab kõvakettana kasutama mälukaarti. EV3 programmeeritavale aju kõvakettale talletatud failid ehk EV3 püsivara kasutavad failid jäävad alles. Naasmiseks tavapärase EV3 kasutuse juurde tuleb aju välja lülitada ning SD kaart pesast eemaldada. Aju uuesti tööle lülitades saab jätkata tavapärase LEGO MINDSTORMS EV3-e kasutamisega. Esmalt pakitakse mälukaardile ev3dev operatsioonisüsteem lahti LEGO MINDSTORMS EV3 programmeritava aju abiga ning seejärel ühendatakse programmeeritava ajuga USB-na ühilduv Wi-Fi kaart. Lõpuks ühendatakse EV3 sobivasse Wi-Fi võrku.

1. Mälukaart sisestatakse EV3 aju SD kaarti pessa ning vajutatakse *Enter* nuppu ehk keskmist nuppu. Korraks süttivad punased LED-tuled ning siis hakkavad tuled vilkuma oranžilt ja samal ajal jookseb ekraanil väikeses fondis tekst. Esmakordsel paigaldusel võib see aega võtta kuni 5 minutit.
2. Paigalduse lõppedes muutub valgus roheliseks ning ekraanile ilmub ev3dev peamenüü. Menüüdes saab liikuda EV3 nuppudega (vaata joonis 15).



Joonis 14. Ev3dev peamenüü.

3. EV3 programmeeritava aju USB pessa ühendatakse USB-na ühilduv Wi-Fi kaart. Kui Wi-Fi kaardil peaks olema sisseehitatud valgus (LED tuluke), siis on ootuspärane, et see veel ei põle.
4. Peamenüüst tuleks valida *„Wireless and Networks“* ning alammenüüst *„Wi-Fi“*. Avanenud aknas on näha valik *„Powered“* ning see pole valitud (kasti sisu pole sama värvi nagu kasti ümbris). See tähendab, et Wi-Fi kaart pole aktiveeritud. Vajutades *„Powered“* peal keskmist nuppu ilmub selle valiku alla uus valik *„Start Scan“*, mille valides hakkab EV3 otsima Wi-Fi võrke. Peale *„Powered“* valimist on ootuspärane, et süttib ka Wi-Fi kaarti valgus selle olemasolul.
5. Kirjeldatud valikute alla on tekkinud tabel *„Networks“*, kus ilmuvad kõik EV3 leitud Wi-Fi võrgud. Sellest menüüst tuleb üles otsida endale sobiv võrk ning vajutada keskmist nuppu.
6. Avatud on võrgu nimega vaade ning menüüs on teine valiku variant *„Connect“*, mille valides saab ühenduda sellesse võrku. Kui Wi-Fi võrk on parooliga kaetud avaneb väiksem aken, kus palutakse sisestada parool selle võrgu jaoks. Vajutades keskmist nuppu teksti välja peal avaneb uus aken. Vaheaknas on tekstiväli, kuhu parooli trükitakse ning selle all klaviatuur, millel saab liikuda noole klahvide abil ning tähemärk trükitakse keskmise nupuga. Tähemärke saab kustutada tagasi-nupuga. Teksti välja ja klaviatuuri vahele jäävad nupud, mis võimaldavad valida suuri või väikseid tähti, numbreid ning sümboleid. Kui

parool on sisestatud kinnitatakse see valides *„OK“* nupu. Parool kinnitatakse uuesti ka väiksemas vaheaknas.

7. EV3 ühendub Wi-Fi võrku ning sellele määratakse IP, mis kuvatakse ekraanil üleval vasakul ääres.

Esmakordne operatsioonisüsteemi paigaldus võib aega võtta umbes 5 minutit. Kui paigaldus ei käivitu või võtab oluliselt rohkem aega võib olla probleem üks järgnevatest:

- EV3 programmeeritava ajuga on ühendatud ka teised seadmed (mootorid, sensorid, USB seadmed) peale operatsioonisüsteemi sisaldava mälukaarti. Tuleks eemaldada teised seadmed on alustada paigaldamist otsast peale.
- Ev3dev operatsioonisüsteem on kaardile paigaldatud vigaselt. Paigaldada ev3dev operatsioonisüsteem mälukaardile uuesti.
- Mälukaart on vigane. Võimalusel korrata paigaldamist mõne teise mälukaartiga.
- EV3 programmeeritava aju toide on puudulik. Kontrollida, et aku oleks töökorras ning et aku oleks piisavalt täis.

Ettevalmistused on nii kaugel, et EV3-le on paigaldatud Debian Linux põhine operatsioonisüsteem ev3dev ning EV3 programmeeritav aju on ühendatud sobivasse Wi-Fi võrku. Kui robotit parasjagu ei kasutata, siis võib selle vabalt ka välja lülitada ilma, et midagi kaotsi läheks. Selleks tuleb minna peamenüüsse ning vajutada tagasi nuppu (nupp, mis asetseb vasakul ekraani all) ning avanevast menüü aknast valida *„Power off“*. Kõik, mis on salvestatud mälukaardile jääb sinna alles, samuti Wi-Fi ühenduse valik, ka seda pole vaja hiljem uuesti seadistada juhul kui kasutate sama Wi-Fi võrku. Taas käivitamine näeb suhteliselt sarnane välja esialgse paigaldusega, kuid peaks oluliselt kiirem olema.

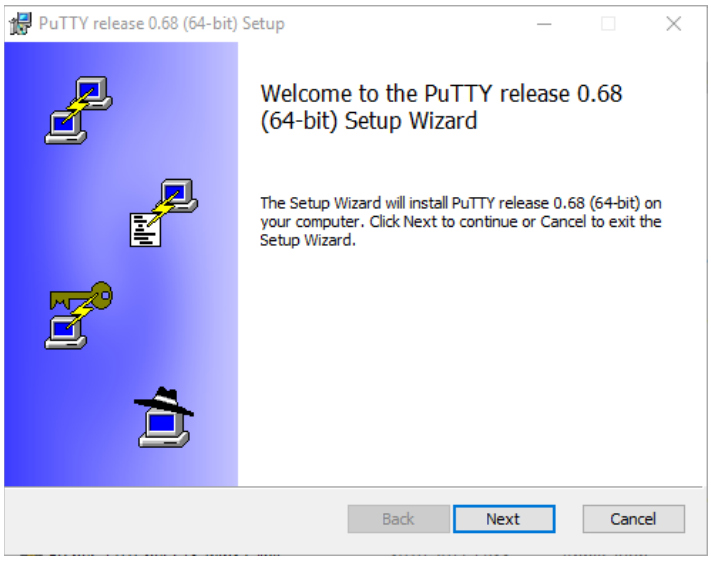
Programmeeritava aju ekraanilt on näha, et aku tase pole kuvatud mitte protsendina, vaid arvuna. Selle põhjuseks on, et aku taset pole võimalik täpselt protsendiks ümber arvutada ning hoopis täpsem on kuvada aku pinge. Aku pinge peaks jääma 8 ja 5 voldi vahele. Aku pinge on 8 volti, siis kui aku on täis ning 5 voldi juures on aku tühi ning lülitab ennast välja. Protsendi arvutamise probleem tuleneb sellest, et aku pinge võib langeda 8-lt voldilt 6,5 voldi peale palju kiiremini kui 6,5-lt voldilt 5 voldini.

LEGO MINDSTORMS EV3 roboti ühendamiseks arvutiga on veel vaja ainult arvutisse paigaldada programm PuTTY.

PuTTY paigaldamine ning EV3-ga ühenduse loomine

Robot on valmis ühendamiseks arvutiga. Nende omavaheliseks suhtluseks kasutame programmi PuTTY abi. PuTTY on Simon Tatham'i loodud tasuta SSH ehk *Secure Shell* klient, mis võimaldab ühendada arvuteid üle võrgu [36]. Programmi abil viiakse läbi suhtlus robotiga, tehakse ära uuendused ja pannakse tööle juba eelnevalt valmis programmeeritud programmid. Programmi saab alla laadida kodulehelt[5]. Lehelt leiab rohelise kasti nimega *„Package files"* ning selle sees on installfaili allalaadimis link nimega: putty-<versiooni_number>-installer.msi 32-bitise programmi jaoks ning putty-64bit-<versiooni_number>-installer.msi 64-bitise programmi jaoks. 64-bitise versioon tuleks valida juhul arvutil on 64-bitine protsessor ning arvutisse on paigaldatud 64-bitine operatsioonisüsteem. Samuti on see turvaline valik kui teate, et arvutil on rohkem kui 4 gigabaiti muutmälu ehk RAM-i, sest 32-bitine süsteem ei toeta rohkem kui 4 GB muutmälu. Kui on teada, et arvuti kasutab 32-bitist süsteemi, siis tuleks valida ka 32-bitine programm. Ebakindluse korral võib alati valida 32-bitise programmi. Kasutades 32-bitist programmi 64-bitisel süsteemil töötab kõik samamoodi, kuigi 64-bitine programm peaks olema mõnevõrra kiirem, kuna kasutab rohkem muutmälu [37]. Kui programm on allalaetud, siis tuleks see ka paigaldada käivitades installfaili administraatori õigustes. Joonistel 15-18 on kujutatud programmi PuTTY paigaldamine.

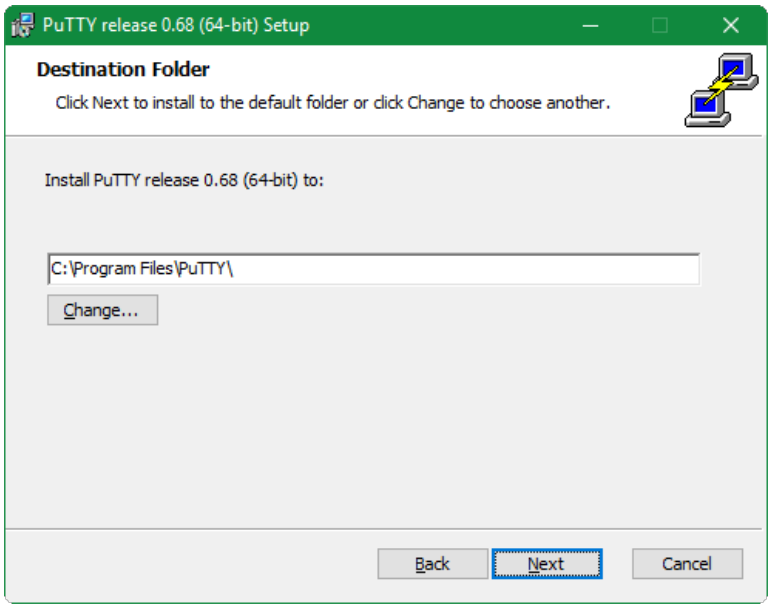
1. Programm PuTTY on käivitatud. Avanenud tervitus aknast saab edasi liikuda nupu *„Next"* abil.



Joonis 15. Programm PuTTY paigalduse tervitus aken.

[5] http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

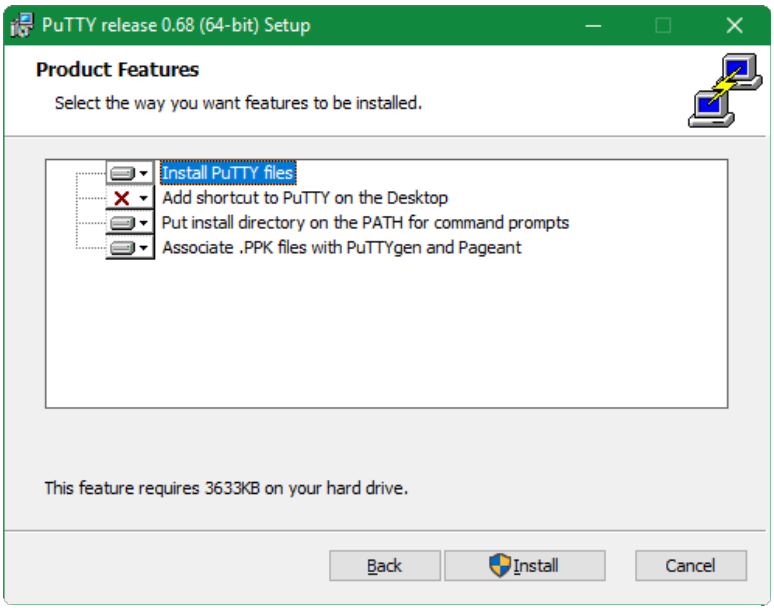
2. Teksti väljal kuvatakse PuTTY paigalduse asukoht. Seda saab muuta nupu „*Change*“ abil. Kui sobiv asukoht on leitud, siis saab edasi liikuda nupu „*Next*“ abil.



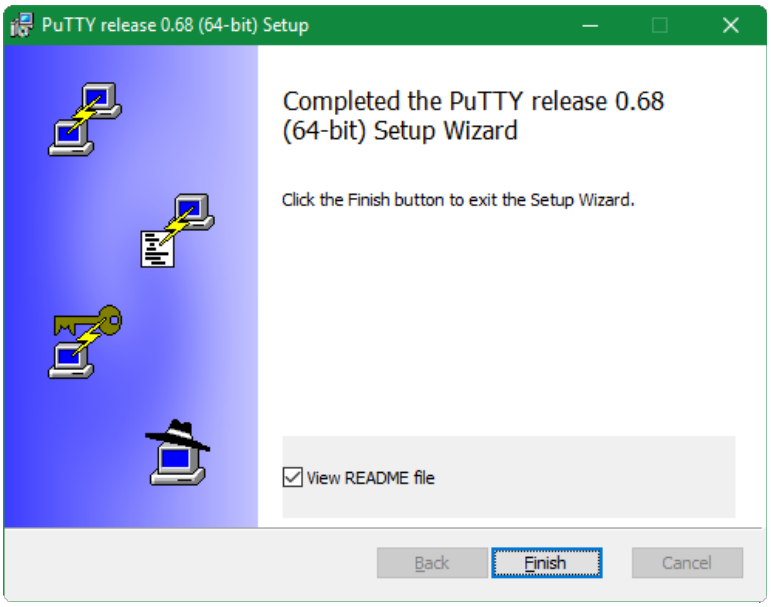
Joonis 16. PuTTY paigalduse asukoha valimine.

3. Siit saab valida sobivaid lisasid PuTTY programmile nagu näiteks töölaua otsetee, mis on vaikimisi välja lülitatud. Samuti kuvatakse, et PuTTY vajab paigalduseks 3633KB kõvaketta ruumi. Paigaldus käivitub kui vajutada nuppu „*Install*“.

4. Kui paigaldus on lõppenud võib linnukese ära võtta „*View README file*“ valiku eest ning programmi sulgeda nupust „*Finish*“.

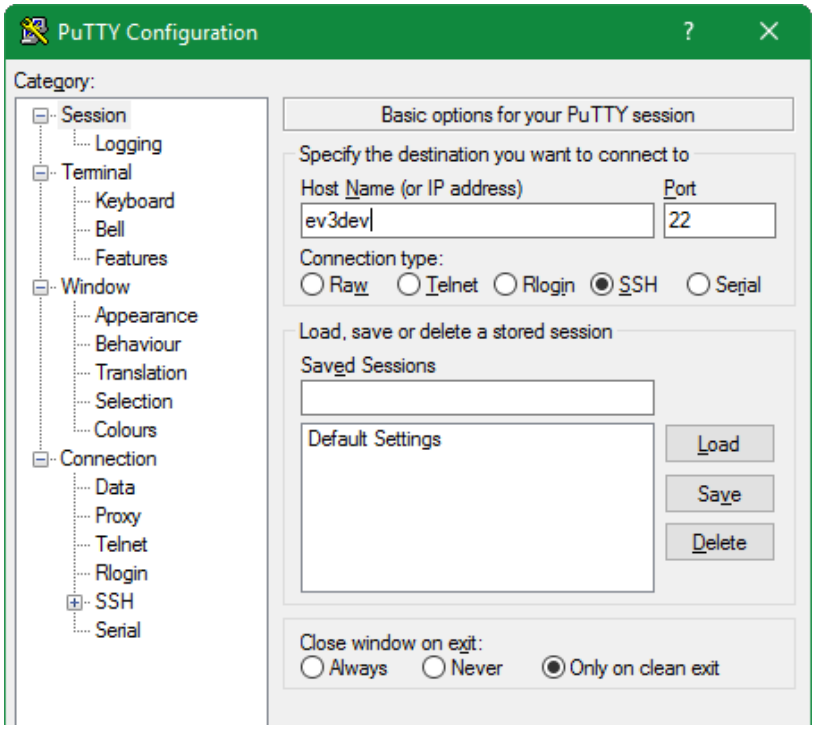


Joonis 17. PuTTY lisandite valiku menüü



Joonis 18. PuTTY paigaldus on lõppenud.

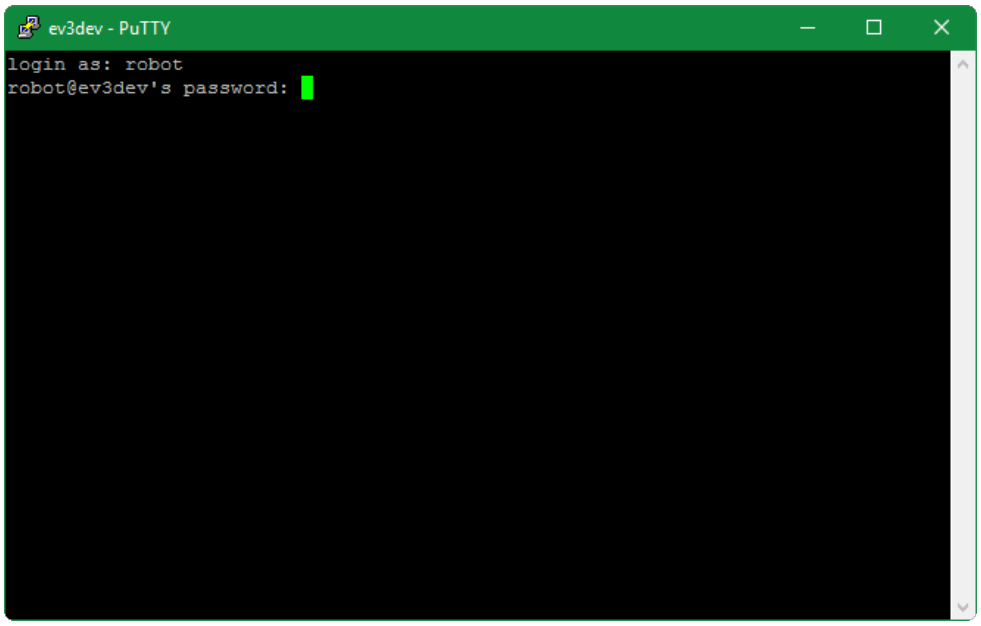
5. Veendutakse, et kasutatav arvuti ning EV3-e programmeeritav aju on ühendatud samasse Wi-Fi võrku.
6. Käivitatakse programm PuTTY. Avanenud konfiguratsiooni aknas tuleb ühenduse võõrustaja nime ehk *„Host Name“* väärtuseks sisestada *ev3dev*. Ühendus luuakse vajutades *„Open“* nuppu (vaata joonis 20).



Joonis 19. PuTTY konfiguratsiooni aken.

7. Esmakordsel ühenduse loomisel võib ilmuda hoiatus, et ühendus ei ole turvaline ning kaasab endaga turvariske. Akna võib sulgeda aktsepteerides riske ning vajutades „*Yes*“ nupule.

8. Avanenud konsooliaknas tuleb ühenduse EV3-ga ühenduse loomiseks sisestada kasutajanimi ning parool (vaata joonis 20). Vaikimis on kasutaja nimi *robot* ja parool *maker*. Parooli sisestamisel seda ei kuvata isegi mitte tärnidena, et kaitsta parooli ning selle pikkust inimeste eest, kes võivad teie ekraani jälgida.



Joonis 20. EV3-ga ühenduse loomine konsooliaknas.

9. Ühendus on loodud ning robot on valmis programmeerimiseks (vaata joonis 21).

Joonis 21. Ühendus EV3-ga on loodud ning robot on valmis kasutamiseks.

Nüüd on robot ühenduses arvutiga ning robotile saab anda käsklusi PuTTY käsurealt. Esmalt võiks teha robotile kontroll uuenduse, et mõni vajalik pakk poleks vananenud. Seda saab teha käsuga *sudo apt-get -y upgrade* ning kui uuendus on lõpetanud tuleks uuendada ka Pythoni python-ev3dev teek ning selleks on käsk *sudo apt-get install --only-upgrade python3-ev3dev* [38]. Kui uuendused on läbi viidud, siis on robot valmis Pythonis programmeerimiseks. Kuna Python on võrdlemisi vähenõudlik programmeerimiskeel, siis on seda üsna lihtne ka kohe käsureal programmeerida. Selleks pole vaja teha muud kui luua Python fail ning kirjutada valmis programmikood käsuga *nano <programmi_nimi>.py*. Kui programm on valmis saab programmi *nano* sulgeda kasutades klahvi kombinatsiooni CTRL+X. Programmi saab jooksutada kasutades käsku *python3 <programmi_nimi>.py*. Programmide jaoks ei ole vaja luua eraldi kaustu, ent soovi korral on see siiski võimalik. Tegemist on Linuxi põhise operatsioonisüsteemiga ning Internetis on palju näiteid, kuidas käsurea abiga faili süsteemis liikuda, kuidas faile ja kaustu luua jne. Käsurealt on võimalik EV3 ka välja lülitada ning selleks tuleb kasutada käsklust *sudo poweroff*. EV3 programmeerimiseks Pythonis saab luua kõik programmid ühte kausta ning programmide tööle panemiseks on vaja ainult käsku *python3 <programmi_nimi>.py*. Antud töös käsureal töötamist täpsemalt ei kirjeldata.

Roboti programmeerimiseks on erinevaid võimalusi ning seda on võimalik teha ka otse käsurealt. Et aga käsuread ja puudulikud kasutajaliidesed tekitavad tihti segadust ja ebameeldivusi, siis kasutame EV3 programmeerimiseks Pythonis JetBrains'i loodud programmi PyCharm.

3.2 Pythoni programmeerimiskeskkond JetBrains'i PyCharm

Kuigi programme Pythonis, nagu ka mitmetes teistes programmeerimiskeeltes, on võimalik kirjutada lihtsa tekstiredaktoriga nagu näiteks Notepad. Programmeerida on siiski lihtsam kasutades mõnda programmeerimiskeskkonda. Korralik programmeerimiskeskkond aitab programmeerijal ennetada lihtsamaid vigu näiteks hoiatades, et koodis on taanded valesti, soovitud meetodite jaoks on puudu importimised jne. JetBrains'i PyCharm on Pythonis programmeerimiseks väga hea tööriist. Siinkohal on sobilik välja tuua PyCharmi kasulikud omadused:

- Teeke saab kasutaja otsida ning lisada läbi PyCharmi kasutajaliidese ning ei pea neid Internetist otsima.
- PyCharm proovib ennustada, mida kasutaja trükib, pakkudes võimalikke meetodite nimesid, mis teeb programmeerimise lihtsamaks kui ei teata meetodi täpset nimetust või soovitakse otsida alternatiive mõnele meetodile.
- Paari nupu vajutusega on võimalik korrastada koodis taandeid, mis on Pythonis hädavajalik ja tühikuid, et kood näeks ilusam välja.
- PyCharm teavitab kui mõni koodi osa on üleliigne.
- Annab kasutajale teada kui tema koodis on midagi valesti, näiteks kui koodis on osa, milleni pole loogiliselt võimalik jõuda, taanded on koodis valed või kui koodis kasutatavad andmetüübid ei sobi kokku.

JetBrains PyCharmil on 2 erinevat versiooni: *Professional* ja *Community*. *Community* versioon on tasuta ning avatud lähtekoodiga. *Community* versioon annab võimaluse kasutada peaaegu kõiki PyCharmi kasulikke omadusi, mis aitavad kaasa EV3 programmeerimisele Pythonis. *Professional* versioon võimaldab luua projekte ka üle SSH ühenduse [39]. Kui projekt on loodud üle SSH ühenduse, siis kaob ära vajadus kasutada käsurida – ehk programmi muutmisel see uuendatakse robotis automaatselt ning programme saab robotis tööle lülitada PyCharmi kasutajaliidesest. Kuigi *Professional* versioon on tasuline, siis on koolidel võimalik taotleda PyCharm *Free Educational* (*Classroom*) litsentsi. See litsents on tasuta ning annab õppeasutustel võimaluse kasutada PyCharm täisversiooni õppe eesmärkidel. Litsentsi taotluse saab edastada JetBrains'i kodulehelt[6]. Töös ei ole siiski eeldatud *Professional* versiooni kasutamist ning pakutakse ka alternatiivset lahendust, kuidas Pythonis kirjutatud programmid saab laadida EV3 programmeeritava aju peale ning käivitada. PyCharm ei eelda kindla Pythoni versiooni olemas olu ning PyCharmis on võimalik korraga seadistada mitu erinevat Pythoni versiooni vastavalt vajadusele. Selleks, et saaks alustada LEGO MINDSTORMS EV3 programmeerimist PyCharmiga Pythonis on vaja paigaldada arvutisse Python, PyCharm ning *Community* versiooni kasutades ka WinSCP, mida kasutatakse programmide kopeerimiseks arvutist robotisse.

1. Arvutisse tuleb paigaldada Python 3 või uuem, sest python-ev3dev teek töötab ainult Python 3.x-ga (x tähistab alamversiooni numbrit) [40]. Kõige uuema Pythoni saab alla

[6] https://www.jetbrains.com/buy/classroom/?product=pycharm

laadida kodulehelt[7]. Taaskord ei ole vahet kas kasutada 32- või 64 bitist Pythonit, mõlemad sobivad antud kontekstis. Käivitades Pythoni paigaldusfaili saab alustada paigaldamist vaikeväärtustega valides „*Install Now*“. Soovi korral saab muuta paigalduse asukohta valides „*Customize installation*“. Kui Python on edukalt paigaldatud tuleb järgmiseks paigaldada PyCharm.

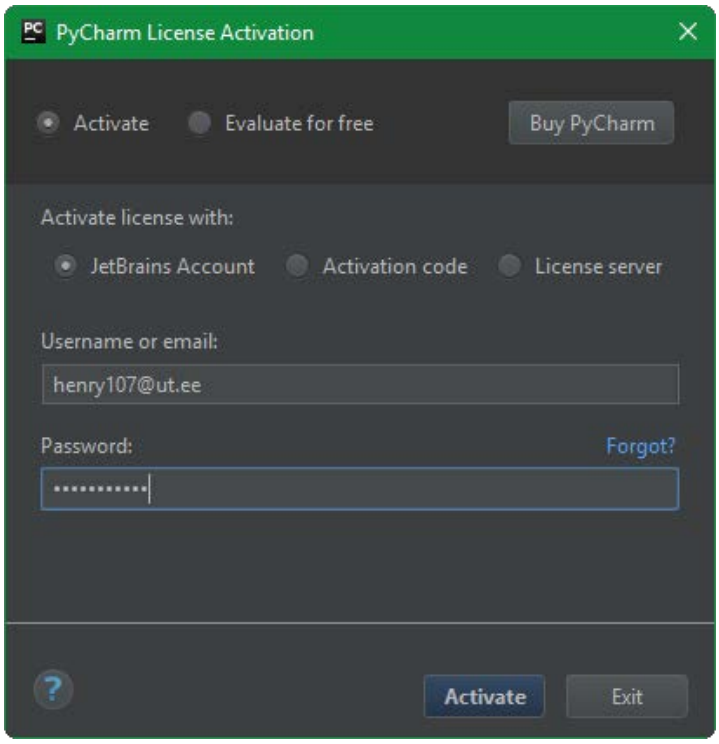
2. PyCharmi saab alla laadida JetBrains'i kodulehelt[8]. Sobiva litsentsi olemas olul tuleks alla laadida *Professional* versioon, vastasel juhul, aga *Community* versioon. Litsents kinnitatakse peale alla laadimist ning paigaldamist, logides kasutajana sisse PyCharmi.
3. PyCharmi paigalduseks tuleb käivitada allalaetud paigaldusfail. Paigaldamiseks tuleb määrata kaust, kuhu PyCharm paigaldatakse. Asukohaks sobib ka vaikeväärtus. Paigaldamisel saab valida 32- ja 64-bitise töölaua otsetee vahel. Teadmiste korral tuleks valida endale sobivam ning teadmiste puudumisel on kindlam valida 32-bitine versioon. Samuti peaks tegema linnukesse „*Create association*“ valikus „.py“ ette. See tagab, et „.py“ laiendiga failide avamiseks kasutatakse vaike valikuna PyCharmi. Kui valikud on tehtud käivitatakse paigaldus. Paigalduse lõppedes ilmub väike aken, kus küsitakse kas kasutaja soovib importida PyCharmi seadistus. Eeldusel, et arvutisse ei ole varem paigaldatud PyCharmi valib kasutaja alumise valiku „*Do not import settings*“, ehk ei soovi importida seadistust ning sulgeb akna vajutades „OK“ nupule. Nüüd algab PyCharmi esmane käivitus ja konfigureerimine. Siinkohal lähevad tasulise ja tasuta PyCharmi versioonide juhendid lahku. Mõlemad juhendid kirjeldavad, kuidas luuakse PyCharmis projekt, kuidas toimub andmevahetus arvuti ja roboti vahel ning kuidas käivitada programm robotis.

[7] https://www.python.org/downloads/
[8] https://www.jetbrains.com/pycharm/

PyCharm Professional esmane käivitamine ning konfigureerimine

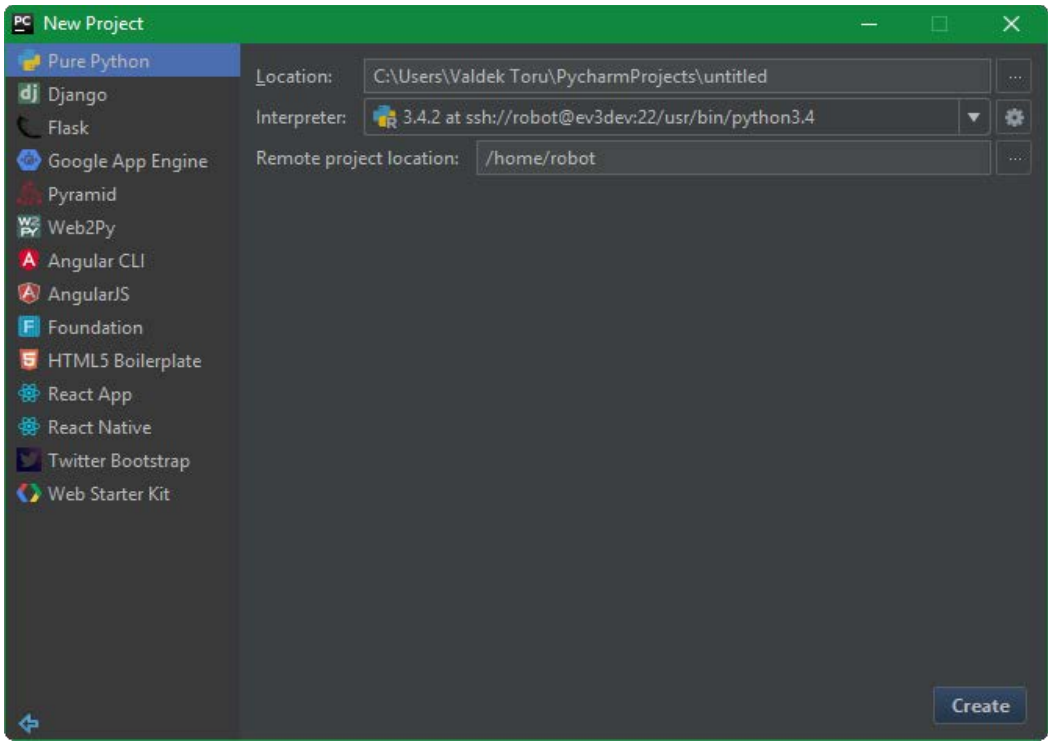
PyCharmi *Professional* versioon on küll tasuline versioon, ent selle tasuta litsentsi on võimalik koolidel ja õppeasutustel JetBrains'i kodulehelt taotleda. PyCharm *Professional* versiooni abil loodud SSH ühendus lihtsustab andmevahetust nii palju, et programmeerimisel ja testimisel pole kasutajal vaja kasutada käsurida üldse.

PyCharmi käivitamise ilmub aken, milles aktiveeritakse litsents logides sisse (vaata joonis 22)



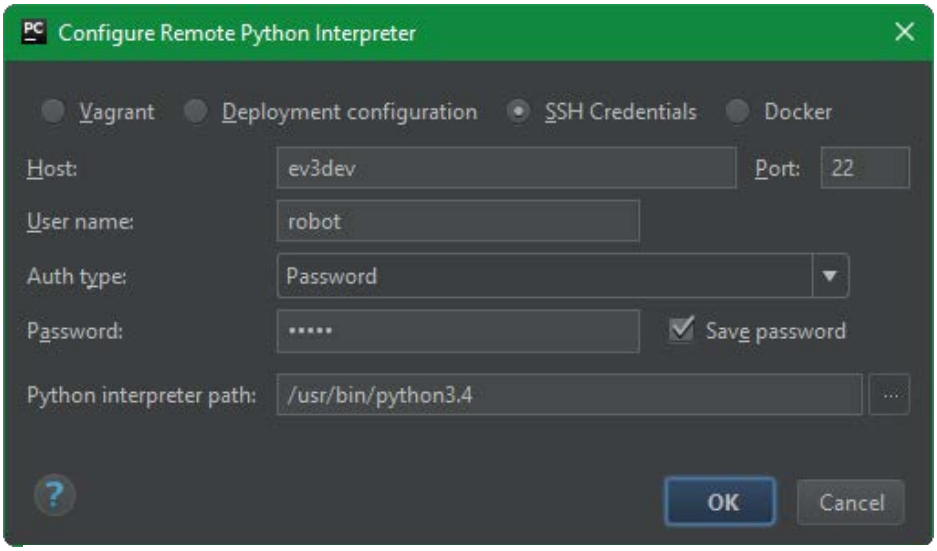
Joonis 22. PyCharmi litsentsi aktiveerimise akna vaade

JetBrains lehe kasutajaga või järgides kodulehelt saadud juhiseid. Kui litsents on aktiveeritud avaneb uus aken, kus kasutaja saab endale sobiliku kasutajaliidese teema ja kirja fondi kombinatsiooni valida ning seejärel suletakse aken vajutades „*OK*" nupule. Nüüd PyCharm aktiveeritud ning ka algne konfiguratsioon on tehtud. Nüüd luuakse näidisprojekt ning PyCharm ühendatakse LEGO MINDSTORMS EV3-ga. Projekti loomiseks valitakse PyCharm tervitus aknast „*Create* New Project". Selle peale avaneb uue projekti konfigureerimise aken nagu kujutatud joonisel 23. Vasakust menüüst valitakse „*Pure Python*", mis tähistab, et tegu on Pythoni projektiga. Paremal pool vaates on näha väli „*Location*", kuhu märgitakse projekti asukoht arvutis. Järgmiseks valitakse „*Interpreter*"- interpretaator, ehk Pythoni versioon, millega hakkab projekt töötama.



Joonis 23. PyCharmi uue projekti konfigureerimise akna vaade.

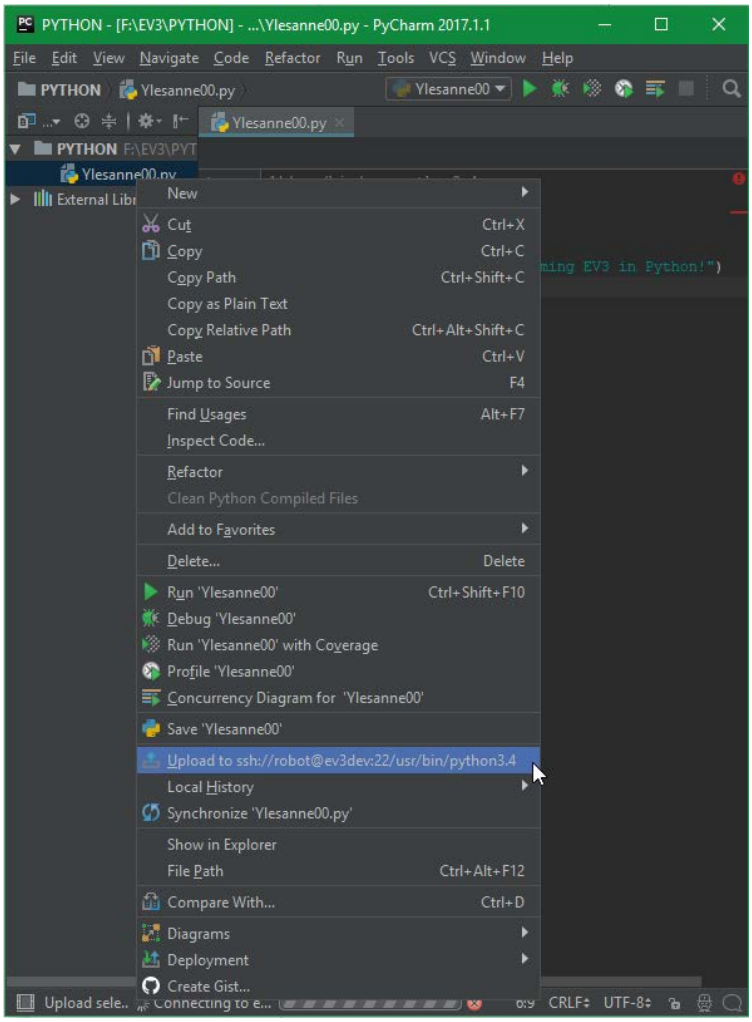
Vaikeväärtusena peaks sellel real olema kuvatud versioon, mis sai paigaldatud arvutisse. Selleks, et ühendada PyCharm robotiga on vaja lisada väline interpretaator, mille saab valida vajutades



Joonis 24. Eemalasuva interpretaatori konfigureerimise akna vaade.

hammasratta nupule ning rippmenüüst valitakse „*Add Remote*“. Avanevast konfiguratsiooni aknast (vaata joonis 24) tuleks valida „*SSH Credentials*“ ning ära täita ühenduseks vajalikud andmed. Ühenduse võõrustajaks on *ev3dev* ning kasutajanimi ja parool on vastavalt *robot* ja *maker*.

Kindlasti tuleb interpretaatoriks määrata */usr/bin/python3.4*, sest lihtsalt */usr/bin/python* tähistab Python 2.x versiooni (x tähistab alamversiooni numbrit). Kui ankeet on täidetud suletakse aken vajutades „*OK*“ nupule. Nüüd on uue projekti konfiguratsiooni aknasse ilmunud veel kolmas väli– „*Remote project location*“, mis tähistab projekti asukohta SD kaardil. Välja väärtuseks määratakse */home/robot*, ehk kasutaja robot kodukaust. Nüüd on ühendus konfigureeritud ning projekt luuakse „*Create*“ nupuga. Testimiseks luuakse Pythoni fail ning sellele määratakse nimeks Ylesanne00. Joonisel 25 on näha, et faili „Ylesanne00“ peal on vajutatud parema hiire klahviga ning avanenud menüüst valitakse „*Upload to ssh://robot@ev3dev:22/usr/bin/python3.4*“, mis tähendab, et fail laetakse üle SSH ühenduse kasutajaga robot võõrustajale ev3dev pordil 22. Kui programm on juba korra üles laetud robotile, siis peaks PyCharm automaatselt hoidma arvuti projektis ja robotil olevas projektis programmi sünkroonis, ehk automaatselt laetakse uuendused üles robotile. Testimiseks kirjutatakse lühike programm. Kõige lihtsam on kasutada *speak* meetodit, mis ütleb välja etteantud lause.



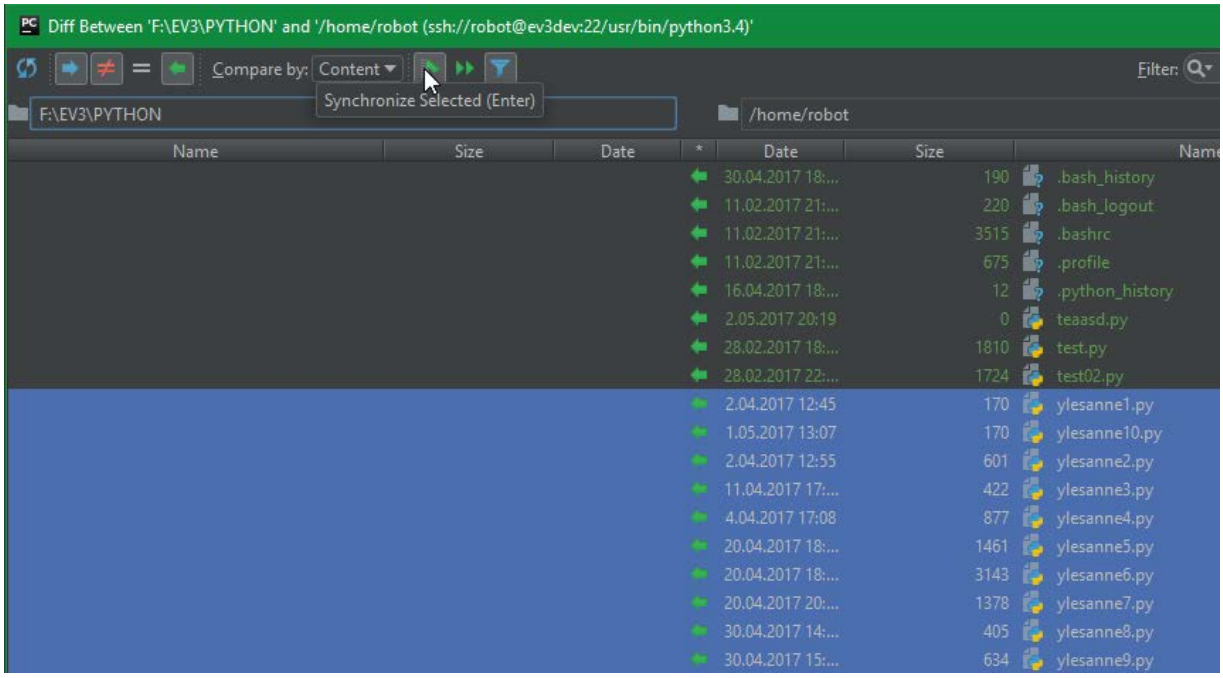
Joonis 25. PyCharmis on loodud uus projekt ning lisatud fail „Ylesanne00.py“

```
1. #!/usr/bin/env python3.4
2. from ev3dev.ev3 import *
3.
4. Sound.speak("Welcome to programming EV3 in Python!").wait()
```

Joonis 26. „Ylesanne00“ koodi näidis.

Näidiskoodis joonisel 26 on näha teegi importimist, millest täpsemalt räägitakse järgmises peatükis. Laseme robotil välja öelda tervituslause „*Welcome to programming EV3 in Python!*“ ehk „Tere tulemast programmeerima EV3-e Pythonis!“. Meetodi *speak* lõpus on ka *wait* meetod, tänu millele ootab programm ära, kuni lause on ette kantud, enne kui programm suletakse. Lause on inglise keelne kuna EV3 nii öelda räägib ainult inglise keelt ning mõnes teises keeles lause ette andes hääldab EV3 sõnu väga imelikult. Programmi saab lihtsalt tööle lülitada vajutades hiire parema klahviga programmil ja menüüst „*Run*“. Peale esmakordset programmi käivitamist peaks „*Run*“ nupp ilmuma ka ülemisel tööribal.

Olukorras, kus programmid on juba robotil olemas, ent arvutis veel pole, saab need ka PyCharmi abil lihtsalt arvuti projekti lisada. Joonisel 26 näha olevast menüüst valitakse „*Deployment*„ ning uuest avanevast menüüst oma korda „*Sync with Deployed to ssh…*“. Avanev aken on kujutatud joonisel 27. Aknast tuleb otsida soovitud programmid ning need siis aktiivseks märkida. Programmid saab arvutis oleva projektiga sünkroniseerida vajutades joonisel 27 kursoriga näidatud nupul. Seejärel kuvatakse programmid ka PyCharmi projektis ning neid saab lihtsasti redigeerida ja käivitada läbi kasutajaliidese



Joonis 27 projekti ja roboti sünkroniseerimiseks avanev failide loend.

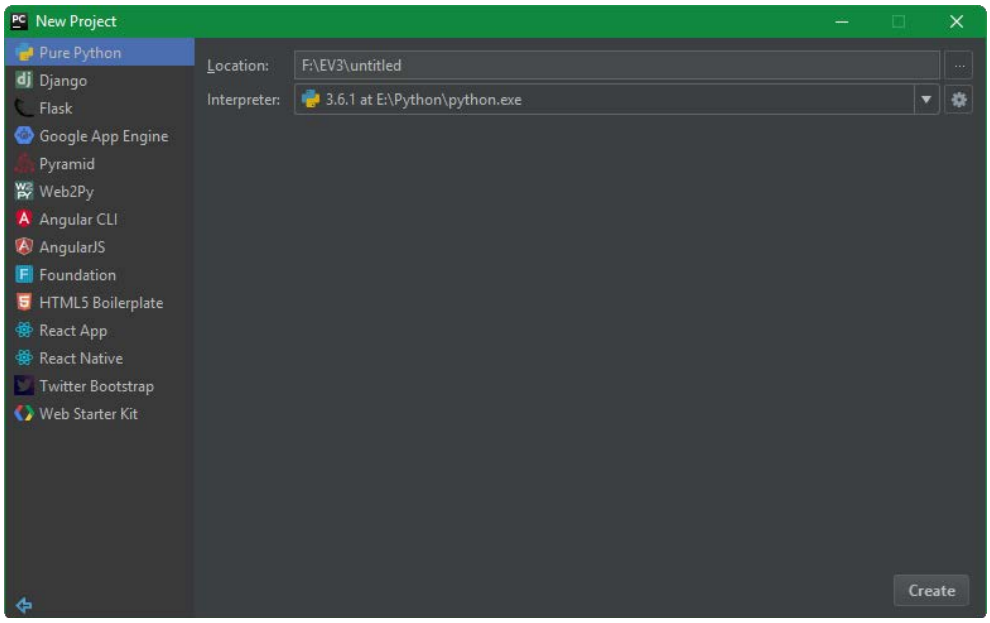
Nüüd on üles seatud ühendus PyCharmi ja LEGO MINDSTORMS EV3 vahel, mis võimaldab lihtsasti liigutada programme arvuti ja roboti vahel ning lubab programmeerimisel ja testimisel vältida käsurida. Selline ühendus imiteerib mõnel määral LEGO MINDSTORMS EV3

programmeerimiskeskkonda, kus kasutaja saab programmeerida ning käivitada robotil jooksva koodi ühest kasutajaliidesest. Järgmises alampeatükis kirjeldame täpsemalt kuidas üles seada ühendus PyCharm *Community* versiooniga nii, et PuTTY käsurealiidese kasutus oleks minimaalne. Kui sobiv ühendus PyCharmi ja roboti jaoks on loodud on programmeerimiseks kõik ettevalmistused tehtud ja ühendused konfigureeritud. Järgmiseks on vaja ehitada sobiv robot ning seda programmeerida, millest räägitakse täpsemalt järgmises peatükis.

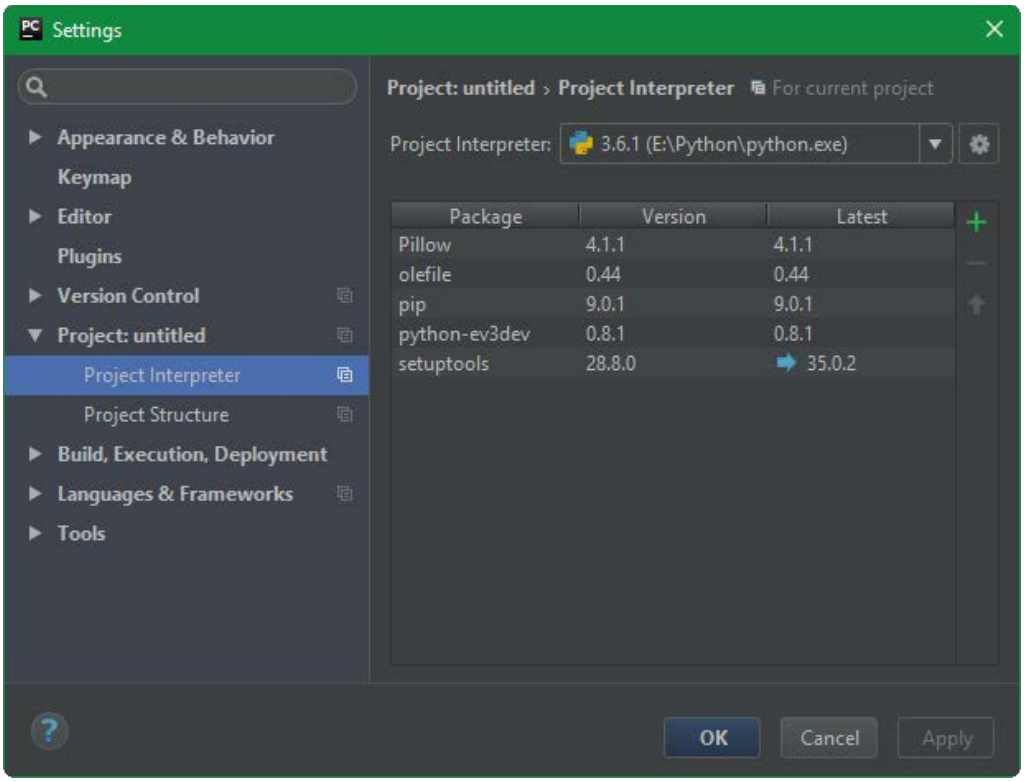
PyCharm Community ja WinSCP esmane käivitamine ja seadistamine

Eelmises alampeatükis seletati täpsemalt kuidas üles seada ühendust PyCharmi ja ev3dev'i vahel, kuid see ei tähenda, et ilma PyCharm *Professional* versioonita on programmeerimine oluliselt raskendatud. Programmeerimist versioon ei muuda. Erinevus tuleneb sellest, et *Community* versioonil, mis on tasuta saadaval, puudub võimalus luua ühendus robotiga, ent suured eelised erinevate programmeerimist lihtsustavate iseärasuste näol on siiski alles. Võimalus on kasutada erinevaid käske PuTTY käsurealiidesel. Süsteemiteadliku kasutaja jaoks ei ole see probleem, ent programmeerimise õpetamisel näiteks robootika ringis oleks see üsna ebamugav. Käesolevas osas seadistatakse ühendus nii, et kasutaja peaks käsurida vähe kasutama ning et programmide jagamine roboti ja PyCharmi vahel oleks lihtne. Kõige pealt luuakse PyCharmis projekt, siis paigaldatakse ja seadistatakse WinSCP ning lõpuks kirjeldatakse kuidas Pythoni programme käivitada PuTTY käsurealt.

Projekti loomiseks valitakse PyCharm tervitus aknast „*Create New Project*“. Selle peale avaneb uue projekti konfigureerimise aken nagu kujutatud joonisel 28. Vasakust menüüst valitakse „*Pure Python*“, mis tähistab, et tegu on Pythoni projektiga. Paremal pool vaates on näha väli „*Location*“, kuhu märgitakse projekti asukoht arvutis. Järgmiseks valitakse „*Interpreter*“- interpretaator, ehk Pythoni versioon, millega hakkab projekt töötama. Projekt luuakse vajutades „*Create*“ nupuga.



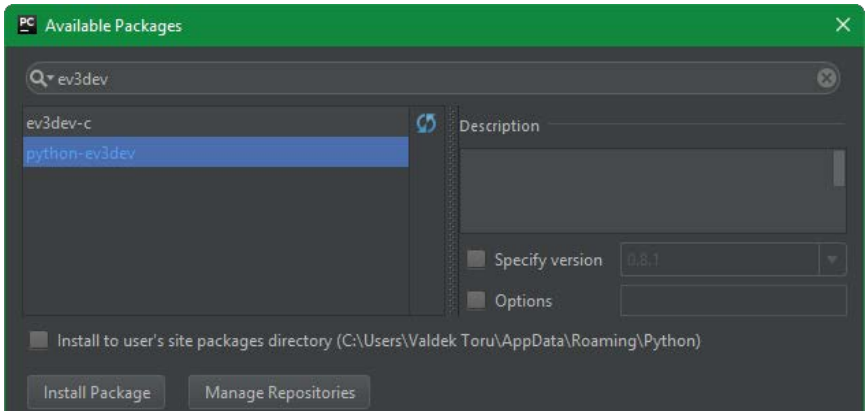
Joonis 28. Uue projekti seadistamise akna vaade.



Joonis 29. Loodud projekti interpretaatori seadistamine.

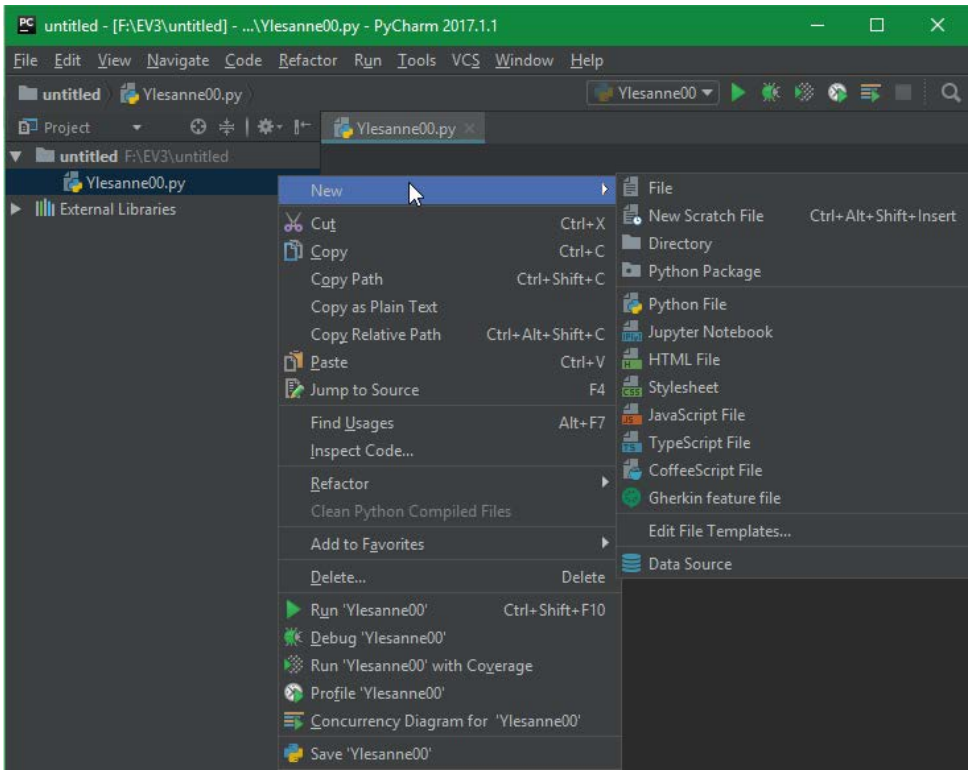
Peale uue projekti loomist tuleb esmalt paigaldada python-ev3dev teek, mis on vajalik, et arvuti tunneks ära loodavates programmides robotiga seotud meetodid. Teegi paigaldamiseks on vaja

avada projekti interpretaatori seaded, mille saab avada valides ülemiselt tööribalt „*File*“ avanenud menüüst „*Settings*“ , avanenud aknast „*Project*: <projekti_nimi>“ ning viimaks „*Project interpreter*“. Projekti interpretaatoriks on määratud arvutisse paigaldatud Python. Interpretaatori



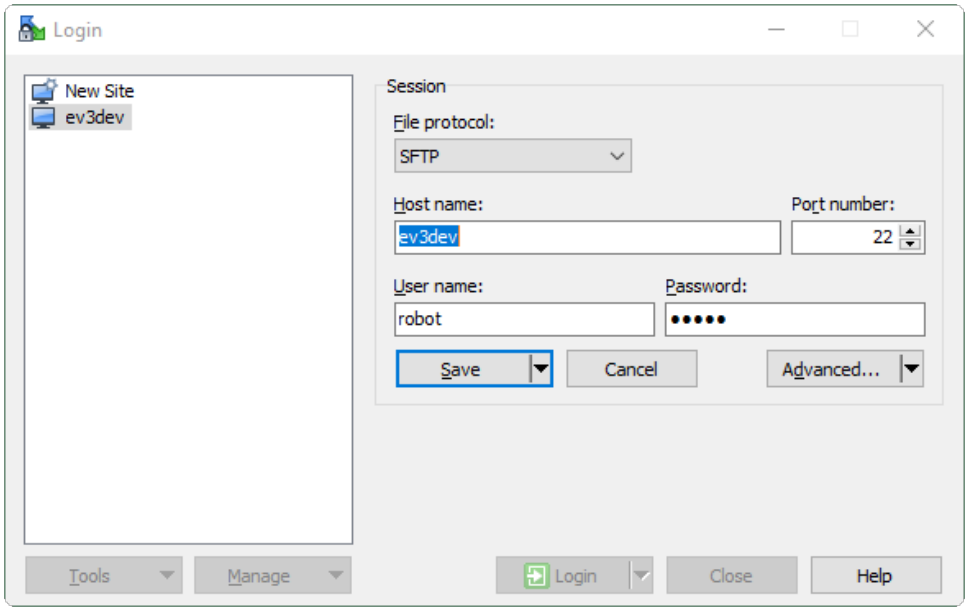
Joonis 30. Uue teegi otsimine ja lisamine.

pakettide nimekirjast paremal on roheline pluss märk, millele vajutades avaneb pakettide otsingu aken nagu joonisel 29. Luubi märgiga tähistatud otsingu ribale kirjutatakse ev3dev ning kuvatavate



Joonis 31. Faili „Ylesanne00“ lisamine ja rippmenüü vaade.

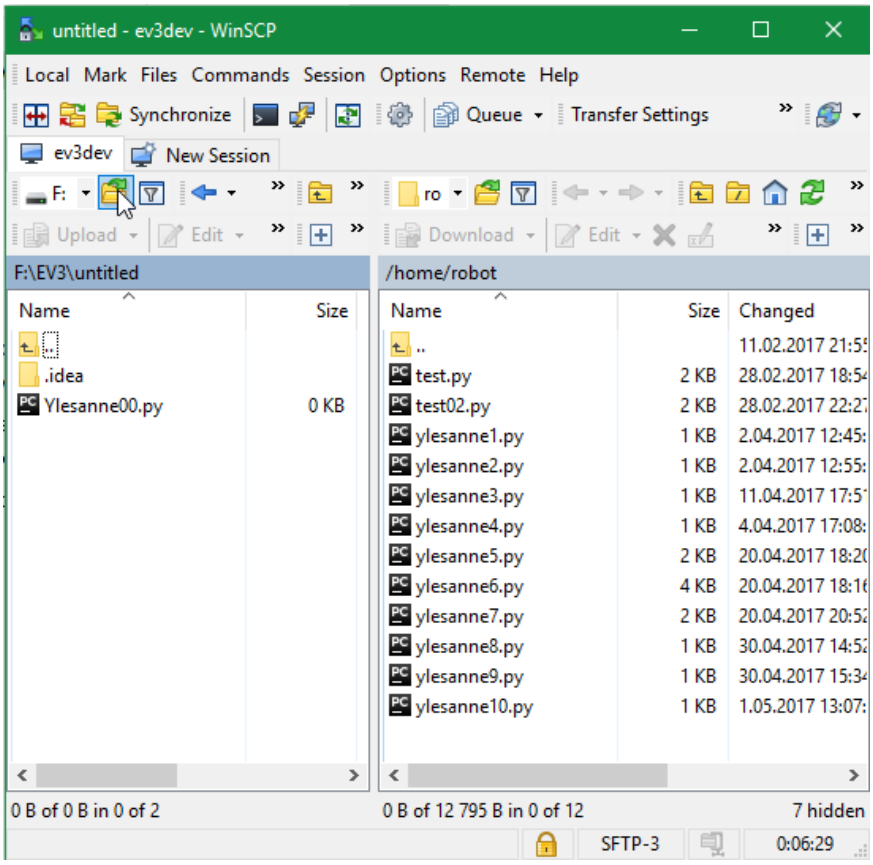
vastuste hulgast märgitakse aktiivseks „*python-ev3dev*“ (vaata joonis 30) ning käivitatakse paigaldus „*Install Package*“ nupuga. Seejärel võib akna sulgeda. Samuti võib sulgeda interpretaatori seadete akna vajutades „*OK*“ nupule. Vajutades projekti kaustal hiire parema klahviga avaneb rippmenüü sarnaselt nagu on näha joonisel 31. Luuakse uus Python fail valides



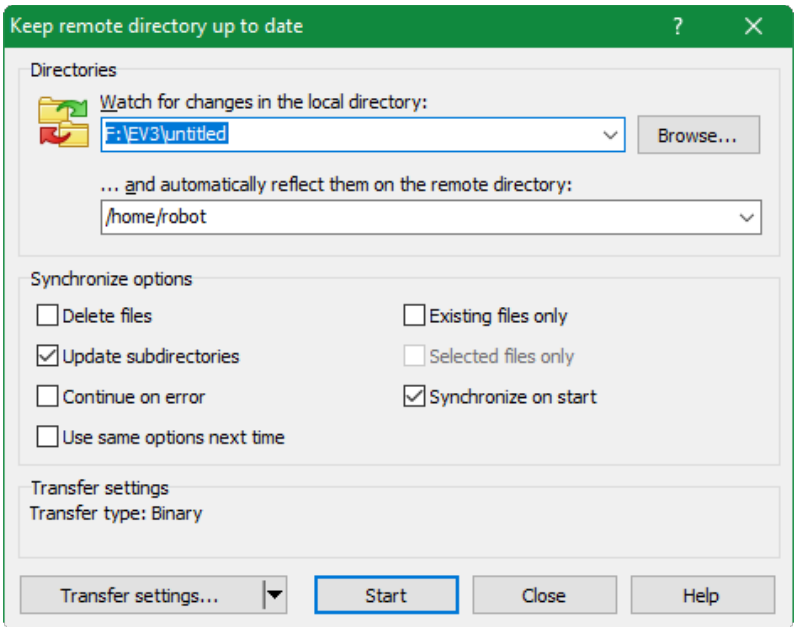
Joonis 32. WinSCP ühenduse seadistamise akna vaade.

menüüst „*New*“ ning seejärel „*Python File*“. Avanenud aknas määratakse programmile nimi

näiteks „Ylesanne00“. Enne programmeerima asumist paigaldatakse WinSCP ja seadistatakse ühendus robotiga. Selleks tuleb esmalt alla laadida WinSCP installfail, mille leiab aadressilt: https://winscp.net/eng/download.php, allalaadimine käivitatakse lingi „*Installation package*“ kaudu. Paigaldamisel on 2 võimalust: „*Typical installation*“ tähendab, et paigalduseks kasutatakse vaikeväärtusi ning „*Custom installation*“ laseb kasutajal valida kuhu programm paigaldatakse ning millised osad paigaldatakse. Peale paigaldust käivitatakse WinSCP ning avaneb ühenduse seadistamise aken nagu joonisel 32. Sisestatakse ühenduseks vajalikud andmed: võõrustajaks on *ev3dev*, kasutajanimi on *robot* ning parool *maker*. Ühenduse saab salvestada „*Save*“ nupuga ning seejärel luuakse ühendus valides „*Login*“. Peale ühenduse loomist avaneb aken nagu joonisel 33, kus vasakul pool on kuvatud arvutis asuv projekti kaust ning vasakul kasutaja robot kodukaust,



Joonis 33. Ühenduse avakuva, kuvatud on projektisisu vasakul ning töös kasutatud roboti programme paremal.



Joonis 34. „Hoia eemalasuv kataloog ajakohane“ akna vaade.

ehk projekti asukoht robotis. Arvutist kuvatavat kausta saab muuta vajutades nupul, mis on pildil märgitud kursoriga. Kui nii arvutis ja robotis on sobiv kaust lahti võib soovituid faile ka lihtsalt lohistada ühe masina kaustast teise masina kausta. Et peale iga uuendust poleks vaja programme käsitsi ümber lohistada lülitatakse sisse automaatne andmete sünkroniseerimine. Selleks avatakse „*Keep remote directory up to date*“ ehk „Hoia eemalasuv kataloog ajakohane“ aken nupust, mis asub joonisel 33 „*Synchronize*“ nupust vasakul. „Hoia eemalasuv kataloog ajakohane“ aken on nähtav joonisel 34. Ülemisel tekstiväljal on kirjeldatud projekti asukoht arvutis, alumisel aga projekti asukoht robotil. „*Start*“ nupuga pakutakse esmalt andmete ühildamine – ehk projektis olevad programmid kopeeritakse robotile ning seejärel jääb WinSCP tahaplaanile jälgima muudatusi projekti kaustas ning kopeerib need üle robotile. Selleks, et programmi käivitada tuleks robotisse ühenduda PuTTY abiga või lülitada programm tööle robotist. Nüüd kirjutakse projekti näidis programm. Täiendatakse juba eelnevalt loodud „Ylesanne00.py“ Python faili. Testimiseks kirjutatakse lühike programm. Kõige lihtsam on kasutada „*speak*“ meetodit, mis ütleb välja etteantud lause.

```
1. #!/usr/bin/env python3.4
2. from ev3dev.ev3 import *
3.
4. Sound.speak("Welcome to programming EV3 in Python!").wait()
```

Joonis 35. „Ylesanne00“ koodi näidis.

Näidiskoodis joonisel 35 on näha teegi importimist, millest täpsemalt räägitakse järgmises peatükis. Laseme robotil välja öelda tervituslause „*Welcome to programming EV3 in Python!*“ ehk „Tere tulemast programmeerima EV3-e Pythonis!“. Meetodi *speak* lõpus on ka *wait* meetod, tänu millele ootab programm ära, kuni lause on ette kantud, enne kui programm suletakse. Lause on inglise keelne kuna EV3 nii öelda räägib ainult inglise keelt ning mõnes teises keeles lause ette

andes hääldab EV3 sõnu väga imelikult. Pärast muutmist laeb WinSCP programmi automaatselt robotisse. Programmi saab käivitada PuTTY'is Python3 abil Selleks tuleb loodud fail Python3 programmile argumendina ette anda kujul *python3* <Programmi_nimi>.py ehk näidis käivitatakse nii: *python3 Ylesanne00.py*. Teine võimalus on programm käivitada kasutades programmeeritava aju ekraani ja nuppe. Peamenüüst tuleb valida „*File Browser*" ning avanenud loetelust saab programmi tööle lülitada keskmisest nupust.

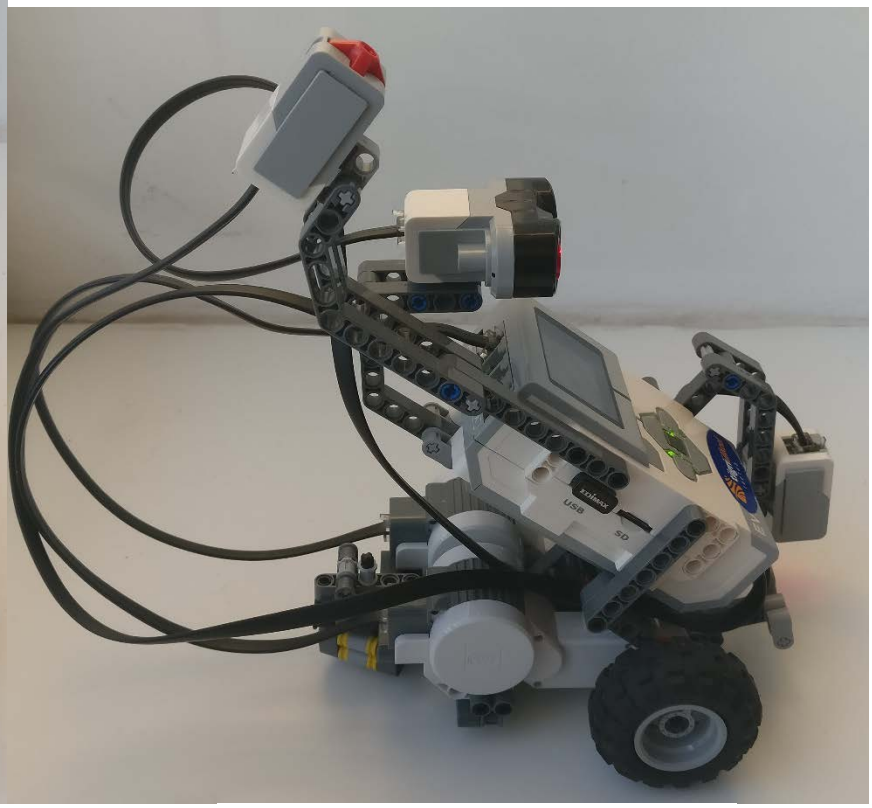
Lõpuks on paigaldatud robotile ev3dev operatsioonisüsteem ning arvutisse on paigaldatud sobiv versioon JetBrains'i PyCharmist. Kõik ettevalmistused on programmeerimiseks tehtud. Nüüd jääb üle ainult robot ehitada ning selle jaoks sobivad programmid programmeerida.

4. EV MINDSTORMS EV3-e programmeerimine Pythonis

Käes olevas peatükis räägitakse täpsemalt LEGO MINDSTORMS EV3-e programmeerimisest Pythonis. Nagu juba eelnevalt on mainitud, siis töös kasutatakse teeki *python-ev3dev* versiooni numbriga 0.8.1 EV3 programmeerimiseks Pythonis. Pythoni kodulehele on teek lisatud 6. veebruar 2017. Teegil on Githubi projekti näol ka koduleht[9]. Teegi dokumentatsioon on leitav ka Githubist[10] ning PDF dokumendina[11]. Antud peatükis tutvustatakse, kuidas kasutada mootoreid, andureid, ekraani, heli funktsiooni, nuppe ning LED tulesid. Töös tutvustatakse mitmeid erinevaid meetodeid ja funktsioone, kuid töö ei käsitle kõiki funktsioone, mis on kirjeldatud dokumentatsioonis.

Töös on kasutatud 3 rattaga baasrobotit, millel on 2 mootorit (vaata jooniseid 36 ja 37). Robot on

Joonis 37. Demorobot eest vaates.



Joonis 36. Demorobot külgvaates.

[9] https://github.com/rhempel/ev3dev-lang-python
[10] http://python-ev3dev.readthedocs.io/en/latest/spec.html
[11] https://media.readthedocs.org/pdf/python-ev3dev/latest/python-ev3dev.pdf

ehitatud NXT komplekti detailidest ning kasutab NXT mootoreid., Programmeeritava ajuna on kasutatu EV3-e oma ning anduritena on roboti külge monteeritud kaugusandur, värviandur ja puuteandur. Andurite ja mootorite puhul ei ole vahet, kas kasutada NXT või EV3-e komplekti kuuluvaid osi, mõlema kompleti andurid töötavad samamoodi.

4.1 Ülesanne 1

Ülesandeks on panna robot sirgjooneliselt liikuma. Robot liigub sirgjooneliselt edasi rataste kolm täispööret.

Lahenduse näidiskood on esitatud joonisel 38. Enne, kui saab panna roboti liikuma, tuleks defineerida mootorid, mida kasutatakse ning kuna EV3 programmeerimiseks on vaja enamat kui Pythoni standardmeetodeid , siis tuleb esmalt importida *ev3dev.ev3* teek Teegi importimine on tähtis selleks, et Python oskaks käsitleda väljakutsustud meetodeid ja funktsioone. Peale importimist defineeritakse objektidena mootorid, mis on ühendatud A ja D väliportidesse. Muutujatena on mootoreid lihtsam hiljem välja kutsuda. Robot pannakse liikuma kutsudes muutujate peal välja funktsioon *run_to_rel_pos*, mis paneb mootori pöörama argumendina etteantud kraadide võrra. Teine etteantud argument on mootori pöörlemise kiirus kraadi/sekundis ühikutes. Viimane argument on *stop_action* ehk peatumiskäsk, mis antakse mootorile kui see on lõpetanud liikumise. Erinevaid peatumiskäske on kolm: *coast, hold* ja *brake*. Käsuga *coast* eemaldatakse mootorile rakendatud jõud ning mootor töötab nii öelda vabal käigul peatumiseni. Käsk *brake* tähendab, et rakendatud jõud eemaldatakse ning samas ka pidurdatakse maha mootori liikumine lühistades mähised. Käsuga *hold* ei võeta mootorilt jõudu ära, vaid mootor üritab hoida hetke positsiooni ning vajadusel hakkab vastu töötama välistele mõjutustele.

```
1.  # Imporditakse kõik osad ev3dev.ev3 teegist.
2.  from ev3dev.ev3 import *
3.
4.
5.  # Defineeritakse muutujad mA ja mD, mis tähistavad mootoried, mis on ühendatud pesade
    sse A ja D.
6.  mA = LargeMotor('outA')
7.  mD = LargeMotor('outD')
8.
9.  # Mootori muutujate mA ja mD peal kutsutakse välja liikumise funktsioon run_to_rel_po
    s.
10. # Meetodile run_to_rel_pos antakse argumetideks pöörded, kiirus ja stop_action.
11. # Antud juhul teeb pöörab mootor 1080 kraadi ehk kolm täis pööret.
12. # Kiiruseks on määratud 180 kraadi sekundis.
```

```
13. # Argiment stop_action on juhis, kuidas mootor käitub kui liikumine on lõppenud.
14. # Antud juhul on lõpp käsuks "brake" ehk pidurda, mis tähendab, et robot pidurdab moo
    torid kinni kui liikumine on lõppenud.
15. mA.run_to_rel_pos(position_sp=1080, speed_sp=180, stop_action="brake")
16. mD.run_to_rel_pos(position_sp=1080, speed_sp=180, stop_action="brake")
```

Joonis 38. Ülesande 1 näidiskood.

4.2 Ülesanne 2

Ülesandeks on panna robot ruudukujulist trajektoori sõitma.

a) Programmeeri funktsioon, mis paneb roboti sirgjooneliselt sõitma kolmeks sekundiks.

b) Programmeeri funktsioon, mis paneb roboti pöörama 90 kraadi vasakule.

c) Kombineeri funktsioonid *while*-tsükli abil nii, et robot liiguks ruudukujulist trajektoori mööda.

Lahenduse näidiskood kasutades *while*-tsüklit on esitatud joonisel 39 ning kasutades rekursiooni joonisel 40. Ülesande alguse imporditakse *ev3dev.ev3* teek ning defineeritakse mootorid objektidena. Esimeseks alamülesandeks on programmeerida funktsioon, mis paneb roboti otse liikuma kolmeks sekundiks. Selleks kasutatakse mõlema mootori peal funktsiooni *run_timed*, mis võtab argumentideks aja, kiiruse ja peatumiskäsu. Argumendina antud aeg on millisekundites, kiirus ikka kraadi/sekundis ning peatumiskäsk *brake*. Teise alamülesandena programmeeritakse funktsioon, mis paneb roboti kohapeal keerama vasakule umbes 90 kraadi. Roboti keeramiseks tuleb tööle panna ainult 1 mootor. Roboti vasakule keeramiseks tuleb tööle lülitada parempoolne mootor ning et pööramine toimuks kohapeal, tuleks vasak mootor täielikult peatada. Keeramiseks kasutatakse juba eelnevalt mainitud *run_to_rel_pos* funktsiooni ning vasaku mootori peatamiseks kasutatakse argumentideta meetodit *stop*. Kui palju tuleb mootorit pöörata oleneb robotist ning mootorist. Seda on võimalik umbkaudselt välja arvutada, selleks on vaja ära mõõta roboti rataste kaugus ning teada saada rehvi ümbermõõt. Argumendi *position* väärtust arvutatakse valemiga: $position = \frac{2r}{d} * 360$, kus *d* tähistab roboti rataste kaugust ja *r* tähistab rehvi ümbermõõtu. Töös kasutatud roboti rataste kaugus oli 11,5 sentimeetrit ja rehvi ümbermõõt 5,6 sentimeetrit ning valemit kasutades tuleb *position* väärtuseks 350,6. Viimaseks ülesande osaks on kasutades kahte eelnevalt programmeeritud funktsiooni panna robot liikuma ruudukujulist trajektoori mööda. Meetodis *ruut* kutsutakse esmalt välja funktsioon *otse* ja oodatakse kuni mootorid on oma töö lõpetanud ning seejärel kutsutakse välja funktsioon *vasakule* ja oodatakse kuni mootor lõpetab töö. Sellist liikumist on vaja korrata neli korda ning selle jaoks lisatakse viimaseks loenduri *i*, mida kontrollitakse *while*-tsükli kontroll osas. Kui loendur on väiksem kui neli, siis korratakse liikumis mustrit ning kui loenduri väärtuseks saab 4, siis töö lõpetatakse. Programmi lõpus kutsutakse meetod *ruut*, mis enda sees kutsub välja teised eelnevalt programmeeritud meetodid ning robot hakkab liikuma trajektoori mööda.

```
1.  # Imporditakse kõik osad ev3dev.ev3 teegist.
2.  from ev3dev.ev3 import *
3.
4.  # Defineeritakse muutujad mA ja mD, mis tähistavad mootoried, mis on ühendatud pesade
    sse A ja D.
5.  mA = LargeMotor('outA')
6.  mD = LargeMotor('outD')
7.
8.  # Ülesandes defineeritakse funktsioonid otse, vasakule ja ruut.
9.  # Nende defineerimine muudab koodi loetavamaks.
10. # Funktsioon otse paneb roboti sirgjooneliselt edasi liikuma kolmeks 3000 millisekund
    iks.
11. def otse():
12.     mA.run_timed(time_sp=3000, speed_sp=360, stop_action="brake")
13.     mD.run_timed(time_sp=3000, speed_sp=360, stop_action="brake")
14.
15. # Funktsioon vasakule paneb roboti kohapeal keerama vasakule umbes 90 kraadi.
16. def vasakule():
17.     mA.run_to_rel_pos(position_sp=351, speed_sp=360, stop_action="brake")
18.     mD.stop()
19.
20. # Funktsioon ruut paneb roboti sõitma ruudukujulist trajektori mööda.
21. # Selleks kasutatakse vaheldumisi funktsiooni otse ja vasakule.
22. # While-tsükli abil korratakse otse liikumist ja vasakule pööramist 4 korda.
23. def ruut():
24.     i = 0
25.     while i != 4:
26.         otse()
27.         mA.wait_while('running')
28.         mD.wait_while('running')
29.         vasakule()
30.         mA.wait_while('running')
31.         i += 1
32.
33. ruut()
```

Joonis 39. Ülesande 2 näidiskood, kasutades *while*-tsüklit.

Ülesande saab lahendada väga edukalt ka rekursiooni kasutades. Miski muu koodi juures ei muutu kui ainult meetod *ruut*. Meetodil on argument *count*, mis loendab kui sügavale on mindud. Kui

count ei ole veel neli, siis kutsutakse uuesti vali meetodid *otse* ja *vasakule* ning ka meetod *ruut* ning välja kutsel suurendatakse argumendi *count* väärtus ühe võrra.

```
20. # Funktsioon ruut paneb roboti sõitma ruudukujulist trajektori mööda.
21. # Selleks kasutatakse vaheldumisi funktsiooni otse ja vasakule.
22. # Rekursiooni abil korratakse otse liikumist ja vasakule pööramist 4 korda.
23. def ruut(count):
24.     if count == 4:
25.         mA.stop()
26.         mD.stop()
27.         exit()
28.     else:
29.         otse()
30.         mA.wait_while('running')
31.         mD.wait_while('running')
32.         vasakule()
33.         mA.wait_while('running')
34.         ruut(count+1)
35. ruut(0)
```

Joonis 40. Ülesande 2 näidiskoodi osa kasutades rekursiooni.

4.3 Ülesanne 3

Ülesandeks on roboti ekraanile kuvada roboti ees seisva takistuse kaugus kasutades kaugusandurit.

Lahenduse näidiskood on esitatud joonisel 41. Ülesandes kasutatakse kauguse mõõtmiseks kaugusandurit. Lisaks *ev3dev.ev3*-le on vaja importida ka *sleep* funktsioon teegi *time* ehk aeg. Tänu sellele saab panna programmi jooksmise määratud ajaks pausile ning mõõtmistulemused kuvatakse ekraanile mõistliku aja tagant. Defineeritakse muutujana kaugusandur ning *assert* meetodi abil kontrollitakse, et see oleks korralikult ühendatud. Kui ühendus roboti ja anduri vahel ei ole korras, tagastatakse *Assert Error* veateade ning veale on lihtsam jälile saada. Kaugusandur lülitatakse sentimeetrite režiimile. Siinkohal tuleb mainida, et töö kirjutamise ajal on kasutatud 0.8.1 versiooni *python-ev3dev*-ist ning sentimeetrite režiim tagastab tulemusi millimeetrites. Roboti ekraan defineeritakse objektiks *lcd* ning roboti nupud defineeritakse objektiks *btn*. Ülesandes on tsükkel vajalik selleks, et robot uuendaks ekraanil andmeid. Tsükli algusesse lisame kontrolli, mis jälgib roboti nuppude vajutusi ning kui suvalist nuppu vajutatakse, siis programm sulgetakse. Funktsioon *draw.text* võtab argumentideks koordinaatide paari ja sõne ning kirjutab selle ekraanile. Ekraanile kirjutatakse tekst „Kaugus on <kaugus sentimeetrites> sentimeetrit.“. Siinkohal avastati defekt, mis väljendub selles, et kuigi mõõterežiimiks on määratud sentimeetrid, siis robot tagastab väärtuse siiski millimeetrites. Sellest saab lihtsalt mööda kui jagada mõõtmistulemus 10-ga ning kuna anduri mõõtmisviga on 3 sentimeetrit, siis tuleks saadud tulemus

ümardada esimese täisarvuni. Jagamine ja ümardamine jätab tulemuse ujukoma arvuks, millelt saab komakoha eemaldada muutes andmetüübi täisarvuks ehk *integer*-iks. Viimase andmetöötlusena on vaja saadud täisarv muuta sõneks ehk *string*-iks. Funktsiooni *draw.text* koordinaadid algavad ekraani ülemisest vasakust nurgast. Peale mõõtmist teeb programm ühe sekundilise pausi, mis on vajalik selleks, et robot ei teeks uusi mõõtmisi liiga kiiresti. Peale pausi uuendatakse ekraan ning alles nüüd ilmuvad ekraanile mõõtmistulemused. Funktsioon *clear* puhastab ekraani, kuid seda pole näha enne kui on ekraanile tehtud uus uuendus. Puhastamine on vajalik selleks, et muidu kirjutataks eelnev tekst koguaeg uuesti üle, mis aga teeb muutava mõõtmistulemuse loetamatuks.

```
1.  # Imporditakse kõik osad ev3dev.ev3 teegist.
2.  # Ajaga seotud teegist time imporditakse meetod sleep.
3.  from ev3dev.ev3 import *
4.  from time import sleep
5.
6.  # Defineeritakse kaugusandur muutujaks us, et hiljem oleks lihtsam välja kutsuda.
7.  # Seejärel kontrollitakse, et kaugusandur oleks korralikult ühendatud ning vastasel k
    orral tagastatakse error.
8.  # Kaugusandur lülitatakse sentimeetrirežiimile.
9.  us = UltrasonicSensor()
10. assert us.connected
11. us.mode = 'US-DIST-CM'
12.
13. # Roboti ekraan defineeritakse muutujaks lcd, et seda oleks hiljem lihtsam välja kuts
    uda.
14. # Roboti nupud defineeritakse muutujaks btn, et seda oleks hiljem lihtsam välja kutsu
    da.
15. lcd = Screen()
16. btn = Button()
17.
18. # Tsükkel on vajalik selleks, et robot andmeid uuendaks.
19. # Tsükli algusesse on lisatud kontroll, mis kuulab roboti nuppude vajutusi ning sulge
    b programmi kui nuppu on vajutatud.
20. # Funktsioon draw võtab argumentideks koordinaadid ning kirjutatava teksti.
21. # Koordinaad X null koht on vasak poolne ekraani äär ning Y nullkoht on ülemine ekraa
    ni äär.
22. # Esimesena joonistatakse valmis kiri "Kaugus on".
23. # Teisena mõõdetakse kaugus, nimest hoolimata, millimeetrites.
24. # Sentimeetrite jaoks jagatakse tulemus 10-
    ga ning ümardatakse nullinda koma kohani ehk esimese täisarvuni.
```

```
25. # Selleks, et vabaneda arvutamise tulemusena tekkinud ühest komakohast muudetakse and
    metüüp täisarvuks ehk integeriks.
26. # Selleks, et arvu ühendada teksti andmetüübiga tuleb muuta arv tekstiks.
27. # Viimasena lisatakse lausesse " sentimeetrit."
28. # Peale rea kirjutamist ootab programm ühe sekundi ennem kui edasi liigub, muidu muut
    uks andmed ekraanil liiga kiiresti.
29. # Seejärel uuendatakse ekraani ning tekst ilmub ekraanile.
30. # Funktsioon clear() puhastab ekraani uuesti ära, kuid seda ei ole näha enne kui teha
    kse uus update väljakutse.
31. # Ilma clear'i kasutamata kirjutatakse tekst kogu aeg üle ning mõõtmistulemuste aseme
    l oleks ekraanil näha musti kaste.
32. while True:
33.     if btn.any():
34.         exit()
35.     lcd.draw.text((0, 0), "Kaugus on " + str(int(round(us.distance_centimeters / 10,
    0))) + " sentimeetrit.")
36.     sleep(1)
37.     lcd.update()
38.     lcd.clear()
```

Joonis 41. Ülesande 3 näidiskood.

4.4 Ülesanne 4

Ülesandeks on panna robot jälgima lähenevat objekti ning kui objekt jõuab liiga lähedale, siis robot pöörab ennast kiiresti ringi. Lisaks tuleb programm sulgeda puuteanduri vajutuse peale.

Lahenduse näidiskood on esitatud joonisel 42. Lahenduses kasutatakse ainult *ev3dev.ev3* teeki, kahte mootorit ning kaugus- ja puuteandurit. Lahenduses ei ole vaja defineerida mitut erinevat funktsiooni, kuid harjutamiseks võib seda teha. Võrdlemisi lihtsalt saab roboti liikumise osa tõsta eraldi funktsiooni sisse. Programmi kogu põhiline osa on ühe *while*-tsükli sees. See on vajalik, et robot teeks pidavalt kontrollmõõtmisi, et mõni objekt ei satuks liiga lähedale. Kaugusanduri abil kontrollitakse ega ükski objekt poleks lähemal kui 25 sentimeetrit ning vastasel korral pöörab robot kiiresti objektile selja ja ütleb: „*Please do not come too close!*“ ehk „Palun ära tule liiga lähedale“. Samuti on valves ka puuteandur ning sellele vajutades sulgetakse programm.

```
1. # Imporditakse kõik osad ev3dev.ev3 teegist.
2. from ev3dev.ev3 import *
```

```
3.
4.  # Defineeritakse muutujad mA ja mD, mis tähistavad mootoried, mis on ühendatud pesade
    sse A ja D.
5.  mA = LargeMotor('outA')
6.  mD = LargeMotor('outD')
7.
8.  # Defineeritakse kaugusandur muutujaks us ning kontrollitakse, et kaugusandur oleks k
    orralikult ühendatud.
9.  # Kaugusandur lülitatakse sentimeetrirežiimile.
10. us = UltrasonicSensor()
11. assert us.connected
12. us.mode = 'US-DIST-CM'
13.
14. # Defineeritakse puuteandur muutujaks ts ning kontrollitakse, et puuteandur oleks kor
    ralikult ühendatud.
15. ts = TouchSensor()
16. assert ts.connected
17.
18. # Kaugusandur on valves ning kui mõni objekt satub lähemale kui 25 cm, siis robot pöö
    rab ennast kiiresti ümber.
19. # Pööramiseks kasutatakse mõlemat mootorit. Mootor A pöörab ratast edasi suunas ning
    mootor D tagasi suunas.
20. # Kui pööramine on lõõpenud annab robot teada:"Please do not come too close!" ehk "Pa
    lun ära tule liiga lähedale.
21. # Puuteanduri alla vajutamisel suletakse programm exit käsuga.
22. while True:
23.     if ts.is_pressed:
24.         exit()
25.     elif us.distance_centimeters < 250:
26.         mA.run_to_rel_pos(position_sp=400, speed_sp=500, stop_action="brake")
27.         mD.run_to_rel_pos(position_sp=-400, speed_sp=500, stop_action="brake")
28.         mA.wait_while('running')
29.         mD.wait_while('running')
30.         Sound.speak("Please do not come too close!")
```

Joonis 42. Ülesande 4 näidiskood.

4.5 Ülesanne 5

Ülesandeks on panna robot objekti jälitama, ent mitte liiga lähedale objektile liikuma. Programm käivitatakse puuteanduri vajutusega.

Lahenduse näidiskood on esitatud joonisel 43. Lahenduses kasutatakse ainult *ev3dev.ev3* teeki, kahte mootorit, kaugusandurit ning programmeeritava aju küljes olevaid nuppe. Esmalt defineeritakse mootorid, andur ja nupud objektidele, et neid oleks hiljem lihtsam välja kutsuda. Roboti liikumine on kirjeldatud meetodis *liigu*, mis võtab argumendiks arvu. Lahenduses kasutatakse arvudena 1 ja -1. See arv korrutatakse läbi roboti liikumise kiirusega ning kui roboti kiirus on positiivne, siis ta liigub edasi ja kui kiirus on negatiivne, siis robot liigub tagasi. Programmis põhiline töö toimub siiski tsüklis. Tsükli sees kontrollitakse pidavalt lähima objekti kaugust robotist. Kui objekt on robotist kaugemal kui 60cm siis robot liigub sellele lähemale, kui aga lähemal kui 40cm siis robot tagurdab objektist kaugemale. Kui objekt jääb nende piiride vahele, siis robot ei liigu. Roboti liikumiseks on kasutatud *run_forever* funktsiooni, mis ei sea piiranguid roboti liikumise lõpule ning programm läbib tsüklit edasi ning kui tuvastatakse tingimuse muutus, siis robot reageerib vastavalt. Roboti saab sulgeda EV3 nuppudele vajutades. Siinkohal tuleb meeles pidada, et kuna liikumiseks kasutatakse *run_forever* funktsioonis, siis tuleb enne programmi sulgemist mootorid peatada, vastasel korral programmi töö lõpeb, ent mootorid töötavad edasi kuna pole saanud käsku, mis muudaks nende tööd.

```
1.  # Imporditakse kõik osad ev3dev.ev3 teegist.
2.  from ev3dev.ev3 import *
3.
4.  # Defineeritakse muutujad mA ja mD, mis tähistavad mootoried, mis on ühendatud pesade
    sse A ja D.
5.  mA = LargeMotor('outA')
6.  mD = LargeMotor('outD')
7.
8.  # Defineeritakse kaugusandur muutujaks us ning kontrollitakse, et kaugusandur oleks k
    orralikult ühendatud.
9.  # Kaugusandur lülitatakse sentimeetrirežiimile.
10. us = UltrasonicSensor()
11. assert us.connected, "Connect a US sensor to any sensor port"
12. us.mode = 'US-DIST-CM'
13.
14. # Roboti nupud defineeritakse muutujaks btn, et seda oleks hiljem lihtsam välja kutsu
    da.
15. btn = Button()
16.
```

```
17. # Defineeritakse funktsioon liigu, mis paneb roboti sõitma nii edasi kui tagurpidi.
18. # Funktsioon võtab argumendiks indikaatori, mis on kas positiivne või negatiivne arv.

19. # Kiirus korrutatakse läbi indikaatori väärtusega.
20. # Kui tulemus on positiivne siis liigub robot edasi ning kui vastus on negatiivne sii
    s liigub robot tagurpidi.
21. # Roboti liikumiseks kasutatakse run_forever meetodit kuna see ei määra robotile liik
    uma hakates liikumise lõppu.
22. # Tänu sellele läbitakse programmi edasi ning tehakse jooksvalt ümber arvutusi ja rea
    geeritakse vastavalt.
23. def liigu(indikaator):
24.     mA.run_forever(speed_sp=(indikaator*500))
25.     mD.run_forever(speed_sp=(indikaator*500))
26.
27. # Kaugusandur on valves. Kui mõni objekt satub lähemale kui 40 cm, siis kutsutakse vä
    lja meetod liigu negatiivse argumendiga.
28. # Kui ükski objekt pole lähemal kui 60 cm siis kutsutakse välja meetod liigu positiiv
    se argumendiga.
29. # Kui lähim objekt on kaugemal kui 40 cm ning lähemal kui 60 cm siis robot seisab pai
    gal.
30. # Roboti programmi saab sulgeda kasutades programmeeritava aju küljes olevaid nuppe.

31. while True:
32.     if us.distance_centimeters > 600:
33.         liigu(1)
34.     elif us.distance_centimeters < 400:
35.         liigu(-1)
36.     elif btn.any():
37.         mA.stop()
38.         mD.stop()
39.         exit()
40.     else:
41.         mA.stop()
42.         mD.stop()
```

Joonis 43. Ülesande 5 näidiskood.

4.6 Ülesanne 6

Ülesandeks on panna robot vältima teel olevaid valgeid takistusi.

a) Programmeeri robot alles startima, siis kui on antud märguanne puuteandurilt.

b) Programmeeri robot tuvastama tema ees olevat valgeid takistusi värvianduriga.

c) Programmeeri robot sõitma sirgjooneliselt ning teel olevate takistuste korral sõidu jätkamiseks kõrvale pöörama.

Lahenduse näidiskood on esitatud joonisel 44. Lahenduses kasutatakse ainult *ev3dev.ev3* teeki, kahte mootorit, programmeeritava aju nuppe ning puute- ja värviandurit. Programmi alguses defineeritakse mootorid, andurind ning nupud objektideks. Nagu ka eelmises ülesandes, kasutatakse edasi liikumiseks funktsiooni *run_forever*, tänu millele saab robot teha muudatusi oma liikumises. Robot jälgib värvianduriga ees olevat teed ning takistuse korral kutsutakse välja funktsioon takistus. Meetodi *takistus* sees pidurdatakse roboti liikumine ning seejärel tagurdatakse rataste ühe täisringi võrra. Tagurdamine on vajalik, et robot ei põrkaks takistusega kokku keeramisel. Kui tagurdamine on lõppenud, teeb robot umbes 90 kraadise vasakpöörde. Kui pööre on lõppenud jätkab programm tsükli läbimist. Programmi saab sulgeda programmeeritava aju küljes olevate nuppudega. Kuna liikumiseks on kasutatud lõpmatut liikumise funktsiooni tuleb mootorid enne programmi sulgemist peatada.

```
1.  # Imporditakse kõik osad ev3dev.ev3 teegist.
2.  from ev3dev.ev3 import *
3.
4.  # Defineeritakse muutujad mA ja mD, mis tähistavad mootoried, mis on ühendatud pesade
    sse A ja D.
5.  mA = LargeMotor('outA')
6.  mD = LargeMotor('outD')
7.
8.  # Defineeritakse puuteandur muutujaks ts ning kontrollitakse, et puuteandur oleks kor
    ralikult ühendatud.
9.  ts = TouchSensor()
10. assert ts.connected
11.
12. # Defineeritakse värviandur muutujaks cs ning kontrollitakse, et värviandur oleks kor
    ralikult ühendatud.
13. # Kaugusandur lülitatakse värvirežiimile.
14. cs = ColorSensor()
15. assert cs.connected
16. cs.mode = 'COL-COLOR'
17.
18. # Roboti nupud defineeritakse muutujaks btn.
19. btn = Button()
20.
```

```
# Defineeritakse meetod otse liikumiseks. Liikumiseks kasutatakse run_forever funktsiooni.
# See tähendab, et robot liigub otse kuni mootoritele antakse uus käsk.
def otse():
    mA.run_forever(speed_sp=360)
    mD.run_forever(speed_sp=360)

# Defineeritakse meetod takistusest ümber põikamiseks. Takistust nähes peatub robot hetkeks.
# Et mitte ümber pööramisel takistusele sisse põrgata tagurdatakse ratta täisringi võrra.
# Tagurdamise lõppedes pöörab robot vasakule  umbes 90 kraadi ning jätkab oma teed.
def takistus():
    mA.stop()
    mD.stop()
    mA.run_to_rel_pos(position_sp=-360, speed_sp=360, stop_action="brake")
    mD.run_to_rel_pos(position_sp=-360, speed_sp=360, stop_action="brake")
    mA.wait_while('running')
    mD.wait_while('running')
    mA.run_to_rel_pos(position_sp=351, speed_sp=360, stop_action="brake")
    mD.stop()
    mA.wait_while('running')

# Põhiline programmi osa ei käivitu enne kui on vajutatud puuteandurile.
# See tähendab, et programm küll töötab, aga robot ei tee midagi enne kui on vajutatud puuteandurile.
# Seejärel hakkab robot liikuma otse ning samal ajal kontrollima ka ees olevaid valgeid takistusi.
# Valgeid takistusi otsitakse värvi anduri abil.
# Takistust märgates tagurdab robot ja pöörab vasakule ning tsüklit hakatakse otsast peale läbima.
# Programm sulgetakse kasutades programmeeritava aju küljes olevaid nuppe.
while True:
    if ts.is_pressed:
        while True:
            if btn.any():
                mA.stop()
                mD.stop()
                exit()
            elif cs.color != 1:
```

```
55.                 otse()
56.             else:
57.                 takistus()
```

Joonis 44. Ülesande 6 näidiskood.

4.7 Ülesanne 7

Ülesandeks on programmeerida robot, mis liigub otse suunas kuni takistuseni ning seejärel pöörab ennast suvalises suunas ümber ja jätkab teed.

Lahenduse näidiskood on esitatud joonisel 45. Lahenduses kasutatakse *ev3dev.ev3* ja *random* teeke, kahte mootorit, programmeeritava aju nuppe ning kaugus- ja puuteandurit. Liikumiseks kasutatakse funktsiooni *run_forever*. Näites on tagurdamiseks kasutatud *run_timed* ehk funktsiooni, mis tagurdab etteantud aja vältel. Siinkohal võiks vabalt kasutada *run_to_rel_pos*, mis liiguks etteantud pöörete võrra. Kui tagurdamine on alustatud oodatakse ära tagurdamise lõpuni enne kui programm hakkab tsüklit edasi läbima. Peale tagurdamist muudab robot suunda. Suuna muutmisel on kasutatud teeki *random*. Selle abil saab lasta arvutil teha nii öelda suvalise valiku etteantud väärtuste hulgast. Muidugi miskit pole täiesti juhuslik ning suvalise arvutamiseks kasutatakse erinevaid võtteid. Antud kontekstis on Pythoni suvalise arvutamisest täiesti piisav. Tekitatakse kaheelemendiline list sisuga vasak ja parem ning lastakse arvutil valida üks väärtus. Seejärel kasutatakse funktsiooni *random.randrange*, mis võtab argumentideks arvu piirkonna alguse, lõpu ja sammu ning tagaplaanil luuakse arvude hulk etteantud vahemikus etteantud sammuga. Samm on kahe kõrvuti oleva elemendi vahe. Vastavalt arvuti tehtud valikutele robot pöörab kas vasakule või paremale ning kui palju robot pöörab ,oleneb eelnevalt suvaliselt valitud arvust. Programm ootab kuni robot on pöörde sooritanud ning jätkab tsükli läbimist. Tsüklis on kolm tingimuslauset. Esimene, mis kontrollib, kas EV3 nuppe on vajutatud ning kui on, siis peatatakse mootorid ning suletakse programm. Teine tingimuslause kontrollib teel olevaid takistusi kaugus- ja puuteanduriga samaaegselt. Kui objekt on lähemal kui 25 cm või värviandur tuvastab valge takistuse, liigutakse kolmandasse tingimuslausesse. Kui teel takistusi ei ole, liigub robot otse ning objekti nähes muudab robot suunda.

```
1.  # Imporditakse kõik osad ev3dev.ev3 teegist.
2.  # imporditakse teek random, mille abil saab valida hulgast suvalisi väärtuseid.
3.  from ev3dev.ev3 import *
4.  import random
5.
6.  # Defineeritakse muutujad mA ja mD, mis tähistavad mootoried, mis on ühendatud pesade
    sse A ja D.
7.  mD = LargeMotor('outD')
```

```
8.  mA = LargeMotor('outA')
9.
10. # Defineeritakse kaugusandur muutujaks us ning kontrollitakse, et kaugusandur oleks k
    orralikult ühendatud.
11. # Kaugusandur lülitatakse sentimeetrirežiimile.
12. us = UltrasonicSensor()
13. assert us.connected
14. us.mode = 'US-DIST-CM'
15.
16. # Defineeritakse värviandur muutujaks cs ning kontrollitakse, et värviandur oleks kor
    ralikult ühendatud.
17. # Kaugusandur lülitatakse värvirežiimile.
18. cs = ColorSensor()
19. assert cs.connected
20. cs.mode = 'COL-COLOR'
21.
22. # Roboti nupud defineeritakse muutujaks btn.
23. btn = Button()
24.
25. # Defineeritakse meetod otse liikumiseks. Liikumiseks kasutatakse run_forever funktsi
    ooni.
26. # See tähendab, et robot liigub otse kuni mootoritele antakse uus käsk.
27. def otse():
28.     mA.run_forever(speed_sp=360)
29.     mD.run_forever(speed_sp=360)
30.
31. # Defineeritakse meetod tagasi liikumiseks. Liikumiseks kasutatakse run_timed funktsi
    ooni.
32. # Robot liigub tagasi 2 sekundit ning peatub.
33. def tagasi():
34.     mA.run_timed(time_sp=2000, speed_sp=-360, stop_action="brake")
35.     mD.run_timed(time_sp=2000, speed_sp=-360, stop_action="brake")
36.     mA.wait_while('running')
37.     mD.wait_while('running')
38.
39. # Defineeritaks meetod roboti suuna muutmiseks. Siin kasutatakse võrdlemisi palju tee
    gi random abi.
40. # Esmalt valitakse suvaliselt suund kummale poole keeratakse.
41. # Seejärel valitakse arv 200-st 700-ni sammuga 50 (ehk siis 200, 250, 300, ...).
42. # Selle arvuga määratakse pööramisel kui mitu kraadi mootor ratast pöörab.
```

```
43. def suund():
44.     i = random.choice(['parem', 'vasak'])
45.     j = random.randrange(200, 700, 50)
46.     print(str(j))
47.     if i == 'parem':
48.         mA.run_to_rel_pos(position_sp=j, speed_sp=360, stop_action="brake")
49.         mD.stop()
50.         mA.wait_while('running')
51.     elif i == 'vasak':
52.         mA.stop()
53.         mD.run_to_rel_pos(position_sp=j, speed_sp=360, stop_action="brake")
54.         mD.wait_while('running')
55.
56. # Tsükli abil kontrollitakse ega ei vajutata roboti nuppe, millega programm sulgetaks
    e.
57. # Seejärel kontrollitakse ega ükski objekt ei asu robotile lähemal kui 25 cm või robo
    ti teel ees poleks tekistust valge teekatte näol.
58. # Valgeid takistusi maast otsitakse värvi anduri abil.
59. # Objektide kaugust kontrollitakse kaugusanduri abil.
60. # Kui need tingimused on täidetud hakkab robot otse liikuma ning tsükkel alustab kont
    rollimist otsast peale.
61. # Kui leidub objekt mis on robotile lähemal kui 25 cm või takistus valge teekatte näo
    l,
62. # siis robot tagurdab ja keerab ennast suvalises suunas ning jätkab teed.
63. while True:
64.     if btn.any():
65.         mA.stop()
66.         mD.stop()
67.         exit()
68.     elif us.distance_centimeters > 250 and not cs.color == 1:
69.         otse()
70.     else:
71.         tagasi()
72.         suund()
```

Joonis 45. Ülesande 7 näidiskood.

4.8 Ülesanne 8

Ülesandeks on programmeerida robot, mis saab sisendeid klaviatuurilt ning reageerib neile vastavalt. Näitena on tehtud mälumäng, kus robot näitab LED-tuledega mustri ette ning kasutaja sisestab värvidele vastavate numbritega sama kombinatsiooni.

Lahenduse näidiskood on esitatud joonisel 46. Lahenduses kasutatakse ev3dev.ev3, import ja time teeki ning roboti LED-tulesid. Programmis on LED-tuled defineeritud objektideks *led*, sest see muudab koodi loetavamaks. Defineeritakse funktsioon *blackout*, mis kustutab ära roboti LED-tuled. Roboti tulesid saab ainult lülitada ükshaaval ehk eraldi tuleb lülitada vasak ja parem tuli. Meetod *kontroll* võtab argumendiks LED-tule värvuse ning lisab tulele vastava numbri listi, mis sisaldab mängu õiget vastust. Meetod *kuvamine* võtab argumendiks listi, mille elemendid on LED-tulede värvid. Ning *for*-tsükliga käiakse kõik elemendid läbi ning roboti tuled lülitatakse vastavat värvi. Seejärel programm ootab 1.5 sekundit ning lülitab tuled välja ning ootab veel 0.3 sekundit enne kui võtab ette uue elemendi. Põhiline mängu osa on *while*-tsükli sees, mille sees on omakorda kaks tähtsamat *while*-tsüklit. Juba enne mängu algust luuakse 2 listi: üks on õiget vastust sisaldav list ning teine sisaldab ettemängitavaid värve. Iga mängu alguses kuvatakse kasutaja konsoolile mängureeglid. Mänguraundi tsükli sees on nimekiri LED-tulede valgusega, millest valitakse juhuslik ning lisatakse kuvamiseks mõeldud listi ning meetodiga *kontroll* lisatakse vastav arvväärtus vastuse listi. Seejärel käivitatakse kuvamine ning tulede kombinatsioon mängitakse kasutajale ette. Peale tulede kustumist küsitakse kasutajalt vastust funktsioon *input* abiga. Kasutaja sisestab vastuse konsoolilt ning vajutab Enter nuppu. Kui vastus on õige algab uus mänguraund, listi lisatakse uus värv jne. Kui vastus on vale, siis mäng lõppeb ning kasutajale pakutakse uut mängu võimalust. Küsimusele „Kas soovid veel proovida ?“ vastab kasutaja kas jah või ei samuti konsoolilt. Jah vastuse korral, algab uus mäng. Eitava vastuse korral programm suletakse ning kui kasutaja vastab midagi muud, siis esitatakse talle küsimus uuesti.

```
1.  # Imporditakse meetod sleep, teek random ning kõik osad ev3dev.ev3 teegist.
2.  import random
3.  from time import sleep
4.  from ev3dev.ev3 import *
5.
6.  # Roboti LED-tuled defineeritakse muutujaks led.
7.  led = Leds()
8.
9.  # Defineeritakse funktsioon, mis lülitab LED-
    tuled vahepeal välja, et tekiks vilkumise efekt.
10. def blackout():
11.     led.set_color(led.RIGHT, led.BLACK)
```

```
12.     led.set_color(led.LEFT, led.BLACK)
13.
14. # Siin lisatakse listi numbreid, mida kasutajalt vastuseks oodatakse.
15. # Siin täiendatud listi vastu pärast võrreldakse kasutaja vastust.
16. # Numbreid lisatakse vastavalt sellele, mis värv on viimasena suvaliselt valitud.
17. def kontroll(ledcolor):
18.     if ledcolor == led.RED:
19.         kontroll_set.append('1')
20.     elif ledcolor == led.GREEN:
21.         kontroll_set.append('2')
22.     elif ledcolor == led.AMBER:
23.         kontroll_set.append('3')
24.     elif ledcolor == led.ORANGE:
25.         kontroll_set.append('4')
26.     elif ledcolor == led.YELLOW:
27.         kontroll_set.append('5')
28.
29. # Defineeritakse funktsioon, milles toimub LED-tulede vilgutamine.
30. # Sisse loetud listist võetakse järjest liikmeid ning muudetakse roboti tuled seda vä
    rvi.
31. # Seejärel programm ootab 1.5 sekundit ning seejärel kustutatakse tuled ning oodataks
    e 0.3 sekundiks.
32. # Peale tulede kustutamist süüdatakse LED-tuled uuesti listist järgmise liikmega.
33. def kuvamine(list):
34.     for i in list:
35.         led.set_color(led.RIGHT, i)
36.         led.set_color(led.LEFT, i)
37.         sleep(1.5)
38.         blackout()
39.         sleep(0.3)
40.
41. # Kogu mängu tegevus toimub siin tsüklis.
42. # Esmalt luuakse 2 tühja listi, kuhu hiljem lisatakse suvaliselt vaitud LED-
    tulede värve.
43. # Teise listi luuakse värvide põhjal kontroll nimekiri numbritest, mida võrreldakse k
    asutaja vastusega.
44. # Iga uue mängu alguses tutvustatakse kasutajale reegleid.
45. # Mängu alguses lülitatakse LED-tuled välja.
46. # Võimalikest tulede nimekirjast valitakse juhuslik värvus ning täiendatakse mõlemat
    loodud listi.
```

```
47. # Seejärel mängitakse tulede järjekord ette ning jäädakse kasutaja vastust ootama.
48. # Kui kasutaja sisestatud numbri kombinatsioon on õige prinditakse konsooli 20 rühja
    rida, et eelmist vastust näha poleks.
49. # Seejärel kiidetakse kasutajat ning algab uus mänguraund.
50. # Kui kasutaja vastab valesti, siis mäng on läbi.
51. # Mängu lõppedes küsitakse kasutajalt kas ta soovib veel mängida.
52. # Jaatava vastuse korral algab uus mäng ning eitava korral programm suletakse.
53. while True:
54.     nimekiri = []
55.     kontroll_set = []
56.     game = True
57.     print("Mängime kõige tavalisemat mälu mängu.")
58.     print("Esmalt tuleb meelde jätta 5 numbri ja värvi paari:")
59.     print("Punane - 1")
60.     print("Roheline - 2")
61.     print("Merevaik (kuldpruun) - 3")
62.     print("Oranž - 4")
63.     print("Kollane - 5")
64.     print("Robot vilgutab oma LED tuledega värvi kombinatsiooni.")
65.     print("Ning sina sisestad konsooli aknasse vastava numbri kombinatsiooni.")
66.     print("Näiteks kui robot vilgutab punane-roheline tuleb sisestada number 12")
67.     print("Hakkame pihta!!!")
68.     sleep(5)
69.     while game:
70.         blackout()
71.         colors = [led.RED, led.GREEN, led.AMBER, led.ORANGE, led.YELLOW]
72.         color = random.choice(colors)
73.         nimekiri.append(color)
74.         kontroll(color)
75.         kuvamine(nimekiri)
76.         blackout()
77.         user_input1 = input("Sisesta värvide järjestus ja vajuta Enter: ")
78.         if str(user_input1) == ''.join(kontroll_set):
79.             x = 0
80.             while x < 20:
81.                 print('\n')
82.                 x += 1
83.             print("Jee, õige proovime veel.")
```

```
84.         else:
85.             print("Oih, valesti läks!")
86.             print("Proovi uuesti, harjutamine teeb meistriks")
87.             print("Õige vastus oli: " + ''.join(kontroll_set))
88.             game = False
89.     vastus = False
90.     while not vastus:
91.         user_input2 = input("Kas soovid veel proovida? (jah/ei): ")
92.         if user_input2 == "jah":
93.             print("Tore, uuesti !!!")
94.             vastus = True
95.         elif user_input2 == "ei":
96.             print("Kahju, nägemiseni")
97.             led.set_color(led.RIGHT, led.GREEN)
98.             led.set_color(led.LEFT, led.GREEN)
99.             exit()
```

Joonis 46. Ülesande 8 näidiskood.

5. Kokkuvõte

Teemaga tutvumiseks aitas autor vabatahtlikuna korraldada 2016 ja 2017 aastal toimunud FIRST LEGO League Eesti võistlusi ja Robomiku Lahingu võistlusi. Autor osaleb ka 25 – 28 mail 2017 toimuval FIRST LEGO League Open Europe Championship üritusel Taanis vabatahtlikuna, et omandada kogemusi ja aidata korraldada samanimeline üritus Eestis 2018 aastal.

2017 aastal võttis FIRST LEGO League võistlusest osa üle 900 inimese. Võistlejad, kes üritusel osalesid on kindlasti läbi teinud ka LEGO MINDSTORMS EV3-ga kaasas käiva õppeprogrammi ning sooviksid proovida midagi enamat. Tänu töö käigus valminud juhendile on robootikaringi juhendajatel võimalus komplekteerida kursused, mille käigus saab noortele õpetada programmeerimiskeelt Python. LEGO MINDSTORMS EV3 robotite abil on võimalik noortele õpetada peaaegu kõiki baasteadmisi Pythonist.

Töö käigus valmis terviklik eestikeelne juhend, kuidas paigaldada ev3dev operatsioonisüsteem LEGO MINDSTORMS EV3 robotile ning kuidas paigaldada JetBrains'i PyCharm nii, et programmeeritud Pythoni failid saaksid võimalikult lihtsasti robotile. Samuti selgitatakse kuidas programmeerida LEGO MINDSTORMS EV3 robotit programmeerimiskeeles Python. Töös selgitatakse kuidas õppeasutused saaksid omandada JetBrains'i litsentsi, et kasutada PyCharm *Professional* versiooni, mille suureks eeliseks on robotiga ühendumine SSH ühendusega. Töös selgitatakse, kuidas seadistada PyCharmi *Community* versiooni ehk tasuta versiooni ja WinSCP programmi, et nende koostööl tekiks sünkroniseeritud ühendus arvuti ja roboti vahel. Selline kombinatsioon kompenseerib PyCharmi *Community* versiooni SSH ühenduse võimaluse puudumist ning tööomadused on võrdlemisi sarnased PyCharmi tasulise *Professional* versiooni omadustega.

Juhendi sihtgrupiks on robootikaringi juhendajad ning see tõttu on juhend töös kirjeldatud mõnevõrra detailsemalt. Töö teeb eriliseks see, et juhendit kuidas programmeerida LEGO MINDSTORMS EV3-e Pythonis kasutades ev3dev operatsioonisüsteemi pole varem eesti keeles avaldatud. Üheks põhjuseks on kindlasti see, et ev3dev avaldati alles 2016 detsembri lõpus. Ka varem on saanud LEGO roboteid programmeerida erinevates programmeerimiskeeltes, ent EV3-e programmerimiseks olid saadaval lahendused, mille litsentsid maksid võrdlemisi palju.

Töös on esitatud 8 erinevat ülesannet, milles on kasutatud kaugus,- puute- ja värviandurit, mootoreid, ning EV3 programmeeritava aju heli, ekraani, nuppe ja LED-tulesid. Esitatud on ka võimalikud hästi kommenteeritud lahendused. Töös esitatud ülesanded algavad lihtsamatest ning lähevad järjest keerulisemaks. Esialgu pannakse robot sirgjooneliselt liikuma ning seejärel lisanduvad ükshaaval pööramised, andurite kasutamine ning EV3 programmeeritava aju funktsioonide kasutamine. Kindlasti ei kajastu töös kõik, mida saab LEGO MINDSTORMS EV3-ga Pythonis teha. Samuti on võimalik kasutada EV3 komplektiga kaasas käivat õppeprogrammi, kuid programmeerimiskeelena kasutada Pythonit.

6. Viidatud kirjandus

[1] LEGO.com About Us - TimeLine 1932 - 1939. https://web.archive.org/web/20101024060020/http://aboutus.lego.com:80/en-us/factsfigures/timeline1930.aspx (09.05.2017)

[2] LEGO | Brickipedia | Fandom powered by Wikia. http://lego.wikia.com/wiki/LEGO (09.05.2017)

[3] LEGO.com About Us - TimeLine 1940 - 1949. https://web.archive.org/web/20101024060025/http://aboutus.lego.com:80/en-us/factsfigures/timeline1940.aspx (09.05.2017)

[4] Kell J. Here's why Mattel ousted its CEO Bryan Stockton | Fortune.com, 2015. http://fortune.com/2015/01/26/heres-why-mattel-ousted-its-ceo-bryan-stockton/ (09.05.2017)

[5] LEGO Education | Brickipedia | Fandom powered by Wikia. http://lego.wikia.com/wiki/LEGO_Education (09.05.2017)

[6] [6] 10831 My First Caterpillar - Products - DUPLO LEGO.com. https://www.lego.com/en-us/duplo/products/my-first-caterpillar-10831 (09.05.2017)

[7] Lego Duplo – Wikipedia. https://en.wikipedia.org/wiki/Lego_Duplo (09.05.2017)

[8] LEGO® 42042 Technic. http://www.happytoys.nl/contents/en-uk/p4786_LEGO%C2%AE_42042_Technic.html (09.05.2017)

[9] Crawler Crane - LEGO Technic - Designer Video 42042 – YouTube, 2015. https://www.youtube.com/watch?v=y3Om9eox0cg (09.05.2017)

[10] About Us - LEGO Education. https://education.lego.com/en-us/about-us (09.05.2017)

[11] ROBOTC.net - Start programming your NXT robotics projects now!. http://www.robotc.net/download/rcx/ (09.05.2017)

[12] History - Mindstorms LEGO.com. https://www.lego.com/en-us/mindstorms/history (09.05.2017)

[13] 9580 LEGO Education WeDo Construction Set | Rapid Online. https://www.rapidonline.com/9580-lego-education-wedo-construction-set-70-6500 (09.05.2017)

[14] LEGO's WeDo: An Innovative Robotic Kit for Children | MIT Media Lab. https://www.media.mit.edu/sponsorship/getting-value/collaborations/wedo (09.05.2017)

[15] Miles S. Lego Education WeDo 2.0 brings Lego robots to the classroom - Pocket-lint, 2016. http://www.pocket-lint.com/news/136404-lego-education-wedo-2-0-brings-lego-robots-to-the-classroom (09.05.2017)

[16] LEGO® Education WeDo 2.0 Core Set. https://education.lego.com/en-us/products/lego-education-wedo-2-0-core-set/45300 (09.05.2017)

[17] WeDo 2.0 downloads – LEGO Education. https://education.lego.com/en-us/downloads/wedo-2 (09.05.2017)

[18] Mindstorms - What is the difference between the NXT and the RCX? – Bricks, 2012. https://bricks.stackexchange.com/questions/446/what-is-the-difference-between-the-nxt-and-the-rcx (09.05.2017)

[19] Laurens, Tutorial: Understanding the difference between NXT set versions – Robotsquare. http://robotsquare.com/2012/02/18/understanding-nxt-versions/ (09.05.2017)

[20] FirmwareVersions – RWTH - Mindstorms NXT Toolbox http://www.mindstorms.rwth-aachen.de/trac/wiki/FirmwareVersions (09.05.2017)
[21] NXT 1.0, 2.0, and Education Inventory Comparison http://static.robotclub.ab.ca/pages/nxt/InventoryComparison/nxt_1_vs_2_vs_edu.html (09.05.2017)
[22] Ultrasonic Sensor | LEGO Shop https://shop.lego.com/en-US/Ultrasonic-Sensor-9846 (09.05.2017)
[23] Sound Sensor | LEGO Shop https://shop.lego.com/en-US/Sound-Sensor-9845 (09.05.2017)
[24] Light Sensor | LEGO Shop https://shop.lego.com/en-US/Light-Sensor-9844 (09.05.2017)
[25] Lego Mindstorms EV3 - Wikipedia https://en.wikipedia.org/wiki/Lego_Mindstorms_EV3 (09.05.2017)
[26] Laurens, The Difference Between LEGO MINDSTORMS EV3 Home Edition (#31313) and LEGO MINDSTORMS Education EV3 (#45544) – Robotsquare, 2017. http://robotsquare.com/2013/11/25/difference-between-ev3-home-edition-and-education-ev3/ (09.05.2017)
[27] Laurens, EV3 Home Edition or EV3 Education Edition: Which one to buy? – Robotsquare. http://robotsquare.com/2014/11/13/ev3-home-edition-or-ev3-education-edition/ (09.05.2017)
[28] Vision and Mission | FIRST. https://www.firstinspires.org/about/vision-and-mission (09.05.2017)
[29] Elementary – LEGO Education. https://education.lego.com/en-us/elementary/explore (09.05.2017)
[30] FIRST LEGO League Eesti statistika 2012 – 2017. https://www.robootika.ee/flleesti/?page_id=2829 (11.05.2017)
[31] Lechner, D. Announcing ev3dev-jessie-2016-12-21 Release, 2016. http://www.ev3dev.org/news/2016/12/21/ev3dev-jessie-2016-12-21-release/ (09.05.2017)
[32] Programming Languages. http://www.ev3dev.org/docs/programming-languages/ (09.05.2017)
[33] Ralph Hempel | LinkedIn https://www.linkedin.com/in/ralph-hempel-8003412/?ppe=1 (09.05.2017)
[34] GitHub - rhempel/ev3dev-lang-python: Pure python bindings for ev3dev https://github.com/rhempel/ev3dev-lang-python (09.05.2017)
[35] AKIT - Andmekaitse ja infoturbe leksikon. http://akit.cyber.ee/term/2486 (11.05.2017)
[36] PuTTY: a free SSH and Telnet client. https://www.chiark.greenend.org.uk/~sgtatham/putty/ (09.05.2017)
[37] PuTTY FAQ. https://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html#faq-32bit-64bit (09.05.2017)
[38] GitHub - rhempel/ev3dev-lang-python: Pure python bindings for ev3dev. https://github.com/rhempel/ev3dev-lang-python#upgrading-this-library (09.05.2017)
[39] PyCharm. https://www.jetbrains.com/pycharm/ (09.05.2017)
[40] GitHub - rhempel/ev3dev-lang-python: Pure python bindings for ev3dev https://github.com/rhempel/ev3dev-lang-python#python-2x-and-python-3x-compatibility (09.05.2017)

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina **Henry Maalinn**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   LEGO MINDSTORMS EV3-e programmeerimine Pythonis,
   mille juhendaja on Anne Villems, Taavi Duvin ja Alo Peets,
   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, **11.05.2017**