TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Tartu kolledž

PRAKTIKUMITÖÖ LOOMINE - SISSEJUHATUS MICROPYTHONIS PROGRAMMEERIMISSE

THE CREATION OF LABORATORY WORK - INTRODUCTION TO PROGRAMMING IN MICROPYTHON

RAKENDUSKÕRGHARIDUSTÖÖ

Üliõpilane: Anna-Liisa Hannus

Üliõpilaskood 183560EDTR

Juhendaja: Ago Rootsi, lektor

Tartu 2022

(Tiitellehe pöördel)

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud.
Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"....." mai 2022

Autor: Anna-Liisa Hannus
/allkirjastatud digitaalselt/

Töö vastab rakenduskõrgharidustööle esitatud nõuetele

"....." mai 2022

Juhendaja: Ago Rootsi
/allkirjastatud digitaalselt/

Kaitsmisele lubatud

"....." mai 2022 .

Kaitsmiskomisjoni esimees .....
/allkirjastatud digitaalselt/

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Anna-Liisa Hannus (sünnikuupäev: 28.09.1998),

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose PRAKTIKUMITÖÖ LOOMINE - SISSEJUHATUS MICROPYTHONIS PROGRAMMEERIMISSE,

mille juhendaja on Ago Rootsi,

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

---

[1] *Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil.*

_____ /allkirjastatud digitaalselt/

_____ (*kuupäev*)

TalTech Inseneriteaduskond Tartu kolledž

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Anna-Liisa Hannus, 183560EDTR)

Õppekava, peaeriala: EDTR17/18 – Telemaatika ja arukad süsteemid, küberfüüsikalised süsteemid

Juhendaja(d): lektor, Ago Rootsi, 56629821

Lõputöö teema:

Praktikumitöö loomine - sissejuhatus MicroPythonis programmeerimisse

The creation of laboratory work - introduction to programming in MicroPython

Lõputöö põhieesmärgid:

1. Uurida tausta programmide ja seadmete kohta, valida välja sobilikud
2. Koostada praktikumijuhend
3. Koguda ja analüüsida tagasiside

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.		
2.		
3.		

Töö keel: eesti **Lõputöö esitamise tähtaeg:** "20" mai 2022.a

Üliõpilane: Anna-Liisa Hannus ..... "....." mai 2022. a
/ allkirjastatud digitaalselt /

Juhendaja: Ago Rootsi ..... "....." mai 2022.a
/ allkirjastatud digitaalselt /

Konsultant: .... "....." mai 2022.a
/ allkirjastatud digitaalselt /

Programmijuht: ..... "....." mai 2022.a
/ allkirjastatud digitaalselt /

Kinnise kaitsmise ja/või lõputöö avalikustamise piirangu tingimused formuleeritakse pöördel

SISUKORD

EESSÕNA 6
Lühendite ja tähiste loetelu 7
SISSEJUHATUS 8
1. LÄHTEÜLESANNE 9
2. METOODIKA 10
3. SOBIVA RIISTVARA VALIMINE PRAKTIKUMIKS 11
3.1 Kasutatavad komponendid 11
3.1.1 Arendusplaat NodeMCU D1 mini ESP8266 baasil 13
3.1.2 DHT11 14
4. MICROPYTHON 16
5. IDE VALIK 17
5.1 Mu Editor 17
5.2 uPyCraft 17
5.3 Thonny 18
6. PRAKTILINE OSA 20
6.1 Praktikumi ülesehitus 20
6.2 Teooria praktikumijuhendis 21
6.3 Praktikumis läbi viidud ülesanded 22
6.3.1 Ettevalmistus 23
6.3.2 Hello World 26
6.3.3 Arendusplaadil olev LED 27
6.3.4 LEDi vilgutamine 28
6.3.5 DHT11 28
6.3.6 DHT11 mõõtmistulemustega graafiku loomine 30
6.3.7 Iseseisev ülesanne 31
7. KATSETAMINE JA TAGASISIDE 32
KOKKUVÕTE 37
SUMMARY 38
KASUTATUD KIRJANDUSE LOETELU 39
LISAD 41

EESSÕNA

Lõputöö teema sõnastati Tallinna Tehnikaülikooli lektori Ago Rootsi poolt, kes on ka mikroprotsessorsüsteemide õppeaine õppejõud. Teemaks oli MicroPythonit tutvustava praktikumitöö loomine. Praktiline osa koostati töö autori poolt ja viidi läbi TalTech Tartu kolledžis.

Autor soovib tänada juhendajat, Ago Rootsit. Samuti soovib autor tänada kaastudengeid nendepoolse toe ja abi eest.

Märksõnad: MicroPython, praktikumi loomine, ESP8266, Thonny, bakalaureusetöö

Lühendite ja tähiste loetelu

IDE - integreeritud programmeerimiskeskkond (ingl k *integrated development environment*)

IoT – targad seadmed, mis suhtlevad omavahel läbi interneti (ingl k *Internet of Things*)

kB - kilobait

MB - megabait

mm - millimeeter

PIN – klemm või kontakt, mikrokontrolleri klemmiviigud arendusplaadi servas

REPL – loe, hinda, väljasta, korda (ing *Read-Evaluate-Print-Loop*)

V - volt

SISSEJUHATUS

Praktikumid erinevates õppeainetes aitavad tudengitel paremini aru saada asjade toimimisest ja annavad neile võimaluse omandatud teoreetilisi teadmisi rakendada. Käesoleva töö eesmärgiks oli koostada küberfüüsikaliste süsteemide tudengitele praktikumitöö mikroprotsessorsüsteemide aines.

Mikroprotsessorsüsteemid on Tallinna Tehnikaülikooli (edaspidi ka TalTech) telemaatika ja arukate süsteemide, peaeriala küberfüüsikalised süsteemid, õppekava kohustuslik aine, mida üldjuhul viiakse läbi 2. kursuse sügissemestril. Õppeaine eesmärgid on anda sissejuhatus mikroprotsessorite ja mikrokontrollerite valdkonda, tutvustada nende rakendusi ning omandada praktilisi kogemusi nende programmeerimises. Mikroprotsessorsüsteemide õppeaines moodustavad programmeerimisega seotud praktikumid olulise osa õppematerjalist.

Käesoleva töö lähteülesanne oli luua praktikumitöö, mis tutvustaks mikrokontrollerite programmeerimist programmeerimiskeeles MicroPython. Praktikumitöö loodi mikroprotsessorsüsteemide õppeaine tarbeks. See koosnes informatsioonist kasutatavate seadmete kohta ja ülesannetest, et tudengid saaksid käsitletava teema kohta mitmekülgseid teadmisi. Viimastel aastatel on mikrokontrollerite valdkonnas jõudsalt arenenud programmeerimiskeskkonnad MicroPython ja LUA, mis on võrgurakenduste programmeerimisel olulisemalt mugavamad ja eeldavad väiksemat programmeerija töömahtu, võrreldes Arduino IDEga. Seetõttu otsustati TalTech Tartu kolledži mikroprotsessorsüsteemide õppeainesse tuua sisse vähemalt üks praktikumitöö, mis käsitleks mikrokontrollerite programmeerimist MicroPython programmeerimiskeskkonnas, kuna varasemad praktikumid on läbi viidud Arduino IDE keskkonnas. Kuna see saab olema esimene MicroPythonit käsitlev praktikum, kuulub ülesandesse ka sobiva kontrolleri ja koodiredaktori valik ning juhend nende esmase seadistamise tabrbeks.

Praktikum viiakse läbi vabatahtlike tudengitega, kes saavad pärast praktikumi läbimist anda tagasisidet praktikumi korraldusele ja loodud juhendile.

Töös kirjeldatakse sobilike komponentide, mikrokontrollerite ja koodiredaktori valikut, praktikumi koostamist, sealhulgas loodud ülesandeid, ning tagasiside analüüsi. Juhend on toodud lisas 1.

1. LÄHTEÜLESANNE

Lõputöö lähteülesanne oli sõnastatud Tallinna Tehnikaülikooli Tartu kolledži lektori Ago Rootsi poolt. Ülesandeks oli TalTech Tartu kolledži mikroprotsessorsüsteemide õppeaine tarbeks luua mikrokontrolleri programmeerimist programmeerimiskeeles Python tutvustav praktikumitöö. Sealjuures tuli arvestada, et see on esimene MicroPythonit käsitlev praktikum, mis tähendas, et praktikum pidi sisaldama ka arvutite ja mikrokontrollerite ettevalmistamise juhendit ja vajalikke linke.

Esmalt tuli tutvuda MicroPythoni kui teemavaldkonnaga, lugeda erinevaid juhendeid ja dokumentatsiooni, saada üldpilt, mida praktikumis kajastama peaks. Seejärel tuli valida sobilik mikrokontroller ja koodiredaktor ja neid katsetada. Võrreldakse mitut erinevat MicroPythoni toega koodiredaktorit - Thonny, Mu Editor ja uPyCraft.

Soovitatav oli kasutada juba mikroprotsessorsüsteemide õppeaine praktikumides kasutusel olevaid kontrollereid, milleks on Arduino Pro Micro ATmegaU4 (edaspidi ka Arduino Pro Micro) ja ESP8266 NodeMCU D1 (edaspidi ka ESP8266) või nende mittesobivusel valida muu mikrokontroller, mille ostuhind ei ületaks 10€. Kuna igal aastal kipub rivist välja minema mingi osa kontrolleritest, on õppeasutustes lihtsam kasutada suuremat kogust sama marki kontrollereid.

Lõputöö tulemusena pidi valmima praktikumitöö, mida saab kasutada mikroprotsessorsüsteemide õppeaine praktikumides. Loodavat praktikumi pidi enne kasutuselevõttu katsetama ka testgrupil, mis viiakse läbi TalTech Tartu kolledži küberfüüsikaliste süsteemide peaeriala tudengitega.

2. METOODIKA

Praktikumitöö loomiseks tuli töö jagada etappideks, et paika panna tegevuste järjekord, sh sobilike komponentide valimine ja töö keerukuse valik.

Peamised etapid olid:

1. MicroPythoni teemavaldkonnaga tutvumine, dokumentatsiooni lugemine, MicroPythoni võimaluste uurimine, MicroPythoni näidisprogrammidega tutvumine;
2. tutvumine ülejäänud mikroprotsessorsüsteemide aine praktikumidega, et mõista loodava praktikumitöö konteksti. Loodav praktikum peaks ühilduma juba olemasolevatega ning vastaks tudengite tasemele;
3. uurimine, kas teistes kõrgkoolides on analoogset praktikumi läbi viidud ning kuidas seal see lahendatud oli;
4. sobiva mikrokontrolleri valimine, valikuteks olid koolis olemasolevad Arduino Pro Micro ATmegaU4 ja ESP8266 NodeMCU D1, nende võrdlus ja analüüs;
5. mikrokontrollerile programmide koostamiseks sobiva koodiredaktori valimine, erinevate koodiredaktorite võrdlus ja analüüs – Mu Editor, uPyCraft IDE, Thonny IDE;
6. praktikumi ülesehituse valik, planeerimine, kui suures osas juhend peaks koosnema teoreetilisest poolest ja kui suures mahus ülesannetest;
7. praktikumijuhendi ning ülesannete, mille keerukus oleks piisav, et tudengid õpiksid midagi uut, koostamine, veebis olemasolevate programmeerimisülesannete otsimine ja nende põhjal uute kombineerimine;
8. praktikumi katsetamine – läbiviimine vabatahtlike Tallinna Tehnikaülikooli Tartu kolledži küberfüüsikaliste süsteemide peaeriala tudengite peal. See on vajalik, et aru saada, kuidas praktikumi läbiviimine toimub kooli tingimustes ning saada tagasisidet praktikumi ülesehituse ja arusaadavuse kohta;
9. katsetulemuste ja tagasiside analüüs, järelduste tegemine – millega tudengid jäid rahule ja millega mitte, mida oleks võinud teha teisiti.

3. SOBIVA RIISTVARA VALIMINE PRAKTIKUMIKS

Enne praktikumitöö loomist tutvus autor teiste mikroprotsessorsüsteemide praktikumidega, et olla kursis, millega tudengid on juba kokku puutunud ja mida õppinud enne MicroPythoni praktikumi läbimist. Eelnevates praktikumides kasutati mikrokontrollerite programmeerimiseks Arduino IDE-t, MicroPythoniga puutuvad tudengid mikroprotsessorsüsteemide õppeaines kokku esmakordselt alles loodavat praktikumi läbides.

Autor otsis ka teiste kõrgkoolide praktikume, kus puututi kokku MicroPythoniga, et saada aimdust, kuidas need on üles ehitatud ja milliseid ülesandeid seal läbiti, kuid jõudis järeldusele, et ühtegi sellist ei ole avalikustatud. See ei tähenda, et käesoleva lõputöö tulemusena valmiv praktikum on ainulaadne, kuid võrdluses Arduino IDE-l baseeruvatega kindlasti haruldasem. MicroPythonit on täpsemalt kirjeldatud peatükis 4.

Kuna praktikum on osa mikroprotsessorsüsteemide õppeainest, valiti praktikumitööle sarnane ülesehitus teiste mikroprotsessorsüsteemide praktikumidega. Praktikum koosneb teoreetilisest osast kasutatavate komponentide kohta ja ülesannetest, mida läbides saab õppida tutvustatud komponentide kasutust ja nende programmeerimist. Praktikum viiakse läbi Tallinna Tehnikaülikooli Tartu kolledži arvutiklassis ja kooli seadmeid kasutades. Praktikumijuhendi kirjeldus on välja toodud peatükis „Praktiline osa“.

3.1 Kasutatavad komponendid

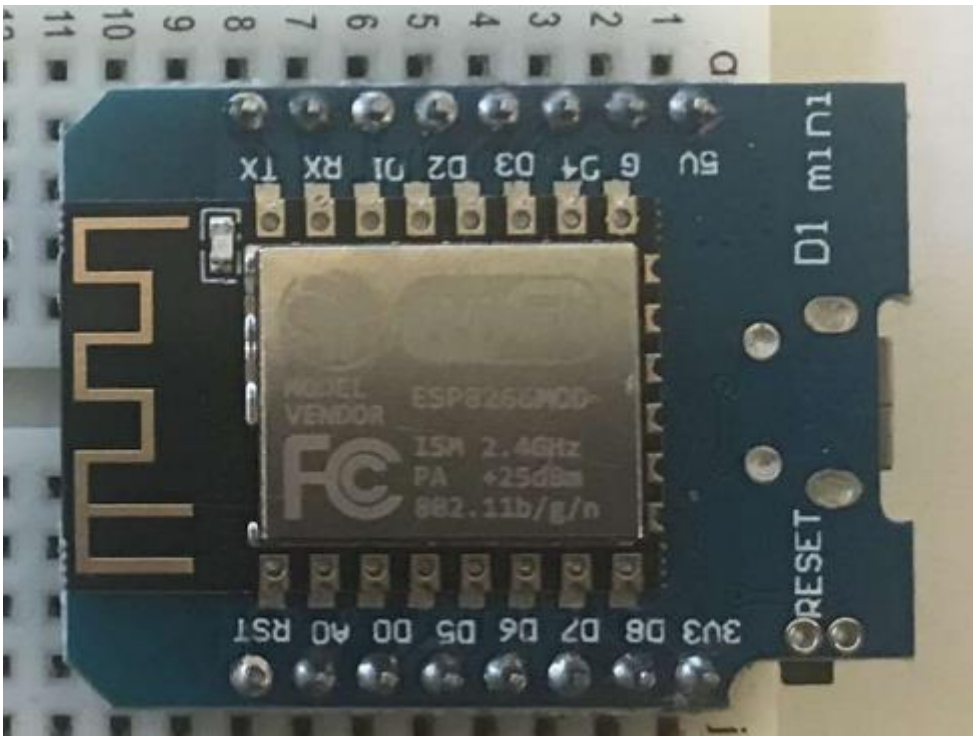
Mikrokontrollereid on mitmeid erinevaid ja erinevate otstarvetega. Mikrokontroller on iseseisev seade, mida saab kasutada millegi monitoorimiseks, juhtimiseks, signaalide töötlemiseks jms. Neid saab kasutada ka mõne muu seadme loomiseks. Mikrokontrolleri peamised komponendid on protsessor, PIN-id (klemmid ühenduste tegemiseks) ja mälu. [1]

Mikrokontrollereid hakati tootma 1970ndatel aastatel kasutamiseks kalkulaatorites ja kassaaparaatides ja algselt kutsuti neid mikroarvutiteks. Tänapäeval toodetakse aastas miljardeid mikrokontrollereid ja neid kasutatakse mitmetes erinevates seadmetes: tööstuses, telekommunikatsioonis, kodumasinates. [1]

Arendusplaat on mikrokontroller, millele on lisatud selle tööks vajalikud komponendid ja plaadi servadesse on lisatud PIN-id, mille kaudu saab arendusplaati elektriskeemi ühendada [2]. Praktikumis kasutamiseks oli vaja valida sobilik arendusplaat koolis olemasolevate seast, ehk tuli võrrelda Arduino Pro Micro ATmegaU4 (joonis 3.1) ja ESP8266 NodeMCU D1 omadusi MicroPythoni miinimumnõuetega. Erinevalt Arduino Pro Microst, on arendusplaadil ESP8266 võrguvõimekus, mistõttu on sellel rohkem perspektiivi edasiste praktikumide jaoks. [3] [4]

Joonis 3.1. Arduino Pro Micro ATmegaU4 pealtvaates [5]

MicroPython vajab arendusplaadil vähemalt 256 kB välkmälu (inglise keeles *flash memory*) [6], kuid Arduino Pro Microl on seda vaid 32 kB [4]. Arendusplaatide põhiparameetrid on välja toodud tabelis 3.1. Arduino Pro Micro vähese välkmälu tõttu võeti praktikumitöös kasutusse ESP8266 (joonis 3.2). MicroPythonit on hetkel võimalik kasutada Arduino arendusplaatidel Nano 33 BLE Sense, Nano 33 BLE, Raspberry Pi Nano RP2040 Connect ja Portenta H7 [7].



Joonis 3.2. ESP8266 NodeMCU D1

Tabel 3.1. Arduino Pro Micro ATmega32U4 ja ESP8266 NodeMCU D1 omadused[4] [5] [8] [9]

Parameeter	Arduino Pro Micro ATmega32U4	ESP8266 NodeMCU D1
Tööpinge (V)	5	3,3
Mõõdud (mm)	33 x 18	68,6 x 53,4
Välkmälu (kB)	32	4000 (4 MB)
Sisend-väljund PINide arv (tk),	20	12, millest kasutada saab 4
millest analoog sisend PINe (tk)	12	1
Sisseehitatud LED	Juhitav LED puudub	Jah, ühendatud PINiga D4 (GPIO 2)
Võrguvõimekus	puudub	WiFi

3.1.1 Arendusplaat NodeMCU D1 mini ESP8266 baasil

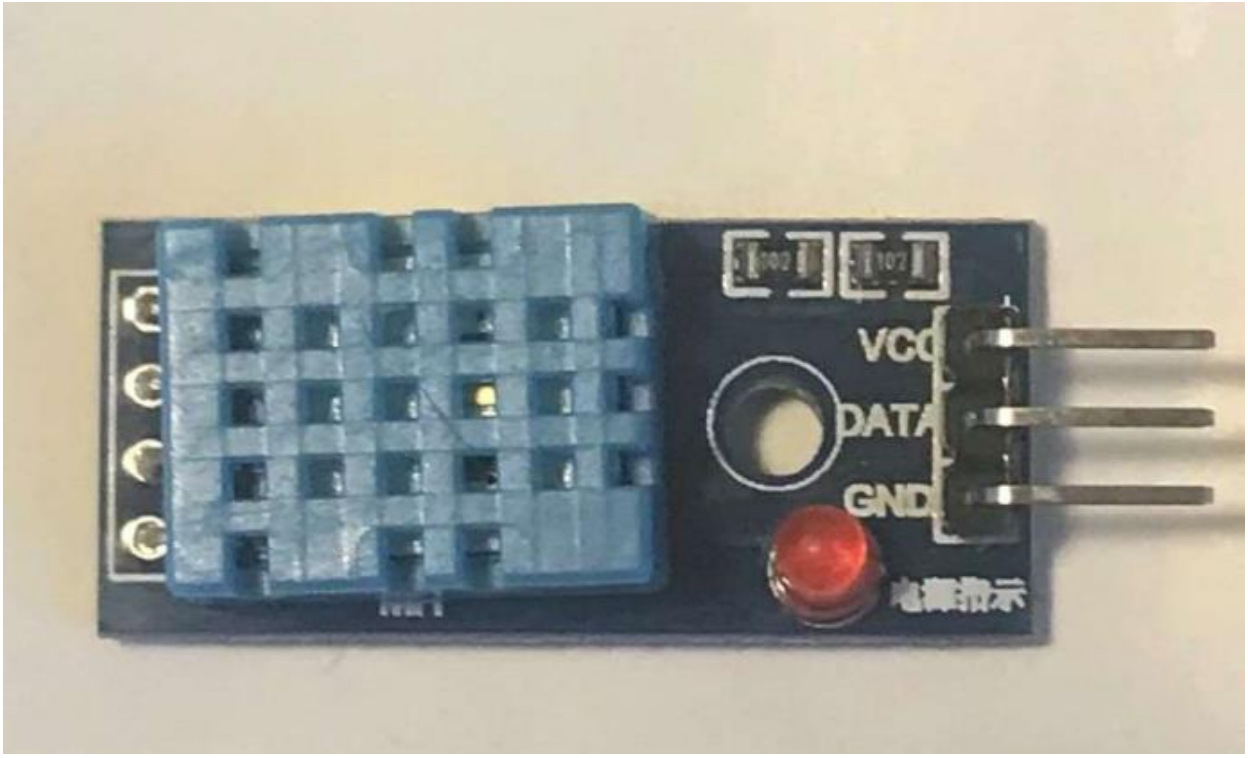
Praktikumis kasutamiseks valiti mikrokontroller ESP8266, mis on levinud seade IoT implementeerimises ning on ka kuluefektiivne. IoT (ingl k *Internet of Things*) on targad seadmed, mis oskavad üksteisega suhelda, kasutades selleks internetti Selle arendamiseks on palju erinevaid võimalusi, millele aitab kaasa ka ESP8266 WiFi

võimekus. [10] Tänu selle suurele kasutajabaasile on veebis saadaval suures koguses dokumentatsiooni ja juhendeid.

ESP8266 ühendamiseks arvutiga on vajalik USB A – Micro-USB juhe, mis võimaldab ka andmeedastust, muidu pole võimalik seda programmeerida. Arvutiga ühendades on antud USB-kaabel ka arendusplaaditoiteühenduseks.

3.1.2 DHT11

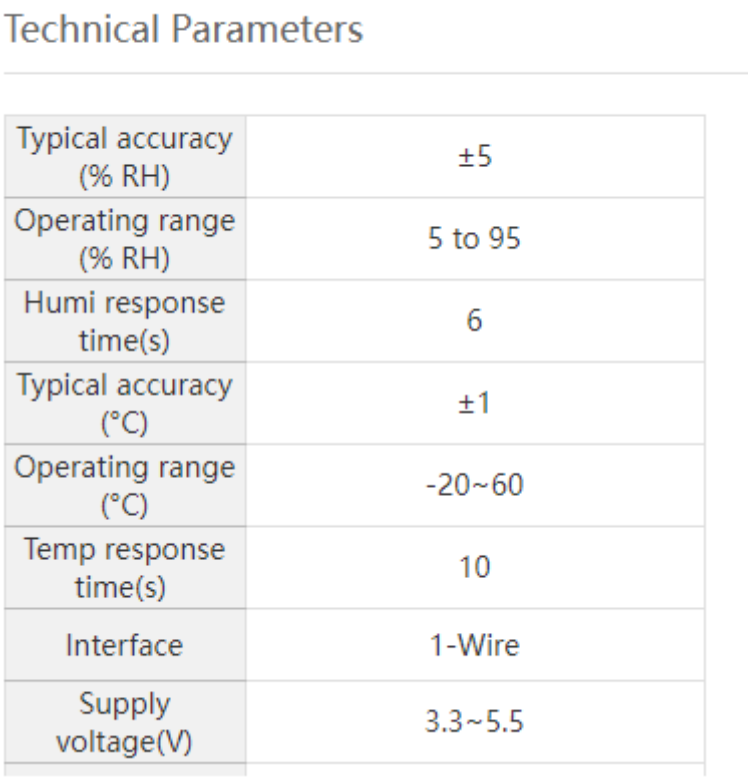
Arendusplaadi ESP8266 ja MicroPythoni rakenduste näitlikustamiseks kasutati ka õhuniiskuse ja -temperatuuriandurit DHT11 (joonis 3.3), mida on lihtne ühendada ja kasutada. Selle plusspooleks oli ka selle olemasolu mikroprotsessorsüsteemide õppeaine praktikumideks ettenähtud komponentide laos.



Joonis 3.3. DHT11

DHT11 on digitaalne õhuniiskuse- ja temperatuuriandur, millel on palju erinevaid kasutusvõimalusi – põllumajanduses, meditsiinis, automaatjuhtimisseadmetes, koduelektroonikas jm. Selle eelisteks on odav hind, pikaajaline stabiilsus, suurepärane kvaliteet. [11]

DHT11 parameetrid on välja toodud joonisel 3.4. Loodavas praktikumis oli nendest olulisimaks DHT11 toitepinge.



Technical Parameters

Typical accuracy (% RH)	±5
Operating range (% RH)	5 to 95
Humi response time(s)	6
Typical accuracy (°C)	±1
Operating range (°C)	-20~60
Temp response time(s)	10
Interface	1-Wire
Supply voltage(V)	3.3~5.5

Joonis 3.4. DHT11 omadused [11]

4. MICROPYTHON

MicroPython (ka µPython) on arendatud programmeerimiskeelest Python 3. See on mõeldud kasutamiseks mikrokontrollerite, sealhulgas arendusplaatide, ja sardsüsteemidega (ingl k *embedded systems*) ning selles koodi kirjutamine on väga sarnane tavalisele Pythonile. [12]

MicroPython on laialdaselt kasutatud, lihtsa ülesehitusega ja kergelt kasutatav programmeerimiskeel. MicroPythoni arendajate eesmärk on elektroonika programmeerimine teha võimalikult lihtsaks, et igaüks saaks selle kasutamisega hakkama. [12]

Programmeerimiskeeles Python koodi kirjutamisel eelis C++ keele, mida kasutab Arduino IDE, ees on selle lihtsus. Programmeerimiskeelt Python on tunduvalt lihtsam õppida, kui C++. Kui kasutaja oskab programmeerimiskeelt Python, on MicroPythoni kasutamine väga lihtne. Samuti on MicroPythoni eeliseks see, et programmi ei pea eraldi kompileerima ja arendusplaadile üles laadima. MicroPythonit on võimalik kasutada erinevatel arendusplaatidel – näiteks pyboard, ESP8266, WiPy, BBC micro:bit, muud Esp32/ESP8266-l põhinevad arendusplaadid jt [12] [13] Kuna ESP32 on ESP8266 järeltulija, on selle MicroPythoni võimalused veel arenemas [12].

MicroPythonis on olemas osa Pythoni teekidest (ingl k *library*), kuid lisaks on veel ka MicroPythoni-spetsiifilised teegid. [14]. Käesoleva töö käigus praktikumitööd luues kasutati teeke „machine", mis on MicroPythoni-spetsiifiline ja „time" .

MicroPythonil on sisseehitatud REPL (ingl k *Read-Evaluate-Print-Loop*) funktsioon, mis loeb sisestatud käsku, hindab ja käivitab käsu, väljastab vajadusel tulemuse ning seejärel ootab uut käsku ja alustab uuesti. See võimaldab käsku täita koheselt peale selle kirjutamist ja „Enter" klahvi vajutamist (joonis 4.1). Seda on võimalik kasutada ka mitmerealiste funktsioonide puhul – näiteks *if*-, *while*- ja *for*-tsüklite korral. [6] [15]

```
>>> print("Hello, World!")
Hello, World!
```

Joonis 4.1. Näide REPL kasutamisest

5. IDE VALIK

MicroPythoni koodi on võimalik kirjutada erinevates koodiredaktorites: Mu Editor, uPyCraft IDE, Thonny IDE. uPyCraft on loodud spetsiaalselt MicroPythoni jaoks, Thonny ja Mu Editor on loodud Pythoni jaoks, kuid toetavad ka MicroPythonit [12]. Mu Editor on loodud eelkõige algajatele, ehk see on lihtsa ülesehitusega [16]. Thonny kohta on veebist võimalik leida palju dokumentatsiooni ja mitmeid erinevaid juhendeid. [12]

5.1 Mu Editor

Mu Editor on lihtne Pythoni koodiredaktor, millel on intuitiivne kasutajaliides ning töötab lisaks Windowsile ka OSX ja Linux operatsioonisüsteemidega [17]. Sellel on MicroPythoni tugi kasutades arendusplaate ESP32 ja ESP8266[18].

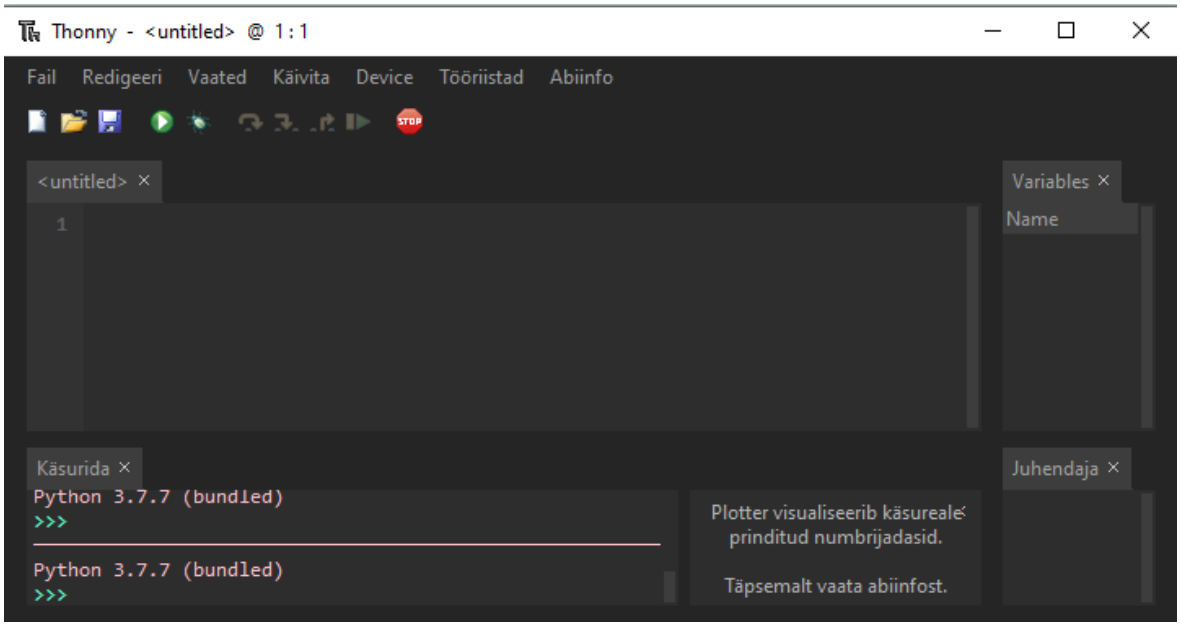
Sellel on sisseehitatud „Tidy" funktsioon, mis kontrollib koodis olevaid vigu – näiteks ridade taandeidja üleliigseid või puuduolevaid tühikuid. Mu Editor koodiredaktoril puudub „STOP"-nupp, mis katkestaks käimasoleva programmi ning kui ESP on hõivatud, ei pruugi sellega saada ühendust ja ühendus seadmega tuleb taasluua manuaalselt.[18]

5.2 uPyCraft

uPyCraft on koodiredaktor, mis disainitud just MicroPythoniga kasutamiseks. Selle negatiivseteks külgedeks on, et käsureale väljastatakse kogu *debugging* informatsioon, mis võib algajatele segadust tekitada ning kui seadmega ei suudeta esimesel korral ühendust luua, tuleb MicroPythoni *firmware* uuesti arendusplaadile laadida, mis on ajakulukas.[18]

5.3 Thonny

Thonny (joonis 5.1) on suuremas osas välja töötatud 2014.-2018. aastatel Tartu Ülikooli arvutiteaduste instituudis spetsiaalselt õppeotstarbeks. See on koodiredaktor, mille kasutamiseks piisab vaid ühest installeerimisfailist ja selle kasutajaliideselt on eemaldatud kõik *feature*'id, mis võivad algajaid segadusse ajada või häirida. Thonny koodiredaktoris on sisseehitatud plotter, mis võimaldab lihtsalt luua graafikuid väljastatud arvude põhjal. [19] Thonny.org



Joonis 5.1. Thonny IDE, plotteri funktsioon on sisse lülitatud. Kuvatõmmis

Praktikumi läbiviimiseks valiti Thonny IDE, kuna sellega on suure tõenäosusega osa tudengitest varem kokku puutunud ning seda on lihtne kasutada. Kasutamist hõlbustab ka võimalus kasutada Thonny IDE-t eesti keeles. Arduino IDE keskkond on samuti levinud redaktor arendusplaatide programmeerimiseks ning selle kohta on mikroprotsessorsüsteemide õppeaines juba mitmeid praktikume. Kuna käesoleva töö eesmärk oli luua praktikum, mis tutvustab MicroPythonit, mida programmeeritakse keeles Python, siis kasutati Pythoni koodiredaktorit.

Thonnys on MicroPythoni *firmware* arendusplaadile laadimiseks olemas sisseehitatud kasutajaliides, mis selle eriti lihtsaks teeb. Thonny suureks boonuseks kasutamisel õppeasutustes või alustatavate programmeerijate puhul on eestikeelse kasutajaliidese võimalus. [18]

MicroPythoni kasutamise üks eesmärke on ka suurema mitmekesisuse pakkumine nii praktikumides kui ka üliõpilaste valmiduse osas laiemas mõttes ning sellega on parem

õppida, kuna on väiksem risk eksida programmi kirjutades, kui programmeerides C-keeltes. Praktikumi peamine eesmärk oli teha sissejuhatus MicroPythonis programmeerimisse ja seetõttu pandi vähem rõhku programmi keerukusele.

6. PRAKTILINE OSA

Praktikumitöö sai teostatud mikroprotsessorsüsteemide aine raames. Praktikumi peamiseks eesmärgiks oli tutvumine MicroPythoni ja selle kasutamisega, kuna eelnevates selle aine praktikumides puututakse kokku põhiliselt Arduino IDE keskkonnaga.

Praktikumi eesmärgiks oli tutvustada tudengitele programmeerimiskeele MicroPython kasutamist arendusplaadi ESP8266 baasil. Praktikumi läbiviimiseks koostati juhend, mis koosnes teooriast ja ülesannetest. Juhend on toodud lisas 1. Autor koostas tudengitelt tagasiside saamiseks veebipõhise küsimustiku Google Forms keskkonnas, mida tudengid said täita pärast praktikumi läbimist.

Juhendi ülesehitus valiti sarnane teiste samalaadsete, ehk siis Arduino IDE praktikumidega. Kuna see oli esimene MicroPythoni praktikum antud kursusel, oli vaja installeerida ka Thonny, laadida alla sobiv *firmware*, laadida alla plaadi draiver, installeerida draiver ja laadida *firmware* plaadile. Juhendi teoreetilise osa loomisel kasutatud allikad on viidatud käesolevas peatükis vastavate alapeatükkide ja punktide juures.

Praktikum viidi läbi 22. aprillil 2022 Tallinna Tehnikaülikooli Tartu Kolledži õppehoones Puiestee 80A ruumis A205. Praktikum oli avalik ja vabatahtlik. Praktikum kestis tund aega ja kokku osales 6 küberfüüsikaliste süsteemide tudengit, kellest 4 õpivad hetkel 1. kursusel ja 2 tudengit 4. kursusel. Esimese kursuse tudengid olid varasemalt läbinud Pythonis programmeerimise kursuse, kuid mitte elektroonikakursust. Neljandal kursusel puudus varasem õppealane kokkupuude Pythoniga, kuid olid läbinud elektroonikakursuse. Töö autor oli praktikumi läbiviimisel juhendaja rollis.

6.1 Praktikumi ülesehitus

Praktikum koosnes teooriast, ühiselt lahendatavatest ülesannetest ja iseseisvast ülesandest. Praktikumis tutvuti arendusplaadiga ESP8266, kuidas seda ühendada arvutiga ja vajadusel sellele installeerida draiverit. Seejärel seoti arendusplaat Thonny IDE-ga ja tutvuti Thonny kasutamisega. Järgmine samm oli MicroPythoni *firmware*'i üleslaadimine arendusplaadile ja pärast seda erinevate programmide kirjutamine

programmeerimiskeeles MicroPython. Praktikumijuhendis välja toodud plaanitud tegevused on näidatud joonisel 6.1.

Praktikumi tegevused:

1. Tutvumine arendusplaadiga ESP8266, selle ühendamine arvutiga ja vajalike draiverite installeerimine
2. Arendusplaadi sidumine Thonny IDEga, Thonnyga tutvumine
3. MicroPythoni *firmware* üleslaadimine arendusplaadile
4. MicroPythoni võimalustega tutvumine ESP8266 arendusplaadil

Joonis 6.1. Juhendis välja toodud praktikumi tegevused

6.2 Teooria praktikumijuhendis

Teoreetiline osa juhendist koosnes praktikumides kasutatud komponentide tutvustamisest ja juhistest arendusplaadile ESP8266 programmeerimiskeeles MicroPython programmi kirjutamiseks.

Jooniselt 6.2 on näha, et juhendis selgitati, et MicroPython on arendatud programmeerimiskeelest Python ja sisaldab osa Pythonis olevatest teekidest ning et see on interpretaatorkeel. Kirjeldatud oli ka otsekäskude režiimi.[12]

MicroPython on arendatud programmeerimiskeelest Python ja sisaldab väikest osa Pythonis olevatest teekidest. See on interpretaatorkeel ja kuna koodi silumise faasis saab programmi tööle panna otse IDE-s, siis ei pea programmi kompileerima ja üles lugema. MicroPythonis on võimalik töötada otsekäskude režiimis, mis tähendab, et käsk täidetakse kohe või antakse veateade, mis kiirendab arendustööd ja kaitseb kontrollerite välkmälu (Flash memory) sagedase ümberkirjutamise eest, mida mikrokontrollerid taluvad vaid piiratud arvu kordi.

Joonis 6.2. MicroPythoni tutvustus juhendis

Seejärel tutvustati ESP8266 (joonis 6.3), toodi välja, et sellel on WiFi võimekus ja juhiti tähelepanu sellele, et suure kasutajabaasi tõttu on arendusplaadil ESP8266 MicroPythoni *firmware* paremini läbi proovitud ja silutud, kui paljude sarnaste arendusplaatide puhul.

ESP8266 on mikrokontroller, mille arendamiseks on palju erinevaid võimalusi. Sellel on olemas WiFi võimekus, mis võimaldab edastada andmeid interneti teel.

ESP8266 kasutamine on levinud ja seetõttu on internetis saadaval suurel hulgal erinevaid materjale selle kasutamise ja võimaluste kohta. Lisaks on võimalik leida mitmeid lahendusi erinevatele probleemidele, mis selle kasutamise käigus tekkida võivad. Suure kasutajabaasi tõttu on ESP8266 arendusplaadil MicroPythoni *firmware* paremini läbi proovitud ja silutud, kui paljude sarnaste mikrokontrollerite puhul.

Joonis 6.3. ESP8266 tutvustus juhendis

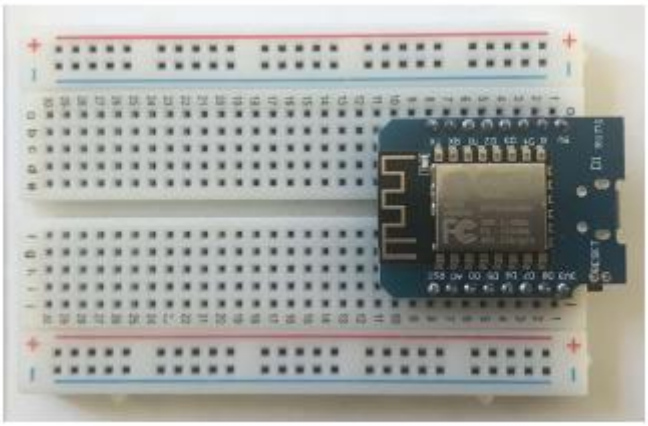
Lisaks tutvustati praktikumijuhendis (joonis 6.4) õhuniiskuse ja -temperatuuriandurit DHT11, mida on lihtne ühendada ja kasutada arendusplaadiga ESP8266.

Kasutame praktikumis ka temperatuuri- ja niiskuseandurit DHT11, mida on lihtne ühendada ja kasutada erinevate mikrokontrolleritega. See töötab pingel 3 kuni 5 V, praktikumis kasutame 3,3V.

Joonis 6.4. DHT11 tutvustus juhendis

6.3 Praktikumis läbi viidud ülesanded

Praktikum koosnes maketeerimisplaadile monteeritud ESP8266 arendusplaadiga (joonis 6.5), Thonny IDE-ga, DHT11-ga ja MicroPythoniga tutvumisest. Praktikumi komplekti kuulusid arendusplaat ESP8266, õhuniiskuse- ja temperatuuriandur DHT11, USB A – Micro-USB juhe ning 3 F-M (ingl k *female to male*, eesti keeles pesa-pistik) Duppont juhet.



Joonis 6.5. ESP8266 NodeMCU D1 monteeritud maketeerimisplaadile

Enne ülesannete juurde asumist olid juhendis välja toodud reeglid (joonis 6.6): mitte teha ühendusi arendusplaadil samal ajal, kui plaat on ühendatud arvutiga ja DHT11 peab ühendama ESP8266 arendusplaadi PINiga 3V3, mitte 5V. Mõlemad reeglid on veelkord välja toodud ka ülesannete juures, kus nendele erilist tähelepanu pöörama peab.

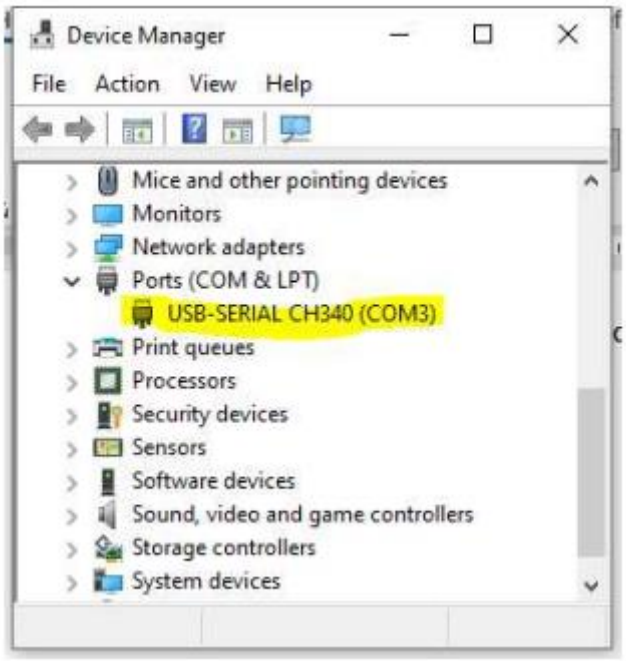
Reeglid

- Mitte teha ühendusi samal ajal kui seade on ühendatud arvutiga
- DHT11 ühendada 3V3 PINiga, mitte 5V

Joonis 6.6. Reeglid praktikumijuhendis

6.3.1 Ettevalmistus

Enne programmeerimise juurde asumist, tuli teha vajalikud ühendused ja installeerida vajalikud rakendused. Esimene samm oli arendusplaadi ESP8266 ühendamine arvutiga kasutades USB kaablit. Pärast ühendamist oli võimalik arvutis, kasutades Device Manageri, vaadata, kas arendusplaat on ühenduses arvutiga ning kas sellel on olemas vajalikud draiverid (joonis 6.7)



Joonis 6.7. Arvutiga ühendatud arendusplaat Device Manageris, millel on vajalik draiver (CH340) installeeritud

Draiveri CH340 puudumisel (joonis 6.8) olid juhendis välja toodud vajalikud sammud selle installeerimiseks koos lingiga, kust draiverit alla laadida saab.[20]

Joonis 6.8. Arvutiga ühendatud arendusplaat Device Manageris, millel puudub vajalik draiver (CH340)

Programmeerimiseks on vajalik ka koodiredaktor, praktikumi läbiviimiseks valiti Thonny, mis on Pythoni koodiredaktor. Thonny installeerimisfaili leidmiseks oli juhendisse lisatud link Thonny kodulehele (joonis 6.9). [19]

Lisaks oli vaja alla laadida MicroPythoni *firmware*, et oleks võimalik kirjutada arendusplaadi jaoks programme kasutades Thonnyt, sobiva faili allalaadimiseks oli lisatud link MicroPythoni kodulehele ESP8266 vahelehele (joonis 6.9) [21].

Järgmiseks sammuks on alla laadida Pythoni IDE Thonny, kui seda veel arvutis pole (https://thonny.org/). Laadida alla Windowsile sobiv installeerimisfail, see käivitada ja installeerida Thonny.

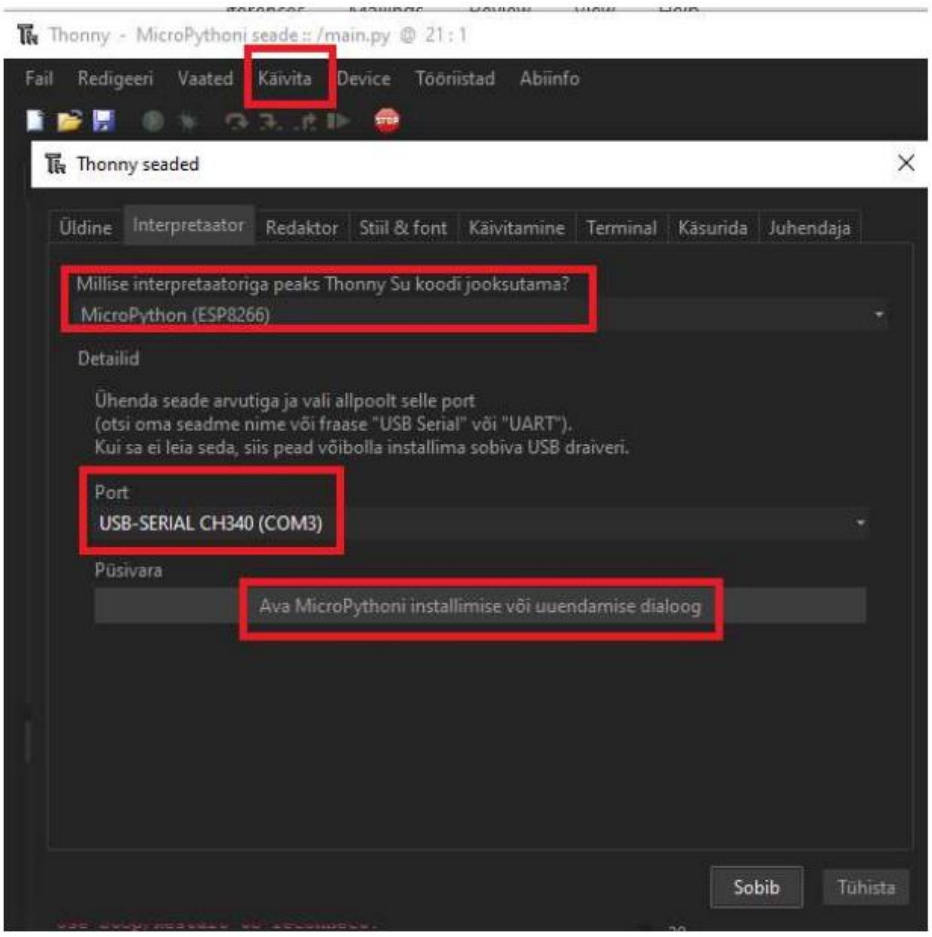
Pärast seda tuleb alla laadida kasutatavale plaadile vastav MicroPythoni *firmware* .bin failina (https://micropython.org/download/esp8266/) (kõige uuem versioon).

2. **MicroPythoni *firmware* üleslaadimine plaadile.**

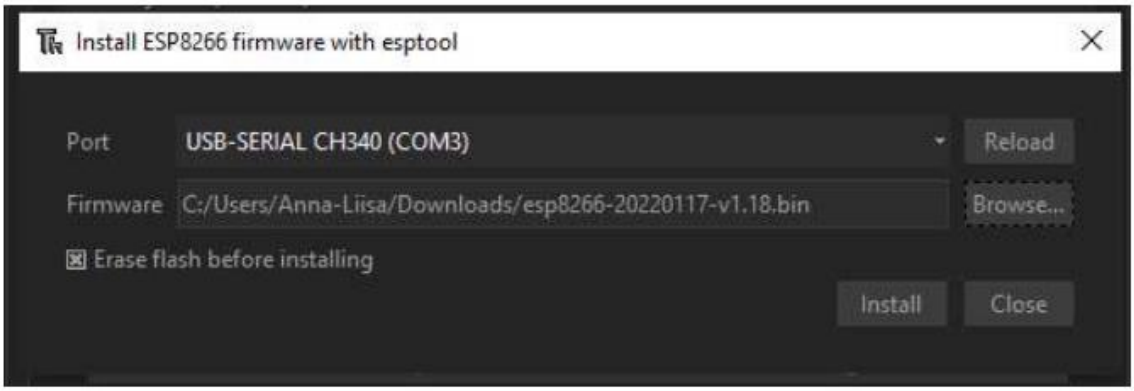
Avada Thonny IDE. Esialgse seadistuse käigus saab valida keele. Juhendi koostamisel on kasutatud eestikeelset Thonnyt.

Joonis 6.9. Thonny koodiredaktori ja MicroPythoni *firmware*-i allalaadimise juhised juhendis

Kui vajalikud draiverid ja programmid olid installeeritud, sai alustada MicroPythoni *firmware*'i paigaldamisega arendusplaadile ESP8266. Selleks oli vaja avada koodiredaktor Thonny IDE ja valida programmi seadetest interpretaator. Interpretaatoriks tuli valida „MicroPython (ESP8266)" ja sai valida ka pordi, mida arendusplaat kasutab arvutiga suhtlemiseks (joonis 6.10). Seejärel tuli avada MicroPythoni installeerimise aken (joonis 6.11) ja seal valida port, mida arendusplaat kasutab ning valida varem alla laaditud .bin fail MicroPythoni *firmware*'iga ja see installeerida arendusplaadile.



Joonis 6.10. Interpretaatori valimine ja MicroPythoni installeerimisakna avamine Thonny seadetes, vajalikud sammud näidatud



Joonis 6.11. MicroPythoni installeerimise aken Thonnys

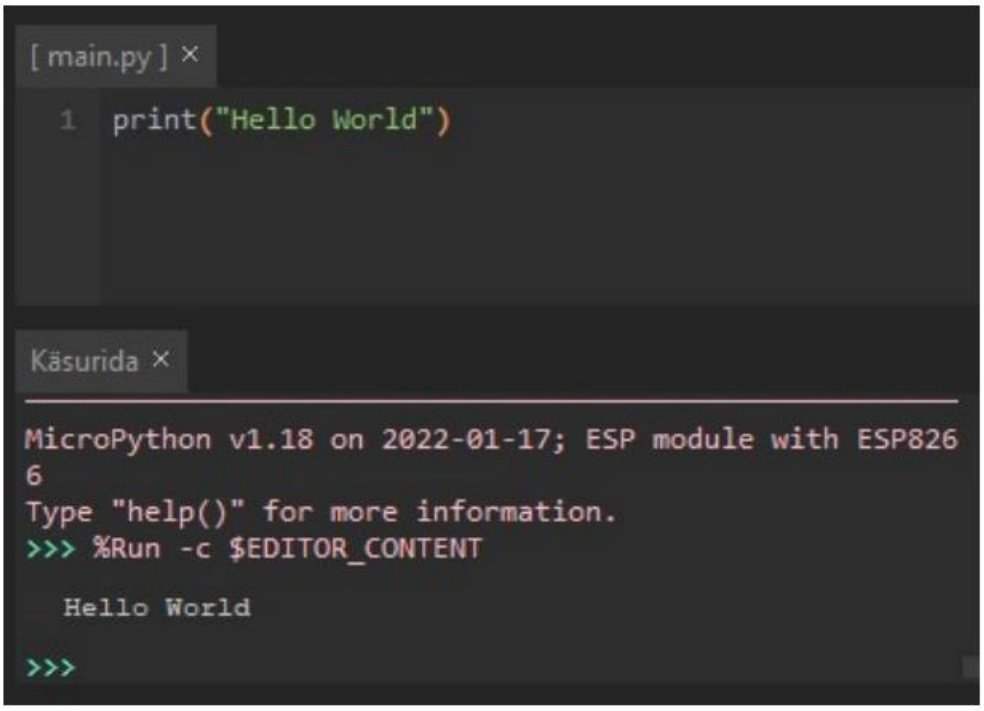
6.3.2 Hello World

Enne programmi koostamist tutvustati juhendis programmi käivitamist ja peatamist (joonis 6.12) Thonnys ning selgitati, millal peaks salvestama programmi arvutisse ja millal MicroPythoni seadmele.

Thonnys saab programmi käivitada vajutades rohelist nuppu „Jooksuta käesolev skript" või vajutada klaviatuuril F5. Programm salvestub automaatselt ka selle käivitamisel. Programmi töö lõpetamiseks on punane nupp „Peata/taaskäivita interpretaator" või klahvikombinatsioon Ctrl+F2.

Joonis 6.12. Programmi käivitamise ja peatamise juhised

Uute programmeerimiskeeltega tutvudes katsetatakse tihtipeale esimese programmina fraasi „Hello World" käsureale väljastamist ning seda kasutati ka käesoleva töö tulemusena valmivas praktikumitöös (joonis 6.13).

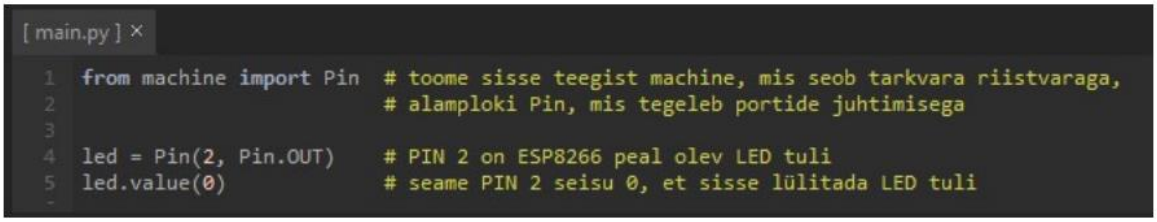


Joonis 6.13. „Hello World" programm Thonnys

6.3.3 Arendusplaadil olev LED

Juhendis selgitati, et arendusplaadile ESP8266 on sisseehitatud LED, mida on võimalik lülitada sisse ja välja seades PIN 2 seisuks kas „0" või „1". Muudetakse PIN seisu, kuna LED-tuli on ühendatud PIN 2 ja toiteklemmi vahele.

Juhendis oli ette näidatud programm, mida koostada (joonis 6.14), et lülitada sisse LED-tuli ESP8266 arendusplaadil. Pärast seda oli ülesandeks LED-tuli taas välja lülitada.



Joonis 6.14. Juhendis näidatud ülesande lahendus

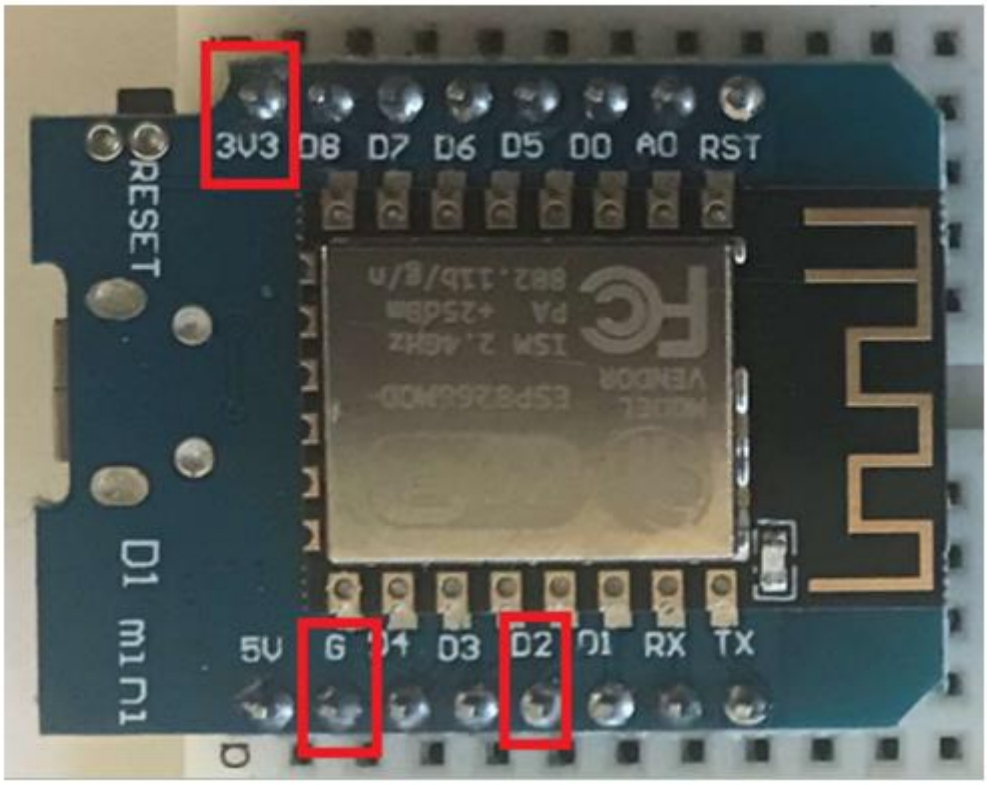
6.3.4 LEDi vilgutamine

LED-tule vilgutamise ülesandes toodi sisse ajafunktsioonide kasutus, et oleks võimalik programmi korrata mingi intervalli tagant. Juhendis oli ette näidatud programm, mis lülitab sisse LED-tule 0,5 sekundiks ning seejärel lülitab välja samaks ajavahemikuks. Kuna see oli praktikumis esimene ülesanne, kus kasutati funktsioone, oli ka välja toodud fakt, et Python-keeles programmeerides peavad funktsioonisisesed read olema taandatud täpselt samale kaugusele, et programm oskaks neid korrektselt lugeda. Vastasel juhul võib saada veateate või ebakorrektselt toimiva programmi.

6.3.5 DHT11

Enne ülesande kirjeldust oli uuesti välja toodud reegel, et ESP8266 arendusplaadiga ei tohi teha ühendusi samal ajal, kui see on ühendatud arvutiga.

DHT11 anduri ühendamine ESP8266 arendusplaadiga oli välja toodud sammudena klemmide kaupa ja vajalikud klemmid olid näidatud joonisega 6.15. Ühendati ESP8266 maaklemm DHT11 maaklemmiga. Andmeedastuseks ühendati DHT11 anduril DATA klemm ESP8266 arendusplaadi klemmiga D2. Seejärel ühendati toiteklemmid – DHT11 anduril VCC ja ESP8266 arendusplaadil 3V3 (3,3V). Selle sammu juures oli ka selgitatud, et DHT11 andurit ei tohi ühendada ESP8266 arendusplaadi klemmiga 5V, kuna andmeside ühildamiseks peavad mikrokontroller ja andur töötama samal toitepingel ja ESP8266 toitepinge on 3,3V.



Joonis 6.15. DHT11 ühendamiseks ESP8266-ga vajalikud PIN-id

Pärast klemmide edukat ühendamist võis ESP8266 taaskord arvutiga ühendada. Et Thonny ja arendusplaadi vahel taasluua ühendus, tuli vajutada nuppu „Peata/taaskäivita interpretaator" või klahvikombinatsiooni Ctrl+F2.

Kuna erinevalt eelnevalt kirjutatud programmidele kasutati ettenäidatud programmis teekide sissetoomiseks „import machine", mitte „from machine import Pin", selgitati enne programmi ära nende erinevus. Seejärel katsetati ettenäidatud programmi (joonis 6.16), mis mõõtis DHT11 kasutades hetke õhuniiskust ja -temperatuuri ning väljastas need käsureale. Programmi loomiseks vajalikud käsud leiti MicroPythoni kasutusjuhendist [22].

[main.py] ×

```
1  import dht                    # vajalik DHT teegi sidumiseks koodiga
2  import machine
3  d = dht.DHT11(machine.Pin(4)) # ühendasime DHT11 infoedastuse juhtme ESP8266 arendusplaadil
4                                # PIN 4-ga, mis on arendusplaadil tähistatud kui „D2"
5
6  d.measure()                   # DHT mõõdab hetkeväärtused
7  print(d.temperature(), "C")   # väljastab mõõdetud temperatuuri Celsiuse kraadides
8  print(d.humidity(), "%")      # väljastab mõõdetud õhuniiskuse protsentides
```

Joonis 6.16. Ülesandes loodav programm

6.3.6 DHT11 mõõtmistulemustega graafiku loomine

Ülesandes oli plaanis kasutada lõpmatut tsüklit mõõtmistulemuste väljastamiseks mingi intervalli tagant, mistõttu enne ülesande juurde asumist oli välja toodud, et DHT11 kahe mõõtmise vahel ei tohi olla aega vähem kui 1 sekund, kuna see ei suuda kiiremini sooritada uut mõõtmist.

Enne ülesande alustamist oli ka juhendatud, kuidas lülitada Thonny IDE-s sisse plotter, et oleks võimalik mõõtmistulemusi jälgida graafikult, mille Thonny ise koostab. Juhendis selgitati, kuidas plotter arvestab erinevaid väljastatud arve, sõltuvalt nende taga olevast ühikust (või selle puudumisest) ja tulemuste väljastamise meetodist – kõik mõõtmistulemused väljastatult 1 reana või iga tulemus eraldi print() funktsiooniga.

Juhendis oli ka soovitatud DHT11 katsetamisel hingata andurisse, mis suurendab lühiajaliselt temperatuuri ja õhuniiskust anduri juures. See kajastub anduri poolt väljastatavates mõõdistes.

Programm, mis väljastab mõõtmistulemusi korduvalt ja selliselt vormistatuna, et sellest tekiks graafik Thonnys, oli juhendis ette näidatud. See programm on näidatud ka joonisel 6.17.

Joonis 6.17. Programm koos mõõtmistulemuse ja nende põhjal plotteris koostatud graafikuga

6.3.7 Iseseisev ülesanne

Praktikumi viimaseks ülesandeks oli koostada ise programm, mille jaoks vajalikud teadmised omandati eelnevaid ülesandeid lahendades. Ülesanne oli koostada programm, mis mõõdab DHT11 kasutades õhuniiskust ning kui niiskuse tase tõuseb üle 50%, lülitatakse sisse arendusplaadil ESP8266 olev LED ning õhuniiskuse taseme langedes 50 protsendini või alla selle, lülitatakse LED taaskord välja.

Juhendi lõpetuseks paluti tudengitel täita tagasisideküsitlus praktikumi ja selle korralduse kohta, mille tulemused on välja toodud järgmises peatükis.

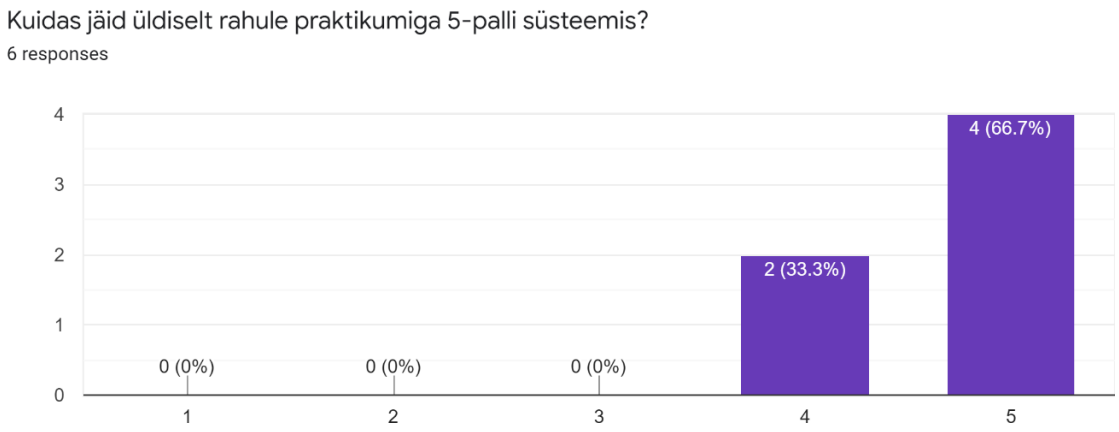
7. KATSETAMINE JA TAGASISIDE

Praktikumi katsetati vabatahtlike küberfüüsikaliste süsteemide eriala tudengitega. Suurem osa vabatahtlikest, neli tudengit kuuest, olid esimese kursuse tudengid, ülejäänud kaks olid neljandal kursusel. Praktikumi viidi läbi 22. aprillil 2022. aastal Tallinna Tehnikaülikooli Tartu kolledži hoones aadressil Puiestee 80A, Tartu ruumis A205 ja võttis aega ligikaudu tund. Enne praktikumi läbiviimist viidi ruumi kõik vajalikud komponendid, käivitati arvutid ja laeti alla praktikumijuhendid.

Praktikumi viis läbi töö autor iseseisvalt juhendaja rollis. Praktikumi alguses tutvustas juhendaja osalejatele praktikumi ja selle loomise eesmärke ning mis tudengeid praktikumis ees ootab. Tutvustati kasutatavaid seadmeid ja lasti tudengitel endale vajalikud komponendid valida. Osalenud tudengid küsisid praktikumi jooksul aktiivselt küsimusi praktikumi, ülesannete ja kasutatavate komponentide kohta.

Praktikumi läbiviimine oli edukas – kõik seadmed töötasid korrapäraselt ja kõik osalenud tudengid läbisid praktikumi ning andsid tagasisidet nii praktikumi korralduse kui ka juhendi kohta. Kõik tudengite esitatud küsimused said ka vastused.

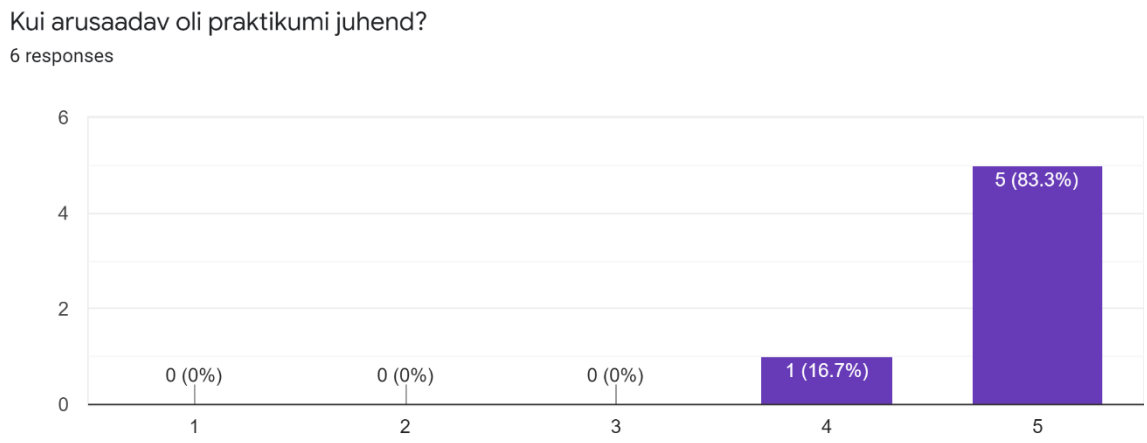
Kuidas jäid üldiselt rahule praktikumiga 5-palli süsteemis?
Esimese küsimusega sooviti teada saada tudengite rahulolu läbitud praktikumi suhtes. Kuuest vastanust neli valis vastuseks 5, ehk „jäin väga rahule" ja kaks valis vastuseks 4, mida võib lugeda kui üsna rahule jäämist. Antud vastuste põhjal võib järeldada, et üldpildis jäid tudengid praktikumiga rahule.



Joonis 7.1. „Kuidas jäid üldiselt rahule praktikumiga 5-palli süsteemis?" küsimuse vastuste jaotus

Kui arusaadav oli praktikumi juhend? Kas ja mis oleks võinud juhendis teisiti olla?

Nende küsimuste põhjal sooviti teha järeldusi, kas praktikumijuhend oli tudengite jaoks piisavalt arusaadav ning saada tagasisidet, mille põhjal edaspidi juhendit muuta või täiendada.Kuuest vastanust viie jaoks oli juhend „väga arusaadav" ehk skaalal 5, ühe tudengi jaoks ei olnud see täiesti arusaadav. Tudengi põhjenduseks oli, et ühes ülesandes oleks võinud olla veel lisaks välja toodud, et on vaja importida üks teek, mis tal endal jäi kahe silma vahele ja seetõttu anti programmi käivitamisel veateade. Suurema osa tudengite tagasiside põhjal oli juhend väga lihtsasti arusaadav.

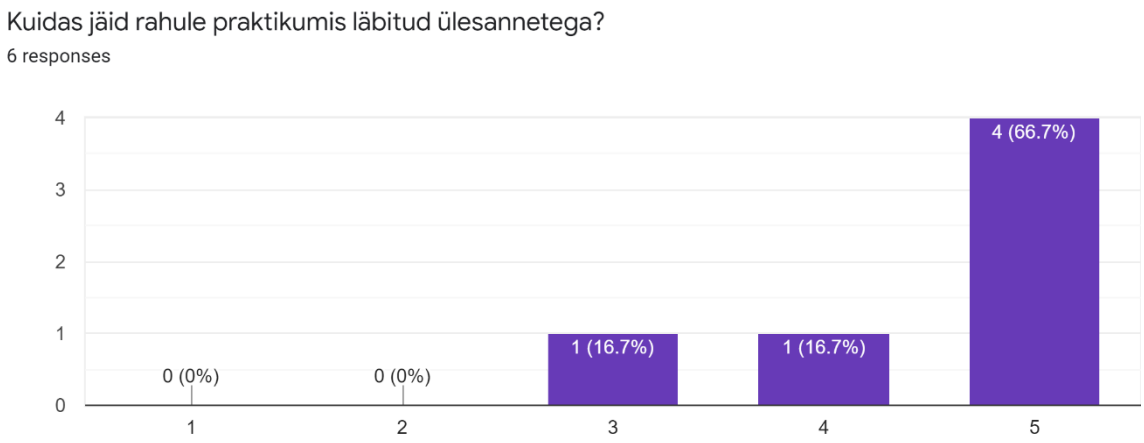


Joonis 7.2. „Kui arusaadav oli praktikumi juhend?" küsimuse vastuste jaotus

Joonis 7.3. Küsimuse „Kas ja mis oleks võinud juhendis teisiti olla?" vastused

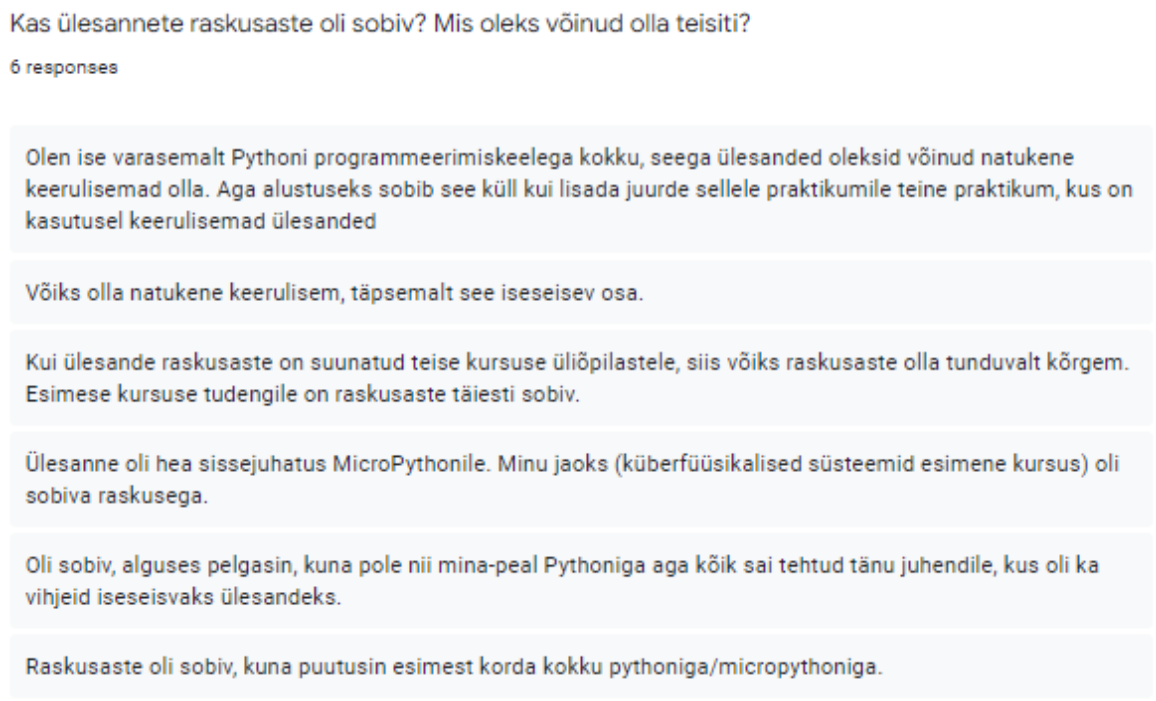
Kuidas jäid rahule praktikumis läbitud ülesannetega? Kas ülesannete raskusaste oli sobiv? Mis oleks võinud olla teisiti?

Nende küsimustega soovis autor teada saada, kas tudengite meelest olid ülesanded ja nende keerukus sobilikud. Kuuest tudengist valdav osa, ehk 4 tudengit valisid vastuseks 5 ehk „jäin väga rahule", üks tudeng valis 4 ja üks valis 3, ehk jäi kahevahele.



Joonis 7.4. Küsimuse „Kuidas jäid rahule praktikumis läbitud ülesannetega?" vastuste jaotus

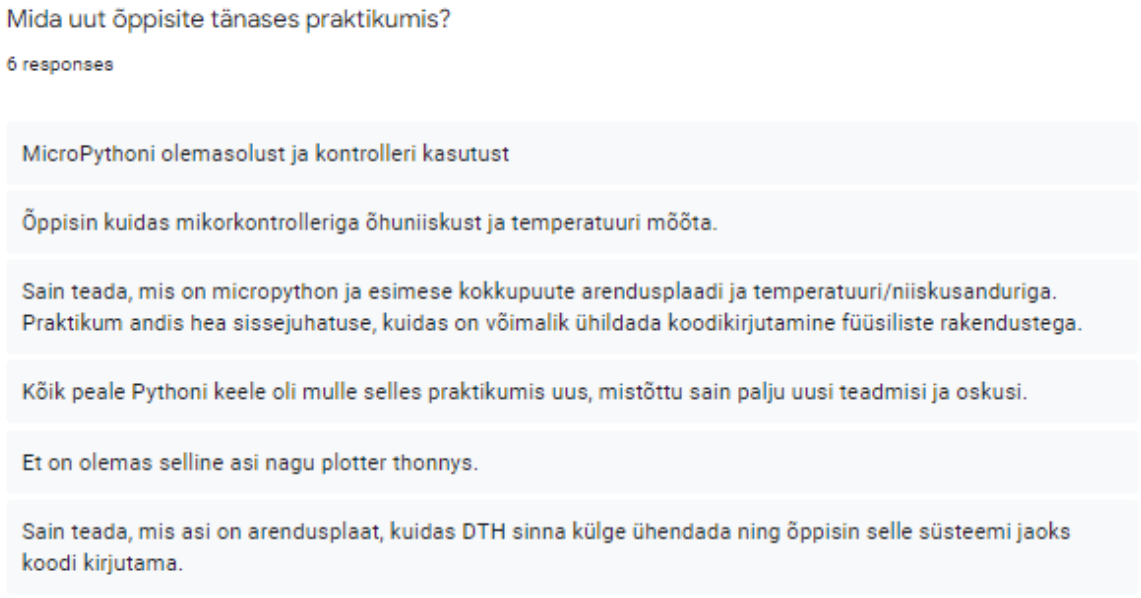
Ülesannete raskusastme kohta oli mitmeid erinevaid vastuseid. Nende jaoks, kes olid varem Pythoniga kokku puutunud ja sellest veel midagi mäletasid, olid programmeerimisülesanded veidi liiga lihtsad. Üldises pildis tundus, et ülesanded oleksid võinud olla raskemad, kuid autori arvates on need sissejuhatuseks MicroPythonisse sobilikud. Olenevalt praktikumis osalejate ettevalmistusest elektroonika ja programmeerimiskeele Python valdkondades, võib praktikumi edasiarendamiseks juhendis olevate ülesannete keerukust tõsta.



Joonis 7.5. Küsimuse „Kas ülesannete raskusaste oli sobiv? Mis oleks võinud olla teisiti?" vastused

Mida uut õppisite tänases praktikumis?

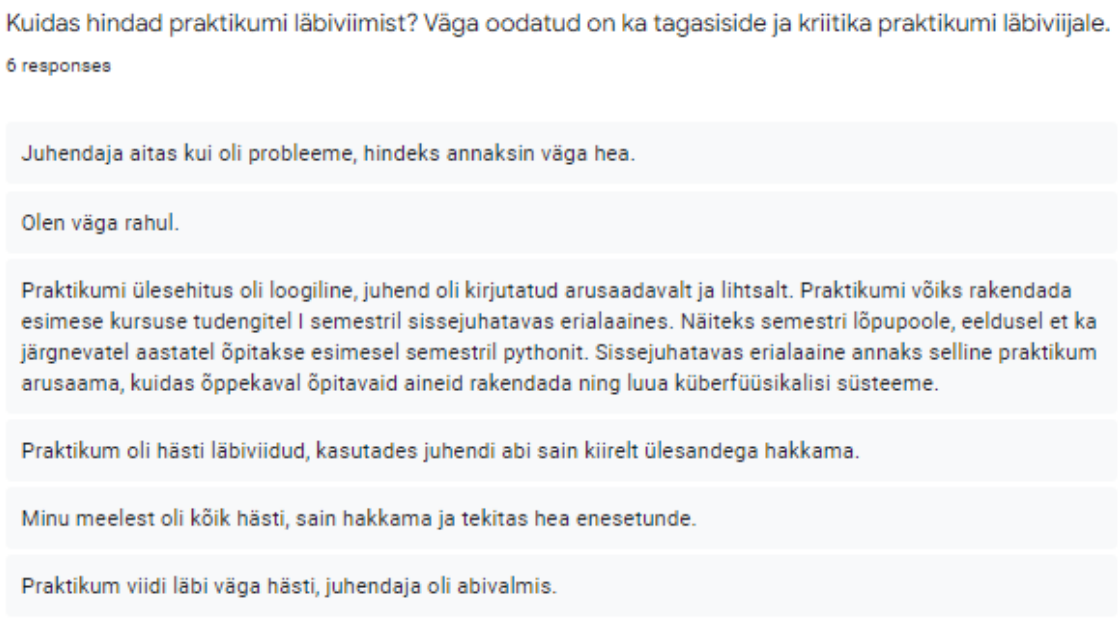
Kuigi küsimus „Mida uut õppisite?" oli tagasisides kohustuslik, siis ükski tudeng ei vastanud „mitte midagi", vaid kõik õppisid midagi uut. Mõned tudengid kuulsid esmakordselt MicroPythoni olemasolust, mõned kasutasid esimest korda õhuniiskuse ja -temperatuuriandurit DHT.



Joonis 7.6. Küsimuse „Mida uut õppisite tänases praktikumis?" vastused

Kuidas hindad praktikumi läbiviimist?

Tudengite sõnul oli praktikumi ülesehitus loogiline ja arusaadav, juhendaja abivalmis. Üks tudengitest arvas, et sellist praktikumitööd võiks läbi viia juba esimesel kursusel, andes juba varakult praktilist kogemust.



Joonis 7.7. Küsimuse „Kuidas hindad praktikumi läbiviimist?" vastused

KOKKUVÕTE

Käesoleva lõputöö lähteülesandeks oli koostada praktikumitöö TalTech Tartu kolledži mikroprotsessorsüsteemide õppeaine tarbeks. Praktikum pidi sisaldama sissejuhatust mikrokontrollerite programmeerimisse programmeerimiskeeles MicroPython. Vajalik oli välja tuua ka vajalike programmide ja draiverite installeerimislingid ja -juhised.

Praktikum koosnes kasutatavate seadmetega tutvumisest ja nende katsetamisest. Praktikumijuhend algas MicroPythonit, arendusplaati ESP8266 NodeMCU D1 ning õhuniiskuse ja -temperatuuriandurit DHT11 tutvustades. Seejärel tutvustati praktikumi eesmärke ja lahendati ülesandeid. Praktikumi viimane ülesanne oli iseseisev. Praktikumi katsetati Tartu kolledži arvutiklassis vabatahtlike tudengitega, kes andsid pärast seda ka tagasisidet, mille põhjal sai loodud praktikumitööd analüüsida ja teha järeldusi. Tudengite tagasisidet koguti Google Forms keskkonnas.

Saadud tagasiside põhjal saab järeldada, et tudengid jäid praktikumi läbiviimise, juhendi ja ülesannetega rahule. Kõik osalenud tudengid läbisid kõik ülesanded ja sooritasid praktikumi edukalt. Praktikumi eduka läbiviimise põhjal võib väita, et loodud praktikumitöö on jõukohane nii 1. kursuse tudengitele, kuid ka vanematele kursustele, kes pole varem Pythonis programmeerinud. Võib järeldada, et juhend oli piisavalt põhjalik ja detailne. Tudengite vastustest selgus, et ülesanded oleksid võinud keerulisemad olla, kuid autori arvates ongi selline ülesannete tase sobilik sissejuhatuseks uude programmeerimiskeelde. Seda kinnitab ka fakt, et kõik osalenud tudengid õppisid antud praktikumis midagi uut.

Muuhulgas sai ka autor kasuliku kogemuse, autor sai võimaluse sügavamalt tutvuda MicroPythoniga ja ESP8266 NodeMCU D1-ga. Sellele lisaks sai esmakordse kogemuse praktikumitöö tarbeks juhendi koostamisest ja läbiviimisest, olles praktikumi katsetamisel iseseisvalt juhendaja rollis.

Töö lähteülesandeks seatud eesmärk koostada MicroPythonit tutvustav praktikum ja seda tudengitega läbi proovida, sai edukalt täidetud. Loodud praktikumitöö edasiarendusena võib koostatud ülesannete keerukust tõsta vastavalt osalejate taustale ja ettevalmistusele selles valdkonnas.

SUMMARY

The aim of this thesis was to create a laboratory work for Microprocessor systems course at TalTech Tartu college. The laboratory work had to include introduction to programming of microcontrollers in MicroPython language. It was necessary to provide installation links and instructions for the programs and drivers needed in the laboratory work.

The laboratory work consisted of getting to know and testing the equipment used and. The guide for the laboratory work began with an introduction to MicroPython, development board ESP8266 NodeMCU D1 and air humidity and temperature sensor DHT11. After that the participants were introduced to the purposes of the laboratory work, they performed the programming tasks shown in the guide. The last task was solved independently. The laboratory work was tested in the computer lab of Tartu college, with volunteer students of cyberphysical systems. The students gave feedback after testing and this feedback was used to analyse the laboratory work and make conclusions. Student feedback was collected in Google Forms environment.

Based on the feedback received, it can be concluded that the students were satisfied with the laboratory work, the instructions and the tasks. All participating students completed the assignments and the laboratory work successfully. Based on the feedback the laboratory work was suitable for first year students, but also for older courses who have not previously programmed in Python. It can be concluded guide was sufficiently detailed and thorough. The students' answers revealed that the tasks could have been more complicated, but the author thinks the difficulty level of the tasks is suitable for introducing a new programming language to students. This is also confirmed by the fact that all the participating students learned something new in this laboratory work.

Among other things, the author also gained useful knowledge by having the opportunity to get acquainted with MicroPython and ESP8266 NodeMCU D1. In addition the author gained a first-hand experience of creating a guide for the laboratory work and being an independent tutor during the laboratory work
The goal set in the beginning of this thesis was successfully accomplished. The author created a laboratory work introducing MicroPython and successfully carried it out on volunteers.

KASUTATUD KIRJANDUSE LOETELU

1. G. Gridling and B. Weiss, "Introduction to Microcontrollers," 2007.
2. D. Ibrahim, "ARM Cortex microcontroller development boards," *Arm-Based Microcontroller Multitasking Projects*, pp. 33–45, Jan. 2021, doi: 10.1016/B978-0-12-821227-1.00003-7.
3. "ESP8266: General Information & Specifications - ESP8266 Shop." https://esp8266-shop.com/esp8266-guide/esp8266-information/
4. "8-bit AVR® Microcontroller with 16/32K Bytes of ISP Flash and USB Controller ATmega16U4 ATmega32U4." http://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/ATMega32U4.pdf
5. "Arduino Pro Micro 5V 16MHz [Generic] :: Micro JPM." https://www.microjpm.com/products/ad36475/
6. "MicroPython - Python for microcontrollers." https://micropython.org/
7. "Python with Arduino Boards | Arduino Documentation | Arduino Documentation." https://docs.arduino.cc/learn/programming/arduino-and-python
8. "WEMOS D1 R2 WIFI ESP8266 Shield Arduino Compatible" https://www.makershop.de/download/d1-wifi-esp8266-board.pdf
9. "ESP8266 WeMos D1 Mini Tutorial." https://diyi0t.com/esp8266-wemos-d1-mini-tutorial/
10. L. Kane, J. Chen, R. Thomas, V. Liu and M. McKague " Security and Performance in IoT: A Balancing Act" https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9133521
11. "DHT11 SIP Packaged Temperature and Humidity Sensor(Discontinued, Replaced By DHT20) -Sensor-Temperature and Humidity-Guangzhou Aosong Electronic Co., Ltd." http://www.aosong.com/en/products-21.html
12. R. Santos and S. Santos, „MicroPython Programming with ESP32 and ESP8266", 2022.
13. "Overview | MicroPython Basics: What is MicroPython? | Adafruit Learning System." https://learn.adafruit.com/micropython-basics-what-is-micropython
14. "MicroPython libraries — MicroPython 1.18 documentation." https://docs.micropython.org/en/latest/library/index.html
15. "Interacting With Python – Real Python." https://realpython.com/interacting-with-python/
16. "About Mu." https://codewith.mu/en/about
17. "Mu: A Python Code Editor — Mu 1.1.1 documentation." https://mu.readthedocs.io/en/latest/

18. "MicroPython IDEs for ESP32 and ESP8266 | Random Nerd Tutorials." https://randomnerdtutorials.com/micropython-ides-esp32-esp8266/
19. "Thonny, Python IDE for beginners." https://thonny.org/
20. "How to Install CH340 Drivers - learn.sparkfun.com." https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all#drivers-if-you-need-them
21. "MicroPython - Python for microcontrollers." https://micropython.org/download/esp8266/
22. "13. Temperature and Humidity — MicroPython 1.18 documentation." https://docs.micropython.org/en/latest/esp8266/tutorial/dht.html

LISAD

Lisa 1 Praktikumijuhend

MicroPython praktikum

Tänases praktikumis tutvume MicroPythoniga, mis on mõeldud arendusplaatidel kasutamiseks. Katsetame seda arendusplaadil ESP8266 ja koodiredaktorina kasutame Thonnyt, mis on Pythoni IDE.

MicroPython on arendatud programmeerimiskeelest Python ja sisaldab väikest osa Pythonis olevatest teekidest. See on interpretaatorkeel ja kuna koodi silumise faasis saab programmi tööle panna otse IDE-s, siis ei pea programmi kompileerima ja üles lugema. MicroPythonis on võimalik töötada otsekäskude režiimis, mis tähendab, et käsk täidetakse kohe või antakse veateade, mis kiirendab arendustööd ja kaitseb kontrollerite välkmälu (Flash memory) sagedase ümberkirjutamise eest, mida mikrokontrollerid taluvad vaid piiratud arvu kordi.

ESP8266 on mikrokontroller, mille arendamiseks on palju erinevaid võimalusi. Sellel on olemas WiFi võimekus, mis võimaldab edastada andmeid interneti teel.

ESP8266 kasutamine on levinud ja seetõttu on internetis saadaval suurel hulgal erinevaid materjale selle kasutamise ja võimaluste kohta. Lisaks on võimalik leida mitmeid lahendusi erinevatele probleemidele, mis selle kasutamise käigus tekkida võivad. Suure kasutajabaasi tõttu on ESP8266 arendusplaadil MicroPythoni *firmware* paremini läbi proovitud ja silutud, kui paljude sarnaste mikrokontrollerite puhul.

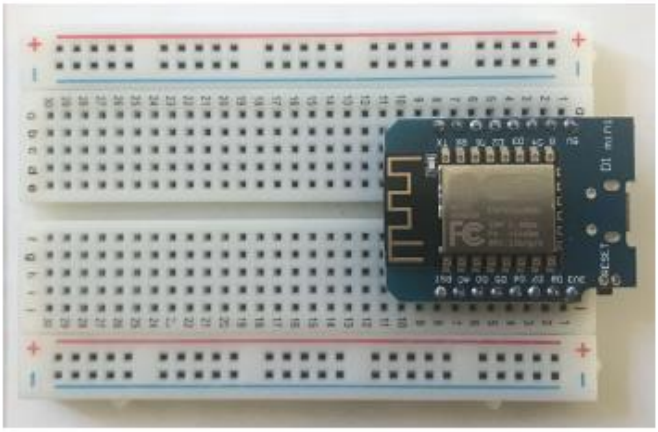
Kasutame praktikumis ka temperatuuri- ja niiskuseandurit DHT11, mida on lihtne ühendada ja kasutada erinevate mikrokontrolleritega. See töötab pingel 3 kuni 5 V, praktikumis kasutame 3,3V.

Praktikumi tegevused:

1. Tutvumine arendusplaadiga ESP8266, selle ühendamine arvutiga ja vajalike draiverite installeerimine
2. Arendusplaadi sidumine Thonny IDEga, Thonnyga tutvumine
3. MicroPythoni *firmware* üleslaadimine arendusplaadile
4. MicroPythoni võimalustega tutvumine ESP8266 arendusplaadil

Praktikumi komplekt

1. Maketeerimisplaadile monteeritud ESP8266



2. USB juhe (Micro USB-USB A)

3. DHT11

4. 3x F-M Duppont juhtmed (female to male, eesti keeles pesa-pistik)

2

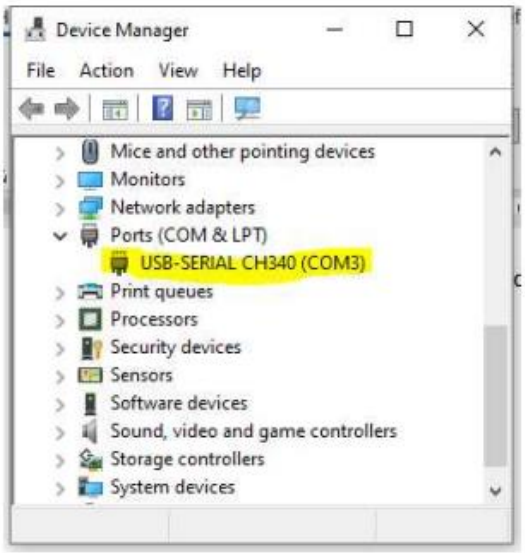
Reeglid

- Mitte teha ühendusi samal ajal kui seade on ühendatud arvutiga
- DHT11 ühendada 3V3 PINiga, mitte 5V

Praktikumi ülesanded

1. Ettevalmistus

Vajadusel lülitada sisse arvuti ja seejärel ühendada arendusplaat kasutades USB – Micro USB juhet. Plaadi ühendust on võimalik kontrollida Device Managerist. Kui plaadil on draiver olemas, ilmub see COM portide alla nimega USB-SERIAL CH340.



Kui arendusplaadil pole draiverit, ilmub arendusplaat kategooriasse *Other Devices* ja draiver tuleb laadida alla internetist ning paigaldada.

Draiveri paigaldamine (vajadusel)

Võib kasutada Google otsingut märksõnadega „CH340 driver" või laadida draiver alla SparkFun leheküljelt. (https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all#drivers-if-you-need-them)

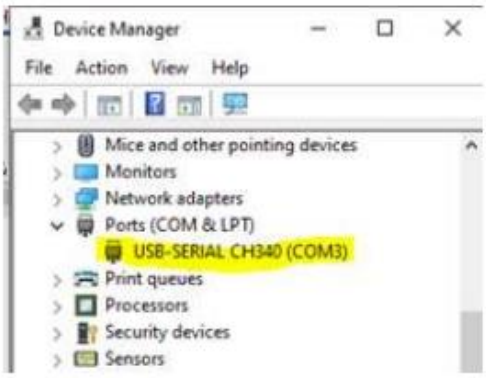
Kui alla laadida .exe fail, võib selle kohe käivitada.

Kui laadida alla .zip fail, siis see tuleb lahti pakkida ja käivitada kaustas olev fail SETUP.EXE.

.exe faili käivitades avaneb aken DriverSetup, seal vajutada nupule INSTALL.

Kui avaneb aken kirjaga „Driver install success!", võib edasi liikuda järgmiste sammude juurde.

Kui draiver on edukalt installeeritud, on arendusplaati näha Device Manageris COM portide all nimega USB-SERIAL CH340 ja nime järel sulgudes on kirjas, millist porti see kasutab.



Järgmiseks sammuks on alla laadida Pythoni IDE Thonny, kui seda veel arvutis pole (https://thonny.org/). Laadida alla Windowsile sobiv installeerimisfail, see käivitada ja installeerida Thonny.

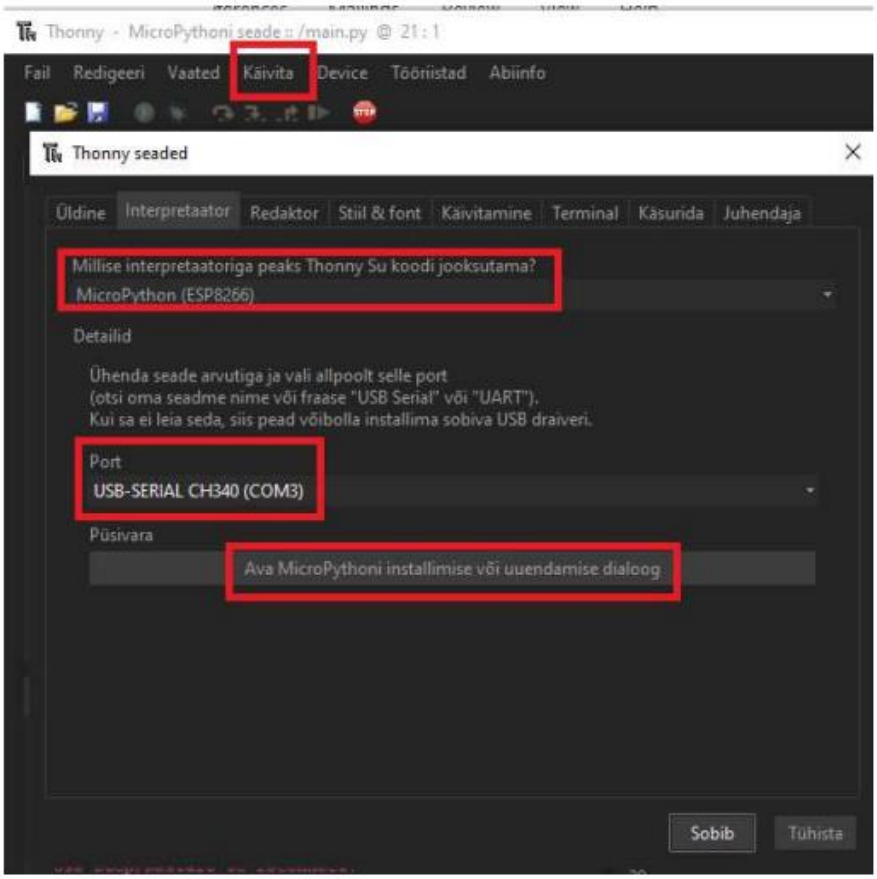
Pärast seda tuleb alla laadida kasutatavale plaadile vastav MicroPythoni *firmware* .bin failina (https://micropython.org/download/esp8266/) (kõige uuem versioon).

2. MicroPythoni *firmware* üleslaadimine plaadile.

Avada Thonny IDE. Esialgse seadistuse käigus saab valida keele. Juhendi koostamisel on kasutatud eestikeelset Thonnyt.

Esimeseks sammuks on valida menüüribalt „Käivita" ja avanenud rippmenüüst avada esimene valik „Vali interpretaator…".

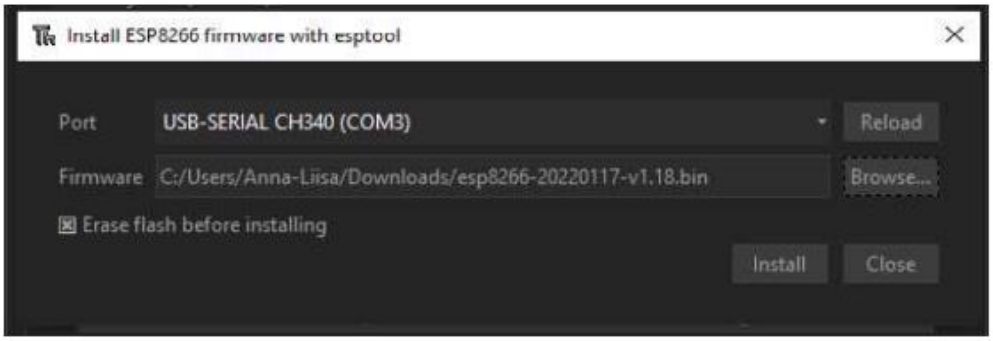
Interpretaator valida „MicroPython (ESP8266)". Pordi valiku võib jätta automaatsele või valida rippmenüüst. Valikuks peaks pakkuma sama porti, mida Device Manageris näha oli.



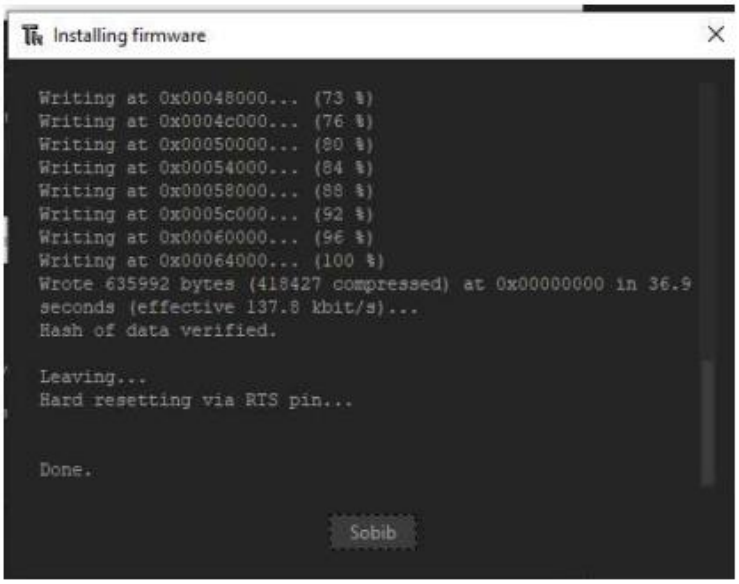
Kui need valikud on tehtud, vajutada nupule „Ava MicroPythoni installimise või uuendamise dialoog".

Avanenud aknas valida port, mida ESP8266 kasutab. Firmware valikuks vajutada nuppu „Browse" ja valida varem alla laetud MicroPythoni .bin fail.

Valiku „Erase flash before installing" ees peaks olema linnuke.



Seejärel vajutada „Install" ja oodata kuni tuleb teade, et protsess on lõpule viidud.



3. MicroPythoni katsetamine

a) Hello World

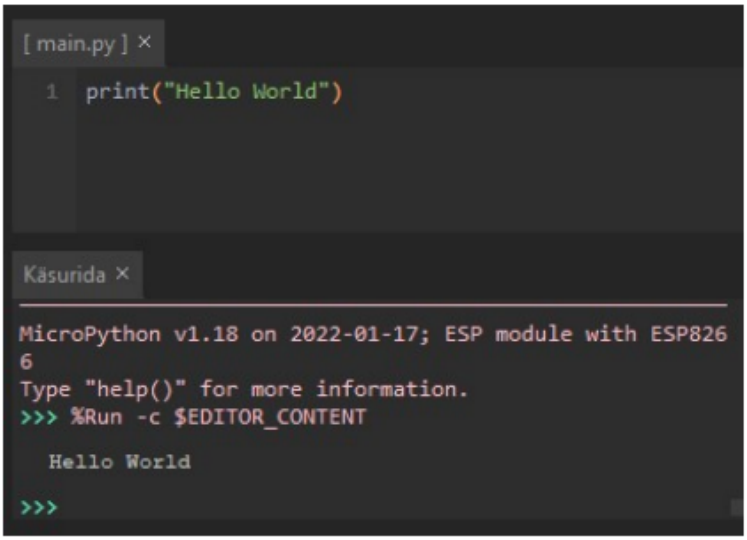
Thonnys saab programmi käivitada vajutades rohelist nuppu „Jooksuta käesolev skript" või vajutada klaviatuuril F5. Programm salvestub automaatselt ka selle käivitamisel. Programmi töö lõpetamiseks on punane nupp „Peata/taaskäivita interpretaator" või klahvikombinatsioon Ctrl+F2.

Erinevate programmeerimiskeeltega alustades on tavaks alustada lihtsa käsu proovimisega ja tihtipeale kasutatakse selleks fraasi „Hello World" väljastamist.

Programmi võib salvestada arvutisse, et säästa mikrokontrolleri välkmälu ja faili nimi pole oluline.

Kui on vajadus programmi käivitada mikrokontrolleril iseseisvalt, tuleks see salvestada MicroPythoni seadmele ja selle nimeks panna „boot.py" või „main.py", kuid tänases praktikumis seda vaja ei lähe.

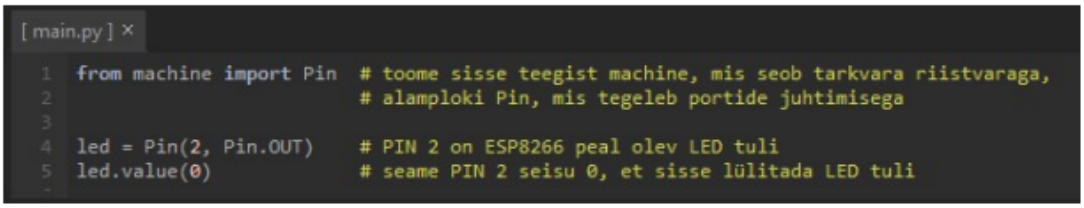
Kirjutame programmi:



b) LED tuli

ESP8266 arendusplaadil on sissehitatud LED tuli, mis on ühendatud PIN 2 ja toiteklemmi vahele. LED tuld saab sisse lülitada seades PIN 2 seisu „0" ja välja lülitada seades PIN 2 seisu „1".

Kirjutame programmi, mis lülitab sisse LED tule ESP8266 arendusplaadil. Uue programmi võib kirjutada samasse faili eelmise programmi asemele.



Seejärel muuta programmi, et LED tuli välja lülitada

c) LED vilgutamine

Kasutades ajafunktsiooni „sleep()" saame programmi panna ootele enne järgmise sammu sooritamist ja korrata tsüklit mingi intervalliga. Kirjutame programmi, mis vilgutab LED tuld iga 0.5 sekundi tagant.

Pythonis on oluline, et funktsiooni sees olevad read oleksid taandatud täpselt samale kaugusele.

```
[ main.py ] ×
1  from machine import Pin
2  from time import sleep        # toome sisse ajafunktsioonide teegi
3
4  led = Pin(2, Pin.OUT)         # muutuja led seotakse PIN 2-ga, mis on seatud väljundiks
5  led.value(0)                  # muutuja LED väärtuseks seatakse 0
6
7  while True:                   # lõpmatu tsükkel
8    led.value(not led.value()) # muutuja LED väärtus muudetakse vastupidiseks (0->1 ja 1->0)
9    sleep(0.5)                  # ootamise aeg (sekundites) enne tsükli kordamist
```

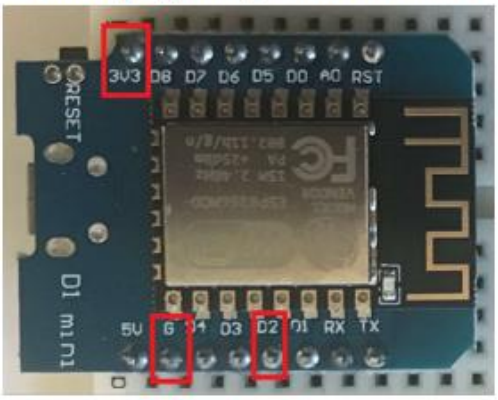
d) DHT11

DHT11 ja arendusplaadi vahel ühenduste tegemise ajaks tuleb kindlasti arendusplaat arvutist lahti ühendada!

DHT ühendamine:

Ühendamiseks on 3 F-M (female to male ehk pesa-pistik) juhet. Pesadega otsad saame ühendada DHT külge ja pistikud saame ühendada ESP8266 külge kasutades maketeerimisplaati. Juhtmete värvid ei ole olulised. PINide, mida kasutame DHT11 ühendamiseks, asukohad on näidatud järgneval joonisel.

1. Ühendame maaklemmi - ground ja ground (DHT11 anduril GND, ESP8266 arendusplaadil G)
2. Ühendame andmeedastuseks DHT11 anduril DATA ja ESP8266 arendusplaadil D2
3. Ühendame toite – DHT11 anduril VCC ja ESP8266 arendusplaadil 3V3 (toite jaoks ei tohi ühendada kindlasti 5V, vaid 3,3V. ESP8266 ja DHT11 andmeside ühildamiseks peavad need töötama samal toitepingel ning ESP8266 kontroller töötab pingel 3,3V).

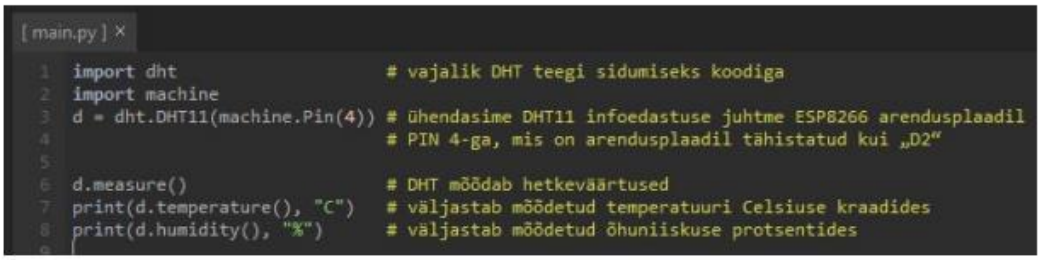


8

Pärast ühenduste loomist võib ESP8266 uuesti arvutiga ühendada ja Thonnys vajutada nuppu „Peata/taaskäivita interpretaator" (STOP), et taasluua ühendus Thonny ja arendusplaadi vahel.
Pärast STOP-nupu vajutamist peaks käsureale ilmuma info, et seade on ühendatud.

Eelnevates sammudes kasutasime „from machine import Pin", mis tõi sisse ainult Pin-ploki machine teegist. Kasutades „import machine", toome sisse terve machine teegi, kuid sel juhul PINe deklareerides tuleb kirjutada „machine.Pin()", ei piisa sellest kui kirjutada „Pin()".

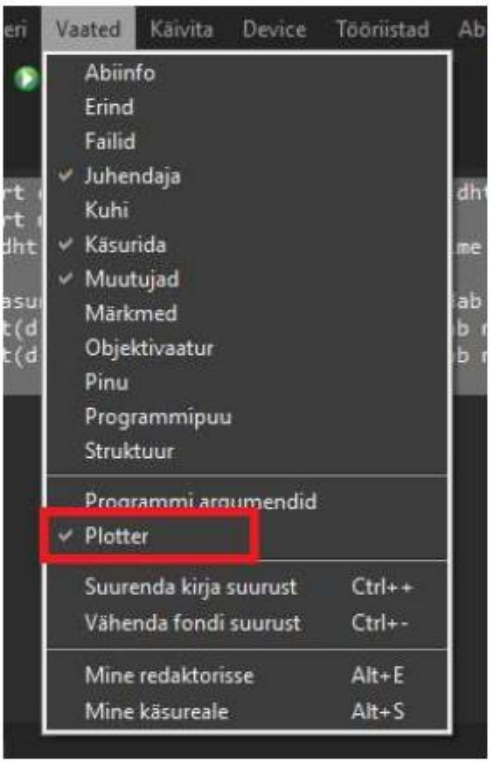
DHT11-ga õhuniiskuse ja temperatuuri mõõtmiseks kirjutame järgneva programmi



e) DHT11 mõõtmistulemustega graafiku loomine

DHT puhul kahe mõõtmise vaheline aeg ei tohi olla alla 1 sekundi, sest DHT ei suuda sellest kiiremini lugeda uusi tulemusi ja saame veateate.

Enne programmi käivitamist lülitada sisse plotter, et Thonny looks käsureale väljastatud tulemuste põhjal graafiku.



Plotterit kasutades on mõistlik mõõtmistulemused käsureale väljastada ühe reana (kasutades ühte „print()" funktsiooni). Selliselt toimides koostab plotter graafikule 2 eri joont, üks kummagi mõõtmistulemuse jaoks. Plotter arvestab ka tulemuse järgi kirjutatud ühikuid, mida kuvatakse graafikul legendina.

Kui tulemused väljastada kordamööda eri ridadele, siis plotter ei koosta käsureale väljastatud tulemustest graafikut.

Kui eemaldada ühikud ja väljastada erinevate mõõtmiste tulemused kordamööda, saame graafiku, mille punktid vastavad kõigile väljastatavatele arvudele ja antud graafik ei ole informatiivne.

Temperatuuri- ja niiskuseanduri mõõtmistulemuste mõjutamiseks võib proovida hingata andurisse.

Kirjutame järgneva programmi

f) Iseseisev ülesanne

Kirjutada programm, mis mõõdab DHT11-ga õhuniiskust ja kui õhuniiskus on üle 50 %, läheb ESP8266 peal olev LED põlema ja kustub jälle kui see langeb 50 -ni või alla selle. Katsetamiseks võib proovida andurisse hingamist.

Näide programmeerimiskeeles Python if-tingimuse kasutamiseks lõpmatus tsüklis.

```
while True:
    if a > 25:
        print(a)
    else:
        print(b)
```