TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kaarel Maidre

Pythonitaolise keele kompilaator Ozobotile

Bakalaureusetöö (9 EAP)

Juhendaja: Aivar Annamaa

Tartu 2017

Pythonitaolise keele kompilaator Ozobotile

Lühikokkuvõte:
Käesoleva bakalaureusetöö eesmärk on luua programmeerimiskeele Python sarnane keel, millega oleks võimalik programmeerida Ozobot robotit. Antud keele näol on tegemist õppevahendiga, mille eesmärk on programmeerimisoskuse õpetamine läbi Ozobot roboti.

Võtmesõnad:
Robootika, Ozobot, kompilaator, Python

CERCS:
P175 Informaatika

Python-like Language Compiler for Ozobot

Abstract:
The purpose of this bachelor thesis is to create a Python-like language for the Ozobot robot, so that the created language could be used as a means to teach programming through the Ozobot robot.

Keywords:
Robotics, Ozobot, compiler, Python

CERCS:
P175 Informatics

Sisukord

Sissejuhatus

Üks nii Eestis kui ka mujal maailmas arenev programmeerimise õpetamise suund on hakata õpetama programmide kirjutamist lastele. Näiteks on Tartu Ülikoolil mitmeid programme, mis on mõeldud just erinevas vanuses kooliealistele lastele programmeerimise oskuste õpetamiseks ja arendamiseks [1]. Kooliealiste, eriti just algkooli laste, puhul on aga üsna oluline, et programmeerimise õppimisel saadav emotsioon oleks positiivne ning lapsed saaksid õppida läbi mänguliste tegevuste.

Üheks võimaluseks lastele programmeerimise mänguliseks tegemisel on kasutada lastele mõeldud roboteid, mida oleks võimalikult lihtne programmeerida. Antud töö eesmärk ongi luua ühele sellisele robotile - Ozobotile - Pythoni keele sarnane keel, millega oleks võimalik Ozoboti lihtsasti programmeerida. Loodud keele eesmärk on olla Pythoni programmerimiskeele põhine edasiarendus Ozoblockly keskkonnast, mis aitaks Ozoboti abiga õpetada programmeerimist ning lihtsustaks ka mahukamate Ozoboti programmide loomist, kuna Ozoblocklys suuremate programmide kokku lohistamine võib olla üsna tülikas.

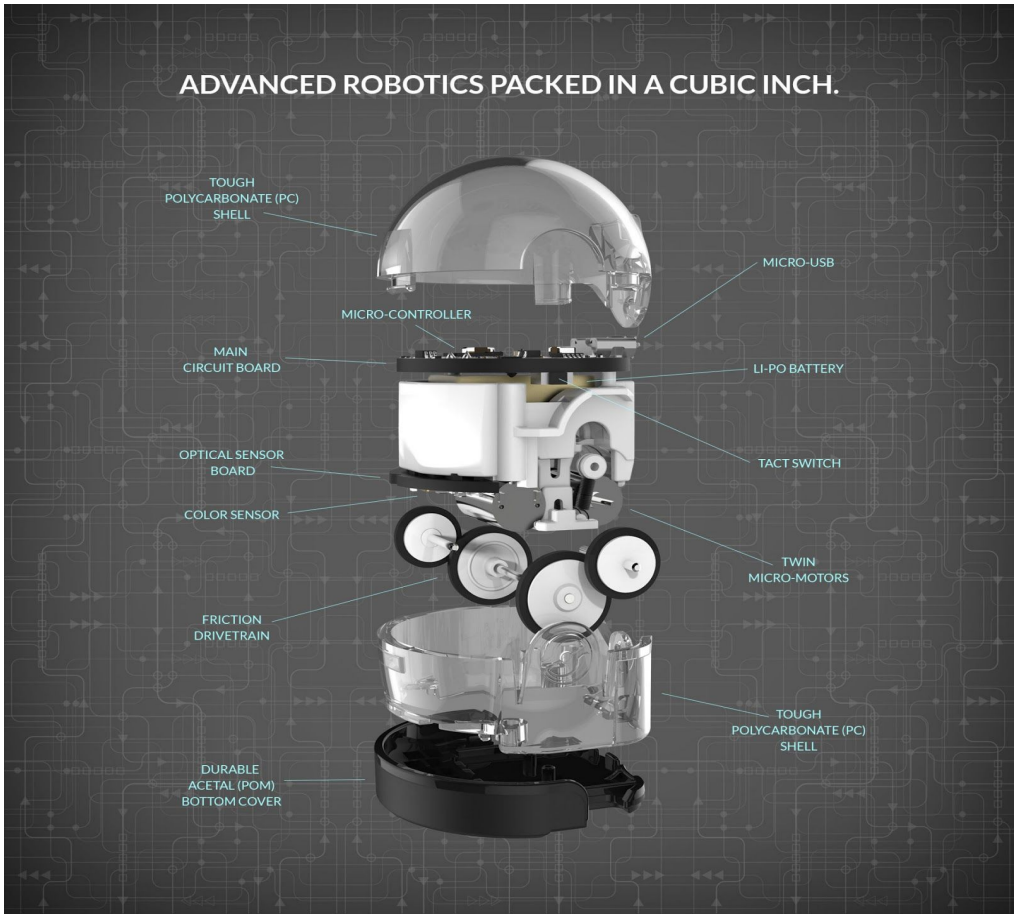
Käesolev töö koosneb kahest osast. Esimeses peatükis antakse ülevaade Ozobot robotist, olemasolevatest Ozoboti programmerimise viisidest ning peatüki lõpus kirjeldatakse lisaks veel robotit nimega Edison ning võrreldakse seda Ozobotiga. Teine peatükk kirjeldab töö praktilist poolt ning annab ülevaate Ozoboti programmi struktuurist ja töö käigus valminud Pythoni sarnasest keelest. Samuti kirjeldatakse peatüki lõpus loodud Thonny pistikprogrammi. Üldisemalt on ka suur osa töö teisest peatükist Ozoboti programmide struktuuri eesti keelne dokumentatsioon, mis enne käesolevat tööd puudus.

1. Ozobot

Antud peatükk annab ülevaate Ozobotist, erinevatest Ozoboti variatsioonidest ning olemasolevatest võimalustest Ozoboti juhtimiseks ning programmeerimiseks. Enamjaolt keskendub antud peatukk Ozoboti variatsioonile Ozobot bit, kuna see on töö praktilises osas kasutatud variant. Lisaks kirjeldatakse antud peatükis robotit nimega Edison ning võrreldakse Edison ja Ozobot roboteid omavahel.

1.1 Ozoboti üldine kirjeldus

Ozobot on väike, paari sentimeetrise läbimõõduga, robot, mille põhiliseks eesmärgiks on loovuse ja programmeerimisoskuste arendamine kuue aastaste ja vanemate laste hulgas [2]. Ozobotist on olemas kolm erinevat versiooni: Ozobot 1.0, Ozobot bit ja Ozobot evo. Kõik antud kolm Ozoboti versiooni on oma suuruse ja eesmärgi põhiselt üsna sarnased, kuid iga järgnev versioon lisab mingi hulga funktsionaalsust nii programmeerimise võimaluste kui ka roboti arukuse ja võimekuse poolest.



Joonis 1. Ozobot bit ehitus [3]

Eeloleval joonisel on näha roboti Ozobot bit ülesehitust. Nagu jooniselt 1 näha on Ozoboti puhul tegemist väikese polükarbonaat kestas oleva robotiga, mis koosneb trükkplaadist, mikrokontrollerist, liitiumpolümeerakust, mikro-usb ühenduspesast, värvisensoritest, lülitist ning liikumiseks vajalikest ratastest ja kahest mikromootorist. Lisaks on Ozobotil LED pirn, mida pole antud joonisel välja toodud.

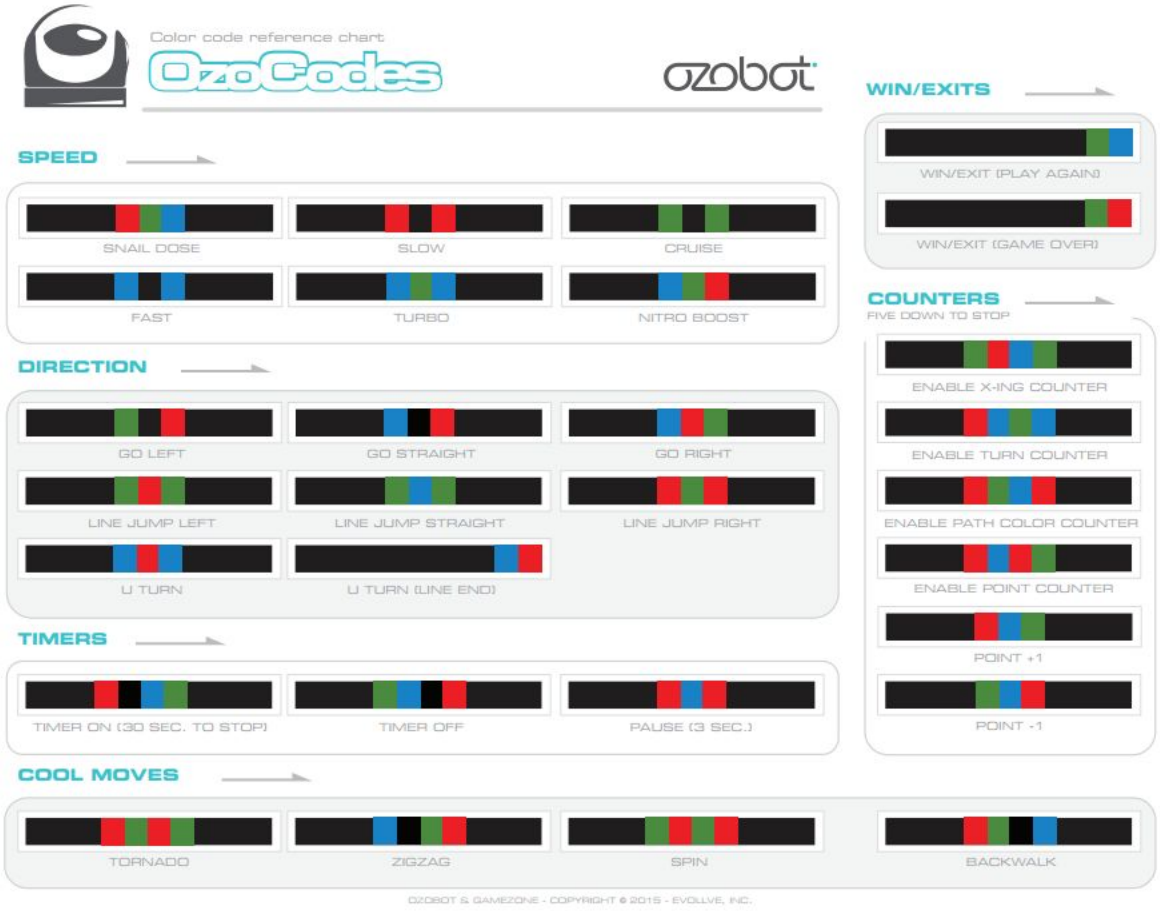
Ozobotil olevat mikro-USB ühenduspesa kasutatakse ainult aku laadimiseks ning muu funktsionaalsus sellel puudub. Ozoboti aku laadimiseks kulub umbes 30-40 minutit ning üks täis laadimine lubab robotit kasutada umbes 90 minutit [4]. Kuigi võiks arvata, et ka Ozoboti programmeerimine käib läbi mikro-USB liidese, siis see nii ei ole ning kõik robotiga seotud juhtimine käib läbi Ozoboti all olevate värvisensorite.

Nagu eelnevalt mainitud, on kõik Ozoboti variandid sarnase ülesehitusega, kuid Ozoboti kõige uuem versioon evo lisab juurde veel infrapuna sensori, mida robot kasutab takistuste vältimiseks, kõlari ning mobiilirakenduse kaudu kaugjuhtimise võimaluse [5]. Ozobot 1.0 ja Ozobot bit suurim erinevus on võimalus robotit programmeerida kasutades Ozoblockly keskkonda ning spetsiaalselt värvikeelt. Lähemalt on Ozoboti programmeerimise ja juhtimise erinevatest võimalustest kirjutatud antud töö järgmises peatükis.

1.2 Olemasolevad võimalused Ozoboti juhtimiseks ja programmeerimiseks

1.2.1 Joone järgimine

Ozoboti põhiline juhtimise ja programmeerimise viis on joone järgimine ning see on ka ainukene roboti juhtimise viis, mida toetavad kõik kolm Ozoboti erinevat varianti. Nimelt vajutades ühe korra Ozoboti küljes olevat lülitit, käivitub robot joone järgimise režiimis. Asetades antud režiimis Ozoboti mustale joonele, hakkab robot mööda seda sõitma. Antud mustade joonte sisse on võimalik asetada teatud värvikoode, mille Ozobot enda all olevate sensoritega ära tunneb ning siis vastavalt loetud koodidele teatavalt käitub.



Joonis 2. Ozoboti joonte värvikoodid [6]

Antud jooniselt on näha kõik erinevad värvikoodid, millega on võimalik Ozoboti käitumist joonte järgimise režiimis muuta. Nagu näha, on võimalik antud värvikoodidega muuta nii roboti kiirust kui ka liikumise suunda ning samuti on olemas käsud taimerite, loendurite, mängu lõpetamise ja teatud liigutuste jaoks.

Kirjeldatud värvikoodidega jooni saab lihtsalt paberile joonistada ning seega on võimalik igaühel väga lihtsate vahenditega luua erinevaid radu, mida Ozobot läbida saab. Samuti on Ozobotiga kaasas üks näiterada, mille peal Ozobot sõita oskab ning mis demonstreerib erinevaid võimalike käske, mida kirjeldatud värvikoodidega võimalik anda on.

Värvikoodidega on võimalik välja mõelda ja mängida erinevaid mänge, kus näiteks mitu erinevat Ozoboti omavahel võistelda saavad. Samuti on Ozoboti kodulehel olemas juba mitmeid joonte põhiseid mänge, mida on võimalik alla laadida ning seejärel lihtsalt värviprinteriga välja printida ja kasutada [7]. Kirjeldatud mängude hulgas on mitmeid erinevaid mänge. Näiteks on olemas

loogikapõhised mängud, kus on ette antud rada, kuhu on sisse jäetud lüngad, mis tuleb raja läbimiseks korrektsete värvikoodidega ära täita. Samuti on olemas mänge kus saavad mitu Ozoboti omavahel võistelda, kes kiiremini raja läbib. Veel saab Ozoboti kasutada mängudes abivahendina ehitades raja, kus robotil on võimalik jõuda juhuslikkuse alusel mitmesse erinevasse lõppolekusse ning siis vastavalt lõppolekule saab vastu võtta juhuslikkusel põhineva otsuse. Eelnevalt kirjeldatud mängud põhinevad suuresti faktile, et juhul kui Ozobotile ei ole öeldud teisiti, siis kohates ristmiku valib Ozobot edasise suuna juhuslikult [8].

Ozoboti on võimalik kasutada ka tahvelarvutitega. Nimelt on olemas nii Google Play's kui ka Apple App Store's kaks ametliku Ozobotile mõeldud rakendust: Ozobot ja OzoGroove. Samuti on olemas Ozoboti kodulehel veebipõhine joonte järgimisel põhinev mängude keskkond [9].

Rakenduse Ozobot Google Play kirjelduses [10] on kirjutatud, et antud rakendus on mõeldud üheksatollise või suurema ekraaniga seadmetele ning lubab mängida kolme erinevat mängu: OzoDraw, OzoLuck ja OzoPath, kasutades selleks ühte või rohkemat Ozoboti. Ozodraw puhul on tegemist joonistamise mänguga, mille eesmärk on loovuse arendamine ja robotiga suhtlemine. Ozoluck on juhusele ülesse ehitatud mäng, kus Ozobot valib mängijate saatuse, mida saab mängida 2-8 mängijaga. OzoPathi näol on tegemist ühe või kahe mängija strateegia mänguga, kus tuleb 5x5 mänguväljal luua juhuslikult tekkivatest plaatidest tee alguspunktist lõpp-punkti.

Kui rakendus Ozobot põhineb joonte järgimisel, siis rakenduse OzoGroove [11] puhul on tegemist rakendusega, mis paneb Ozoboti muusika järgi tantsima. Antud efekt saavutatakse programmeerides Ozoboti spetsiaalse värvikeelega, mis lubab Ozoboti kontrollida ilma, et Ozobot peaks järgima kindlaid jooni ning millega saab Ozoboti peale laadida programme, mida on võimalik hiljem käivitada. Antud programmeerimise meetodit kirjeldab järgmine peatükk.

1.2.2 Värvikeel ja Ozoblockly

Teine meetod Ozoboti programmeerimiseks on kasutada spetsiaalselt värvikeelt, mis ei ole vastavuses eelmises peatükis kirjeldatud värvikoodidega. Antud programmeerimise meetod ei ole saadaval Ozobot 1.0 versioonil vaid ainult Ozobot bit ja evo versioonidel ning on ka peamine erinevus Ozobot 1.0 ja Ozobot bit vahel.

Antud meetodi kasutamiseks on olemas Google Blockly'l põhinev veebikeskkond Ozoblockly [12]. Ozoblockly on keskkond, mis lubab erinevaid plokke lohistades luua erinevaid Ozoboti programme ning neid programme läbi Ozoboti värvikeele robotisse laadida. Antud keskkond annab Ozoboti kontrollimiseks palju rohkem tööriistu, kui eelnevalt kirjeldatud joonte järgimine ning käesoleva bakalaureusetöö eesmärk ongi just luua Pythoni sarnane programmeerimiskeel,

millega saaks Ozoboti programmeerida samade funktsionaalsustega, mida pakub Ozoblockly keskkond.

Joonis 3. Ozoblockly keskkond [12]

Joonis 3 kujutab Ozoblockly programmeerimiskeskkonda, mis nagu juba mainitud lubab luua Ozoboti programme ja valmis olevaid programme läbi värvikeele Ozoboti peale laadida.

Ozoblockly keskkonna üleval vasakus servas on võimalik valida, kas kirjutada programme Ozobot bit või evo versioonile. Ozobot bit ja evo versiooni keskkonnad on väga sarnased, kuid kuna evo versioonil on lisaks veel kõlar ning infrapuna sensor, siis valides Ozoblockly's evo versiooni, lisanduvad keskkonda heli ja infrapuna sensori võimalused.

Roboti versiooni valiku all on käskude tulp, mis on omakorda jaotatud plokkideks, kust on võimalik käsklusi ekraani keskel olevasse alasse lohistada. Nagu jooniselt 3 näha on võimalik antud käskude tulpa kuvada nelja erineva raskusastme kaupa: *Novice*, *Beginner, Intermediate* ja *Advanced*. Olemas on ka viies raskusaste *Master*, kuid antud aste ei olnud veel töö kirjutamise hetkel saadaval. Iga järgnev raskusaste toob esile rohkem funktsionaalsust, kuid on samas ilmselt programmeerimisega alustavale inimesele raskemini hoomatav. Näiteks *Novice* raskusastme korral on näha ainult kolm käskude plokki: *Movement, Light Effects* ja *Wait* ning kõik plokkides olevad valikud on lihtsad ikoonid, kuid kõige kõrgema raskusastme peal on näha kümme erinevat käskude plokki: *Movement, Line Navigation, Light Effects, Timing, Terminate, Logic, Loops, Math, Variables* ja *Functions*.

Movement ploki all on kõik liikumisega seotud käsud, mis ei kuulu joonte järgimise alla. Antud ploki alt on võimalik anda käske Ozoboti liikumise kiiruse, suuna ja pööramiste kohta. Samuti on võimalik ka Ozobot seisma jätta ning lugeda hetkel Ozoboti all oleva pinna värvi.

Line navigation ploki all on kõik käsud, mis on seotud joonte järgimisega. Seadistada saab joone peal sõitmise kiirust, lugeda joone värvi, valida ristmikul õiget suunda ja kontrollida, kas ristmikul on üldse teatud suund olemas.

Light Effects ploki seest on võimalik määrata Ozoboti peal oleva LED lambi värvi või see üldse välja lülitada. Värvi on võimalik sättida nii juhuslikult kui ka sisestades punase, sinise ja rohelise värvi väärtused skaalal 0 kuni 127.

Timing ploki alt on võimalik panna Ozobot valitud ajaks ootama, enne järgmise käsu täitma asumist.

Terminate ploki alt saab Ozoboti programmi lõpetada ning valida, mis režiimi Ozobot pärast lõpetamist jääb, kas välja lülitatud, passiivne või joonte järgimine.

Logic ploki sees on kõik loogika konstruktsioonid nagu näiteks tingimuslausete ehitamine ja võrduste ning võrratuste kontrollid.

Loops ploki abil on võimalik luua programmis tsükleid.

Math plokk sisaldab matemaatikaga seotud konstruktsioone nagu konstandid, tehted, absoluutväärtuse või jäägi arvutamine ja etteantud vahemikus juhuarvu genereerimine.

Variables plokk lubab luua uusi muutujaid, anda neile numbrilisi või sensori väärtusi ning olemasolevaid muutujaid lugeda.

Functions ploki abil on võimalik luua funktsioone, mida saab hiljem välja kutsuda, millele saab anda kaasa parameetreid ning millel võib olla ka tagastusväärtus.

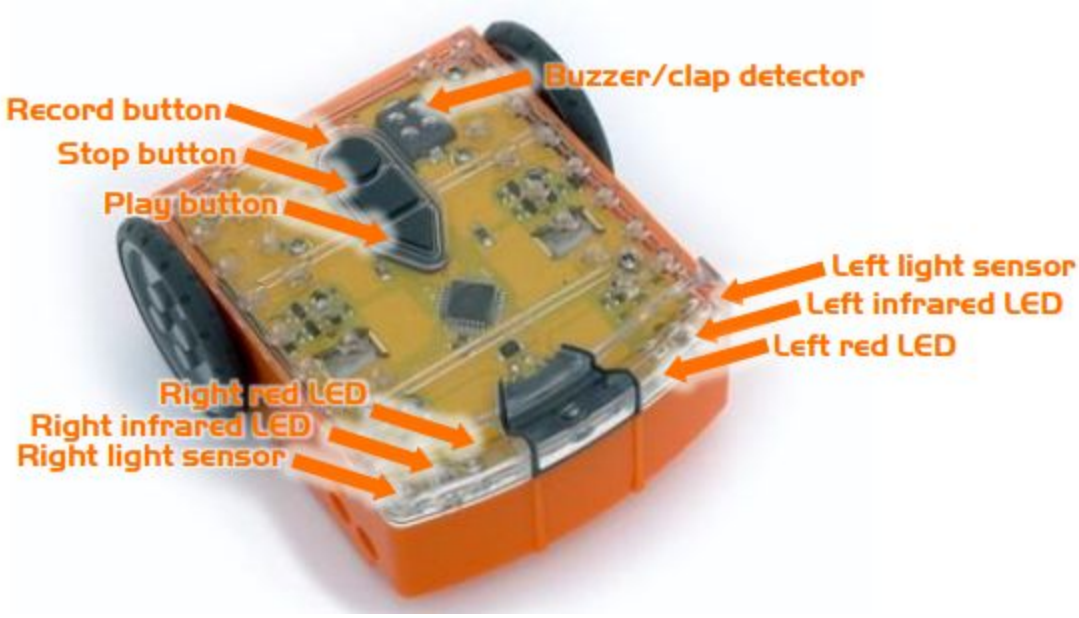
Programmi valmimisel on võimalik valmis programm Ozoboti laadida. Selleks tuleb Ozobot ühe lüliti vajutusega käivitada ning seejärel asetada ekraani all olevale valgele Ozoboti kujulisele alale ning vajutada nuppu “Load bit”. “Load bit” nupule vajutades hakkab kirjeldatud valge ala kiirelt värve vahetades vilkuma. Antud vilkumine ongi Ozoboti värvikeel, mida Ozobot enda all oleva sensoriga loeb. Eduka laadimise korral vilgub Ozobot terve programmi laadimise ajal roheliselt ning pärast programmi Ozoboti laadimist on võimalik laetud programm käivitada Ozoboti lülitit topeltvajutusega. Ebaõnnestunud programmi laadimise korral muutub Ozoboti

värv punaseks. Sellisel juhul tuleb kontrollida, et kasutatav ekraan oleks puhas ja ekraani heledus piisavalt kõrge. Samuti on võimalik Ozoboti kalibreerida. Selleks tuleb Ozoboti lülitit kaks sekundit all hoida, mille peale Ozobot korraks valget värvi vilgutab. Seejärel tuleb Ozobot asetada kirjeldatud valgele alale ning oodata mõni sekund kuni Ozobot end kalibreerib.

1.3 Sarnased lahendused

Antud jaotus kirjeldab Ozobotile sarnast robotit nimega Edison, ning võrdleb neid kahte robotit omavahel. Jaotuse lõpus on ka lühidalt kirjeldatud Edisoni võimalusest programmeerida robotit programmeeriskeeles Python, mis on sarnane võimalus, mida loob käesoleva töö praktiline pool Ozobotile.

Sarnaselt Ozobotile on Edisoni puhul tegemist robotiga, mille eesmärk on lastele ja noortele programmeerimise õpetamine.



Joonis 4. Edison robot [13]

Jooniselt 4 on näha roboti ehitus ja info roboti peal olevate sensorite ning nuppude kohta. Antud jooniselt ja Edisoni tehniliste andmete lehelt [14] on näha, et roboti esiotsas on nii paremal kui vasakul pool üks punane LED, üks infrapuna LED ja üks valgussensor. Lisaks on Edisonil veel peal heliandur, millega on võimalik robotit plaksutamise teel juhtida ning mis on võimeline heli tagasi mängima. Seadmel on veel kolm nuppu, millega on võimalik alustada robotile juhiste ehk programmide laadimist ning alustada ja lõpetada käimasoleva programmi täitmist. Lisaks on roboti all valgussensor, mis koosneb fototransistori ja punase LED tule kombinatsioonist, mida joonisel 4 näha ei ole.

Kirjeldatud tehniliste andmete poolest erineb Edison Ozobotist kõige rohkem oma audio ja infrapuna sensoritega. Nimelt pole esimesel kahel Ozoboti versioonil antud tüüpi sensoreid ja kuigi kõige uuemal Ozobotil, Ozobot evol, on olemas nii audio kui ka infrapuna sensorid, siis Ozobot evo puhul on need sensorid ainult väljundiks. Edisoni puhul suudavad mõlemad sensorid vastu võtta sisendeid, mistõttu on võimalik Edisoni juhtida nii plaksutades kui ka näiteks tavalise televiisori puldiga [13].

Üheks suurimaks erinevuseks Edisoni ja Ozoboti vahel on Edisoni ühilduvus LEGOga. Nimelt on Edisoni kodulehe LEGO sektsioonis [15] kirjutatud, et robotil on nii peal, külgedel kui ka all LEGOga ühilduvad ühendused. Antud ühendusi on võimalik näha jooniselt 4 ning lubavad seadet kasutada ühe ilmselt kõige tuntuma ja levinuma mängutootega üle maailma. Samuti lubab LEGO ühilduvus füüsiliselt ühendada omavahel mitmeid Edison roboteid.

Sarnaselt Ozobotile on Edison võimeline sõitma mööda musta joont, kasutades enda all olevat valgussensorit. Küll aga erinevalt Ozobotist, ei ole Edison võimeline tuvastama enda all oleva sensoriga erinevaid värve. Nagu enne mainitud, koosneb Edisoni all olev sensor punasest LED tulest ja fototransistorist. Punane led tuli valgustab fototransistori all olevat ala ning fototransistor mõõdab pinna pealt tagasipeegelduva valguse tugevust. See tähendab, et Edison on võimeline ainult eristama üksteisest heledaid ja tumedaid pindu ning mitte erinevaid värve nagu Ozobot. Seetõttu ei ole võimalik Edisoni käitumist muuta joone järgimise ajal nii nagu see on võimalik Ozobotiga. Küll aga suudab Edison antud sensori abil tuvastada nelja erinevat vöötkoodi, millest üle sõites on võimalik Edison seadistada teatud operatsiooni režiimi. [13]

Edisoni programmeerimiseks on olemas Ozoblocklyga sarnane keskkond nimega EdWare [16]. Sarnaselt Ozoblocklyle on võimalik antud keskkonnas kasutajal valida vasakul olevast menüüst erinevaid käsklusi ning neid käsklusi lohistades on võimalik ehitada programme, mida on võimalik ka EdWare keskkonnast Edisoni peale laadida.

Nagu Ozobotil, käib Edisoni robotile programmi laadimine läbi roboti all oleva valgussensori, kuid erinevalt Ozobotist ei käi see programmi tähistavat värvikoodi ekraani kaudu vilgutades. Edisoni tehniliste andmete [14] järgi teisendatakse programm helifailiks ning Edisoni peale programmi laadimiseks on olemas eraldi kaabel nimega EdComm. Antud EdComm kaabel tuleb ühendada ühest otsast arvuti kõrvaklappi pessa ning teisest otsast Edisoni all oleva valgussensori külge. Programmi laadides liigub signaal mööda EdComm kaablit selle otsas oleva led tuleni, kus sisenev audio signaal teisendatakse valgussignaaliks, mida kaabli otsas olev led tuli roboti fototransistorile edasi annab.

Lisaks on võimalik Edison programmeerida programmeerimiskeeles Python. Selleks on olemas veebikeskkond EdPy [17]. EdPy näol on tegemist integreeritud arenduskeskkonnaga, kus on võimalik kirjutada programmeerimiskeele Python koodi ning kasutada Edisoni programmeerimiseks Pythoni moodulit Ed. EdPy keskkond lubab sarnaselt EdWare keskkonnale sealt otse programme Edisoni robotile laadida läbi eelnevalt kirjeldatud meetodi.

EdPy väljatuleku teadaande [18] järgi on antud keskkonna eesmärk õpetada läbi Edisoni õpilastele reaalset programmeerimiskeelt, mille oskusi saaksid õpilased edasises elus kasutada enda professionaalses karjääris. Lisaks on antud keskkonnas mitmed programmeerimist abistavad vahendid nagu automaatne koodi lõpetamine ja mitmed abistavad tekstid seletamaks, mida mingi funktsioon teeb.

Võrdluses Ozobotiga, puudus Ozobotil enne käesoleva töö valmimist Edisoni EdPy'le sarnane Pythonil põhinev programmeerimise viis.

2. Ozobotile Pythoni taolise keele loomine

Käesolevas peatükis kirjeldatakse bakalaureusetöö raames loodud keelt, millega on võimalik Ozoboti programmeerida. Antud peatükk annab ülevaate nii Ozoboti programmide ülesehitusest kui ka loomeprotsessis tehtud sammudest ja kasutatud võtetest ning tehnoloogiatest.

Antud peatükis kirjutatu kehtib Ozoboti bit versiooni puhul, kuna see oli ainukene roboti versioon, mis töö autoril töö käigus kasutada oli. Samuti on töö raames valminud keelt võimalik kasutada ainult Ozobot bit versiooniga ning valminud keele kohandamine ka Ozobot evo robotile oleks üks võimalike töö edasiarendusi.

Ozoboti programmide ülesehitus ei ole loojate poolt avalikustatud. Seega puudub avalik allikas, mis kirjeldaks Ozoboti programmi struktuuri ning oleks samal ajal täielik. Samuti ei õnnestunud luua antud teemal kontakti Ozoboti arendajatega, kelle käest oleks võimalik olnud roboti programmide struktuuri kohta pärida. Küll aga on olemas *github*i repositoorium [19], kus on üsna detailselt, kuid mitte täielikult kirjeldatud Ozoboti programmi struktuuri. Antud bakalaureusetöö toetub suuresti nimetatud repositooriumile ning kogu antud peatükk on kirjutatud antud allika põhjal.

Käesolevas peatükis on palju räägitud baitidest ning baitkoodist. Eristamaks baite tavalistest arvudest, on kõik antud peatükis olevad baitide väärtused standardsel kujul `0xXX`, kus XX tähistab baidi kahekohalist väärtust kuueteistkümnendsüsteemis ehk vahemikus 00 kuni ff.

Töö raames valminud kompilaatori ja baitkoodi värvikeelde tõlkimise lähtekood on saadaval lisas 1.

2.1 Ozoboti programmi ülesehitus

Ozoboti programm ei ole kõige madalamal tasemel midagi muud, kui lihtsalt üks baidijada, mis tõlgitakse Ozobotile arusaadavasse värvikeelde, mida on võimalik robotile läbi tema all olevate sensorite laadida.

Eelnevalt kirjeldatud repostiooriumi põhjal näeb Ozoboti programmi üldine struktuur baitkoodi tasemel välja järgmine: V, U, X, Y, Z, …, C. Antud struktuuri võib vaadelda kui programmi käskude ümber olevat ümbrikut, milles V ja U tähistavad arvatavasti versiooni numbrit, X, Y ja Z on programmi pikkusest sõltuvad muutujad, C on programmi kontrollsumma ning punktiiriga tähistatud ala on käskude jada, millele vastavalt Ozobot käitub. Versiooni numbrit tähistavad V ja U on alati väärtustega `0x01` ja `0x03`. Z puhul on tegemist programmi käskude jada pikkusega ning Y puhul on ka ilmselt tegemist programmi pikkust tähistava baidiga juhul, kui ühest baidist

ei peaks piisama programmi pikkuse määratlemisel. X on samuti seotud programmi pikkusega ning on alati 219 miinus programmi pikkus. Programmi viimane bait C on kontrollsumma, mille eesmärk on tuvastada programmi valesti lugemist. Kontrollsumma puhul on tegemist allakaoga (*underflow*) baidiga, mille algväärtus on null ja mille lõppväärtus arvutatakse järjest lahutades kõik programmi baidid alustades esimesest versiooni numbrist.

Kasutades aga eelnevalt kirjeldatud repositooriumis olevat programmi FlashReader, mis aitab Ozoboti värvikoodi baitkoodiks tõlkida, leidis käesoleva töö autor, et eelnevalt kirjeldatud Ozoboti programmi struktuur ei ole täielikult korrektne. Nimelt kuigi bait V on alati väärtusega `0x01` nagu kirjeldatud, siis sellele järgnev bait U ei tähista mitte osa versiooninumbrist vaid tähistab seda mitu korda on järgneval baidil X tekkinud allakadu. Iga kord kui baidil X tekib allakadu väheneb baidi U väärtus ühe võrra ning seetõttu ei ole pikemate programmide puhul antud väärtus alati `0x03` nagu eelpool kirjeldatud. Seega on Ozoboti programmi pikkusest sõltuvad neli baiti ning baidid U, X ja X, Y sisuliselt duplikeerivad üksteist ning nende ebakõla korral ka Ozobotile programmi laadimine ebaõnnestub. Arvatavasti tähistavad baitide U ja X algväärtused 3 ja 219 Ozoboti programmi maksimaalset pikkust ehk Ozobotil oleva kasutatava mälu suurust.

Ozoboti puhul on tegemist pinul põhineva masinaga, kus käsu argumendid salvestatakse enne käsu käivitamist pinusse. Näiteks on Ozoboti LED pirni muutmiseks käsk `0xb8`, mis võtab kolm argumenti punase, rohelise ja sinise värvi väärtusteks. Seega näiteks jada `0x7f 0x00 0x00 0xb8` muudaks Ozoboti LED pirni punaseks väärtustega 127 (`0x7f`), 0 (`0x00`), 0 (`0x00`).

Baidid väiksemad kui 128 ehk `0x00` kuni `0x7f` loetakse arvudeks ning lükatakse programmi käigus pinusse. Baidid, mis on suuremad või võrdsed kui 128 ehk vahemikus `0x80` kuni `0xff` on käsklused. Ozoboti programm toetab ka negatiivsed väärtuseid. Negatiivse väärtuse kasutamiseks saadetakse välja soovitud negatiivse arvu pöördväärtus, millest on lahutatud üks, ning seejärel pöördväärtust tähistav käsk `0x8b`. Näiteks kõige väiksem arv, mida on võimalik Ozoboti programmis kasutada on -128, mis baitkoodis oleks `0x7f 0x8b`. Seega on Ozoboti programmi kirjutamisel võimalik kasutada täisarvulisi väärtusi vahemikus -128 kuni 127. Ozobot toetab ainult täisarvulisi väärtusi.

Lisas 2 olev tabel kirjeldab lühidalt kõiki teadaolevaid baitkoodi käskluseid. Nagu eelnevalt mainitud ei ole arusaam Ozoboti programmi toimimisest täielik ning seega on antud tabelis kirjeid, mille kohta puudub täpsem arusaam. Küll aga on arusaam Ozoboti programmi struktuurist piisav, et käesolev bakalaureuse töö oleks võimalik ning terviklikult teostatav, sest puudu olev informatsioon teatavate baitkoodide täpse vajalikkuse ning eesmärgi kohta on kompilaatori ehitamisel võrdlemisi ebavajalik. Ozoboti programmi alamstruktuure käsitletakse peatükis 2.3.

2.2 Parser

Parseri puhul on tegemist kompileerimise protsessis kasutatava tarkvaraga, mis võtab sisendiks kirjutatud programmi tekstilisel kujul ning koostab antud sisendist programmi kujutava puukujulise andmestruktuuri vastavalt kasutatava keele süntaksi reeglitele. Sellist struktuuri nimetatakse süntaksipuuks ning abstraktseks süntaksipuuks nimetatakse süntaksipuud, mis ei sisalda midagi üleliigset vaid ainult informatsiooni, mis on programmi tähenduse edasi andmiseks hädavajalik.

Antud töö käigus loodud keel kasutab ära programmeerimiskeele Python parserit, kutsudes välja Python keele standardteegis oleva mooduli *ast* meetodit *parse*, millele antakse sisendiks teksti kujul kirjutatud programm ning, mis tagastab abstraktse süntaksipuu. Antud abstraktsest süntaksipuust loob järgmises peatükis kirjeldatud kompilaator Ozobotile arusaadava baitkoodi, mis pärast värvikeelde tõlkimist on terviklik Ozoboti programm, mida on võimalik Ozobotile laadida. Kuna loodud keel kasutab Python keele parserit, siis on ka loodud keele süntaksi reeglid üldjoontes samad, nagu programmeerimiskeeles Python. Loodud keelt võib vaadelda ka kui Python keele alamhulka, millele on lisatud teatud Ozoboti funktsionaalsust võimaldavad sisseehitatud funktsioonid.

2.3 Kompilaatori loomine

Kompilaatori puhul on tegemist tarkvara komponendiga, mis eelmises jaotuses kirjeldatud abstraktset süntaksipuud läbides tõlgib antud puud baitkoodiks, mis on kompileerimisprotsessi lõpptulemus. Antud peatükk kirjeldab loodud keele võimekust ning seda kuidas kõiki keele struktuure ja millisesse baitkoodi tõlgitakse.

Kompilaatori loomisel on Rasmus Saksi nõusolekul kasutatud tema *crumblepy* repositooriumit [20], kuna tegemist on samuti Pythoni taolise keele kompilaatoriga, mis on mõeldud Crumble kontrolleri jaoks. Antud repositooriumi põhjal on loodud käesoleva töö käigus valminud kompilaatori üldine struktuur, mida on vastavalt vajadusele kohandatud.

2.3.1 Liikumine ja LEDpirn

Ozoboti peal oleva LED pirni värvi muutmiseks on keelde sisseehitatud funktsioon *color*, mis võtab endale kolm parameetrit: punase, rohelise ja sinise värvi tugevuse märkimiseks. Näiteks, kui on soov muuta Ozoboti LED võimalikult eredaks punaseks, tuleks välja kutsuda funktsioon *color* väärtustega 127, 0, 0 ehk `color(127,0,0)`. Ozoboti LED tule värvi muutmise käsuks on baitkoodis bait 0xb8 ning seega kompileerimise protsessis kompileeritakse nimetatud *color*

funktsioon kujule `x y z 0xb8`, kus x, y ja z on vastavalt funktsioonile ette antud punase, rohelise ja sinise värvi väärtused baitkujul.

Ozoboti liikumise kontrollimiseks on keelde sisseehitatud kolm lihtsamat funktsiooni: *move*, *rotate* ja *wheels*.

Move funktsioon lubab Ozobotil sirgjooneliselt liikuda vastavalt etteantud kiirusele ja distantsile. Nimelt võtab *move* funktsioon parameetriteks kaks väärtust distantsi ja kiiruse määramiseks. Distants on määratud millimeetrites ja kiirus ühikuga millimeetrit sekundis. Seega näiteks funktsiooni *move* väljakutse kujul `move(50, 30)` liigutaks Ozoboti edasi 50mm kiirusega 30mm/s. Baitkoodi tasemel on ozoboti edasi liikumise käsu baidiks `0x9e` ning seega kirjeldatud *move* käsk tõlgitakse kompilaatori poolt baitkoodiks `x y 0x9e`, kus x ja y on vastavalt funktsioonile antud distantsi ja kiiruse parameetrid baitkujul. Üks või ka mõlemad funktsioonile *move* antavad parameetrid võivad olla negatiivsed. Sellisel juhul liigub Ozobot ette antud distantsi ja seda ette antud kiirusega, kuid lihtsalt teises suunas ehk tagurpidi. Negatiivsete väärtuste baitkoodi tõlkimisest on kirjutatud lähemalt järgmises peatükis 2.3.2

Rotate funktsiooniga saab Ozoboti pöörata ning sarnaselt *move* käsule võtab antud funktsioon kaks parameetrit: pöördenurga ja kiiruse. Pöördenurga ühikuks on kraadid ja kiiruse ühikuks jällegi mm/s. Baitkoodis on Ozoboti pööramise käsu baidiks `0x98`, ehk funktsiooni `rotate(x, y)` väljakutse tõlgitakse kompileerimisel baitkoodi `x y 0x98`, kus jällegi x ja y on vastavalt funktsioonile *rotate* antud nurga ja kiiruse parameetrid baitkujul. Ozobot pöörab ennast vastupäeva, kuid, sarnaselt *move* käsule, andes funktsioonile ükskõik kumma või mõlemad väärtused negatiivsed, pöörab Ozobot ennast teistpidi ehk päripäeva. Kuna maksimaalne Ozoboti poolt arusaadav väärtus on 127, siis ei ole võimalik ühe *rotate* funktsiooni väljakutsega teha rohkem kui 127 kraadine pööre. Näiteks 180 pöörde tegemiseks võib lihtsalt *rotate* funktsiooni välja kutsuda kaks korda järjest nii, et funktsiooni väljakutsete pöördenurgad annaksid tulemuseks 180 kraadi.

Viimaseks lihtsamaks Ozoboti liikumist muutvaks funktsiooniks on funktsioon *wheels*. Antud funktsiooniga on võimalik paika panna Ozoboti vasaku ja parema ratta kiirus. Nimelt võtab funktsioon parameetriteks kaks arvu: vasaku ja parema ratta kiiruse mm/s.
Baitkoodis on rataste kiiruse määramise käsuks bait `0x9f`. Seega tõlgitakse antud funktsioon kompilaatori poolt baitkoodi `x y 0x9f`, kus jällegi x ja y on funktsioonile antud parameetrid baitkujul. Sarnaselt eelnevale kahele funktsioonile võib üks või mõlemad funktsioonile antud parameetrid olla negatiivsed, mis juhul vastav Ozoboti ratas hakkab pöörlema teises suunas.

Lisaks eeltoodud kolmele funktsioonile on loodud keeles veel mõningad Ozoboti liikumisega seotud funktsionaalsused, mis on seotud joonte järgimisega. Antud võimalustest on kirjutatud lähemalt peatükis 2.3.6

2.3.2 Muutujad, matemaatilised tehted ja funktsioonid

Muutujate loomiseks ja väärtustamiseks on tulenevalt Pythoni parseri kasutamisest loodud keeles kasutusel standardne "=" operaator. Näiteks kui on soov luua muutuja nimega x, mille väärtus on 5, siis näeks see välja järgmine: `x = 5`. Samuti on loodud keeles toetatud ka Pythoni süntaks mitme muutuja korraga loomiseks või väärtustamiseks kujul `x = y = 5`. Antud kirjaviis loob kaks muutujat x ja y ning väärtustab need mõlemad väärtusega 5. Ozoboti programmid toetavad andmetüübina ainult täisarve, seega saab muutujaid väärtustada ainult täisarvudega.

Baitkoodi tasemel on muutujate väärtustamiseks olemas käsk `0x93`, mis võtab kaks parameetrit: muutuja väärtus ja aadress. Näiteks tõlgitakse avaldis `x = 5` baitkoodi `0x05 0x2a 0x93`, kus `0x05` on muutuja väärtus ning `0x2a` on muutuja aadress. Igal loodud muutujal on oma aadress ning kompilaator peab sõnastik tüüpi andmestruktuuris meeles kõikide loodud muutujate nimesid ja neile vastavaid aadresse, et muutujaid oleks võimalik hiljem kasutada või ümber väärtustada. Kompilaator hakkab muutujaid salvestama aadressitesse alates aadressist `0x2a` ning peab meeles esimest vaba aadressi, kuhu saab vajadusel uue muutuja salvestada. Näiteks, kui meil on järgnev kolme realine kood:

```
x = 5
y = 7
x = 3
```

Siis kompilaator tõlgib selle järgnevasse baitkoodi:

```
0x05 0x2a 0x93 0x07 0x2b 0x93 0x03 0x2a 0x93
```

Muutuja väärtuse lugemiseks on baitkoodis käsk `0x92`, mis võtab ainukeseks argumendiks muutuja aadressi. Näiteks järgnev kaherealine kood:

```
x = 30
move(50, x)
```

kompileeruks järgnevaks baitkoodiks:

```
0x1e 0x2a 0x93 0x32 0x2a 0x92 0x9e,
```

kus esimesed kolm baiti väärtustavad muutuja aadressil `0x2a` väärtusega `0x1e` (30), neljas bait `0x32` on käsu `0x9e` (*move*) esimene parameeter (50) ning viies ja kuues bait on muutuja väärtuse lugemine aadressilt `0x2a`, mis omakorda on *move* käsu (`0x9e`) teiseks parameetriks.

Ozobot toetab viite matemaatilist tehet ning kõik need on baitkoodis kujul `x y op`, kus x ja y on vastavalt tehte vasak ja parem argument ning *op* on kas liitmine (`0x85`), lahutamine (`0x86`), korrutamine (`0x87`), jagamine (`0x88`) või jäägi leidmine (`0x89`). Näiteks tehe `1 + 2` on baitkoodis `0x01 0x02 0x85`. Lisaks toetab Ozobot ka unaarset miinust kujul `0xXX 0x8b`, kus XX on soovitud negatiivse arvu absoluutväärtus, millest on lahutatud üks. Näiteks `-6` on baitkoodis `0x05 0x8b`.

Lisaks nimetatud tehetele on loodud keeles kaks sisseehitatud matemaatilist funktsiooni: *abs* ja *random*. Nendest esimene funktsioon on absoluutväärtuse arvutamiseks, mis võtab parameetriks väärtuse, mille absoluutväärtus on vaja teada saada ning on baitkoodis kujul `x 0xa8`, kus x on parameetriks antud väärtus. Funktsioon *random* on juhuarvu genereerimise funktsioon, mis võtab parameetriteks alumise ja ülemise piiri, mille vahel juhuarvu vaja genereerida on. Baitkoodi tõlgituna on funktsioon *random* kujul `x y 0x8c`, kus x on soovitud vahemiku ülempiir ja y alampiir.

2.3.3 Loogika ja tingimuslaused

Loogilised väärtused tõene ja väär on Ozoboti baitkoodis lihtsalt vastavalt baidid `0x01` ja `0x00`. Kuna loodud keel kasutab Pythoni parserit, siis on ka loodud keeles antud loogilised väärtused realiseeritud sõnadega *True* ja *False*.

Ozobot toetab kolme loogilist tehet: konjunktsioon, disjunktsioon ja eitus. Konjunktsioon ja disjunktsioon on baitkoodis samasugusel kujul nagu eelnevas peatükis kirjeldatud matemaatilised tehted, ehk kujul `x y op`, kus *op* on kas konjunktsioon (`0xa2`) või disjunktsioon (`0xa3`) ning eitus on kujul `x 0x8a`, kus x ja y vastavalt loogiliste tehete argumendid. Tulenevalt Pythoni parserist on loodud keeles antud loogilised tehted realiseeritud vastavalt sõnadega *and*, *or* ja *not*.

Lisaks toetab Ozobot kolme võrdlus operatsiooni, kujul `x y op`, kus x ja y on vastavast võrdluse vasak ja parem pool ning *op* on kas võrdumine (`0xa4`), suurem võrdne (`0x9c`) või suurem kui (`0x9d`). Võttes antud operatsioonidest eituse saame genereerida baitkoodi mitte võrdumise, väiksem kui ja väiksem võrdne kohta. Näiteks koodirida `1 <= 4` kompileerub baitkoodiks `0x01 0x04 0x9d 0x8a`

Kasutades lisas 2 olevaid lühendeid, on tingimuslausete üldkuju Ozoboti baitkoodis järgmine:

```
A if x 0x97 P jump y 0x97 Q
```

Kus A on tingimuslause tingimus ehk mingi tõeväärtus või mingit tõeväärtust omav avaldis, x tähistab seda mitu baiti tuleb hüpata juhul kui A on väär, P on tingimuslause n-ö tõese ehk *if* poole keha, y tähistab seda mitu baiti tuleb hüpata pärast P täitmist ning Q on tingimuslause n-ö väära ehk *else* poole keha.

Näiteks järgnev koodilõik:

```
if 1 <= 2:
    color(0, 0, 0)
else:
    color(127, 127, 127)
```

Kompileerub järgnevaks baidijadaks:

```
0x01 0x02 0x9d 0x8a 0x80 0x0a 0x97 0x00 0x00 0x00 0xb8 0xba 0x07 0x97 0x7f 0x7f
0x7f 0xb8
```

Asendades selguse mõttes antud jadas olevad käsud lisas 2 olevate lühenditega saame:

```
0x01 0x02 < not if 0x0a 0x97 0x0 0x0 0x0 led jump 0x07 0x97 0x7f 0x7f 0x7f led
```

Antud jadas on esimesed neli baiti tingimuslause tõeväärtust omav avaldis ehk avaldise `1 <= 2` baitkuju ning seejärel on käsklus *if* (`0x8a`), mis vastavalt eelnevale tõeväärtusele, kas hüppab tingimuslause *else* poole kehasse või täidab tingimuslause *if* poole keha. Pärast käsklust *if* on bait `0x0a`, mis tähistab tingimuslause *else* poole keha kaugust *if* käsklusest. Antud baidi järgi hüppab programmi täitmine väära tõeväärtuse puhul tingimuslause *else* poole kehasse ning jätab täitmata *if* poole keha. Antud näites on *else* poole keha kümne baidi kaugusel *if* baidist ning seega on *if* baidile järgnev bait väärtusega 0a ehk kümnendsüsteemis 10. Edasi tuleb bait `0x97`, mille täpne eesmärk ei ole teada, kuid tundub tähistavat tingimuslause kummagi poole keha algust. Edasi järgneb tingimuslause *if* poole keha, ehk antud näite puhul funktsiooni `color(0, 0, 0)` baitkuju. Seejärel on *if* poole kehale veel otsa pandud käsklus *jump* (`0xba`), mis võtab argumendiks järgneva baidi ehk hüppe pikkuse. *Jump* käsklus hüppab programmis edasi sellele järgneva baidi võrra ning see pannakse *if* poole kehale otsa, et hüpata programmi täitmisega üle *else* poole keha juhul, kui täitmisele läks just *if* poole keha. Seejärel on jälle bait `0x97` ning lõpetuseks tingimuslause *else* poole keha ehk antud näite puhul funktsiooni *`color(127, 127, 127)`* baitkuju.

Tingimuslauses võib *else* haru puududa, mis puhul ei lisa loodud kompilaator *if* poole kehale otsa ebavajaliku käsku *jump* koos baidiga `0x97`, kuigi Ozoblockly keskkond seda mingil põhjusel

teeb. Samuti võib tingimuslause konstruktsioone panna üksteise sisse, kuid loodud kompilaator ei toeta Pythoni süntaksit *elif*.

2.3.4 Tsüklid

Loodud keel toetab Pythoni süntaksist ainult *while*-tsükleid. Ozoboti programmi tsüklid ei ole midagi muud, kui ilma *else* keha pooleta tingimuslaused, mille *if* poole keha lõppu on lisatud hüpe tagasi tingimuslause algusesse. Loodud kompilaator optimeerib veel lõpmatut ehk `while True:` süntaksiga tsüklit, jättes tingimuslause üldse baitkoodist välja ning ka `while False:` süntaksiga tsüklit, jättes selle üldse kompileerimata, kuna väära tingimusega tsükli keha ei täideta mitte kunagi.

Ozoboti programmis hüppamiseks on käsklus *jump* (`0xba`), mis hüppab vastavalt järgnevale baidile. Negatiivse ehk tagasi hüppe puhul ei ole aga käsule *jump* järgnev mitte negatiivne arv, nagu kirjeldatud peatükis 2.3.2, vaid allavooga null bait, millest on maha lahutatud tagasi hüppe pikkus.

Näiteks järgnev tsükkel:

```
while True:
    wheels(30, 30)
```

Kompileerub järgnevaks baitkoodiks:

```
0x1e 0x1e 0x9f 0xba 0xfd
```

Nagu näha on tegemist lihtsalt funktsiooni *`wheels(30, 30)`* baitkujuga, millele on lisatud käsk *jump* parameetriga `0xfd`. Antud juhul on tsükli keha suurus kolm baiti ning fd on kümnendsüsteemis arv 253 ehk 256 - 3.

2.3.5 Programmi lõpetamine ja ootamine

Programmi töö lõpetamiseks on loodud keeles olemas funktsioon *terminate*. Antud funktsiooni baitkuju on `x 0xae`, kus x on funktsioonile antav parameeter märkimaks, millisesse opereerimise režiimi peaks Ozobot pärast programmi lõpetamist jääma. *Terminate* funktsioonile võib anda ühe kolmest parameetrist, mis on keelde sisseehitatud konstandid *OFF* (`0x00`), *FOLLOW* (`0x01`) ja *IDLE* (`0x02`). Andes antud funktsioonile parameetriks konstandi *OFF* lülitub Ozoboti pärast antud käsku täielikult välja. Konstandi *FOLLOW* puhul lülitub Ozobot pärast lõpetamise käsku joonte järgimise režiimi ning konstandi *IDLE* puhul lülitub Ozobot ooterežiimi. Tulenevalt funktsioonide struktuurist Ozoboti baitkoodis (vt ptk 2.3.7), peab iga programm lõppema käsuga

`0xae`. Seega juhul kui Ozoboti programmi lõppu jäetakse funktsioon *terminate* panemata, lisab loodud kompilaator selle automaatselt väärtusega *OFF*.

Ozoboti programmis pauside tegemiseks ja ootamiseks on keelde ehitatud funktsioon *wait*, mis võtab kaks parameetrit: pausi pikkuse sekundites ja sentisekundites (1 sentisekund on võrde 10 millisekundiga). Näiteks kutsudes välja funktsiooni *wait* parameetritega 2 ja 30 ootab Ozobot enne järgmise käsu täitmist 2,3 sekundit. Kirjeldatud funktsioon tõlgitakse baitkoodi erinevalt sõltuvalt funktsioonile antavatest parameetritest. Kui funktsioonile antav esimene ehk sekundeid tähistav parameeter on 0, siis kompileerub funktsioon lihtsalt baitkoodi `x 0x9b`, kus x on funktsioonile antud teine ehk sentisekundeid tähistav parameeter. Kui aga esimene funktsioonile antav parameeter ei ole 0, siis kompileerub antud funktsioon järgnevaks baitkoodiks:

```
X 0x64 0x9b 0x01 0x86 0x94 0x00 0x9d 0x8a 0x80 0xf8 0x97 0x96 Y 0x9b,
```

kus X ja Y tähistavad vastavalt funktsioonile antud esimest ja teist parameetrit. Asendades antud baitkoodis käsud lisas 2 olevate lühenditega saame:

```
X 0x64 wait 0x01 - dup 0x00 > not if 0xf8 0x97 drop Y wait
```

Antud baitkoodi puhul lükatakse kõigepealt väärtus X programmi pinusse, misjärel oodatakse sekund (`0x64 wait`). Seejärel lahutatakse pinu tipus olevast väärtusest 1 ja duplikeeritakse antud tulemus pinusse. Pärast seda kontrollitakse, kas antud väärtus on väiksem või võrde kui 0. Juhul, kui antud võrdus on tõene, siis täidetakse tingimuslause keha, kus kustutatakse käsuga *drop* pinu pealmine väärtus ning kutsutakse välja veel viimane *wait* käsklus funktsioonile antud teise parameetriga. Kui antud võrdus pole tõene, hüpatakse tagasi esimese *wait* käsuni ning korratakse antud protsessi seni, kuni antud võrdus saab tõeseks. Lühemalt öeldes on antud konstruktsioon lihtsalt tsükkel, mis täidab sekundi pikkuseid *wait* käske vastavalt nii palju, kui on funktsiooni esimene parameeter ning seejärel täidab lõpus veel viimase *wait* käsu funktsioonile antud teise parameetriga.

2.3.6 Joonte järgimine ja värvisensori lugemine

Loodud keelde on sisseehitatud kolm suuremat joonte järgimisega seotud funktsiooni. Kõik kolm funktsiooni on oma baitkujult keerukad ning täpne arusaam nende baitkoodi konstruktsioonidest puudub.

Esimene nendest kolmest funktsioonist on *follow_line_to_intersect_or_end*. Antud funktsiooni puhul liigub Ozobot mööda joont, kuni järgmise ristmikuni või joone lõpuni. Juhul, kui Ozobot pole antud funktsiooni väljakutse hetkel mõne joone peal ei tee Ozobot midagi, kuna ei suuda leida joont mida järgida. Antud funktsioon tõlgitakse järgnevaks baidijadaks:

```
0x01 0xa0 0xac 0xad 0x9a 0x10 0xa4 0x80 0xfd 0x00 0xa0 0x01 0x29 0x93
```

Teine joonte järgimisega seotud keelde ehitatud funktsioon on *move_straight_until_line*. Selle funktsiooni väljakutsel liigub Ozobot otse seni, kuni leiab mõne joone. Kirjeldatud funktsioon võtab parameetriks ühe väärtuse: liikumise kiiruse. Funktsioon tõlgitakse kompilaatori poole järgnevaks baidijadaks:

```
X 0x94 0x94 0x9f 0xac 0x08 0x92 0x80 0xfa 0x97 0x96 0x00 0x00 0x9f 0xc6 0x01
0xa0 0xac 0xad 0x9a 0x10 0xa4 0x80 0xfd 0x97 0x00 0xa0 0x01 0x29 0x93,
```

kus X on funktsioonile antud parameeter.

Viimane joonte järgimisega seotud keerukam funktsioon on *pick_direction*. Antud funktsioon võimaldab ristimikul olles valida edasi liikumise suuna ning eeldab, et funktsiooni väljakutse hetkel on Ozobot juba ristmikul. Juhul, kui Ozobot ei ole antud funktsiooni väljakutse hetkel ristmikul või kui ei ole võimalik valida etteantud suunda, siis hakkab Ozobot vilkuma kordamööda sinise ja punase värviga andmaks märku, et on tekkinud vigane olukord, millest Ozobot aru ei saa ning programmi töö lõpetatakse. Nagu mainitud tuleb kirjeldatud funktsioonile anda kaasa üks parameeter, mis tähistab valitavat suunda. Suuna jaoks on loodud keelde sisseehitatud neli konstanti *STRAIGHT* (`0x01`), *LEFT* (`0x02`), *BACK* (`0x08`) ja *RIGHT* (`0x04`) tähistamaks suundasid otse, vasakule, tagasi ja paremale. Antud funktsioonile antav parameeter peab olema üks neljast nimetatud konstandist. Funktsioon kompileerub järgnevaks baitkoodiks:

```
X 0x94 0x10 0x92 0x81 0x8a 0xb7 0x29 0x92 0x8a 0xb7 0x1f 0x93 0x01 0xa0 0xad
0x9a 0x14 0xa4 0x80 0xfd 0x00 0xa00 0x00 0x29 0x93,
```

kus X on funktsioonile antud parameeter.

Nagu eelnevalt mainitud, siis puudub täpne arusaam kuidas eelnevad kolm funktsiooni baitkoodi tasemel töötavad ning kirjeldatud konstruktsioonides on võimalik näha mitmeid lisas 2 kirjeldatud käske, mille eesmärk ja funktsionaalsus on teadmata. Sellegi poolest on Ozoboti võimalik antud käsklustega juhtida ning täpne arusaam kirjeldatud konstruktsioonidest pole Ozoboti programmerimise mõttes vajalik.

Joonte järgimisega on seotud veel mõned lihtsamad funktsioonid. Esimene nendest on *there_is_way*. Antud funktsiooniga saab kontrollida, kas ristmikul olles on olemas tee mingis suunas ning on põhiliselt mõeldud kasutamiseks tingimuslausetes. Antud funktsioon võtab parameetriks ühe neljast keelde ehitatud suuna konstandist nagu eelnevalt kirjeldatud funktsioon *pick_direction* ning kompileerub baitkoodiks `0x10 0x92 x 0x81`, kus x on funktsioonile antud

parameeter. Nagu näha antud baitkoodist on siin olemas käsk `0x92`, mis on juba eelnevalt kirjeldatud muutuja väärtuse lugemise käsk. Ilmselt hoiab Ozobot sisemiselt aadressil `0x10` mingit väärtust ning käsuga `0x81` võrdleb antud aadressilt loetud väärtust funktsioonile antud väärtusega ning tulemuseks on kas tõene või väär väärtus.

Lisaks on võimalik seadistada ja lugeda joone järgimise kiirust. Selleks on vastavalt funktsioonid *set_line_speed* ja *get_line_speed*. Ozobot hoiab sisemiselt joone järgimise kiiruse väärtust aadressil `0x18` ning seega on antud funktsioonide puhul tegemist antud aadressil oleva väärtuse muutmise ja lugemisega ehk kompileeruvad vastavalt peatükis 2.3.2 kirjutatule baitkoodideks `x 0x18 0x93` ja `0x18 0x92`, kus x on funktsioonile *set_line_speed* antud kiiruse parameeter. Ilma joone järgimise kiirust määramata tundub Ozoboti joone järgimise vaikimisi kiiruseks olema 30mm/s

Ozoboti programmi puhul on võimalik lugeda tema all olevat värvi. Selleks on loodud keeles olemas kaks erinevat funktsiooni, millest esimene on *get_surface_color*. Sarnaselt joone järgimise kiirusele on ka Ozoboti all oleva pinna värv salvestatud lihtsalt muutujana aadressile 0x0e. Seega kompileerub antud funktsioon lihtsalt baitkoodiks `0x0e 0x92`. Ozobot on võimeline eristama kaheksat erinevat värvi: must, valge, punane, sinine, roheline, kollane, tsüaan ja fuksiinpunane. Selleks, et nimetatud funktsiooni oleks võimalik kasutada näiteks tingimuslauses, on keelde ehitatud sisse antud värve tähistavad konstandid, mis on vastavalt: *BLACK*, *WHITE*, *RED*, *BLUE*, *GREEN*, *YELLOW*, *CYAN* ja *MAGENTA*.

Lisaks on veel funktsioon *get_intersect_or_line_end_color*. Sarnaselt eelnevalt kirjeldatud funktsioonile on tegemist Ozoboti all oleva pinna värvi lugemist teostava funktsiooniga, kuid on mõeldud spetsiaalselt just joone järgimisel ristmike ja joonte lõppude värvide lugemiseks. Sarnaselt eelmisele funktsioonile on ka siin tegemist lihtsalt aadressilt (`0x0f`) väärtuse lugemisega ning antud funktsioon kompileerub seega baitkoodiks `0x0f 0x92`. Antud funktsiooni tulemust võib võrrelda samade konstantidega nagu funktsiooni *get_surface_color* tulemust. Põhjus, miks Ozobotil on olemas kaks erinevat võimalust tegemaks pealtnäha sama asja on teadmata.

2.3.7 Funktsioonid

Loodud keel toetab lihtsamate funktsioonide loomist ja väljakutsumist. Kuna töö on juba üsna mahukas ja täielikult funktsioonide loomise võimekuse kompilaatorisse implementeerimine on keeruline ja mahukas, siis on loodud keeles tugi ainult ilma parameetrite ja tagastusväärtusteta funktsioonidele.

Ozoboti programmi baitkoodis on funktsioonid baidijadad, mis lisatakse Ozoboti programmi lõppu ning kutsutakse välja käsuga *call* (`0x90`). Antud käsk hüppab programmi täitmise järjega funktsiooni algusesse, vastavalt käsu järel olevale kahele baidile, mis tähistavad funktsiooni keha alguse aadressi ehk kaugust programmi algusest. Lisaks on iga funktsiooni keha lõpus tagastuskäsk `0x91`, mille peale programmi täitmisjärg naaseb pärast käsku *call* olevate aadressit tähistavate baitide taha. Antud tagastuskäsuga on tehniliselt võimalik ka Ozoboti programmis tagastada väärtusi, kuid nagu mainitud, siis loodud kompilaator seda funktsionaalsust ei toeta.

Tulenevalt jällegi Pythoni parseri kasutusest toimub loodud keeles funktsioonide defineerimine märksõnaga *def*. Lähtudes eelnevast kompileerub kood:

```
def f():
    wheels(x, x)
    wait(0, 100)

x = 50
f()
```

Järgnevaks baitkoodiks:

```
0x32 0x2a 0x93 0x90 0x0 0x8 0x0 0xae 0x2a 0x92 0x2a 0x92 0x9f 0x64 0x9b 0x91
```

Kasutades jällegi lisas 2 olevaid lühendeid saame:

```
0x32 0x2a set call 0x00 0x08 0x00 end 0x2a get 0x2a get wheels 0x64 wait ;
```

Antud baitkoodist on näha, et esmalt väärtustatakse muutuja ning seejärel kutsutakse välja funktsioon, mis on kaheksa baidi kaugusel programmi algusest. Antud näitest on näha ka see, et funktsioonidel puudub enda skoop. Seega on kõik loodud keeles olevad muutujad globaalsed, mis lubab funktsiooni sees kasutada muutujaid, mis on väärtustatud väljaspool funktsiooni ning samuti on võimalik väärtustada funktsiooni sees muutujaid, mida saab hiljem väljaspool funktsiooni kasutada, eeldusel muidugi, et funktsioon üldse välja kutsutakse. Lisaks on antud näitest ka näha, miks peab iga programmi lõpus olema käsklus *end*: juhul kui see puuduks, siis langeks programmi täitmine pärast oodatavat lõppu esimesse funktsiooni.

Funktsiooni definitsioonid jätab kompilaator meelde sõnastik tüüpi andmestruktuuri ning need kompileeritakse programmi lõppu pärast seda, kui ülejäänud programm on juba valmis kompileeritud. Samuti peab kompilaator meeles juba kompileeritud funktsioone ja nende alguse aadresse, selleks et ühtegi funktsiooni ei kompileeritaks mitut korda juhul kui ühe ja sama funktsiooni väljakutseid on programmis mitu tükki.

Funktsiooni väljakutsed kompileeritakse kahes osas, kuna esmasel kompileerimisel ei ole funktsiooni väljakutse hetkel veel teada, kui pikk on valmiv programm ning seega ei ole teada ka funktsiooni väljakutse aadress. Seega esmasel kompileerimisel lisatakse funktsiooni väljakutsel käsu *call* asemel sõnena funktsiooni nimi ning hetkel veel teadamata aadressi tähistamiseks kaks nullbaiti. Pärast esmast kompileerimist käib kompilaator veel valminud baitkoodi üle ning kohates sõne, asendatakse antud sõne käsuga *call* ning kontrollitakse, kas antud funktsioon on juba programmi lõppu kompileeritud. Juhul kui on, siis on kompilaatoril vastava funktsiooni aadress meeles ning eelnevalt kompileeritud nullbaidid asendatakse vastavalt funktsiooni aadressile. Juhul kui antud funktsioon ei ole veel kompileeritud ehk antud funktsiooni väljakutse on programmis esmakordne, kompileeritakse funktsioon programmi lõppu ning salvestatakse funktsiooni aadress ja asendatakse vastavalt eelnevalt kompileeritud nullbaidid.

2.4 Värvikeelde tõlkimine

Selleks, et koostatud Ozoboti programmi roboti peal kasutada, tuleb see tõlkida Ozobotile arusaadavasse värvikeelde, mida on võimalik Ozoboti all oleva sensori kaudu ekraanil värve vilgutades robotile laadida.

Ozoboti värvikeel koosneb järgnevatest värvidest (sulgudes värvile vastav lühend ja seitsmendsüsteemi arv):

- Must (K, 0);
- Punane (R, 1);
- Roheline (G, 2);
- Kollane (Y, 3);
- Sinine (B, 4);
- Fuksiinpunane (M, 5);
- Tsüaan (C, 6).

Lisaks nendele seitsmele põhivärvile on eritähendusega kasutusel ka valge (W).

Valmis oleva baitkoodi tõlkimiseks on vaja kõik baidid tõlkida kolmekohalisteks seitsmendsüsteemi arvudeks. Pärast seda asendatakse lihtsalt iga arv vastava värviga. Näiteks bait 0x08 on seitsmendsüsteemis arv 011, mis teisendub värvikoodiks *KRR*. Selliselt teisendatakse kõik baidijada baidid ning tulemuseks on programmi kujutav värvijada.

Pärast kirjeldatud värvijada loomist tuleb tekkinud jada uuest läbi käia ning asendada kõik värvide järjestikused kordused valge värviga. Nimelt tajub Ozobot värvide muutumist ning seega ei saa lõplikus jadas olla sama värvi kaks korda järjest. Seetõttu kasutatakse valget värvi korduste märkimiseks. Näiteks kui võtta värvijada *KKKKRR*, siis sellest saaks *KWKWRW*.

Iga Ozoboti programmi kujutava värvijada ees ja taga on veel vastavalt värvijadad *CRY CYM CRW* ja *CMW*, mis on n-ö staatilised raamid iga Ozoboti programmi ümber. Pärast värvijadale antud raamide lisamist on Ozoboti programm lõplikult värvikeelde tõlgitud ning robotile antud programmi laadimiseks tuleb valminud värvijada vilgutada arvutiekraanilt Ozoboti all olevatele värvisensoritele sagedusega 20 hertsi.

2.5 Thonny pistikprogramm

Thonny on käesoleva töö juhendaja, Aivar Annamaa, poolt programmeerimiskeele Python jaoks loodud integreeritud arenduskeskkond. Thonny kodulehe [21] järgi on tegemist algajatele suunatud tarkvaraga, mille eesmärk on võimalikult palju abistada algajaid programmeerijaid programmeerimise ja keele Python õppimisel, tehes näiteks vajaliku tarkvara paigaldamise võimalikult lihtsaks ning omades mitmeid funktsionaalsusi, mis võimaldavad algajatel nii programmi tööst kui ka tekkinud vigadest kergesti aru saada.

Käesoleva töö üheks tulemiks on ka Thonny pistikprogramm (lähtekood lisas 3), mis lubab luua eelnevalt kirjeldatud Pythoni taolises keeles Ozoboti programme ning neid seejärel kergesti Ozoboti peale laadida. Pistikprogrammi Thonny'sse lisamiseks tuleb Thonny menüüs "Tools" valida valik "Manage plug-ins..." ning seejärel kirjutada tekkinud otsingukasti "thonny-ozobot". Thonny peaks selle peale antud pistikprogrammi üles leidma ning vajutades nuppu "Install" selle Thonny'sse lisama. Pärast Thonny taaskäivitamist peaks "Tools" menüüsse tekkima valik "Create ozobot color sequence", millele vajutades kompileeritakse ja tõlgitakse värvikeelde Thonny tekstiaknas olev kood ning õnnestumise korral tekib ekraanile valge alaga ja nupuga "Load" aken. Vajutades ühekordselt Ozoboti lülitit ning asetades Ozoboti vastu ekraanil olevat nimetatud valget ala ja seejärel vajutades nuppu "Load" ongi võimalik Ozoboti peale laadida kirjutatud programm.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua Ozobot robotile programmeerimiskeele Python sarnane keel, millega oleks võimalik Ozoboti programmeerida ning, mida saaks kasutada õpivahendina lastele ja noortele programmeerimise õpetamiseks läbi Ozobot roboti.

Töö põhiliseks tulemiks on valminud kompilaator, mis lubab Pythoni sarnase keelega programmeerida robotit Ozobot bit. Loodud keel toetab peaaegu kogu Ozoblockly funktsionaalsust s.t lubab tekstipõhiselt luua samaväärseid programme kui Ozoblockly keskkond. Antud kompilaatori mugavamaks kasutamiseks valmis ka Thonny pistikprogramm, millega on lihtne programme luua ning neid ka Ozoboti peale laadida. Lisaks seisneb käesoleva töö väärtus ka Ozoboti programmide struktuuri kirjelduses ja eestikeelse dokumentatsiooni loomises.

Kuna käesolev töö keskendus Ozobot bit robotile, siis oleks ühe edasiarendusena võimalik antud töös valminud keelele juurde lisada Ozobot evo programmeerimise võimekus, koos antud robotiga lisanduvate funktsionaalsustega. Samuti oleks võimalik tegeleda Ozoboti programmi edasise pöördkonstrueerimisega, kuna nagu töös mainitud ei ole Ozoboti prorgammi toimimise kõik elemendid ja konstruktsioonid täielikult teada ning arusaadavad.

Viidatud kirjandus

[1] Tartu Ülikool, Programmeerimine lastele ja noortele. https://courses.cs.ut.ee/kids (22.03.2017)

[2] Ozobot http://ozobot.com/ (04.04.2017)

[3] Ozobot bit http://ozobot.com/products/ozobot-bit (06.04.2017)

[4] Ozobot, Frequently asked questions http://ozobot.com/support/faq (06.04.2017)

[5] Ozobot evo http://ozobot.com/products/ozobot-evo (04.04.2017)

[6] Ozobot, OzoCodes http://play.ozobot.com/print/guides/ozobot-ozocodes-reference.pdf (08.04.2017)

[7] Ozobot, print games http://ozobot.com/play/print-games (08.04.2017)

[8] Ozobot, draw games http://ozobot.com/play/drawing-games (08.04.2017)

[9] Ozobot, web games http://ozobot.com/play/web-games (09.04.2017)

[10] Google Play, Ozobot https://play.google.com/store/apps/details?id=com.evollve.ozobot&rdid=com.evollve.ozobot (09.04.2017)

[11] Google Play, OzoGroove https://play.google.com/store/apps/details?id=com.evollve.ozogroove (09.04.2017)

[12] Ozoblockly http://ozoblockly.com (09.04.2017)

[13] Edison, Your EdVenture into Robotics 10 Lesson plans https://meetedison.com/wp-content/uploads/2015/04/Your-EdVenture-into-Robotics-10-Lesson-Plans.pdf (04.05.2017)

[14] Edison, Technical https://meetedison.com/technical/ (04.05.2017)

[15] Edison, Edison Robot works with LEGO bricks
https://meetedison.com/works-with-lego-bricks-robotics/ (04.05.2017)

[16] EdWare http://edwareapp.com (04.05.2017)

[17] EdPy http://www.edpyapp.com (04.05.2017)

[18] Meet EdPy – Python programming for Edison
https://meetedison.com/meet-edpy-Python-programming-edison/ (04.05.2017)

[19] Reverse engineering the ozobot https://github.com/AshleyF/ozobot (02.05.2017)

[20] Raskmus Saks, crumblepy https://github.com/rasmussaks/crumblepy (26.04.2017)

[21] Thonny http://thonny.org/ (07.05.2017)

Lisad

1. Pythonitaolise keele kompilaatori ja värvikeelde tõlkimise lähtekood

Käesoleva töö raames loodud keele kompilaatori ja värvikeelde tõlkimise lähtekood, mis on kättesaadaval ka aadressil https://github.com/Kaarel94/Ozobot-Python. Sisaldab lähtekoodi kaustas *ozopython*.

2. Ozoboti baitkoodid ja nende kirjeldused

Tabel 1. Ozoboti baitkoodid ja nende kirjeldused [19]

Bait	Märk / Lühend	Lühiselgitus
0x80	*if*	Nn. “Kui .. siis” operaator. Kasutatakse tingimuslausete ehitamisel.
0x83	~	bitieitus
0x85	+	liitmismärk
0x86	-	lahutamismärk
0x87	*	korrutusmärk
0x88	/	jagamismärk
0x89	*mod*	Jäägiga jagamise operaator
0x8a	*not*	Loogiline eitus
0x8b	*neg*	Pöördarvu operaator
0x8c	*rand*	Juhuarvu loomise käsk
0x90	*call*	Funktsiooni väljakutse käsk
0x91	;	Funktsiooni tagastamise käsk
0x92	*get*	Muutujate ja Ozoboti sensorite lugemise käsk
0x93	*set*	Muutujate väärtustamise käsk
0x94	*dup*	Pinu pealmise elemendi duplikeerimise käsk

0x96	*drop*	Pinu pealmise elemendi eemaldamise käsk
0x97	?	Teadmata eesmärk. Kasutatakse tsüklite konstrueerimisel
0x98	*turn*	Pööramise käsk
0x9a	?	Teadmata eesmärk
0x9b	*wait*	Ootamise käsk
0x9c	>=	Suurem võrdne
0x9d	>	Suurem kui
0x9e	*move*	Liikumise käsk
0x9f	*wheels*	Rataste kiiruse seadmise käsk
0xa0	?	Teadmata eesmärk
0xa2	*and*	Loogiline ja ehk konjunktsioon
0xa3	*or*	Loogiline või ehk disjunktsioon
0xa4	=	võrdumine
0xa5	*pick*	Pinust väärtuse valimise käsk
0xa6	*put*	Pinusse väärtuse panemise käsk
0xa7	*pop*	Pinust pealmise elementi võtmise käsk
0xa8	abs	Absoluutväärtuse käsk
0xa9	?	Teadmata eesmärk

`0xaa`	?	Teadmata eesmärk
`0xac`	?	Teadmata eesmärk
`0xad`	?	Teadmata eesmärk
`0xae`	*end*	Programmi lõpetamise käsk
`0xb8`	*led*	Led pirni värvi muutmise käsk
`0xba`	*jump*	Programmi täitmise järje hüppamise käsk
`0xc6`	?	Teadmata eesmärk

3. Thonny pistikprogramm

Käesoleva töö raames loodud Thonny pistikprogramm, mis on kättesaadaval ka aadressil https://bitbucket.org/kaarel94/thonny-ozobot. Sisaldab lähtekoodina faili *__init__.py* kaustas *thonnycontrib/ozobot*.

4. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Kaarel Maidre**,

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

Pythonitaolise keele kompilaator Ozobotile,

mille juhendaja on Aivar Annama,

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**