TARTU ÜLIKOOL

Arvutiteaduse instituut

Informaatika õppekava

Kauri Raba

Arenduskeskkonnale Thonny micro:bit toe lisamine

Bakalaureusetöö (9 EAP)

Juhendaja: Aivar Annamaa

Tartu 2017

Arenduskeskkonnale Thonny micro:bit toe lisamine

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärgiks on luua Thonny integreeritud programmeerimiskeskkonna jaoks pistikprogramm, mille abil on võimalik programmeerida miniarvutit BBC micro:bit. BBC micro:bit on programmeeritav miniarvuti, mis on eelkõige loodud lastele digitaalse maailma avastamiseks ning neis antud valdkonna vastu huvi tekitamiseks. Pistikprogramm võimaldab kasutaja poolt kirjutatud programmi miniarvutil käivitada ilma seda miniarvuti välk-püsimällu salvestamata, mis on küllaltki aeganõudev protsess.

Võtmesõnad: Thonny, arenduskeskkond, pistikprogramm, BBC micro:bit, programmeerimine

CERCS: P175 - Informaatika, süsteemiteooria

Adding micro:bit support for Thonny IDE

Abstract:

The goal of this Thesis is to create a plug-in for Thonny integrated development environment which allows to program BBC micro:bit. BBC micro:bit is a programmable minicomputer primarily created for children to introduce them digital world and to make them interested in that field. This plug-in allows users to execute programs on minicomputer without flashing, which is time-consuming process.

Keywords: Thonny, IDE, plug-in, BBC micro:bit, programming

CERCS: P175 - Informatics, systems theory

Sisukord

Sissejuhatus

Programmeerimine on tänapäeva kiirelt arenevas ühiskonnas väga tähtsal kohal. Seda tõestavad nii Eestis kui ka mujal maailmas näiteks erinevad töökuulutused, kus otsitakse järjest rohkem IT-valdkonna spetsialiste.

BBC algatusel hakati 2012. aastal kavandama uut lastele mõeldud miniarvutit BBC micro:bit, mille abil loodetakse noortes tekitada huvi tehnoloogia vastu ning üles kasvatada uus põlvkond spetsialiste, kes hakkavad tulevikus vähendama tehnoloogiasektoris valitsevat spetsialistide puudust [1].

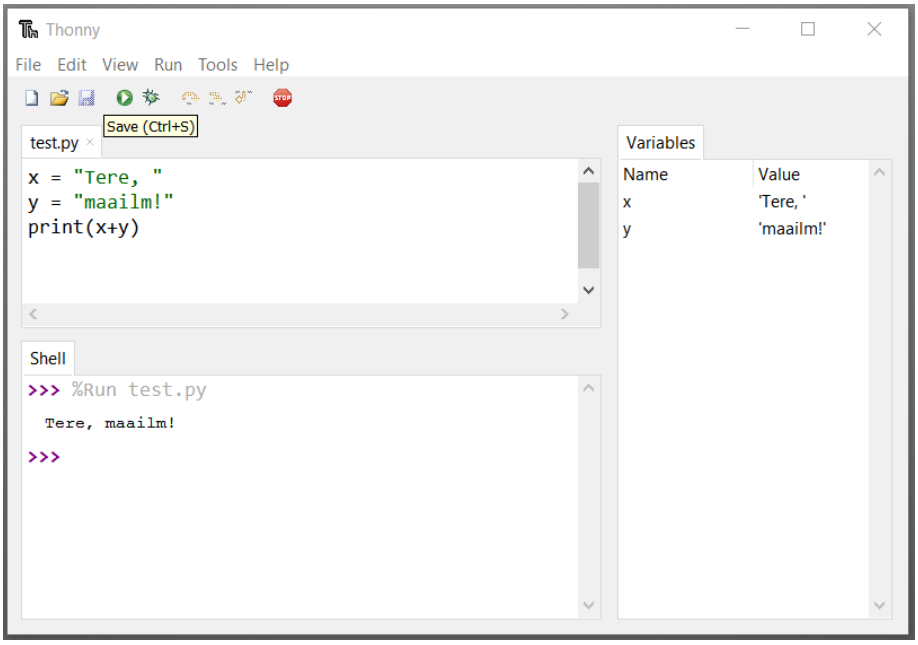
Thonny on integreeritud programmeerimiskeskkond (ingl k *IDE – Integrated Development Environment*), mida kasutatakse peamiselt programmeerimise õppimiseks ning mis on Tartu Ülikoolis kasutusel programmeerimise algkursustel [2].

Käesoleva bakalaureusetöö eesmärgiks on luua Thonny programmeerimiskeskkonnale pistikprogramm, mis lisab sellele funktsionaalsuse micro:biti programmeerimiseks. Töö teema valik tulenes bakalaureusetöö autori huvist miniarvutite vastu ning soovist luua midagi, mis võiks kellelegi programmeerimise õppimisel kasulik olla.

Bakalaureusetöö esimeses peatükis antakse lühiülevaade Thonny integreeritud programmeerimiskeskkonnast. Teises peatükis tutvustatakse miniarvutit micro:bit. Kolmas peatükk kirjeldab lühidalt olemasolevat lahendust Mu, mis oli autorile pistikprogrammi loomisel eeskujuks. Neljandas ja viiendas peatükis antakse ülevaade bakalaureusetöö raames valminud pistikprogrammist ning selle valmimise protsessist. Kuuendas peatükis antakse soovitusi edasiseks pistikprogrammi arendamiseks.

1. Integreeritud programmeerimiskeskkond Thonny

Thonny (vt Joonis 1) on Tartu Ülikooli arvutiteaduse instituudis loodud integreeritud programmeerimiskeskkond, mis on eelkõige mõeldud algajatele programmeerimise õppimiseks programmeerimiskeeles Python [3]. See on avatud lähtekoodiga ning kõigile vabalt kättesaadav [4,5].

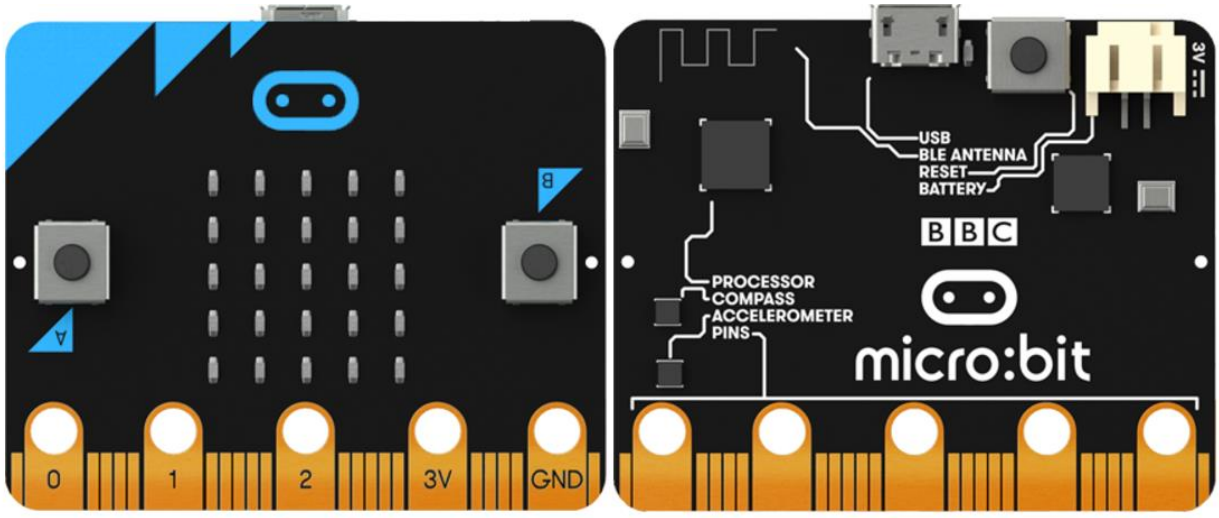


Joonis 1. Ekraanitõmmis Thonny kasutajaliidesest

Thonny keskendub lihtsusele ning selle kasutamiseks ei ole vaja Pythonit eraldi installeerida, kuna see on Thonnysse sisseehitatud [5]. Aivar Annamaa [3] on kirjutanud, et Thonny jaoks on alles jäetud ainult kõige olulisem funktsionaalsus ning lisaks programmide tavalisele käivitamisele on olemas ka erinevaid vahendeid kasutaja poolt kirjutatud programmide töökäigu visualiseerimiseks samm-sammulisel käivitamisel. Nendeks on näiteks muutujate väärtusete kuvamine ja funktsioonide väljakutsete visualiseerimine igal sammul. Üks Thonny eripäradest see, et Pythoni interaktiivse käsurea vaade (ingl k *shell*) on alati avatud, ning see julgustab algajaid programmeerijaid sellel väiksemaid katsetusi läbi viima. Thonny toetab ka kolmandate osapoolte poolt loodud pistikprogramme, mis teeb antud bakalaureusetöö teostamise lihtsamaks.

2. Miniarvuti BBC micro:bit

BBC [1] andmetel alustasid nad koos teiste koostööpartneritega BBC micro:biti (edaspidi micro:bit) kavandamist 2012. aastal. Selle miniarvuti (vt Joonis 2) eesmärgiks on tulevikus leevendada Suurbritannias valitsevat tehnoloogiavaldkonna spetsialistide puudust. Micro:bit on eelkõige mõeldud lastele, kuid on vägagi sobilik ka vanematele huvilistele, et tutvustada neile programmeerimist ning tekitada neis huvi teaduse ja tehnoloogia vastu. Suurbritannias jagati 2016. aastal 11–12-aastastele koolilastele ja nende õpetajatele tasuta ligi üks miljon micro:biti.



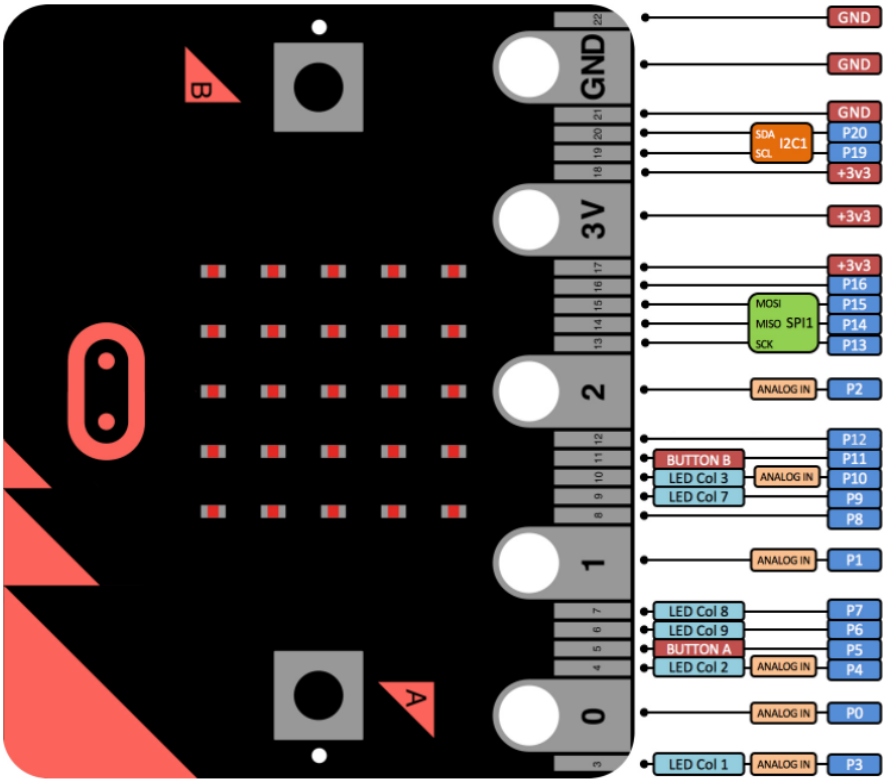
Joonis 2. Micro:biti esikülg (vasakul) ja tagakülg [6]

2.1 Riistvara

Micro:bitil on 16 MHz taktsagedusega protsessor, 16 kB suvapöördusmälu ning 256 kB välkmälu programmide salvestamiseks [7]. Programmeerimise ja digitaalse maailma avastamise huvitavamaks tegemiseks on seadmele lisatud veel erinevaid riistvaralisi komponente, mida micro:biti programmeerijad saavad oma programmides lihtsasti kasutada. Nendeks on näiteks erinevad metallkontaktid, valgusdioodid, nupud, kompass, kiirendusandur. Pikemalt on igast komponendist juttu järgmistes alapeatükkides.

2.1.1 Ühenduse loomise võimalused teiste seadmetega

Micro:biti alumisel serval on kokku 25 metallkontakti. Viis neist on suuremad ja ülejäänud 20 väiksemad. Suuremad on nimetatud järgnevalt: 0, 1, 2, 3V ja GND. Micro:biti metallkontakte tutvustava veebilehe [8] põhjal on esimesed kolm kontakti neist üldotstarbelised sisend-väljundid, mille abil saab lugeda või saata informatsiooni teistele micro:bitiga ühendatud seadmetele. 3V kontakti kasutatakse kas elektrivoolu edasiandmiseks teistele micro:bitiga ühendatud seadmetele või micro:biti enda vooluringi ühendamiseks, kui puudub muu vooluallikas. GND on maandus, mille kaudu lõpetatakse vooluring. Ülejäänud 20-st väiksemast kontaktist täidavad mõned mitut ülesannet. Näiteks on mõned kontaktid otseühenduses micro:biti sisemiste komponentidega ning samas on ka sisend-väljundid (vt Joonis 3).

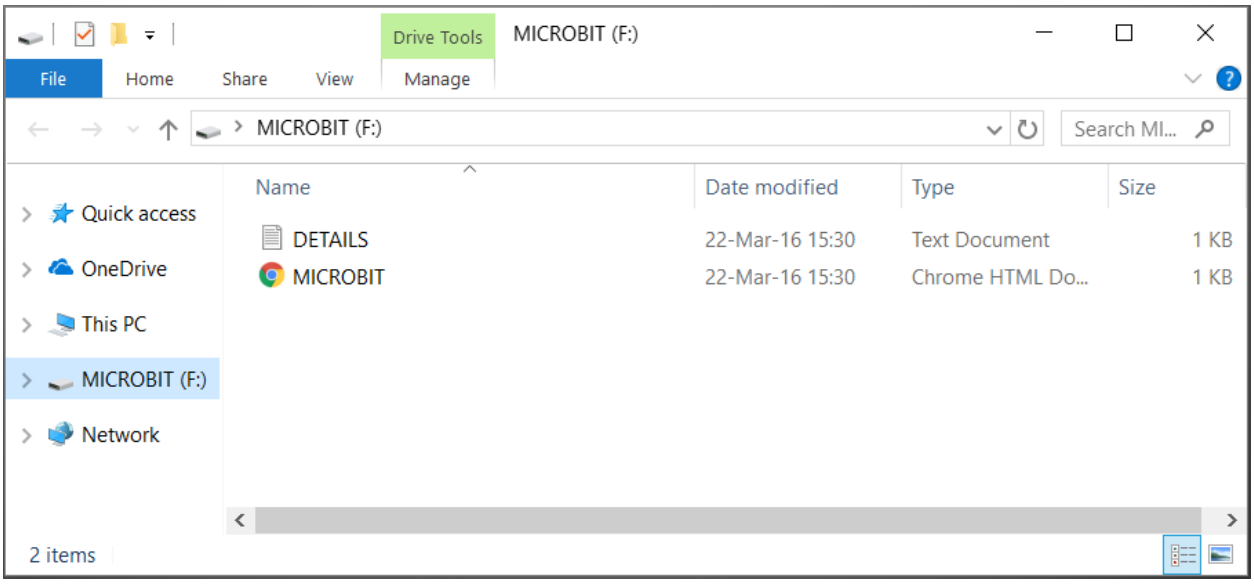


Joonis 3. Micro:biti metallkontaktid [8]

Micro:biti tagaküljel asub Micro USB pesa, mille kaudu saab micro:biti tavaarvutiga ühendada. Sellel on kolm peamist ülesannet: informatsiooni vahetamine läbi jadasiini, programmide

salvestamine micro:biti välk-püsimällu (ingl k *flashing*) [9] ning micro:biti ühendamine vooluringi [10].

Programmide salvestamise lihtsustamiseks miniarvuti välk-püsimällu on tehtud nii, et kui micro:bit USB kaabliga tavaarvuti külge ühendada, siis miniarvuti käitub välise andmekandjana, nagu näiteks USB mälupulk (vt Joonis 4). Tegelikult on tegemist virtuaalse andmekandjaga, mida päriselt ei eksisteeri ning micro:biti programmeerimiseks piisab vaid sellest, kui kopeerida .hex laiendiga fail, mis sisaldab kuueteistkümnendsüsteemi tõlgitud programmi, selle virtuaalse andmekandja peale [9]. Sellist .hex laiendiga faili on võimalik luua erinevates veebipõhistes micro:biti programmeerimiskeskkondades.



Joonis 4. Micro:biti tuvastamine välise andmekandjana

Micro:bit toetab ka juhtmevaba Bluetooth ühendust, mille abil on võimalik saata signaale või neid vastu võtta teistelt seadmetelt, mis samuti Bluetooth ühendust toetavad [10].

2.1.2 Valgusdioodid

Micro:biti esiküljelt leiab valgusdioodidest (ingl k *LED – light-emitting diode*) moodustatud 5 x 5 maatriksi, mis on võimeline kuvama tähti, numbreid ja erinevaid kujutisi ning tähthaaval ka sõnu

ja lauseid [11]. Igat valgusdioodi saab kontrollida ka individuaalselt sisse- või väljalülitamisega ja nende ereduse muutmisega [10].

2.1.3 Nupud

Micro:biti riistvara tutvustava veebilehe põhjal [10] asuvad miniarvuti esiküljel kaks programmeeritavat nuppu (A ja B), mille vajutamist ja vajutuse vabastamist micro:bit tuvastada suudab. Tagaküljel on veel ka taaskäivitamise nupp, mis on süsteemi nupp ja mis ei ole programmeeritav. Sellele vajutades taasalustatakse micro:biti välk-püsimälus oleva programmi täitmist algusest. Taaskäivitamise nupu abil on võimalik micro:biti ka hooldusolekusse viia.

2.1.4 Kompass

Kompassi abil suudab micro:bit tuvastada enda liikumise suunda ning seda infot saab kasutada programmides või saata edasi teistele micro:bitiga ühenduses olevatele seadmetele [10].

2.1.5 Kiirendusandur

Kiirendusanduri abiga saab micro:bit tuvastada enda kiiruse muutuseid ruumis ning selle abil suudetakse tuvastada ka mõningaid tegevusi, nagu näiteks micro:biti raputamist, kallutamist ja vabalangemist [10].

3. Micro:biti programmeerimine

Micro:biti programmeerimiseks on erinevaid võimalusi ja keeli. Nendeks on näiteks Microsoft Touch Developi menüüpõhine programmide koostamine, plokkprogrammeerimise võimalus ning programmide koostamine programmeerimiskeeltes Javascript ja MicroPython [12]. Viimasele programmeerimiskeelele toetub antud lõputöö praktiline osa, kuna Thonny on mõeldud Pythoni programmide kirjutamiseks.

3.1 MicroPython

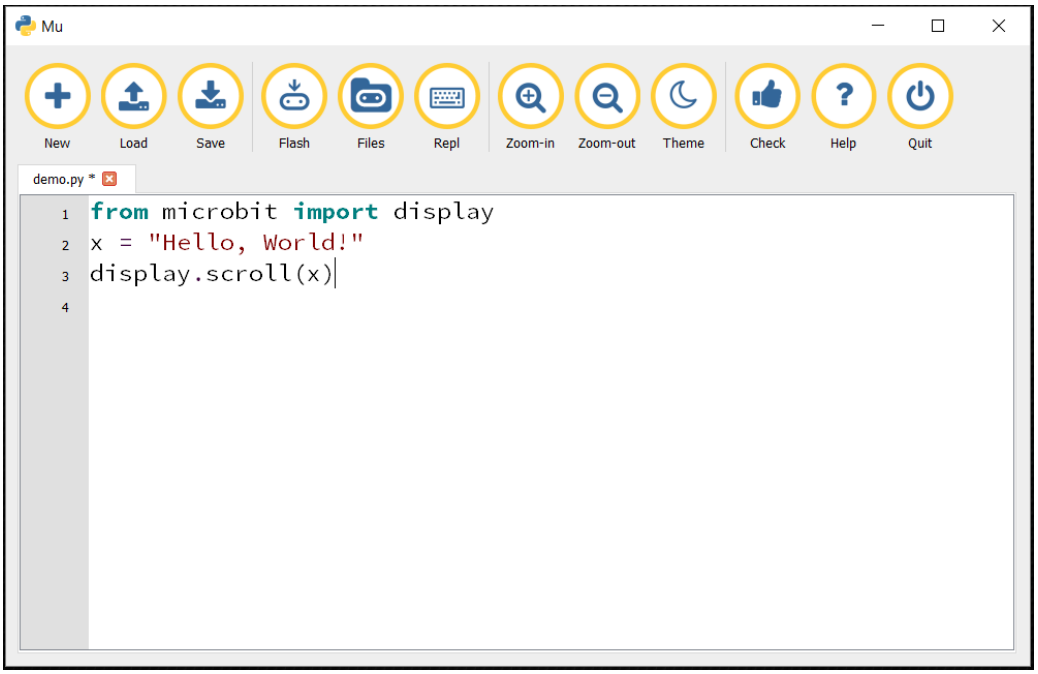
MicroPython on avatud lähtekoodiga programmeerimiskeel, mis on sarnane Python 3 programmeerimiskeelele, kuna see sisaldab endas alamhulka Python 3 standardteegist ning on sarnase süntaksiga [13]. MicroPython on loodud ja optimeeritud spetsiaalselt mikrokontrollerite ja miniarvutite jaoks ning selle standardteegist on välja jäetud komponendid, mida ei saa miniarvutitel kasutada näiteks rohke mälu hõivamise või mõne riistvaralise komponendi puudumise tõttu [14].

Micro:biti arendajate seltsi [15] põhjal luuakse Micro:biti programmeerimiseks masinkoodi sisaldav .hex laiendiga fail, mis sisaldab kompileeritud MicroPythoni keelt. Kasutaja poolt kirjutatud programm kompileeritakse ja lisatakse eelnevalt mainitud failile lisaks ning seejärel salvestatakse kogu masinkood micro:biti välk-püsimällu. Esmalt käivitatakse micro:bitil kasutaja poolt kirjutatud programm. Kui programm lõpetab töö, siis käivitub MicroPythoni interaktiivne käsurida (ingl k *REPL – Read-Eval-Print-Loop*) mille kaudu on võimalik MicroPythoni interpretaatoris koodi käivitada. Käsurida saab kasutada ka ilma kasutajapoolse programmi lisamiseta, sel juhul käivitub interaktiivne käsurida kohe peale micro:biti käivitamist. Interaktiivse käsurea kasutamiseks on vaja luua jadaühendus kasutaja arvuti ning micro:biti vahel.

3.2 Programmeerimiskeskkond Mu

Mu on avatud lähtekoodiga ning platvormiülene programmeerimiskeskkond, mis on mõeldud MicroPythoni programmide kirjutamiseks micro:bitile [16]. Mu keskendub lihtsusele, kuna see on loodud eelkõige algajatele programmeerimise õppimiseks ning lihtsuse tagamiseks on keskkonnale lisatud ainult kõige tähtsam funktsionaalsus koos intuitiivse kasutajaliidesega [17].

Joonisel 5 on näha, et Mu programmeerimiskeskkonnas on micro:biti programmeerimisega seotud funktsionaalsusi kolm. Esmalt saab koodiredaktoris olevat koodi micro:biti välk-püsimällu salvestada ühe nupuvajutusega. Teiseks on olemas micro:biti failisüsteemi vaade, mille abil saab miniarvuti failisüsteemi lisada faile ning neid sealt kustutada. Kolmandaks on olemas ka interaktiivse käsurea kasutamise võimalus.



Joonis 5. Ekraanitõmmis Mu kasutajaliidesest

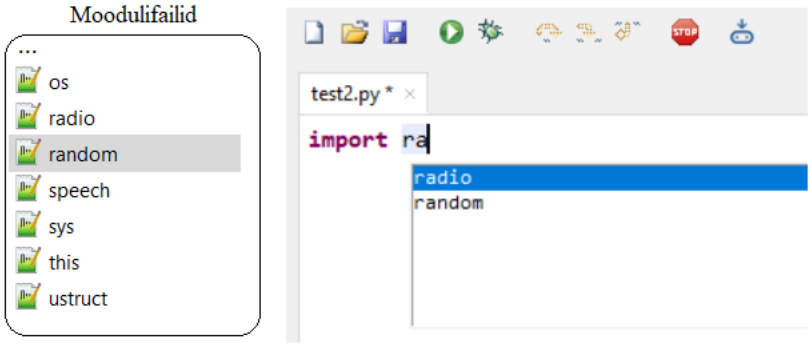
Antud programmeerimiskeskkond on käesoleva bakalaureusetöö autorile Thonny pistikprogrammi loomisel peamiseks eeskujuks.

4. Thonny pistikprogrammi ülevaade

Käesolev peatükk annab ülevaate Thonny pistikprogrammi funktsionaalsusest ning kirjeldab lühidalt, millised protsessid pistikprogrammi kasutamisel süsteemisiseselt toimuvad. Lõpus tuuakse välja ka pistikprogrammi eelised ning puudused programmeerimiskeskkonna Mu ees. Pistikprogrammi installeerimis- ning kasutusjuhend on lisatud bakalaureusetöö lisadesse.

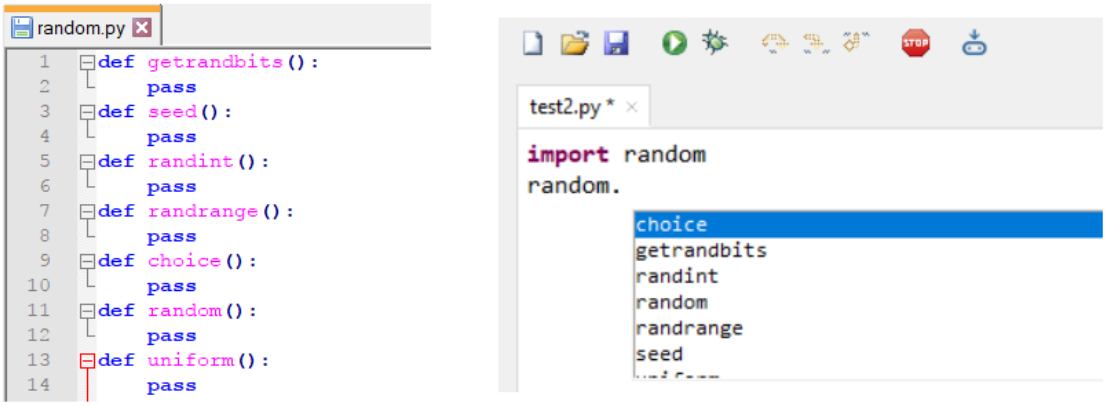
4.1 Koodi kirjutamine, käivitamine ning peatamine

Thonny pistikprogramm toetab koodiredaktoris automaatset sõnade lõpetamist (ingl k *autocomplete*), mida saab kasutada vajutades klahvikombinatsiooni „Ctrl+Space“. Automaatne sõnade lõpetamine valmis Jedi teegi abil, mis oskab aktiivset koodiredaktori faili ja Pythoni teeke analüüsida ning selle põhjal teha pakkumisi, kuidas mingit sõna lõpetada [18].



Joonis 6. Pakkumiste tegemine moodulifailide põhjal

Jedi analüüsimise jaoks valmis hulk Pythoni faile, mis esindavad MicroPythonis olevaid mooduleid. Need failid sisaldavad kõiki klasse ning funktsioone, nagu MicroPythoni moodulidki, kuid funktsioonid on lisatud nendesse ilma kehadeta, sest antud juhul on oluline ainult moodulifailide olemasolu ning nende struktuur. Joonis 6 näitab, kuidas Jedi teeb mooduli importimisel moodulifailide põhjal pakkumisi ning joonis 7 näitab moodulifaili *random.py* sisu ja seda, et Jedi teeb pakkumisi antud moodulifaili sisu põhjal.



Joonis 7. Pakkumiste tegemine moodulifaili sisu põhjal

Thonny pistikprogramm võimaldab kasutajal käivitada MicroPythonis kirjutatud koodi ilma seda miniarvuti välk-püsimällu salvestamata. Selle eelis on see, et kirjutatud programmi testimiseks ei kulu nii palju aega, kui muidu välk-püsimällu salvestamise protsessi käigus kulub. Programmi käivitamiseks tuleb vajutada klaviatuuril nuppu „F5“ või kasutajaliidesel olevat rohelist „*Run current script*“ nuppu.

Programmi käivitamise eel tehakse selle jaoks ettevalmistusi, mis koosnevad neljast sammust. Esmalt saadetakse läbi jadaliidese micro:bitile taaskäivituse signaal, kuna on oluline, et eelnevatest koodi käivitamistest või käsurea kasutamisest ei oleks miniarvuti mällu alles jäänud muutujaid, mis võiks programmi täitmist mõjutada. Teiseks saadetakse kohe peale taaskäivitust signaal, mis peatab micro:bitil oleva programmi täitmise, et miniarvuti mällu ei tekiks uuesti muutujaid. Kolmandaks viiakse micro:bitil olev MicroPython spetsiaalsesse olekusse, mida nimetatakse *raw_mode*'ks. See olek lihtsustab programmide käivitamist ning eraldab tavalised programmi täitmisel tekkinud väljundid veateadetest. Neljandaks saadetakse funktsiooni „*__print_msg__*“ definitsioon (vt Joonis 8), mis on vajalik kasutaja programmi poolt antud väljundite ning Thonnyle mõeldud sõnumite eristamiseks.

```
def __print_msg__(msg):
    print('<ForThonny>'+repr(msg)+'</ForThonny>', end='')
```

Joonis 8. Micro:bitil defineeritav funktsioon

Alles siis, kui kõik eelnevalt kirjeldatud toimingud on tehtud, võetakse koodiredaktorist kasutaja poolt kirjutatud kood, mis saadetakse miniarvutil oleva MicroPythoni interpretaatori käsureale, kus see käivitatakse koheselt ilma programmi miniarvuti välk-püsimällu salvestamata.

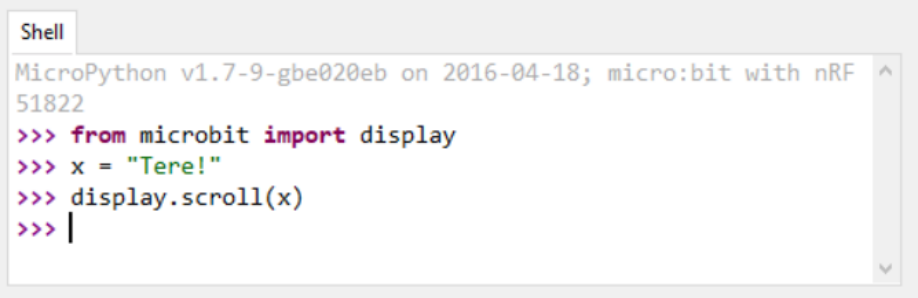
Koodi käivitamise puhul on veel ka see eelis, et vigase programmi puhul kuvatakse veateade koheselt interaktiivse käsurea aknasse. Micro:bit on võimeline välk-püsimällu salvestatud programmide vigu ka ise oma ekraanil kuvama, kuid selle ekraani väiksuste tõttu kuvatakse kogu teade tähthaaval ning terve sõnumi lugemine võib võtta üpris kaua aega.

Programmi peatamiseks (näiteks juhul, kui programm koosneb lõpmatust tsüklist) tuleb vajutada klaviatuuril klahvikombinatsiooni „Ctrl+F2“ või Thonny kasutajaliidesel punast „*Interrupt/Reset*“ nuppu. Esmakordsel vajutamisel katkestatakse ainult programmi täitmine ilma taaskäivitust tegemata. Kui koodiredaktoris olev programm ei ole käivitatud või selle täitmine on katkestatud, siis „*Interrupt/Reset*“ nupu vajutamisel tehakse micro:bitile taaskäivitus ning mälu puhastatakse muutujatest. Antud juhul toimib taaskäivitus täpselt samamoodi, nagu koodi käivitamise eel tehtav taaskäivitus. „*Interrupt/Reset*“ nuppu saab kasutada alati, kui on vajadus miniarvutile taaskäivitus teha ning mälu puhastada.

4.2 Interaktiivne käsurida

Thonnyl on olemas interaktiivse käsurea vaade, mis on tavaliselt Thonny käivitamisel avatud. Kui mingisugusel põhjusel antud vaade ei ole avatud, siis on seda võimalik sisse lülitada Thonny menüüst „View“ valikuga „Shell“.

Interaktiivsel käsureal saab teha erinevaid väiksemaid katsetusi, nagu näiteks luua uusi muutujaid, kasutada erinevaid funktsioone ning väärtustada avaldisi. Kõik kasutaja poolt sisestatud read saadetakse micro:bitil olevale MicroPythoni interpretaatori käsureale, kus sisestatud rida käivitatakse. Joonisel 9 oleva näite sisestamisel interaktiivsele käsureale kuvatakse micro:biti ekraanil muutuja x väärtus.



Joonis 9. Interaktiivse käsurea vaade

Interaktiivsele käsureale on samuti lisatud automaatne sõnade lõpetamise funktsionaalsus, mis sel puhul ei ole loodud Jedi teegi abil, kuid see on võimeline lõpetama globaalses skoobis olevaid nimesid ning nende nimede atribuute. Esmalt kontrollitakse pistikprogrammis, kas kasutaja üritab lõpetada mõnda nime või atribuuti. Vastavalt sellele kasutatakse MicroPythonisse sisseehitatud funktsiooni *dir*. Ilma argumentideta tagastab see funktsioon loendi globaalses skoobis olevate nimedega. Kui argumendiks anda mõni objekt, siis tagastatakse loend kõikide selle objekti atribuutidega. Eelmainitud funktsiooni väljakutse saadetakse MicroPythoni interpretaatori käsureale kindalt formaaditud sõnumina, mis saadab peale *dir* funktsiooni käivitamist pistikprogrammile tagasi nimede loendi. Seejärel valitakse välja sobivad pakkumised ning kuvatakse need kasutajale.

4.3 Globaalsed muutujad

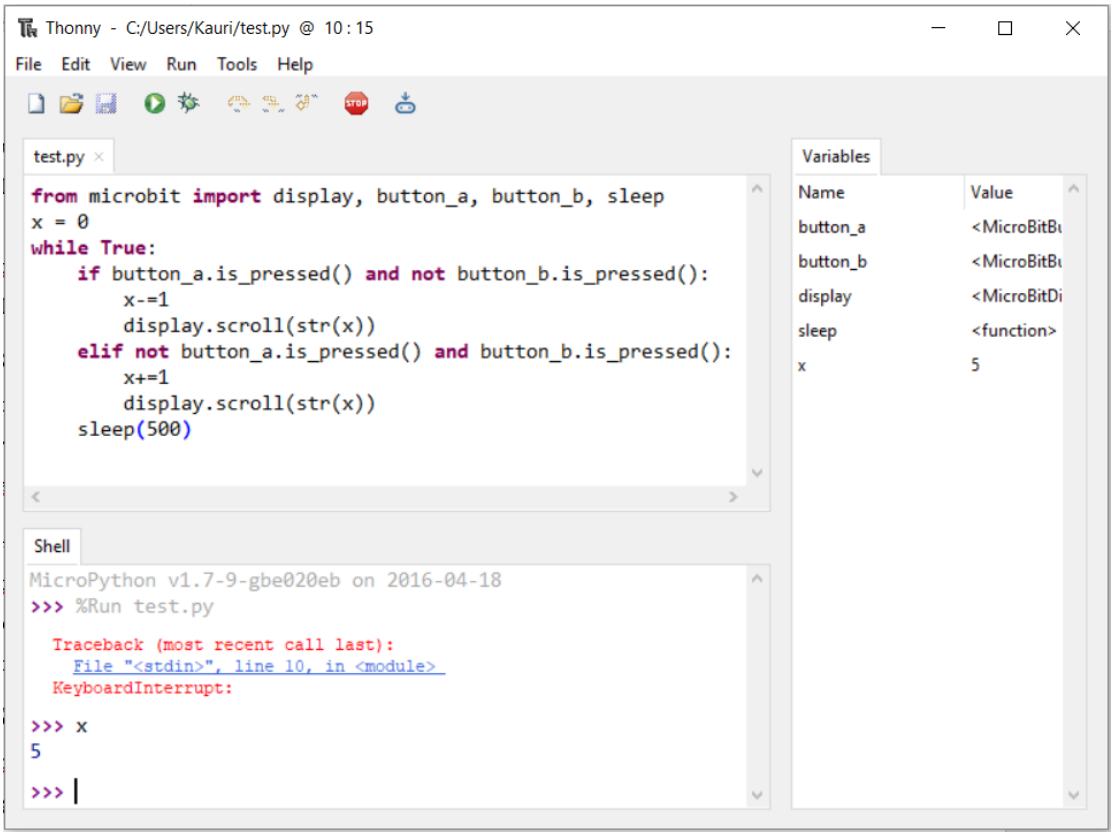
Pistikprogramm on võimeline micro:bitilt koguma infot globaalsete muutujate kohta ning seda infot kuvama Thonnysse sisseehitatud muutujate vaates, mille saab sisse lülitada menüüst „*View*“ valikuga „*Variables*“.

```
__print_msg__({'message_type':'Globals',
               'module_name' : '__main__',
               'globals':{x:repr(globals()[x]) for x in globals() if not x.startswith('__')}})
```

Joonis 10. Micro:bitile saadetav sõnum globaalsete muutujate pärimiseks

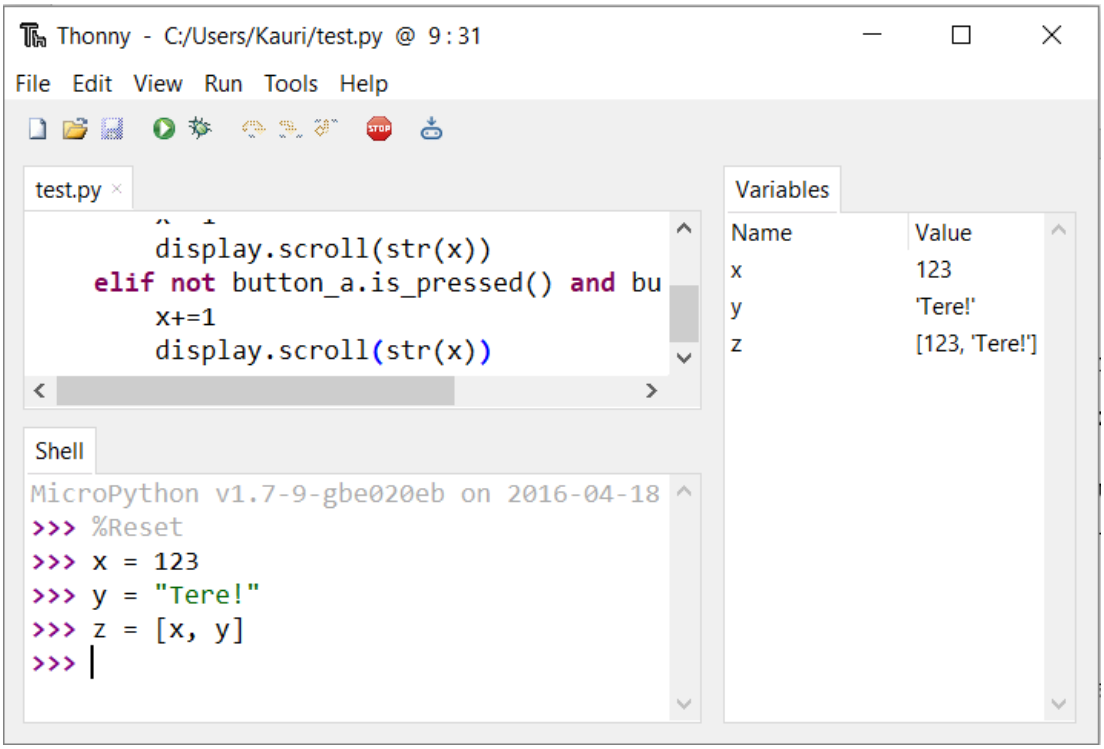
Globaalsete muutujate kohta info saamiseks saadetakse micro:bitile sõnum, mis sisaldab sõnastiku komprehensiooni (ingl k *dictionary comprehension*) koos MicroPythonisse sisseehitatud *globals* funktsiooniga (vt Joonis 10). Globaalsete muutujate hulka ei arvestata selliseid muutujaid, mis

algavad kahekordse alakriipsuga. Selle abil välditakse info kogumist erinevate süsteemis olemasolevate muutujate kohta, mida kasutaja ei ole ise loonud.



Joonis 11. Katkestatud programm

Joonisel 11 näidatakse katkestatud programmi ning selle muutujate väärtusi katkestuse toimumise ajahetkel. Antud näite puhul muutub muutuja x väärtus väiksemaks või suuremaks vastavalt sellele, kas kasutaja hoiab all micro:bitil olevat nuppu A või B. Programmi katkestamise hetkel oli muutuja x väärtus 5, mida on näha globaalsete muutujate vaatest. Seda tõestab ka interaktiivne käsurida, kui selle abil küsida muutuja x väärtust.



Joonis 12. Muutujate loomine käsureal ning nende info globaalsete muutujate vaates

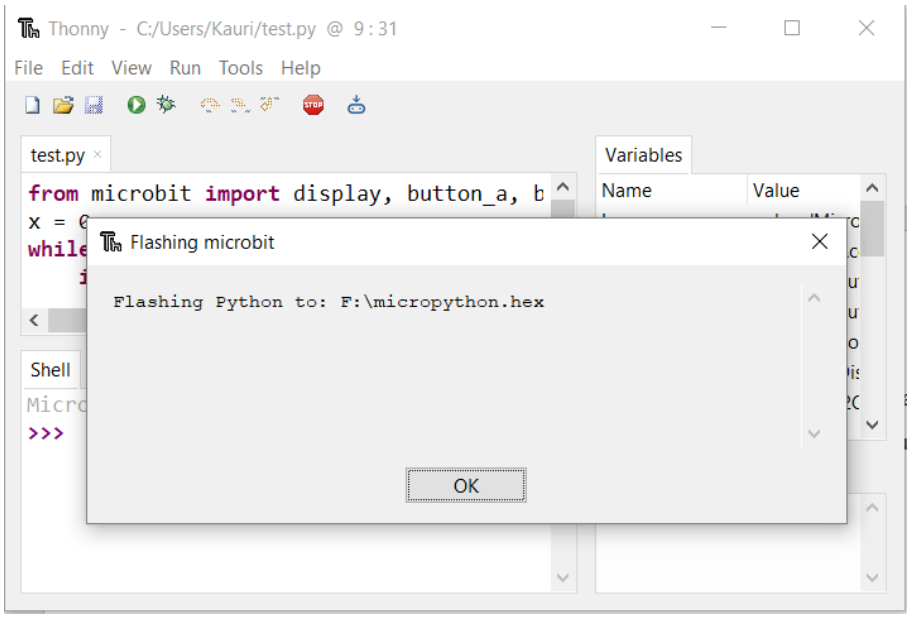
Globaalseid muutujaid kuvatakse muutujate vaatesse ka siis, kui kasutaja kasutab interaktiivset käsurida, luues sellega uusi muutujaid, nagu on näidatud joonisel 12.

4.4 Programmi salvestamine micro:bitile

Thonnys on võimalik MicroPythoni käituskeskkonda salvestada miniarvuti välk-püsimällu kahel erineval moel. Esiteks saab seda teha salvestades sinna ainult puhas MicroPythoni käituskeskkond. Selle protsessi käivitamiseks tuleb kasutajal vajutada menüüs „*Tools*“ käsule „*Flash REPL to BBC micro:bit*“. Seda tuleks teha näiteks siis, kui miniarvutisse ei ole veel salvestatud programmeerimiseks vajalikku MicroPythoni käituskeskkonda. Teiseks saab koos MicroPythoni käituskeskkonnaga salvestada miniarvuti välk-püsimällu ka kasutaja poolt kirjutatud koodi, mida käivitatakse miniarvuti igal taaskäivitamisel. Selle jaoks lisab pistikprogramm Thonny kasutajaliidesele uue nupu „*Flash current script to BBC micro:bit*“, mis on joonisel 13 kõige parempoolsem ning millele vajutamine käivitab eelnimetatud protsessi. Alternatiivselt võib kasutada ka klahvikombinatsiooni „Ctrl+M“ või valida menüüst „Tools“ käsk „*Flash current script to BBC micro:bit*“.

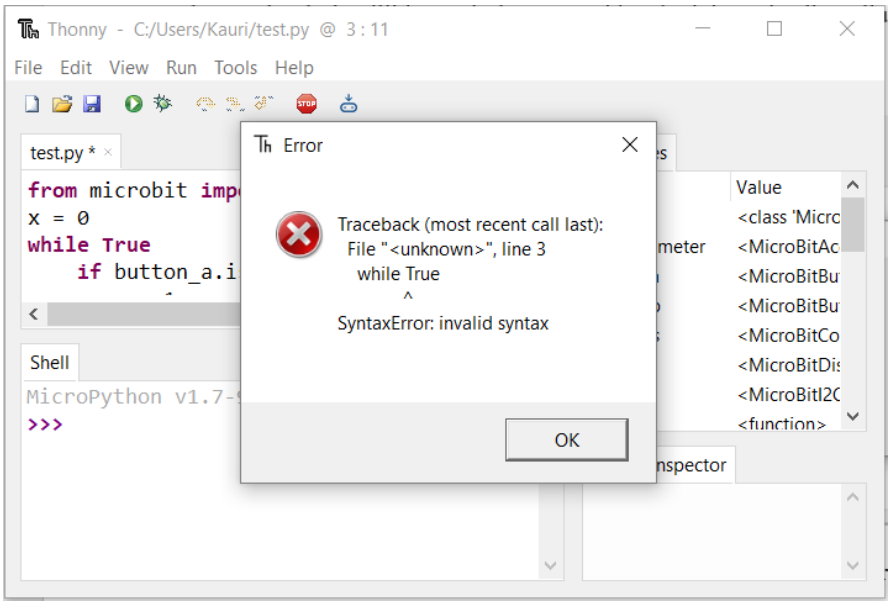
Joonis 13. Thonny kasutajaliidesel olevad nupud

Eelmises lõigus kirjeldatud välk-püsimallu salvestamine valmis uFlash [19] utiliidi abiga. Välk-püsimällu salvestamise alustamiseks luuakse Thonnys uus alamprotsess, mille kaudu kasutatakse tavaarvuti operatsioonisüsteemi käsureal uFlash utiliiti. Joonisel 14 on näidatud miniarvuti välk-püsimällu salvestamise protsessi. Miniarvuti välk-püsimällu salvestamine võtab ligikaudu 10 sekundit aega, kuna MicroPythoni käitussüsteem on küllaltki mahukas ja see tuleb viia läbi jadaliidese iga kord, kui kasutaja salvestab seda miniarvuti välk-püsimällu kas koos enda poolt kirjutatud koodiga või ilma selleta.



Joonis 14. Välk-püsimällu salvestamise protsess

Kui kasutaja soovib oma kirjutatud programmi salvestada miniarvuti välk-püsimällu, siis enne salvestamise protsessi kontrollitakse kirjutatud programmi süntaksit. Kui süntaksis leidub vigu, siis salvestamisprotsessi ei alustata ning kasutajale kuvatakse uues aknas vastav teade.



Joonis 15. Veateade programmi salvestamisel miniarvutile

Joonisel 15 on näha katkestatud välk-püsimällu salvestamise protsessi, mida põhjustas süntaksi viga kasutaja poolt kirjutatud koodis. Antud näites on *while*-tsükli päisest puudu seda lõpetav koolon ning pistikprogramm kuvab kasutajale vastava teate.

4.5 Pistikprogrammi ja Mu võrdlus

Thonny pistikprogrammil on nii eeliseid kui ka puuduseid populaarse Mu programmeerimiskeskkonna kõrval. Tabel 1 on välja toodud kõige tähtsamad erinevused mõlema programmi osas.

Tabel 1. Pistikprogrammi ja Mu võrdlus

Võrreldav omadus	**Thonny pistikprogramm**	**Mu programmeerimiskeskkond**
Failisüsteemi vaade	Puudub	Eksisteerib failisüsteemi vaade, mille abil on võimalik faile lisada ja kustutada micro:biti failisüsteemis.
Muutujate väärtuste vaade	Olemas	Puudub

Arendustsükkel	Koodi käivitamise funktsionaalsuse olemasolu tõttu ei pea programmi testimiseks ilmtingimata salvestama seda miniarvuti välk-püsimällu, mis on küllaltki ajakulukas protsess. Programmi käivitamine pistikprogrammis võtab aega ligikaudu ühe sekundi.	Koodi testimiseks peab selle salvestama koos MicroPythoni käitussüsteemiga miniarvuti mällu, mis võtab umbkaudu 10 sekundit aega.
Automaatne sõnade lõpetamine	Koodiredaktori puhul on kasutatud Jedi teeki, mis suudab analüüsida koodiredaktoris olevat koodi ja moodulifaile ning nende põhjal teha sobivaid pakkumisi. Interaktiive käsurea puhul suudetakse lõpetada globaalses skoobis olevate muutujate nimesid ning nende atribuute.	Koodiredaktoris võib pakkumiste sees olla ka valikuid, mis tegelikult ei sobi antud konteksti (näiteks pakutakse mõnele objektile mõne teise objekti funktsioone). Interaktiivne käsurida suhtleb otse MicroPython'i interpretaatoriga, millel on olemas sisseehitatud automaatne sõnade lõpetamine. See suudab lõpetada globaalses skoobis olevate muutujate nimesid ning nende atribuute.
Jadaühendus	Harva esineb anomaaliaid	Stabiilne

Kokkuvõtteks võib öelda, et pistikprogrammi suurim eelis programmeerimiskeskkonna Mu ees on kordades kiirem arendustsükkel, kuna pistikprogramm võimaldab kirjutatud programmi käivitamist ilma seda miniarvuti välk-püsimällu salvestamata. Olulisimaks puuduseks on kohati ebastabiilne jadaühendus. Kuna Thonny pistikprogramm kasutab jadaliidese kaudu suhtlust erinevate funktsionaalsuste tõttu intensiivsemalt kui Mu, siis stabiilsema jadaühenduse loomine ongi keerulisem. Peamiseks mustriks anomaaliate tekkimisel jadaühenduses tundub olevat see, kui mõlemad osapooled, nii pistikprogramm kui ka micro:bit üritavad samaaegselt saata teineteisele sõnumeid. Siiski ei tundu see olevat ainus põhjus anomaaliate tekkimiseks. Teisi tekkepõhjuseid ei ole veel avastatud.

5. Tööprotsess

Käesolev peatükk kirjeldab pistikprogrammi loomise protsessi. Sealhulgas tuuakse välja ebaõnnestumised ning alternatiivsed meetodid, mida pistikprogrammi valmimisel kasutati.

Tööd alustati olemasolevate lahendustega tutvudes ning antud pistikprogrammi loomisel oli autorile suureks eeskujuks Mu programmeerimiskeskkond. Järgevalt tutvuti utiliidi/teegiga uFlash, mis sisaldab vahendeid micro:bitile programmide salvestamiseks [19] ning pySerial teegiga, mis võimaldab luua jadaühendusi väliste seadmetega [20].

Enne Thonny pistikprogrammi loomise alustamist anti bakalaureusetöö autorile töö juhendaja poolt harjutusülesandeid. Need seisnesid Pythoni programmi loomises, mille abil läbi jadaliidese micro:bitiga suheldakse. Jadaühenduse loomine osutus keerulisemaks, kui alguses paistis. Nimelt oli probleem selles, et saadetud sõnumid ei jõudnud iga kord selliselt kohale, nagu nad oleksid pidanud jõudma. Probleemi lahendas sellel hetkel sõnumite jupikaupa saatmine, ehk ühe sõnumi tükeldamine juppideks ning seejärel nende saatmine micro:bitile. Kuna sõnumite lõppe tähistasid kindlad sümbolid, siis micro:bitil ei teki probleeme vastu võetud juppide kokku panemisega ning saadud sõnumite käivitamisega.

Järgmisena tutvus autor pistikprogramme puudutava Thonny lähtekoodiga, et saada aimu, kuidas Thonny sõnumeid saadab ning neid vastu võtab. Seejärel asus autor pistikprogrammi kirjutama. Esmalt valmis jadaühenduse loomise kood pySerial teegi abiga, mille kaudu sai micro:bitiga ühenduse luua. Kui kood jadaühenduse loomiseks ja kasutamiseks oli valmis, asus autor kirjutama pistikprogrammi osa, mis tegeleb Thonny sõnumite saatmise, püüdmise ning nende töötlemisega. Esialgne lahendus saatis micro:bitile Thonny spetsiifilise interaktiivse käsurea simulatsiooni, mille ülesanne oli lugeda sissetulevaid sõnumeid, neis olevat koodi käivitada ning vastused spetsiaalselt Thonny jaoks vormindatud sõnumina tagasi saata. Kuna miniarvuti on vägagi piiratud ressurssidega, siis peagi hakkasid ilmuma veateated miniarvuti mälu puuduse kohta. Oli selge, et tuleb leida muu lahendus, kuna peale kasutaja kirjutatud koodi saadeti micro:bitile korraga veel ka palju muud koodi, mis tegeleb Thonny suhtlusega ning see kõik hõivab miniarvutil lisamälu.

Alternatiivne lahendus kasutas MicroPythoni interpretaatori oma käsurida ning programmide lihtsamaks käivitamiseks kasutati MicroPythoni *raw_mode* olekut. *Raw_mode* oleku katsetamisel selgus veel, et see suudab tagastada tavalised programmi väljundid ja veateated eraldatuna, mis teeb info töötlemise ning kuvamise pistikprogrammis lihtsamaks.

Järgmisena loodi funktsionaalsus programmide salvestamiseks micro:biti välk-püsimällu, mis erilisi raskusi ei tekitanud, kuna uFlash utiliit sisaldas kõiki vajalike vahendeid selle teostamiseks.

Viimase funktsionaalsusena lisati sõnade automaatne lõpetamine nii interaktiivse käsurea vaatele kui ka koodiredaktorile. Esmalt valmis antud funktsionaalsus interaktiivse käsurea vaate jaoks MicroPythonisse sisseehitatud *dir* funktsiooni abil ning kohe peale seda ka koodiredaktori jaoks Jedi teegi abil. Jedi poolt tehtavate pakkumiste jaoks tuli luua hulk moodulifaile, mis sisaldavad infot MicroPythonis olevate moodulite, nende klasside ning funktsioonide kohta. Nii interaktiivse käsurea vaate kui ka koodiredaktori jaoks valminud funktsionaalsuse valmimine kulges eriliste probleemideta.

Pistikprogrammi arendamise käigus tuli ette ka mitmeid uuendusi Thonny poole pealt, millega kohati muutus Thonny käitumine erinevate protsesside puhul. Näiteks võib tuua „Interrupt/Reset“ nupu muutuse, mis vanemas Thonny versioonis oli nimega „Stop/Reset“ ning mis ei katkestanud käivitatud programmi täitmist esmakordsel vajutamisel, vaid tegi kohe taaskäivituse. Muudatused nõudsid pistikprogrammi puhul nendega kaasa minemist ning koodi ümberkirjutamist uuenduste poolt mõjutatud kohtades. Üldiselt tõsisemaid probleeme Thonny uuenduste puhul ei tekkinud ning pistikprogrammi arendamine kulges sujuvalt.

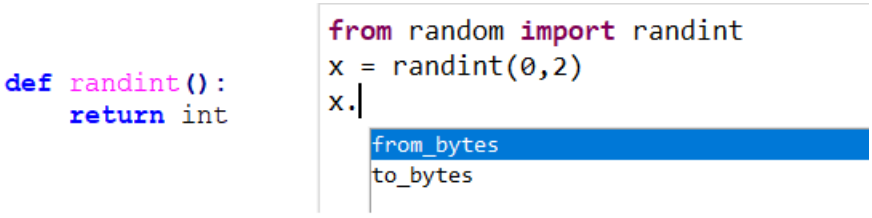
6. Edasised arendamisvõimalused

Käesolev peatükk toob välja mõningad olulisemad tähelepanekud ning ideed, mida Thonny pistikprogrammi edasiarendamisel võiks arvesse võtta.

Enamus Thonny pistikprogrammis olevast funktsionaalsusest kasutab toimimiseks jadaühendust ning seetõttu on jadaühenduse laitmatu töötamine kõige tähtsam osa antud pistikprogrammi puhul. Kuna kohati on täheldatud anomaaliaid Thonny pistikprogrammi kasutamisel, mis puudutavad jadaühendust, siis esmakorras tuleks otsida põhjuseid, mis neid anomaaliaid tekitavad ning need põhjused kõrvaldada.

Lõputöö algfaasis oli üheks ideeks ka micro:biti jaoks implementeerida Thonny põhifunktsionaalsuse alla kuuluv silur (ingl k *debugger*). Kuna siluri implementeerimine osutus oluliselt keerulisemaks ning mahukamaks, kui see esialgu paistis, siis otsustati see osa ajapuuduse tõttu lõputööst välja jätta ning keskenduti peamiselt micro:biti programmeerimisele ning välk-püsimällu salvestamisele. Seega jääb siluri implementeerimine tulevikuplaanidesse.

Koodiredaktorisse lisatud automaatse sõnade lõpetamise funktsionaalsuse saaks teha veelgi täpsemaks, kui lisada Jedi jaoks loodud MicroPythoni moodulifailide funktsioonidele tagastusväärtused. Näiteks võib tuua MicroPythonis oleva *random* mooduli funktsiooni *randint*. On teada, et *randint* funktsioon tagastab alati täisarvu, seega võib Jedi jaoks loodud moodulifailis olevale *randint* funktsioonile lisada tagastusväärtuseks täisarvu, ehk *int* tüübi. Joonisel 16 on näha, et peale tagastusväärtuse lisamist pakub Jedi muutujale x ainult täisarvu funktsioone.



Joonis 16. Jedi poolt tehtavad pakkumised peale tagastusväärtuse lisamist funktsioonile *randint*

Ideid ning soovitusi tulevikuplaanideks on mitmeid ning lõputöö autor on huvitatud pistikprogrammi edasiarendamisest.

7. Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli luua Thonny integreeritud programmeerimiskeskkonna jaoks pistikprogramm, mis lisab keskkonnale miniarvuti micro:bit toe. Töö käigus valmisid micro:biti programmeerimiseks mitmed funktsionaalsused, nagu näiteks koodi käivitamine ilma välk-püsimällu salvestamiseta, interaktiivse käsurea kasutamine, muutujate info kuvamine muutujate vaates, automaatne sõnade lõpetamine nii interaktiivsel käsureal kui ka koodiredaktoris ning kasutaja programmi salvestamine miniarvuti välk-püsimällu.

Pistikprogrammi arendamisel oli autorile suureks eeskujuks Mu programmeerimiskeskkond. Valminud pistikprogrammil on nii eeliseid kui ka puuduseid Mu programmeerimiskeskkonna kõrval. Olulisim pistikprogrammi eelis on kordades kiirem arendustsükkel ning puuduseks jadaühenduses esinevad anomaaliad.

Lõputöö autor on huvitatud tulevikus pistikprogrammi edasi arendamisest, et lisada juurde funktsionaalsust, mis bakalaureusetöö raames mahukuse tõttu lisamata jäi.

8. Viidatud kirjandus

[1] BBC. The BBC Micro:bit. http://www.bbc.co.uk/programmes/articles/4hVG2Br1W1LKCmw8nSm9WnQ/the-bbc-micro-bit (09.05.2017)

[2] Tartu Ülikooli arvutiteaduse instituut. Thonny. https://courses.cs.ut.ee/2017/eprogalused/Main/Thonny (09.05.2017)

[3] Annamaa, A. Introducing Thonny, a Python IDE for Learning Programming. Proceedings of the 15th Koli Calling International Conference on Computing Education Research. Koli, Finland. 2015. p 117-121.

[4] Thonny koodirepositoorium. https://bitbucket.org/plas/thonny/ (09.05.2017)

[5] Thonny koduleht. http://thonny.org/ (09.05.2017)

[6] Micro:bit. All the bits that make up your BBC micro:bit. http://microbit.org/about/ (09.05.2017)

[7] ARMmbed. BBC micro:bit. https://developer.mbed.org/platforms/Microbit/ (09.05.2017)

[8] Micro:bit. BBC micro:bit pins https://www.microbit.co.uk/device/pins (09.05.2017)

[9] Micro:bit: developer community. DAPlink and the USB interface. http://tech.microbit.org/software/daplink-interface/ (09.05.2017)

[10] Micro:bit. Hardware. https://www.microbit.co.uk/device (09.05.2017)

[11] Micro:bit. LED screen. https://www.microbit.co.uk/device/screen (09.05.2017)

[12] Micro:bit. Power your imagination with code. http://microbit.org/code/ (09.05.2017)

[13] MicroPythoni koodirepositoorium. https://github.com/micropython/micropython (09.05.2017)

[14] MicroPythoni koodirepositooriumi viki. Differences. https://github.com/micropython/micropython/wiki/Differences (09.05.2017)

[15] Micro:bit: developer community. Micro Python. http://tech.microbit.org/software/micropython/ (09.05.2017)

[16] Mu koodirepositoorium. https://github.com/mu-editor/mu (09.05.2017)

[17] Mu koduleht. https://codewith.mu/ (09.05.2017)

[18] Jedi koduleht. http://jedi.readthedocs.io (09.05.2017)

[19] uFlash'i koodirepositoorium. https://github.com/ntoll/uflash (10.05.2017)

[20] pySeriali dokumentatsioon. pySerial. http://pyserial.readthedocs.io/en/latest/pyserial.html (10.05.2017)

Lisad

Lisa 1. Pistikprogrammi installeerimis- ning seadistusjuhend

Pistikprogrammi ingliskeelne installeerimis- ning seadistusjuhend on kättesaadav järgnevalt aadressilt: https://bitbucket.org/KauriRaba/thonny-microbit/wiki/installation-guide

Lisa 2. Pistikprogrammi kasutusjuhend

Pistikprogrammi ingliskeelne kasutusjuhend on kättesaadav järgnevalt aadressilt: https://bitbucket.org/KauriRaba/thonny-microbit/wiki/manual

Lisa 3. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Kauri Raba**,

(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Arenduskeskkonnale Thonny micro:bit toe lisamine**,

   (*lõputöö pealkiri*)

   mille juhendaja on **Aivar Annamaa**,

   (*juhendaja nimi*)

   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**