Tartu Ülikool

Loodus- ja täppisteaduste valdkond

Tehnoloogiainstituut

Kaarel Rhede

Aine ”Nutilahenduste praktikum” värskendamine

Bakalaureusetöö (12 EAP)
Arvutitehnika eriala

Juhendaja: kaasprofessor Heiki Kasemägi

Tartu 2022

Resümee/Abstract

Aine ”Nutilahenduste praktikum” värskendamine

Käesoleva bakalureusetöö eesmärgiks on Tartu Ülikooli arvutitehnika bakalaureuseõppekava kohustusliku aine ”Nutilahenduste praktikum” uuendamine. Uuendatakse nii praktikumi riistvara kui ka praktiliste tööde juhendeid.

Seni ühendati praktikumitöödes seadmed omavahel maketeerimislaua ja juhtmete abil. Kuna maketeerimislauad ei pea vastu sadadele seadmete paigaldamistele, mille tõttu ühendused katkevad ega ole stabiilsed, siis disainitakse bakalaureusetöö raames arendusplaat, millel on ESP32 mikrokontroller koos sisseehitatud WiFi ja Bluetoothiga, asendades Arduino Nano arendusplaati. Plaadil on ka spetsialiseeritud pesad põhiliste praktikumis kasutatavate välisseadmete ühendamiseks, samuti mitut tüüpi LED-id, potentsiomeeter ja kvadratuurkodeerija esmasteks lihtsamateks sisend-väljundseadmeteks.

Koos arendusplaadiga loomisega värskendatakse 5 praktikumitöö juhendid, et need oleksid kooskõlas uuenenud riistvaraga. Luuakse ka 2 täiesti uut praktikumitööd. Nii riistvara kui ka juhendeid testiti esimest korda ”Nutilahneduste praktikumis” 2022. aasta kevadsemestril. Aine eelnenud aastal pooleli jätnud osalejad hindasid riistvaraproblemidele kulunud aega väiksemaks kui varasemalt, ülejäänutest pidas ajakulu riistvaraprobleemidele liiga suureks ca. 28 %. Ca. 53 % vastanutest leidis, et ülesanded on piisavalt ühemõttelised ja/või lahtiseletatud. Töös analüüsitakse ka ilmnenud riistavaraprobleeme.

Töo tulemusena valmis 40 arendusplaati ekseplari ja 7st praktikumitöö juhendist koosnev komplekt. Töö tulemused on ”Nutilahenduste praktikumis” aktiivses kasutuses.

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia; T170 Elektroonika; S270 Pedagoogika ja didaktika

Märksõnad: arvutid, asjade internet, nutilahendused

Update on course ”Smart solutions”

The aim of this bachelor’s thesis is to update the compulsory subject ”Smart Solutions” in the bachelor’s curriculum of computer engineering at the University of Tartu. The update will include both hardware used in labs, aswell as lab instructions.

Until now, the devices in use were connected to each other by means of a breadboard and wires. As the breadboard do not withstand hundreds of device installations, resulting in faulty and unstable connections, the bachelor’s thesis will develop a development board with an ESP32 microcontroller, that has built-in WiFi and Bluetooth, replacing the Arduino Nano development board. The board also has specialized sockets for connecting the main peripherals used in the practical sessions, as well as several types of LEDs, a potentiometer and a quadrature encoder for basic input/output devices.

Along with the creation of the development board, the 5 lab instructons will be updated to be compatible with the updated hardware. 2 completely new practical works will also be created. Both the hardware and the instructions were tested for the first time in the ”Smart Solutions” in the spring semester of 2022. Participants who dropped the subject in the previous year estimated the time spent on hardware problems to be less than before, 28% of the rest of students considered the time spent on hardware problems as too high. Ca. 53% of students considered the tasks to be sufficiently unambiguous and / or clear. The work also analyzes the hardware problems that have occurred.

As a result of the work, a set of 40 development boards and 7 lab instructions was completed. The results of the work are actively used in the ”Smart Solutions” course.

CERCS: T120 Systems engineering, computer technology; T170 Electronics; S270 Pedagogy and didactics

Keywords: computers, Iot, smart solutions

Sisukord

Resümee/Abstract **2**

Lühendid, konstandid, mõisted **6**

1 Sissejuhatus **7**

1.1 Nutilahenduste praktikum 7

1.2 ”Nutilahenduste praktikumis” üleskerkinud probleemid 8

1.2.1 Probleemid riistvaraga 9

1.2.2 Probleemid praktikumijuhenditega 10

1.3 Töö eesmärgid 11

2 Ülevaade olemasolevatest arendusplaatidest **13**

2.1 Arendusplaadid Tartu Ülikooli mõningates riistvarakursustes 13

3 Nõuded **16**

3.1 Nõuded arendusplaadile 16

3.2 Nõuded juhenditele 16

4 Riistvara arendamine **18**

4.1 Põhikomponendid 18

4.2 Komponentide valiku põhimõtted ja kaalutlused 18

4.3 Riistvara tootmine 20

4.4 Arendusplaadi maksumus 21

5 Juhendite loomine **23**

5.1 Raspberry Pi praktikum 24

5.2 *Socketi* praktikum 24

5.3 ESP32 ja UART praktikum 24

5.4 ESP32 arendusplaadi praktikum 25

5.5 Bluetooth praktikum 25

5.6 I2C praktikum 25

5.7 WiFi praktikum 25

6 2021/22 kevadsemestri tagasiside **26**

7 Kokkuvõte **28**

Kokkuvõte **28**

Lisad **35**

A Arendusplaadi dokumentatsioon **36**

B Raspberry Pi praktikum **40**

C Socketite praktikum **51**

D UART ja ESP32 praktikum **59**

E ESP32 arendusplaadi DEV32 perifeeriaseadmete praktikum **66**

F Bluetooth ja AT käsud praktikum **70**

G I2C praktikum **76**

H WiFi praktikum **81**

Lihtlitsents **85**

Lühendid, konstandid, mõisted

Breadboard - maketeerimislaud

RasPi - Raspberry Pi

IoT - *Internet of Things* ehk asjade internet

nutipraks - ”Nutilahenduste praktikum”

USART - *Universal Synchronous and Asynchronous Receiver-Transmitter* ehk universaalne sünkroon-asünkroonne saatja-vastuvõtja

I2C - *Inter-Integrated Circuit* ehk mitme võimaliku ülemaga jadasiin

ToF - *Time of Flight* sensor ehk laserkaugusandur

PIR - *Passive Infrared* sensor ehk passiivne infrapunaandur

LCD - *Liquid Crystal Display* ehk vedelkristallkuvar

OLED - *Organic Light-Emitting Diode* ehk orgaaniline valgusdiood

LED - *Light Emitting Diode* ehk valgusdiood

RGB - *Red-Green-Blue* ehk puna-rohe-sinine värviruum

OOP - *Object Oriented Programming* ehk objektorienteeritud programmeerimine

1 Sissejuhatus

Asjade internet ehk nutistu on väga kiiresti kasvav kogum nutikatest seadmetest. Viimaste aastate jooksul on nutistu kasvanud endas sisaldama üle 10 miljardi seadme ning lähiaastatel oodatakse selle kasvu kiirenemist [1]. Nutistu on aina enam saamas keskmise inimese igapäevaelu osaks paljuski tänu oma eelistele, millest tänapäeva maailmas kaks olulisemat on andmete kättesaadavus ja neid kasutades ülesannete automatiseerimine. Näiteks on võimalik nutika kliimaseadmega kuumal ajal kodust ära oldud ajal säästa energiat nii, et kodust lahkudes lülitatakse seadmed välja ja veidi enne naasmist kaugjuhtimise teel uuesti sisse. Samuti on tihti sellistele seadmetele sisseehitatud õppimisvõime ehk peale lühikest kasutusaega õpib seade kasutusharjumuse ja eemaldab vajaduse käsitsi lülituste järele [2]. Poes müüdavast koduelektroonikast muutub aina suurem hulk nutikaks ning paljud neist oskavad juba omavahel informatsiooni jagada, et kõiki ühendatud seadmeid efektiivsemaks muuta [3]. Ühendatus toob endaga paratamatult kaasa aga palju küsimusi privaatsuse ja turvalisuse kohta [4]. Ühelt poolt vaadates on nutikad seadmed ja tark kodu väga mugav viis, kuidas oma elukohta turvalisemaks muuta, näiteks kasutades nutikaid seireseadmeid, mille abil saab telefonist oma kodu turvakaameratel pilku peal hoida [5]. Samas on potentsiaalse ründaja jaoks spionaaž ainult nii keeruline, kui on ligipääs vastavale ühendusele [6]. Lisaks sellele on tarbijate seas umbusaldus korporatsioonide vastu, mille veebilehed, rakendused ja seadmed koguvad andmeid oma kasutajate kohta [7]. Selleks, et see valdkond saaks kasvada ja oma praegused probleemid ületada, on vaja inimesi, kes oskavad sääraseid lahendusi luua ja teha seda turvaliselt [8]

1.1 Nutilahenduste praktikum

Tartu Ülikooli arvutitehnika eriala [9] sisaldab aineid füüsikast elektroonikani, digitaalsest loogikast kõrgtaseme programmeerimiseni ja palju muud. Aine ”Nutilahenduste praktikum” [10] eesmärgiks on tutvustada esimese aasta üliõpilastele erinevaid sensoreid, täitureid, kasutajaliideseid, anda üliõpilastele praktiline kogemus seadmete liidestamisel ja süsteemide loomisel probleemide lahendamiseks. Kursuse maht on 3 EAP. Reeglina toimub igal nädalal üks 4-tunnine praktikum sellele rühmale, kuhu üliõpilane on ennast registreerinud. Praktikuminädalaid on kokku 16, mille jooksul peaks valmis jõudma 7-8 praktikumitöö ja lõpuprojektiga. Kodune töö hõlmab enamasti juhenditega tutvumist enne praktikumitööd. Kursus algab sissejuhatava loenguga. Praktikumi kestel on üliõpilasel võimalik küsida nõu praktikumijuhendajalt (2 ühe

rühma st. kuni 12 üliõpilase kohta) ja tööga valmissaamisel lasta see juhendajal ka kohe ära hinnata. Praktikumides loodud tarkvara laaditakse üles Tartu Ülikooli gitlabi [11]. Moodle'sse laetakse nii kirjalikus kui ka videovormis lühike lõpuprojekti kirjeldus ja aruanne, mida, kuidas ja milleks loodud seade või süsteem teeb või tegutseb.

Kursusel osalemiseks on vajalik programmeerimisoskus vähemalt keeles Python [12], millele hiljem lisandub C++ läbi Arduino IDE [13]. Lisaks sellele on veel vajalikud algteadmised elektrist ja elektroonikast. Praktikumides liigutakse LEDide vilgutamisest kuni sensorite ja muude seadmetega suhtlemiseni, mis päädivad reaalsete probleemide iseseisva lahendamisega. Puututakse kokku klient-server-ideoloogiaga ja selle rakendamisega, saadakse kogemus Linuxi kasutamisest. Luuakse ühendusi arvutivõrgus olevate seadmete vahel ja käsitletakse, kuidas neid turvaliseks muuta, visualiseeritakse kogutud andmeid. Luuakse traadita võrke, millega saab ühendada erinevaid nutiseadmeid nagu nutitelefon ja tahvelarvuti. Õpitakse lugema toodete andmelehti ja saama aru elektriskeemidest. Puututakse kokku erinevate anduritega, näiteks temperatuuriandur [14], ultraheli-kaugusandur [15], laser-kaugusandur [16]. Andmete visualiseerimiseks on kasutuses OLED- ja LCD-ekraanid [17], 7-segmendilised LED-numbrid [18]. Andmeedastustehnoloogiatest õpitakse tundma sinihammast ehk *Bluetoothi* [19], WiFi-t [20], USART-i [21], I2C-d [22]. Tutvutakse erinevate seadmetega, näiteks reaalajakellad [23] ja protsessorivälised *Bluetooth* moodulid [24].

1.2 ”Nutilahenduste praktikumis” üleskerkinud probleemid

Esimene suur probleem ”Nutilahenduste praktikumi” aines oli riistvara töökindluse puudumine. Selle peamisteks põhjusteks olid kasutusel oleva riistvara mitmekesisus ja hulk ning seadmete omavahelisteks ühendamiseks võimalike kombinatsioonide arv. Sellest tulenevalt võis üliõpilasel kuluda tunde probleemide otsimiseks, mille põhjuseks oli kellegi poolt eelnevalt kahjustatud komponent ning juhendajatel puudus võimekus kõiki komponente juhendamise kõrvalt kontrollida. Eriti probleemsed olid praktikumides laialdaselt kasutatud maketeerimislauad, mille töökindlus oli paljude üliõpilaste hooletu kasutamise tõttu alla igasugust arvestust.

Teine kõige teravamalt silma jäänud probleem oli üliõpilase üldine arusaamine. Seda põhjustas osaliselt ebausaldusväärne riistvara, mis pani üliõpilased enda väljamõeldud lahenduses kahtlema ja osaliselt juhendid ja näidiskoodid. Viimased andsid üliõpilasele osad ehitusklotsid ette nii, et nad ei pidanud aru saama, kuidas seadmed täpselt töötavad. Juhendites polnud paljudel juhtudel materjale, mis seletaksid ammendavalt lahti seadme põhimõtte ja töötamise, mille asemel mitmel juhul oli räägitud selle aine ulatusest väljas olevatest või triviaalsetest teemadest (nt. joonisel 1.1).

4 Raspberry Pi seadistamine arvutivõrku kasutama

Vali alljärgnevate võrguühenduste seast endale sobivaim ja ühenda RasPi võrku.

- Esimeseks võimaluseks RasPi võrku saada on ühendada ta RJ45-kaabliga lokaalvõrku. Tavaliselt saab RasPi siis DHCP-ga endale IP ja on võimeline välisvõrguga suhtlema.
- Avalik WiFi-võrk. Sinna ühendumiseks (kui see ei toimu automaatselt) tuleks avada avakuva ülemises osas olev võrguikoon ja valida seal saadaolevate juurdepääsupunktide seast sobiv (joonis Fig. **2**). Kui juurdepääsupunkt on kaitstud salasõnaga, siis kuvatakse ka dialoogiaken selle sisestamiseks. Tavaliselt jätab Rasbian meelde viimase WiFi-võrgu, mida kasutati ja järgmisel taaskäivitamisel püüab automaatselt sellega ühenduda.
- Tekitada nutitelefoni või tahvelarvutiga lokaalne WiFi-võrk ja ühendada RasPi sellesse võrku.

Kuna alati pole graafiline kasutajaliides või isegi sisend-väljundseadmed olemas RasPi konfigureerimiseks, siis saab seda teha ka otse mikroSD-l konfiguratsioonifaile muutes.

Praktikumis ühendu WiFi-võrgu külge, mille SSID ja salasõna juhendaja ette annab. Võimalik, et selleks tuleb RasPi WiFi konfiguratsioonifaili muuda. Siis on juhend selleks saadaval moodle'is.

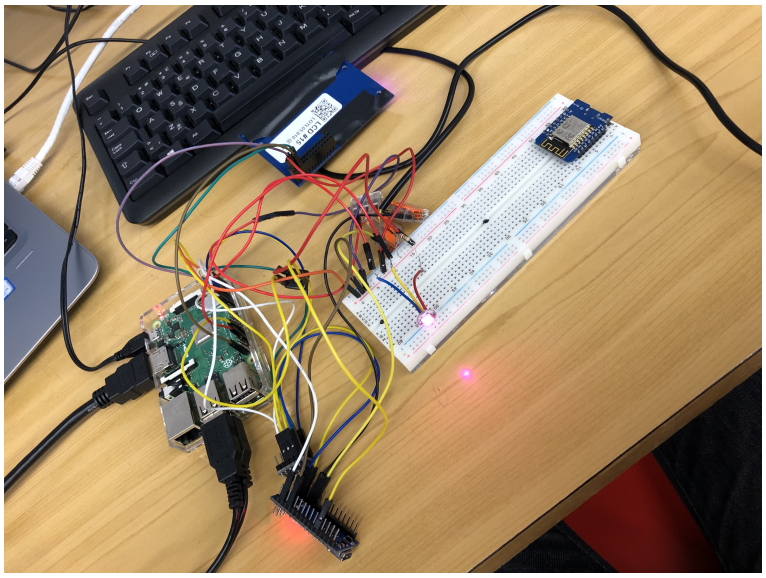
Joonis 1.1: Esimese praktikumi vana juhendi lõik

1.2.1 Probleemid riistvaraga

Eelmistel aastatel kasutati "Nutilahenduste praktikumi" aines lisaks Raspberry Pi 4B platvormile [25] veel kuni kahte võrdlemisi vana arendusplaati: Atmega 328P protsessoril [26] põhinevat Arduino Nanot [27] ja ESP8266 protsessoril [28] põhinevat LOLIN D1 Minit [29] WiFi praktikumi jaoks, kuna antud kiip saab ühenduda WiFi-võrkudesse. Praeguses maailmas on ESP8266 platvorm peaaegu täielikult asendatud uuema ESP32 [30] protsessoriga, kuna peaaegu sama hinna eest saab igas mõttes parema protsessori, millele on lisatud võimekusi. Kuigi Atmega 328P on väga populaarne protsessor [26], ei ole ta tänapäevaste targa kodu ja asjade interneti seadmete jaoks tihti piisav, sest tegemist on 8-bitisel arhitektuuril põhineva protsessoriga, millel on väga vähe perifeeriaseadmeid. Näiteks on riistvaraliselt toetatud ainult 1 USART, mis on kasutusel ka programmeerimisliidesena, seega on välise USART liidest kasutava seadmega ühendamisel vaja kasutada tarkvaralist USARTi, mis on väga ressursinõudlik ja piiratud kiirusega. Sellel protsessoril põhinevaid tooteid kasutavad aktiivselt isetegijad ja algajad tänu väga suurele hulgale internetis saadaval olevatele näidiskoodidele ja õpetustele. Praktikumis tekitas probleeme asjaolu, et nendele protsessoritele kirjutatud programmid ei olnud alati ühilduvad, mis tekitas segadust üliõpilasete hulgas. Sellest tekkis mõte liikuda uuemale platvormile, mis on piisavalt algajasõbralik, ehk omades suurt hulka saadaval olevaid teeke, näidiseid ja lahendusi probleemidele, kuid ka piisavalt võimekas, et olla levinud tööstuses, et üliõpilasel hiljem sellest kasu oleks.

Suurim probleem riistvaraga tulenes maketeerimislaudade (ing. *breadboard*) [31] töökindlusest. Nimelt kulus üliõpilasel tihtipeale rohkem aega vigaste maketeerimislauda ühenduse leidmisele, kui ülesandes käsitletava seadmega tutvumisele. Eriti oli see probleem, kui üliõpilane ei jõudnud praktikumiülesande jaoks vajalikku koodi ühe praktikumiajaga valmis ning pidi samad komponendid järgmine kord uuesti ühendama (joonis 1.2), mis raiskas väärtuslikku praktikumiaega. See oli ajamahukas ka juhendajatele, kes pidid pidevalt vastama küsimusele "miks see ei tööta,

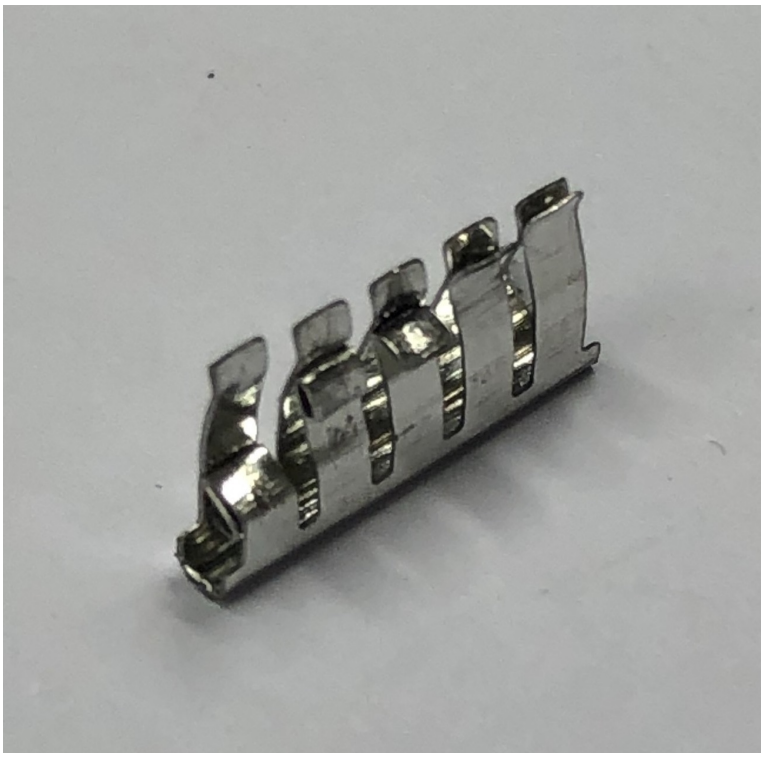
eelmine kord oli samamoodi ja töötas?”. Näitlikustamaks probleemi ulatust, võttis autor ühe suvalise kasutusel olnud maketeerimislaua ja loendas kokku klemmliistud, eristades neid selle järgi, mitu klemmi olid silmnähtavalt deformeerunud (joonis 1.3). Sellised klemmid võivad üldse mitte ühendust anda või võivad ühendused olla ebastabiilsed, näiteks vastu minnes kaob ühendus ära. Antud maketeermislaua 162 klemmliistust 43 olid vigastusteta (joonis 1.4). Ülejäänute hulgast leidis autor kokku 132 vigastatud klemmi ehk 20.95 % kõigist ühenduskohtadest olid ebausaldusväärsed.



Joonis 1.2: Näide pratikumiülesande lahendusest juhtmetega

1.2.2 Probleemid praktikumijuhenditega

Praktikumijuhendites tekitas üliõpilasele raskusi juhendite ja ülesannete vähene seotus. Eelnev praktikumide struktuur nägi välja selline, et Moodle lehel, mis on jagatud praktikumideks, olid tekstina lahti seletatud ülesande eesmärgid. Moodle lehel oli ka linke juhenditele, milles oli infot vahemikus ühe seadme tutvustus kuni terve praktikumi teema koos põhjaliku taustaga ja ka palju muud. Sellest tulenevalt olid juhendite pikkused vahemikus 1 kuni 24 lehekülge, mitmel juhul oli ühes praktikumis vajalik info laiali mitmes erinevas failis. Sellest tekkis probleem, et ei üliõpilasel ega juhendajatel ei olnud kõikehõlmavat ülevaadet materjalide sisust ja tihtipeale ka sellest, millised materjalid üldse olemas on. Sealt omakorda ei olnud juhendajatel võimalik viidata lõikudele või materjalidele, kus üliõpilaste enamlevinud küsimused juba lahti seletatud on, mis raiskas jällegi kõigi osapoolte aega.



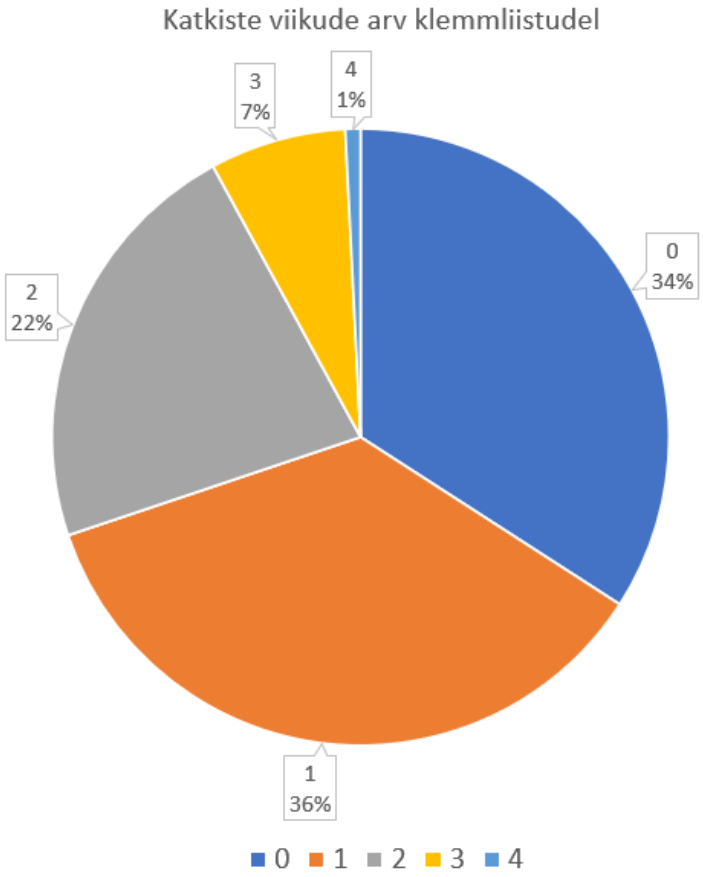
Joonis 1.3: Näide deformeerunud kelmmliistust

1.3 Töö eesmärgid

Toetudes asjaolule, et autor on ise ”Nutilahenduste praktikumi” kunagi läbinud ja selle aine pratkikume ka ise juhendanud, millest tulenevaid kogemusi on eelpool kirjeldatud, siis on käesoleva töö eesmärgiks ”Nutilahenduste praktikumi” riistvaralise poole, praktikumitööde ja nende juurde kuuluvate juhendite uuendamine, vajadusel ka uute tööde loomine.

Riistvaralise poole pealt on eesmärgiks luua kompaktne arendusplaat, mis minimeeriks vajaduse kasutada erinevate seadmete ühendamiseks kõikvõimalikke juhtmeid ja maketeerimisplaati. Plaadil võiks olla spetsiaalsed pesad praktikumitöödes enimkasutatavate seadmete ühendamiseks.

Kuna uue loodava arendusplaadiga ei pruugi olemasolevad juhendid ühilduda, siis tuleb ka luua uued praktikumijuhendid, mis toetaks loodavat arendusplaati. Tõenäoliselt osutub vajalikuks lisada 1-2 uut praktikumitööd, eelkõige arendusplaadiga integreeritud seadmete ja andmesiinide kasutamise tundmaõppimiseks.



Joonis 1.4: Maketeerimislaua viikude olukord

2 Ülevaade olemasolevatest arendusplaatidest

Arendusplaadi idee põhineb lihtsasti taasprogrammeeritavatel protsessoritel, mida kasutatakse, et arendusprotsessi käigus testida süsteemi komponentide sobivust ja/või arendada tarkvara päris toote jaoks. See on välja arenenud primitiivsetest ühekordselt programmeeritavatest protsessoritest, kuid programmide testimine oli väga kallis, seega keskenduti teoreetiliselt programmide silumisele. Tänapäeval, kui protsessoreid on võimalik kiiresti ja peaaegu piiramatu arv kordi programmeerida, on tihti lihtsam ja kiirem arendusprotsessis testimisele keskenduda [32].

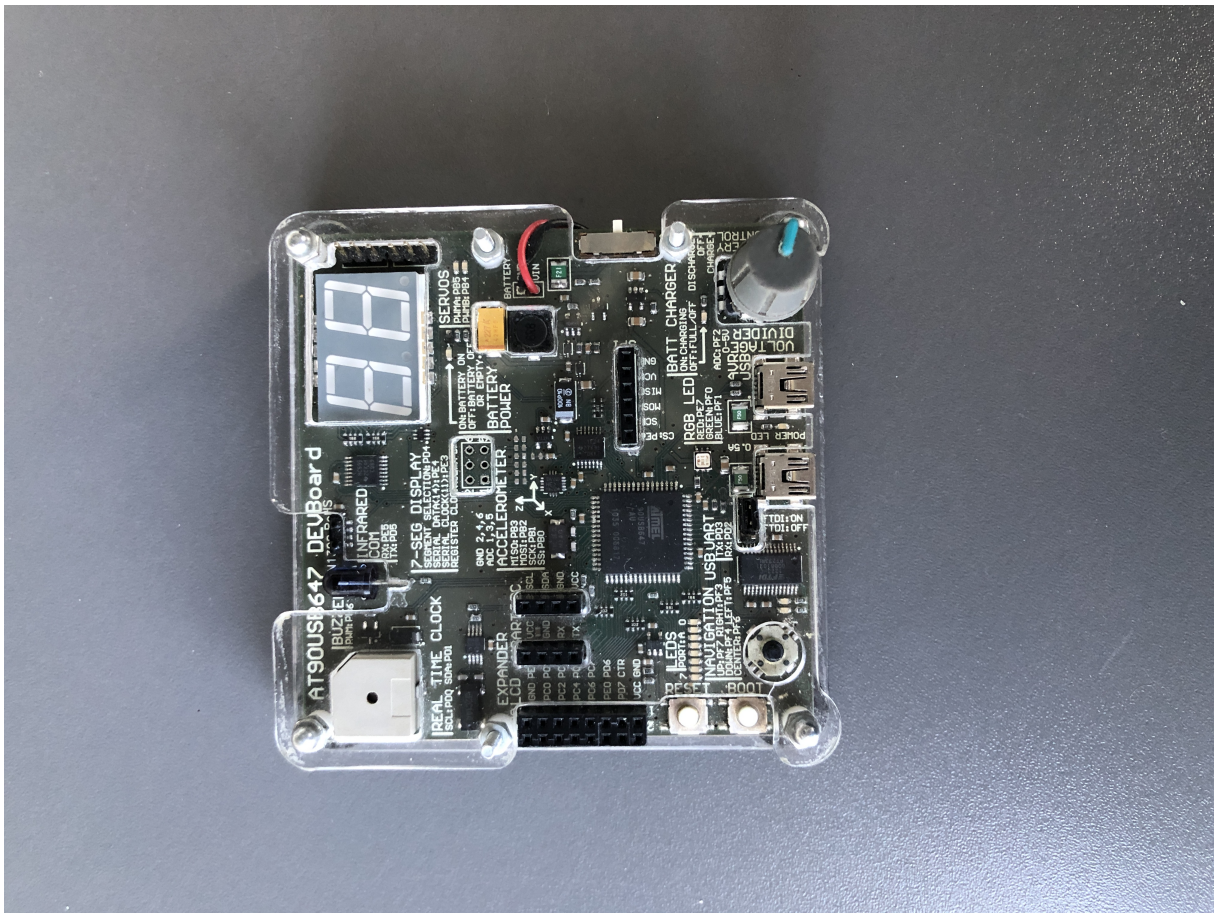
Mõningad tingimused, mida riistvaraplatvormi luues silmas pidada, oleksid järgmised [33]:

- rakenduste jaoks sobilik mikrokontroller või -protsessor;
- toiteahel plaadil olevate seadmete toitmiseks;
- põhilised sisendseadmed nagu nupud, sensorid jne;
- põhilised väljundseadmed nagu LED-id, LCD-d, mootorid jne;
- mikroprotsessori programeerimisliides;
- suhtlus- ja kasutajaliides nagu USB, traadita side võimalus jne

2.1 Arendusplaadid Tartu Ülikooli mõningates riistvarakursustes

Tartu Ülikooli riistvarakursustes nagu ilmselt paljudes teistes ülikoolidest üle maailma, kasutatakse õige mitut arendusplaati. Aine ”Manussüsteemid” kasutab ST toodetud arendusplaati STM32F429I-DISCO koos spetsiaalselt disainitud laiendusplaadiga, mis ühildub MikroBus [34] standardit kasutavate sensoritega. ”Nutilahenduste praktikumis” on juba välja kujunenud oma sensorite komplekt ning ei oleks mõistlik neid kõiki välja vahetada.

Õppeaines ”Mikroprotsessorid” kasutatakse spetsiaalselt loodud arendusplaati (joonis 2.1), mis

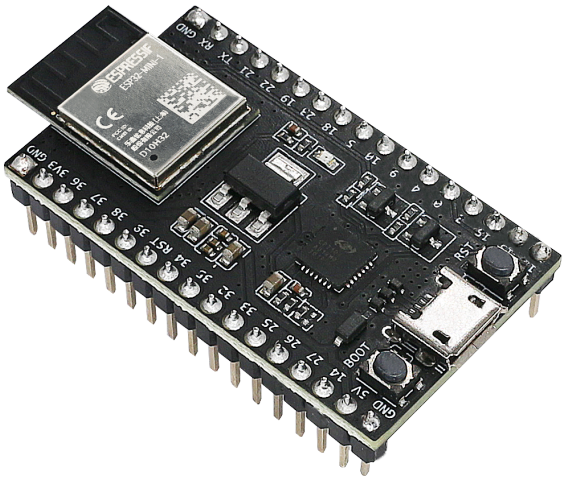


Joonis 2.1: Aine ”Mikroprotsessorid” arendusplaat

põhineb AVR arhitektuuriga kontrolleril AT90USB647 [35]. Sellel arendusplaadil on kasutajale kättesaadavad I2C, UART ja SPI ühendused ja 13 GPIO viiku, kuid sensorite ühendamiseks peaks siiski kasutama maketeerimislauda. Selle protsessori ümber puudub ka arvestatav kommuun ning Arduino IDE liidese viimane uuendus sellele protsessoriperekonnale avaldati 2013. aastal [36].

Kummalgi nendest arendusplaatidest ei ole võimekust lisaseadmeteta luua juhtmevabu ühendusi Bluetoothi ega WiFi võrkudega, mida ”Nutilahenduste praktikumis” mitmes töös kasutatakse. Arendusplaadid, mis on mõeldud kasutamiseks analoogselt sellega, mida ”Nutilahenduste praktikum” vajab, on lahtimurdelaud (*breakout board*). Ühe sellise lahtimurdelaua näiteks on ESP32 jaoks loodud ESP32-DevKitM-1 [37] (joonis 2.2), millega analoogsed arendusplaadid olid nutipraksis varasemalt kasutusel põhjusel, et oli võimalik ühendada kõiki vabalt valitud lisaseadmeid. Sellise arendusplaadi miinus siinkohal on aga just vajadus kasutada maketeerimislauda, et ühendusi luua.

ESP32 platvormile on loodud suur kogus erineva eesmärgiga arendusplaate [38]. Need on enamasti suunatud kindla funktsionaalsuse saavutamiseks või perifeeriaseadme testimiseks.



Joonis 2.2: ESP32-DevKitM-1

Näiteks on olemas arendusplaat ESP-EYE [39], mis on mõeldud masinnägemise ja -kuulmise testimiseks ja ESP32-Sense-Kit, mis on mõeldud puutetundlike viikude testimiseks kasutajaliidestes [40]. Nimetatud plaatidel on olemas ka praktikumides vajaminevad RGB LED ja 7-segmendiline LED, kuid puudub võimalus ilma maketeerimislauda või vaheplaati kasutamata praktikumitöödes vajalikke mooduleid liidestada.

3 Nõuded

3.1 Nõuded arendusplaadile

- Saab programmeerida kasutades Arduino IDE tarkvara [13]
- Lõviosa ühendustest on võimalik teha maketeerimislauda kasutamata
- Võimaldab kasutada järgmisi praktikumitöödes kasutatavaid mooduleid:
  - HC05 Bluetooth moodul
  - VL53l0x ToF andur
  - MPU6050 güroskoop
  - HC-SR04 ultraheliandur
  - Adafruiti PIR andur
  - PCF8574 I2C mooduliga LCD
  - OLED
- Läbi nihkeregistri kontrollitav 7-segmendiline LED kasutajale väljundi andmiseks
- Kasutajalt sisendi saamiseks potentsiomeeter, kooder ja nupp
- Individuaalselt juhtitavad RBG LED-id
- Võimalus lisaseadmeteta ühenduda WiFi võrku
- On olemas aktiivne kogukond, kes sama protsessorit samal arendusplatvormil (Arduino IDE) kasutavad.

3.2 Nõuded juhenditele

- Vastavad loodavale riistvarale
- Sissejuhatav materjali peab olema esitatud nii, et see oleks läbiloetav 10 minutiga

- Näidiskoode nii vähe kui võimalik, et üliõpilased peaksid ise rohkem mõtlema ja aru saama, mida nende programm teeb
- Piisavalt viiteid riistvara ametlikule dokumentatsioonile

4 Riistvara arendamine

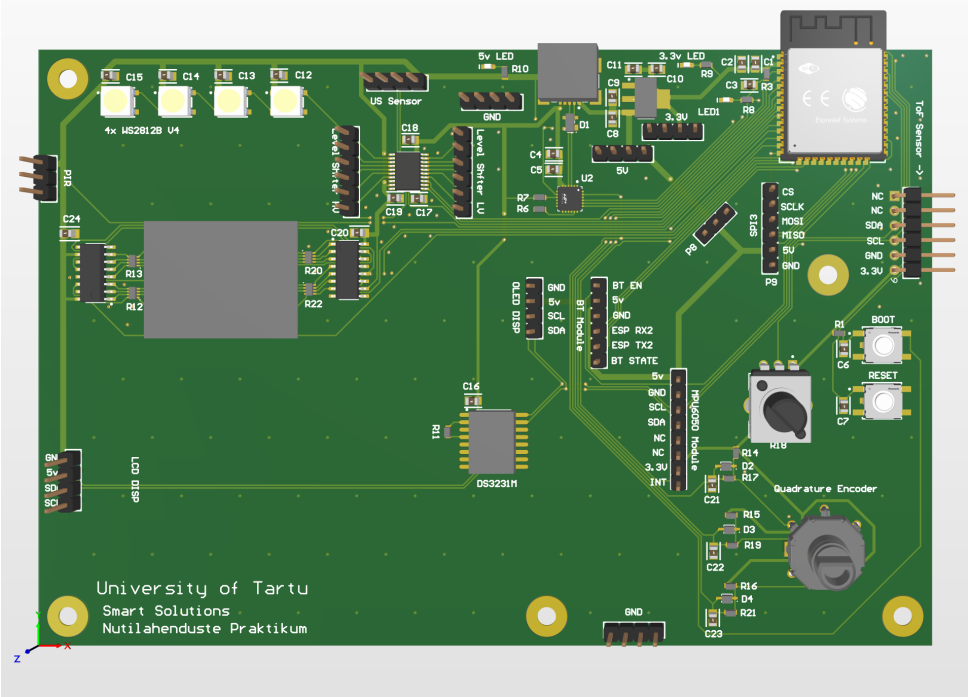
4.1 Põhikomponendid

Arendusplaat DEV32 koosneb järgmistest põhikomponentidest:

- mini-USB ühendus;
- 2 tk. 7-segmendilisi LED-e LTD-6940HR;
- potentsiomeeter PTV09A-4020F-A203;
- kvadratuurenkooder EC12D1564402;
- protsessor ESP32-WROOM-32;
- USB-UART sild CP2102N-A02-GQFN24;
- lineaarne pingeregulaator LD1117S33TR;
- reaalajakell DS3231M+TRL;
- 4 tk. individuaalselt adresseeritavaid RGB LED-e WS2812v4;
- tasememuundur TXB0108PWR;
- 2 tk. nihkeregistreid SN74HC595D.

4.2 Komponentide valiku põhimõtted ja kaalutlused

Nutipraksis eelnevalt kasutusel olnud ESP8266 protsessoriga oli probleemiks väike GPIO viikude arv ja vähene perifeeriaseadmete valik, mis ei võimaldanud seda kõikides praktikumitöödes kasutada. Seevastu ESP32, mis on selle protsessori järeltulija, on võimekam ja sellel on rohkem GPIO viike, perifeeriaseadmeid ja lisandunud on Bluetooth võimekus. ESP32 platvorm on laialdaselt kasutuses hobikasutajate seas ja ka kommertsrakendustes. Näiteks kasutab seda platvormi Sonoffi Switchman M5 tark lüliti ja muud selle firma tooted [41].



Joonis 4.1: DEV 32 trükiplaat

Komponentide valikul oli üheks kaalutlemiskohaks nende saadavus, sest maailmas oli arendamise ajal just pead tõstnud kiibikriis ja tarneraskused. Seetõttu on komponente tellitud Farnelli [42], Mouseri [43] ja TME [44] veebipoodidest. Plaati disainides kasutas autor relevantseid andmelehti, peamiselt ESP32 riistvara disainijuhendit [45], ESP32 andmelehte [30] ja CP2102 USB-UART silla kiibi andmelehte [46].

Arendusplaat saab tavakasutuses toite mini-USB ühendusest, üle mille toimub ka protsessori programmeerimine. Mini-USB valiti, kuna see on kasutajakogemuse järgi välismõjudele, st. kasutamisele füüsiliselt vastupidavam kui mikro-USB [47]. USB väljundpinge on 5 V, kuid protsessor ja osad moodulid vajavad toiteks 3.3 V. Pinge reguleerimiseks on arendusplaadil LD1117S33TR [48] lineaarne madala langusega pingeregulaator. Programmeerimine toimub ESP32 protsessoril üle UART0 liidese, seega on vajalik USB-UART sild. Selleks valiti CP2102 kiip [49]. Programmi peale laadimine toimub ESP-IDF tarkvara kasutades automaatselt, kasutades protsessori alglaadurisse saamiseks UARTi RTS- ja DTR-signaale. ESP-IDF [50] tarkvara on Espressifi arendusraamistik, mida DEV32 plaadil kasutatakse läbi Arduino IDE [13] arenduskeskkonna kasutajale märkamatult.

Arendusplaadil (joonis 4.1) on potentsiomeeter ja nupuga kvadratuurkooder [51], et kasutajalt sisendit saada. Väljundseadmeteks on plaadil üks tavaline LED, kaks 7-segmendist LED-i, mis on ühendatud läbi nihkeregistrite ja 4 individuaalselt juhitavat RGB LED-i. RBG LED-ide kontrollimiseks on vajalik 5 V loogikapinge, kuid ESP32 väljund on ainult 3.3 V. Selle probleemi

lahendamiseks on plaadil tasememuundur, et erinevate pingetasemetega sisendeid ja väljundeid ühitada. Arendusplaadi dokumentatsioon on lisas A.

Sensorite ja muude praktikumis kasutusel olevate moodulitega liidestumiseks on arendusplaadil lisamoodulitega sobituvad viigupäised, kuhu moodulid saab lihtsasti ühendada.

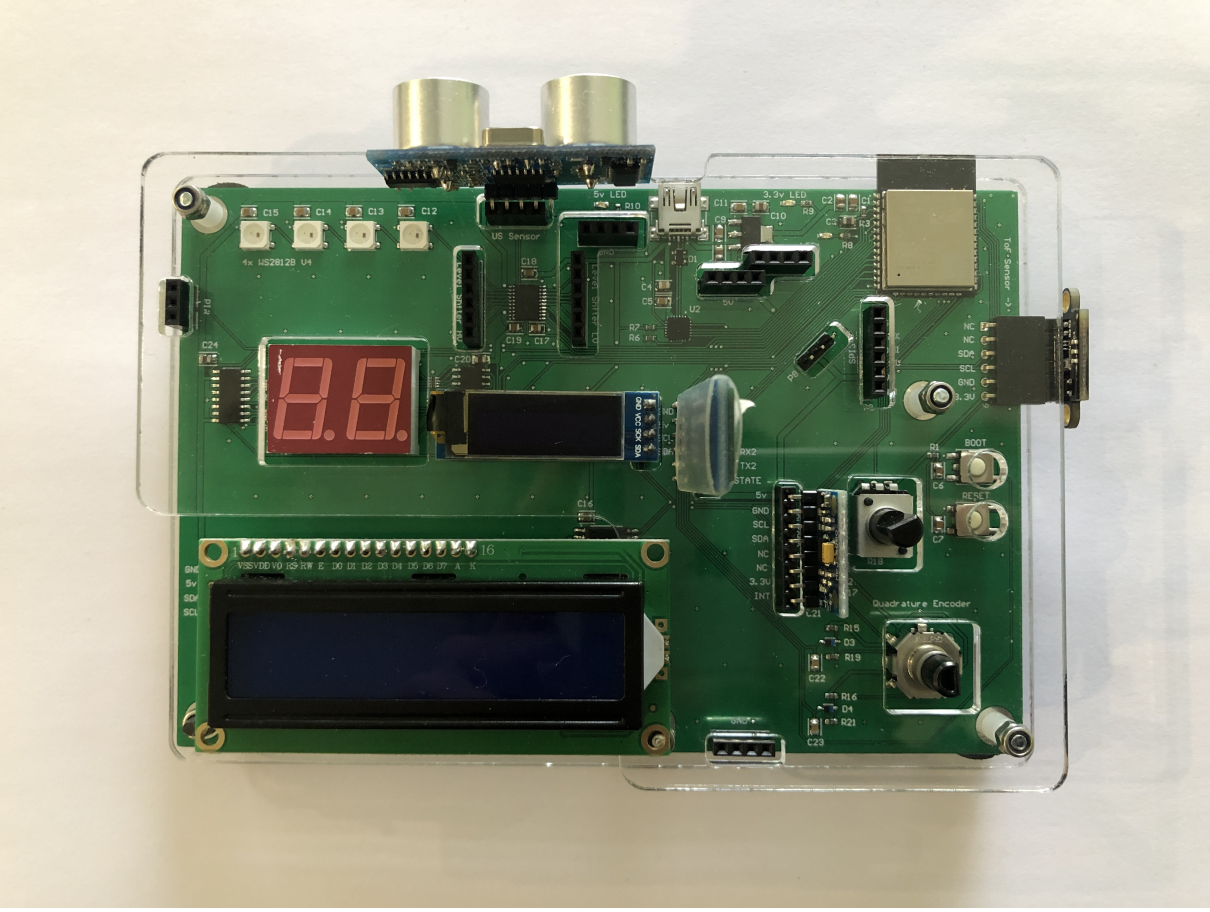
4.3 Riistvara tootmine

Arendusplaadi DEV32 prototüüp disainiti kursuse ”Riistvaraprojekt” raames. Selleks kasutati tarkvara Altium Designer 2021.2.2 [52]. Prototüüp ja hiljem ka kasutusse läinud plaadid (joonis 4.2) telliti firmast JLCPCB [53], mistõttu plaadi loomisel arvestati tootjafirma võimekusega.

Praktikumides kasutuses olev riistvara 4.2 on toodetud kasutades LPKF Protoflow E [54] *reflow* ahju SMD komponentide jaoks, THT komponendid on joodetud käsitsi.

Plaadid on altpoolt ja ülaltpool kaetud 3 mm paksuse akrüülklaasist [55] plaadiga. Alumine plaat on 10 mm võrra laiem trükiplaadi mõõtmetest ja sisaldab 3 mm läbimõõduga kruviauke, mis on kohakuti trükiplaadis olevatega. Nurgad on ümardatud raadiusega 5 mm. Ülemine plaat on samuti mõõtmetes trükiplaadist 10 mm laiem, sisaldades nii trükiplaadi kui ka alumise plaadiga kohakuti olevaid 3 mm läbimõõduga kruviauke. Plaadis on avaused viigupäistele, 7-segmendilistele LED-idele, potentsioneetrile, kvadratuurkooderile ja sisselõiked BOOT- ja RESET-nuppude ja USB-pesa kasutamiseks. Kõigi avauste sisenurgad on ümardatud kõverusraadiusega 2 mm, välja arvatud nuppude sisselõige, mille nurgad on kõverusraadiusega 5 mm. Ülemisel plaadil on ”LCD DISP”-viigupäise kohal 80x47 mm suurune sisselõige I2C-adapteriga 1602 LCD paigaldamiseks.

Trükiplaat ja akrüülklaasist katted on omavahel ühendatud M3x24 poltidega, välja arvatud LCD sisselõike kohal olev polt, mis on M3x16. Samuti on erinev ühe pikema külje keskmine polt, mis on M3x28 ja mõeldud toetama ühest nurgast ka LCD-d. Poldipea ja alumise plaadi vahel on kummipuks [56]. Alumise plaadi ja PCB vahel on 5 mm kõrgune plastikpuks, mis on ka PCB ja ülemise plaadi vahel. Poldid on pealt kinnitatud M3 lukustava mutriga.



Joonis 4.2: Valminud arendusplaat

4.4 Arendusplaadi maksumus

Loomulikult ei valmi ükski tehniline seade ilma kulutusteta. Tabelis 4.1 on toodud ära ühe arendusplaadi eksemplari komponentide maksumused ja plaadi komponentide kogumaksumus, mis on 29.87 €. Siia pole arvestatud plaadi disainimiseks ja toomiseks kulunud aega. Arvestuslikult kulus plaadi disainimiseks 50 töötundi ja ühe plaadi tootmiseks kuni 2 töötundi. Viimane koosneb peamiselt THT-komponentide käsitsi jootmisele kuulunud ajast, jootepasta pealekandmisest ja SMD komponentide ladumisest.

Tabel 4.1: Ühe plaadi maksumus

Komponent	Tüüp	Kogus	Hind €
ESD kaitsediood	PRTR5V0U2X	1	0.408
Schottky diood	BAT5403WE6327HTSA1	3	0.239
Nupp	EVQQ2W02W	2	0.236
Tasememuundur	TXB0108PWR	1	1.350

Tabel jätkub järgmisel leheküljel

Tabel 4.1 Ühe plaadi maksumus – eelmise lehekülje jätk

Komponent	Tüüp	Kogus	Hind €
Takisti	CRG0603F1K0	2	0.004
Takisti	CRG0603J10K	10	0.002
Takisti	CRG0603F4K7	2	0.003
Kondensaator	C0805C104Z5VACTU	13	0.017
Kondensaator	GRM219R61C226ME15L	2	0.198
Kondensaator	C0805C105Z3VACTU	3	0.065
Kondensaator	885012107018	1	0.146
Kondensaator	CL21B103KBANNNC	3	0.026
Kruvi	M3x28	1	0.220
Kruvi	M3x24	4	0.200
Kummijalg	3094BLACK	4	0.272
Mutter	M3- N5A4-S50	4	0.115
Reaalajakell	DS3231M+TRL	1	6.390
USB-UART sild	CP2102N-A02-GQFN24	1	2.050
Takistmassiiv	CAT10-151J4LF	4	0.048
Nihkeregister	74HCT595S16-13	2	0.258
Kvadratuurenkooder	EC12D1564402	1	1.380
Potentsiomeeter	PTV09A-4020F-A203	1	0.847
2x 7seg LED	LTD-6940HR	1	1.080
LED	150060SS75000	3	0.116
Mini-USB	KMBX-SMT-5S-S-30TR	1	1.080
Viigupäis	ZL263-4SG	1	0.100
Viigupäis	ZL262-4SG	5	0.054
Viigupäis	-6SG	3	0.062
Viigupäis	ZL263-6SG	1	0.139
Viigupäis	ZL262-8SG	1	0.082
Viigupäis	ZL262-3SG	1	0.089
Viigupäis	ZL201-03G	1	0.030
RGB LED	WS2812B-V4	4	0.186
Lineaarregulaator	LD1117S33TR	1	0.504
Protsessor	ESP32-WROOM-32E-N16	1	3.420
Puks	D01475	8	0.043
Trükkplaat		1	3.713
		Summa:	29.87

5 Juhendite loomine

Aine ülesannete ja juhendite uuendamisel lähtutakse eesmärgist anda üliõpilasele täpsem arusaam, kuidas praktikumis õpitavad asjad töötavad. Selle saavutamiseks kasutatakse meetodeid Browni ja Wilsoni uuringust [57]. Eelkõige on eesmärk vastavalt nende 6. punktile jagada ülesanded rohkemateks alamülesanneteks, et anda üliõpilastele praktiline kogemus asjade sammsammult üles ehitamisel, kontrollides seejuures iga sammu töötavust ja arusaama sellest. Selline protsess ei neid, kes ei saanud pikkade ülesande püstituste seast aru, mida täpselt neilt oodatakse. See põhjustas omakorda probleeme ülesande esitamisel, kui tuli välja üksikasi, mis vajas tähelepanu programmi ülesehituse faasis. Samuti on eesmärk kohandada materjalid vastama Browni ja Wilsoni punktile 9, hoida alles kursuse seni suurim õpiväljund: info iseseisev otsimine ja probleemide lahendamine. Sellel eesmärgil on soov lisada juhenditesse vihjeid, kust internetist usaldusväärset informatsiooni leida, kuidas lugeda ja saada aru elektriskeemidest ning kuidas riistvaralähedasel programmeerimisel vigu leida.

Juhendid kirjutati kasutades tarkvara AsciiDocotor, mis võimaldab mugavalt väljastada html-faili, kus ei ole raisatud ruumi pdf-failidest tuttavatele leheküljevahedele. Samuti näeb väljund hea välja. Ühe üliõpilase võrdlus eelnevate aastate juhenditega oli:

> *”Näevad uuemad välja, ei näe välja nagu oleks pärit aastast 1998”.*

Kursuses kasutatavad juhendid on järgmised:

1. Raspberry Pi praktikum
2. *Socketi* praktikum
3. ESP32 ja UART praktikum
4. ESP32 arendusplaadi praktikum
5. Bluetooth praktikum
6. I2C praktikum
7. WiFi praktikum

5.1 Raspberry Pi praktikum

Selle praktikumitöö (lisa B) eesmärk on üles seada hiljem vajaminevad tööriistad. Alustatakse git keskkonna ülesseadmisest ja edasi liigutakse Raspi juhtmevabalt ülesseadmisele. Seejärel peab üliõpilane lahendama kaks programmeerimisülesannet, milles tutvutakse LEDi-de ja nupu tööpõhimõtte ja kasutamisega GPIO.RPi [58] teegi abil. Juhendisse lisati tutvustus 7-segmendise LEDi, *pull-down(up)* takisti ja git-i keskkonna [11] kohta. Lisaks uuendati juhendis tutvustusi Raspberry Pi, GPIO, LED-i ja nupu kohta. Sellest aastast liigutati materjalide hoidmine gitlabi keskkonda [11]. Praktikumiülesannete hulka lisati repositooriumi loomine praktikumis kirjutatud koodide hoiustamiseks. Gitlabi kasutamisele üleminek on põhimõtteline muudatus aine struktuuris, sest seni hoidsid osalised oma loodud tarkvara individuaalsel mikroSD-kaardil, kuhu pärast iga praktikumi selle jooksul kirjutatud kood salvestati. Iga praktikumitöö arvestatud saamiseks tuli valmis tarkvara ja/või muu info skeemide, piltide või videote näol laadida üles kursuse Moodle'i lehele. Moodle ülesannete sektsioon ei ole aga selleks mõeldud ja kasutajakogemus jättis soovida. Gitlabis antakse juhendajatele õigused, et oleks võimalik osaliste üleslaaditud koode kontrollida ja kursuse lõpus arhiveerida.

5.2 *Socketi* praktikum

Selle praktikumitöö (lisa C) eesmärk on tutvuda interneti protokolli kasutavate ühenduspesade ehk *socketitega*. Praktikumitöö raames loob üliõpilane *socket*-serveri ja -kliendi ning lahendab õpib *socket*eid kasutades lõimedega töötama. Viimaks tehakse läbi SSH-tunnelite kasutamine. Juhendis uuendati *socket*i, TCP ja UDP protokollide ja klient-server arhitektuuri tutvustust ning lisati juurde lõimede, portide ja OOP-i tutvustused. Uuendati SSH-tunnelite selgitusi, sest antud töös luuakse nii lokaal- kui ka kaugtunnelid arvutivõrgus olevate seadmete vahel.

5.3 ESP32 ja UART praktikum

See praktikum (lisa D) asendab varasemat Arduino praktikumi. Praktikumitöö raames tutvustatakse üliõpilasele ESP32 platvormi, Arduino IDE arenduskeskkonda ja ESP32 arendusraamistikku. Tutvutakse USART jadaliidesega, mida kasutatakse andmevahetuseks Raspi ja DEV32 arendusplaadi vahel. Analoog-digitaalmuunduri abil tutvutakse pingejaguriga potentsiomeetri näitel. Selle praktikumi sissejuhatusse lisati juurde tutvustused Arduino IDE, ESP32 platvormi, ESP-IDF arendusraamistiku ja pingejaguri kohta. Sissejuhatust USART liidese kohta täiendati ja uuendati. Autor liigutas sellest praktikumist välja LCD ühendamise ja selle liidestamise hilisemasse I2C praktikumi, et vältida teekide kasutamist teadmata, kuidas need töötavad.

5.4 ESP32 arendusplaadi praktikum

Tegemist on uue praktikumitööga (lisa E), mille raames tutvub üliõpilane arendusplaadil olevate sisend-väljundseadmetega. Selles praktikumis on üks ülesanne vanadest praktikumidest ja lisatud juurde nihkeregistrite ja kvadratuurkooderi osa, tulenevalt nende laialdasest kasutusest nutikate seadmete hulgas. Selle praktikumi raames tutvustatakse arendusplaadil olevaid RGB LEDe, nihkeregistreid ja kvadratuurkooderit. Praktikumi raames tuleb kirjutada programmid kvadratuurkooderilt sisendi saamiseks, RGB LEDidel väljundi näitamiseks ja nihkeregistreid kasutades 7-segmendistel LEDidel väljundi näitamiseks.

5.5 Bluetooth praktikum

See praktikum (lisa F) on sissejuhatus Bluetooth ühendustesse. Selle raames tutvustatakse üliõpilasele Bluetooth sidet, ühenduse osapooli ja AT käske moodulite konfigureerimiseks. Selle praktikumi raames ühendab üliõpilane Bloetoothi kasutades Raspi ja telefoni, siis ESP32 ja telefoni ning lõpuks kaks ESP32 plaati, millest üks kasutab välist HC05 Bluetooth moodulit. Selles praktikumis asendati kahes ülesandes väline HC06 või HC05 orja režiimis töötav moodul ESP32 sisemise Bluetooth mooduliga. Lisati juurde selgitav osa AT-käskude kohta ja värskendati sissejuhatust Bluetoothi.

5.6 I2C praktikum

Tegemist on uue praktikumitööga (lisa G), mille vajadus tulenes eelnevalt kõrvale jäänud I2C protokollist. Üliõpilase jaoks tähendas selle käima panemine ainult internetist sobiva teegi leidmist. Selle parandamiseks koguti eelnevalt mitmes praktikumis laiali olnud I2C seadmete ülesanded. Praktikumi algusesse lisati ülevaade I2C protokolli olemusest ja tööst ning uuendati kasutusel olevate seadmete tutvustusi. Lisandus ülesanne, kus üliõpilane peab andmelehest ise üles leidma seadme aadressi ja registrid, kus vajalik info asub, misjärel peab üliõpilane alustama I2C vestlust ja registrite sisu välja lugema.

5.7 WiFi praktikum

Selles praktikumis (lisa H) tutvutakse WiFi ühendusega ja kasutades *socket*eid luuakse mitme kliendiga suhtlev server. Siia praktikumi lisati kasutuselolevate sensorite kirjeldused ja jagati ülesanded väiksemateks osadeks laiali. Ülesannete lahendamise käigus saab üliõpilane aru, et tegemist on samasuguste ühenduste otspunktidega nagu teises praktikumis käsitleti ja tekib võrdlus, kuidas sarnaste programmide implementatsioonid platvormide vahel erinevad.

6 2021/22 kevadsemestri tagasiside

2022. aasta kevadsemestril registreerus ”Nutilahenduste praktikumi” algselt 114 üliõpilast, nendest 78 on viimase kuu jooksul aktiivsed olnud. Autor viis kõigi osalejate seas läbi küsitluse, millele sai kokku 49 vastust, neist 34 eestikeelsetelt üliõpilastelt ja 15 inglisekeelsetelt. Vastanutest ainult kolm eestikeelset üliõpilast olid seda ainet varem võtnud. Kõik kolm varem ainet läbinud üliõpilast nõustusid, et aine juhendid olid paremad kui eelnevatel aastatel, kusjuures $\frac{2}{3}$ vastasid, et on tunduvalt parem. Mitmendat korda läbijad hindasid riistvaraprobleemidele kulunud aega väiksemaks kui eelneval korral.

> *”Juhendid on ütlustatud vomi ja sisuga. Lisandunud on sissejuhatav teooria läbitava materjali koha, mis on hea - annab aimu, millega praktikumis tegu. Lisandunud on juhendisse näidiskoodi. Näiteks signaali kuvamise osa, millele varem võis kuluda liiga palju aega, et õigeid vahendeid leida. Soovitavad lõpptulemused on rohkem lahti kirjutatud.”* - See on praktikumijuhendaja hinnang kursusele.

Kõige tihedamini välja toodud punkt küsitlusele vastanud üliõpilaste tagasisides oli kursuse maht. Seda mainiti kokku 10 korda. Eestikeelsete vastanute seas hinnati kursuse mahuks keskmiselt 83 tundi, mis on 1 tund vähem, kui 3 EAP kursus eeldaks, sealjuures suurim hinnang oli 160 tundi ja vähim hinnang 35 tundi. Inglisekeelsed vastanud hindasid kursuse mahuks keskmiselt 106 tundi, neist üks üliõpilane hindas kursuse mahuks 288 tundi, mis ei ole reaalne hinnang, kuna kõige rohkem praktikumides kohal käinud üliõpilane registreeris oma kohalolu 26 praktikumis, mis oleks 104 tundi kontakttööd ja selles aines ei ole kodutöid. Kui see anomaalia eemaldada, on ingliskeelsete vastanud üliõpilaste keskmise mahu hinnang 93 tundi, mis on 9 tundi üle 3 EAP normi.

Vastanud üliõpilaste tagasisidest selgus, et õppeeesmärgid olid 29 eestikeelse üliõpilase arvates selged ning 2 üliõpilase arvates jäid ebaselgeks. Ingliskeelsetest vastajatest hindasid õppeesmärke ebaselgeks 5 üliõpilast ja selgeks 6 üliõpilast.

Väitega, et kursuse materjalid on organiseeritud, nõustus 17 eestikeelset küsimustikule vastanud üliõpilast ja 5 ingliskeelset. Kokku 43 küsitlusele vastanud üliõpilast nõustus, et kursuse materjalid olid loogilised, 4 olid vastu.

Tagasisides hinnati kasutusel oleva riistvara tööpõhimõtete lahtiseletamine üldiselt heaks, selle

väitega nõustus 31 vastanud üliõpilast, vastu oli 8. Riistvara dokumentatsiooni hindasid üliõpilased samuti hästi arusaadavaks, sellega ei nõustunud ainult 4 üliõpilast. Riistvara probleemide lahendamisele kulunud aega pidasid 13 üliõpilast liiga suureks.

Tagasisidest juhenditele selgus, et ülesanded ei olnud piisavalt ühemõtteliselt ja/või selgelt lahti seletatud, sellise tagasiside andsid kokku 23 üliõpilast. Ingliskeelsete üliõpilaste vastustest tuli välja, et nende jaoks oli puudu näitekoode. Sellest järeldab autor, et nende programmeerimistaust on eestikeelsete tudengitega võrreldes tagasihoidlikum ning on vaja viia sisse muudatusi, et neid järele aidata.

Tagasiside andnud tudengitest 69% hindas, et peale aine läbimist nende huvi selles aines käsitletud teemade üle kasvas, 22% vastanutest hindas, et nende huvi ei muutunud, nende hulgast 82% olid enne kursust juba huvilised. Ainult 3 vastanud üliõpilast ütlesid, et nende huvi kursuse teemade vastu vähenes peale selle läbimist, kõikide nende tagasisides toodi välja, et kursuse maht oli nende arvates liiga suur.

Praktikumide jooksul ilmnes üks suurem probleem arendusplaatidega. Mingitel tingimustel on võimalik USB-UART silla kiip läbi põletada, mis lühistab kokku 5 V ja 3.3 V siinid ja sellest tulenevalt põleb läbi ka protsessor. See probleem on praeguseks ilmnenud ainult arendusplaatidel, millel ei olnud veel kaitsvat pleksiklaasi ümber, millest tulenevalt kahtlustab autor põhjusena staatilist elektrit. Antud probleem ilmnes 13 nädala jooksul kokku viiel arendusplaadil. Probleem ei ilmnenud kordagi autori testimise käigus, aga esines ühe juhendaja käes. Üliõpilased, kellel probleem ilmnes, ei saanud tihitipeale isegi aru, et midagi halvasti on enne, kui uue programmi pealelaadimine ebaõnnestus, seega ei õnnestunud neilt kasulikku informatsiooni saada.

7 Kokkuvõte

Bakalaureusetöö tulemusena valmis komplekt arendusplaate ja 7 praktikumi juhendid. Sellega võib tõdeda, et töös püsitatud eesmärgid said täidetud. Nii loodud riistvara kui ka juhendid läbisid 2022. aasta kevadsemestril esimese tuleproovi ”Nutilahenduste praktikumis” ja praeguseks hetkes on isegi suuremas osas kasutatavad, kui toodetud riistvara silmas pidada.

Riistvara töökindluse probleemid muutusid selle töö tulemusena palju lihtsamini kontrollitavaks. Üleüldiselt vähenes ka ühendusprobleemidele kulunud ajaressurss. Kahjustatud komponentide testimine muutus väga lihtsaks, on võimalik võtta suvaline arendusplaat, mis kogu riistvara funktsionaaluse läbi testib. Selle tulemusena olid ainult üksikud juhtumid, kus katkine riistvara jõudis tudengite kätte.

Kohandatud juhendid, mis küsisid üliõpilastelt rohkem selle kohta, kuidas nende kood töötab ja mida teevad ja tähendavad spetsiifilised osad selles, aitasid autori hinnangul üliõpilastel paremini aru saada, kuidas kasutusel olnud protokollid ja seadmed töötavad. Võrreldes möödunud õppeaastaga märkasid juhendajad ja töö autor, et tänavu kasutasid üliõpilased rohkem ametlikku dokumentatsiooni.

Tänuavaldused

Heiki Kasemägi, juhendamise ja nõu eest kogu protsessi käigus.

Teet Tilk ja Renno Raudmäe komponentide tellimise ja tootmisega abistamise eest.

Kerli Rhede keelekorrektuuri eest.

Viited

[1] S. Sinha, "State of iot 2021: Number of connected iot devices growing 9% to 12.3 billion globally, cellular iot now surpassing 2 billion." https://iot-analytics.com/number-connected-iot-devices/, 2021. refereeritud 25 Oktoober 2021.

[2] Congressional Research Service, "The internet of things (iot): An overview." https://sgp.fas.org/crs/misc/IF11239.pdf.

[3] A. Snead, "Beginner info diy smart home gadgets smart hvac options explained: 2 upgrades to save you money." https://smarterhomeguide.com/smart-hvac/#:~:text=What%20Is%20Smart%20HVAC%3F,of%20custom%20settings%20and%20schedules.

[4] Fibaro, "Smart hvac perfect climate in your home." https://www.intechopen.com/chapters/75630.

[5] Ring, "Smart security here, there, everywhere.." https://eu.ring.com/pages/security-cameras.

[6] Bitdefender, "Ring video doorbell pro under the scope." https://www.bitdefender.com/files/News/CaseStudies/study/294/Bitdefender-WhitePaper-RDoor-CREA3949-en-EN-GenericUse.pdf.

[7] robu.in, "Internet of things – iot advantages and disadvantages." https://iot-analytics.com/number-connected-iot-devices/, 2021. refereeritud 25 Oktoober 2021.

[8] E. E. LIIT, "Visioon eestist aastaks 2035." https://www.estonianelectronics.eu/EETL%20Visioon%202035_.pdf.

[9] "Arvutitehnika õppekava." https://www.tuit.ut.ee/et/arvutitehnika-oppekava.

[10] "Nutilahenduste praktikum (3 eap)loti.05.060." https://ois2.ut.ee/#/courses/LOTI.05.060/version/45ffb6d4-fe01-86ea-b57b-74791093c432/details.

[11] "Gitlab." https://gitlab.ut.ee/.

[12] "Python." https://www.python.org/.

[13] Arduino, "Arduino ide." https://www.arduino.cc/en/software.

[14] MaximIntegrated, "Ds18b20." https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[15] "How hc-sr04 ultrasonic sensor works & interface it with arduino." https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/.

[16] "Time-of-flight ranging sensor." https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html.

[17] ElectronicWings, "Ssd1306 oled display." https://www.electronicwings.com/sensors-modules/ssd1306-oled-display.

[18] "Ltd-6940hr datasheet." https://www.mouser.ee/datasheet/2/239/liteon_LTD-6940HR-1175224.pdf.

[19] https://www.bluetooth.com/.

[20] Cisco, "What is wi-fi?." https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html.

[21] "Universal synchronous asynchronous receive/transmit usart." https://www.ti.com/sc/docs/products/micro/msp430/userguid/ag_12.pdf.

[22] J. B. Jonathan Valdez, "Understanding the i2c bus." https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1652853721761&ref_url=https%253A%252F%252Fwww.google.com%252F.

[23] "Ds3231." https://datasheets.maximintegrated.com/en/ds/DS3231.pdf.

[24] ElectronicWings, "Bluetooth module hc-05." https://www.electronicwings.com/sensors-modules/bluetooth-module-hc-05-.

[25] R. P. Foundation, "Raspberry pi 4b." https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, 2022. vaadatud 14 Mai 2022.

[26] "Atmega328p." https://www.microchip.com/en-us/product/ATmega328P.

[27] T. A. Team, "Arduino nano." https://store-usa.arduino.cc/products/arduino-nano/, 2022. vaadatud 14 Mai 2022.

[28] "Esp8266." https://www.espressif.com/en/products/socs/esp8266.

[29] E. Systems, "Lolin d1 mini." https://www.wemos.cc/en/latest/d1/d1_mini.html, 2022. vaadatud 14 Mai 2022.

[30] E. Systems, "Esp32 series datasheet." https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf, 2022. vaadatud 14 Mai 2022.

[31] J. M-SHORT, "How to use a breadboard." https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard/all.

[32] "Stm32 mcu & mpu eval tools." https://www.st.com/en/evaluation-tools/stm32-mcu-mpu-eval-tools.html.

[33] "How to select your next development board." https://www.electronicsforu.com/buyers-guides/hardware-buyers-guide/select-next-development-board.

[34] "Mikrobus." https://www.mikroe.com/mikrobus.

[35] "At90usb647." https://www.microchip.com/en-us/product/AT90USB647.

[36] P. D. C. A. Church, "At90usb64x/128x support files for arduino." https://github.com/DynamicPerception/ArduinoAT90USB.

[37] Espressif, "Esp32-devkitm-1." https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/user-guide-devkitm-1.html.

[38] Espressif, "Development boards." https://www.espressif.com/en/products/devkits?id=ESP32.

[39] Espressif, "Esp-eye." https://www.espressif.com/en/products/devkits/esp-eye/overview.

[40] Espressif, "Esp32-sense-kit." https://docs.espressif.com/projects/espressif-esp-dev-kits/en/latest/esp32/esp32-sense-kit/user_guide.html.

[41] SONOFF, "Switchman m5 product specification." https://sonoff.tech/wp-content/uploads/2022/03/%E4%BA%A7%E5%93%81%E5%8F%82%E6%95%B0%E8%A1%A8-M5-V1.1-20220311.pdf, 2022. vaadatud 14 Mai 2022.

[42] "Farnell." https://ee.farnell.com/.

[43] "Mouser." https://www.mouser.ee/.

[44] "Tme." https://www.tme.eu/en/.

[45] Espressif, "Esp32 hardware design guidlines." https://www.espressif.com/sites/default/files/documentation/esp32_hardware_design_guidelines_en.pdf.

[46] S. Labs, "Usbxpress™ family cp2102n data sheet." https://www.mouser.ee/datasheet/2/368/cp2102n_datasheet-1634912.pdf.

[47] Theydiffer, "Difference between mini usb and micro usb." https://theydiffer.com/difference-between-mini-usb-and-micro-usb/.

[48] "Ld1117 datasheet." https://www.st.com/resource/en/datasheet/ld1117.pdf.

[49] "Cp2102n family datasheet." https://www.mouser.ee/datasheet/2/368/cp2102n_datasheet-1634912.pdf.

[50] Espressif, "Esp-idf." https://www.espressif.com/en/products/sdks/esp-idf#:~:text=ESP%2DIDF%20is%20Espressif's%20official,as%20C%20and%20C%2B%2B.

[51] J. S. Cook, "What is quadrature? linear & rotational encoding methods." https://www.arrow.com/en/research-and-events/articles/linear-and-rotational-encoding-methods.

[52] Altium, "Altium designer." https://www.altium.com/altium-designer.

[53] "Jlcpcb." https://jlcpcb.com/.

[54] "Protoflow e." https://www.tokimeku.com/products/lpkf-protoflow-e.

[55] "Läbipaistev plexiglas® gs ja xt." http://proplastik.ee/plexiglas/labipaistev-plexiglas-gs-ja-xt/.

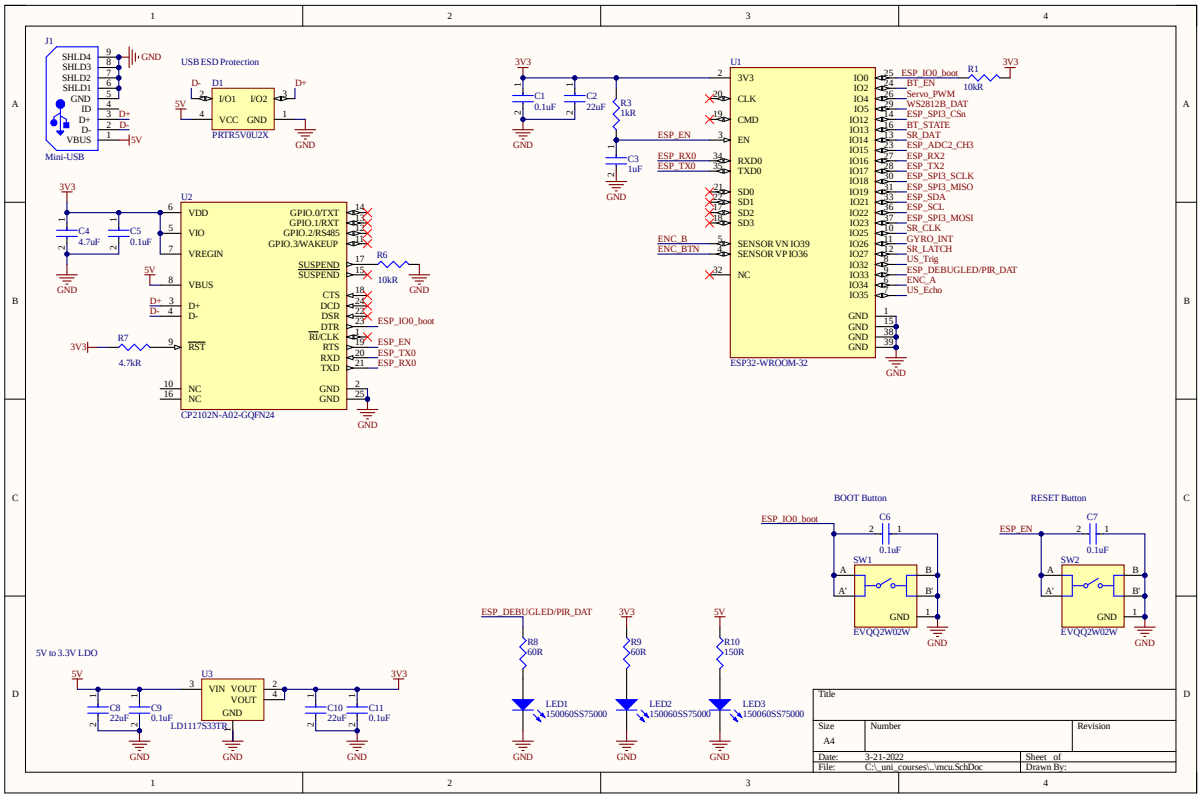
[56] "Kummipuks." https://ee.farnell.com/multicomp/3094black/feet-rubber-16x9-5mm-pk100-black/dp/320237?ost=320237.

[57] G. W. Neil C.C. Brown, "Ten quick tips for teaching programming," *PLoS Computational Biology*, vol. 14, 2018.

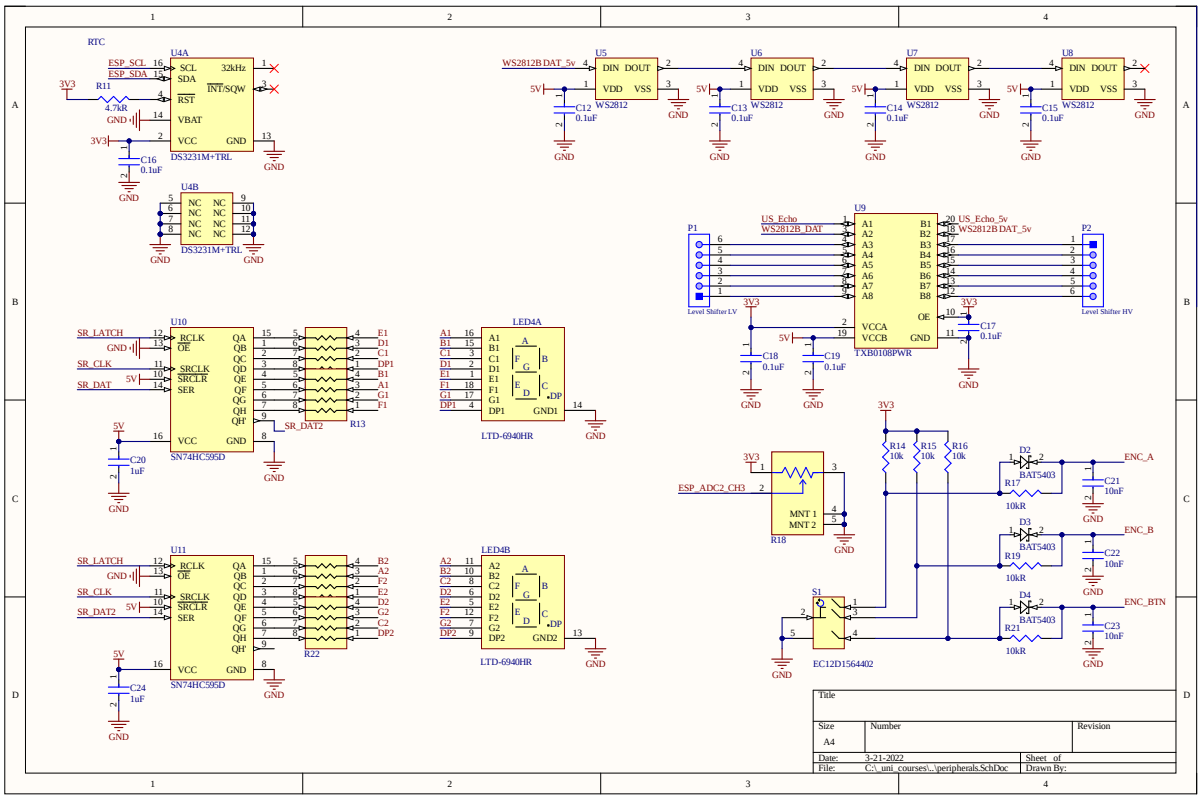
[58] "Gpio.rpi." `https://pypi.org/project/RPi.GPIO/`.

Lisad

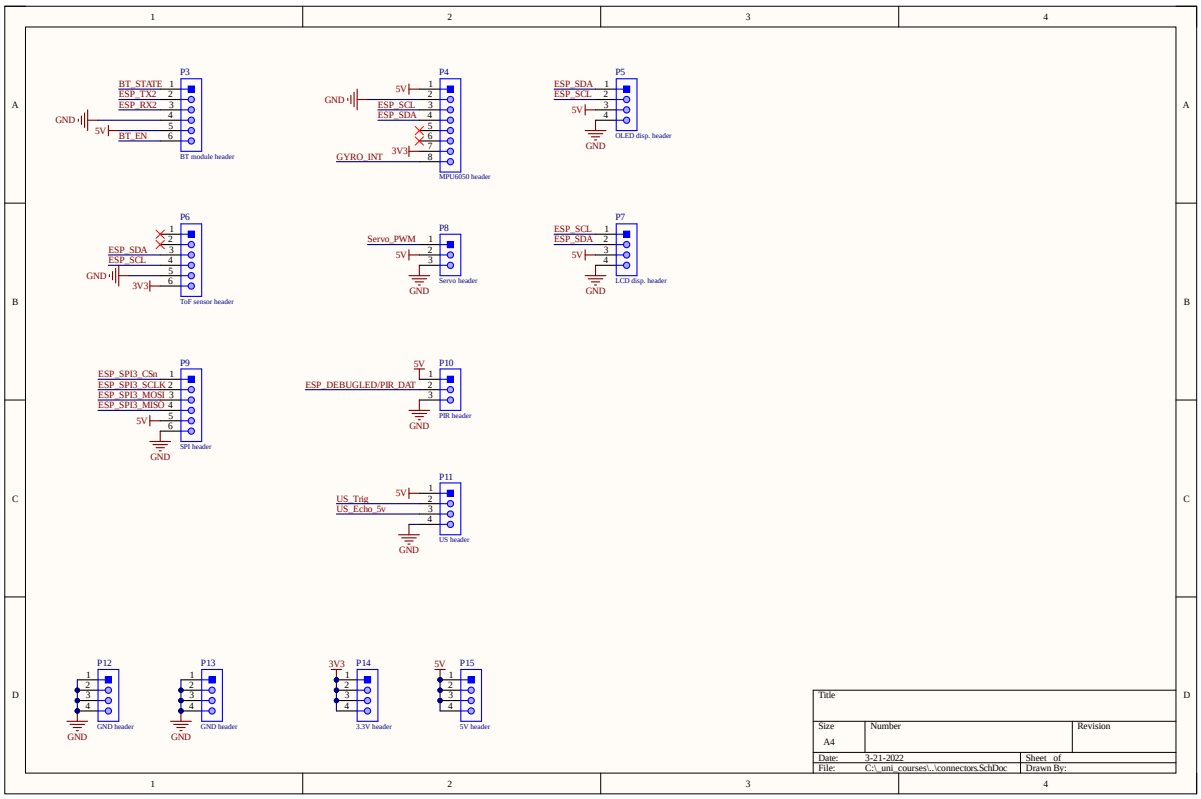
A Arendusplaadi dokumentatsioon



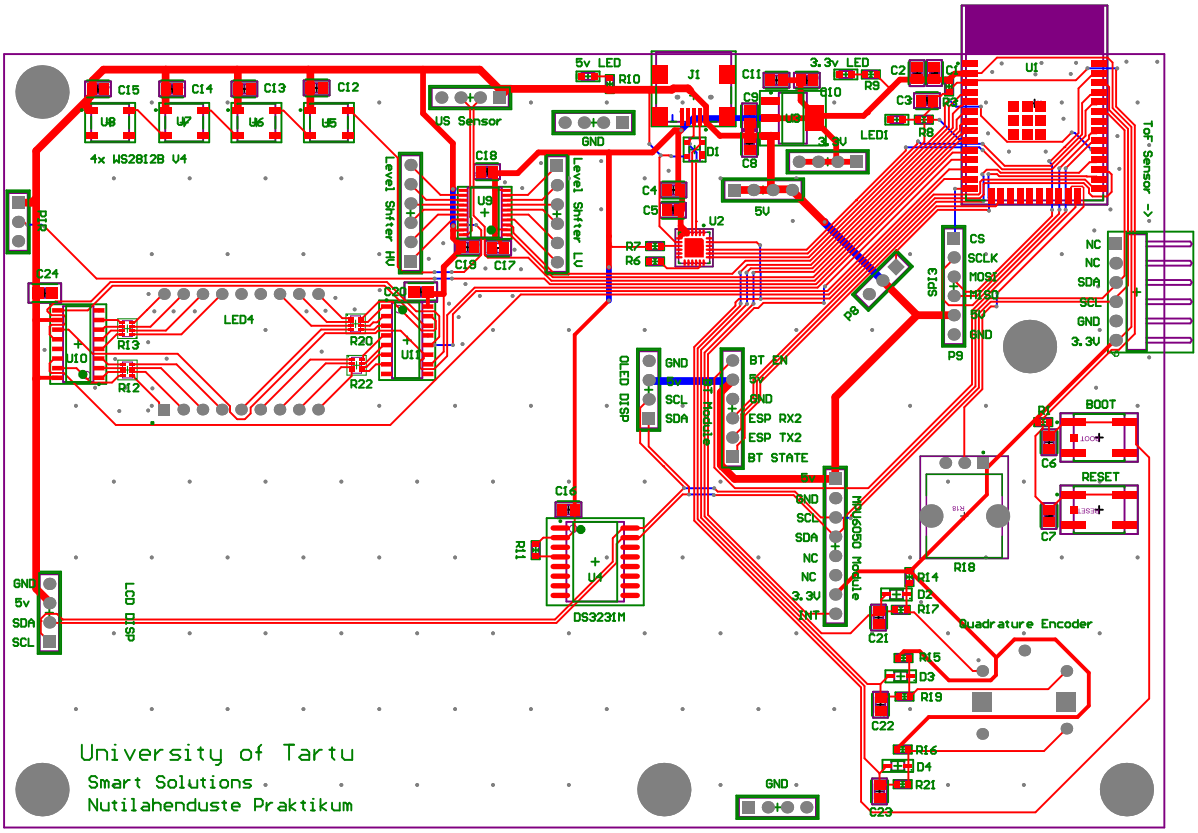
Joonis A.1: Arendusplaadi dokumentatsioon lk 1



Joonis A.2: Arendusplaadi dokumentatsioon lk 2



Joonis A.3: Arendusplaadi dokumentatsioon lk 3



Joonis A.4: Arendusplaadi dokumentatsioon lk 4

B Raspberry Pi praktikum

1 Raspberry Pi praktikum

1.1 Ülevaade

See praktikum tutvustab Raspberry Pi arvutit. Juhend sisaldab endas ülesannete kirjeldusi koos pseudokoodide ja vihjetega. Töö sujuvuseks on välja toodud ka probleemseid kohti.

1.2 Vajalikud vahendid

1. Internetiga arvuti
2. Raspberry Pi
3. microSD kaart
4. Maketeerimislaud (breadboard)
5. 10-segmendiline valgusdiood ja takisti
6. Nupp
7. 7-segmendiline valgusdiood

1.3 Praktikumis esmakordselt kasutusele tulevate teemade tutvustused

1. **Raspberry Pi**, edaspidi RasPi, on ARM arhitektuuril põhinev kompaktarvuti. Selle protsessor on paljuski sarnane Androidiga telefonides leitavale, kuna põhinevad samal arhitektuuril. RasPi't eristab telefonidest ja tavalistest x86-arhitektuuril põhinevatest arvutitest tema sisseehitatud GPIO-port (üldotstarbelised sisend-väljundviigud). RasPi operatsioonisüsteem ja failid asuvad välisel andmekandjal, milleks on enamasti microSD kaart, kuid on võimalik bootida ka USB-ühendusega andmekandjalt.

   GPIO (*General-Purpose Input/Ouput*) ehk üldostarbelised sisend-väljundviigud on tarkvaraliselt kontrollitavad viigud, mille külge saab kasutaja ühendada erinevaid elektroonikakomponete ja -seadmeid. RasPi GPIO viigud töötavad pingel kuni 3.3 V. GPIO-viigud

töötavad pinge mõõtmise peal, mitte voolu mõõtmise peal. Seda on kasulik meeles pidada pull-up/down takistite korral.

2. **Maketeerimislaud** ehk *breadboard* on elektroonika prototüüpimiseks kasutatav vahend. See võimaldab luua ühendusi komponentide vahel ilma jooteid kasutamata. See teeb plaadi korduvkasutatavaks. Kahjuks ei ole breadboardid ideaalsed ning nende ühendused tekitavad tihtipeale probleeme. Selle vältimiseks tuleb nendega töötades mitte kasutada üleliigset ning kõverate viikudega komponente sinna mitte toppida. Kui on probleeme halbade ühendustega, siis saab alati küsida juhendajalt multimeetrit, et võimalik viga üles otsida.

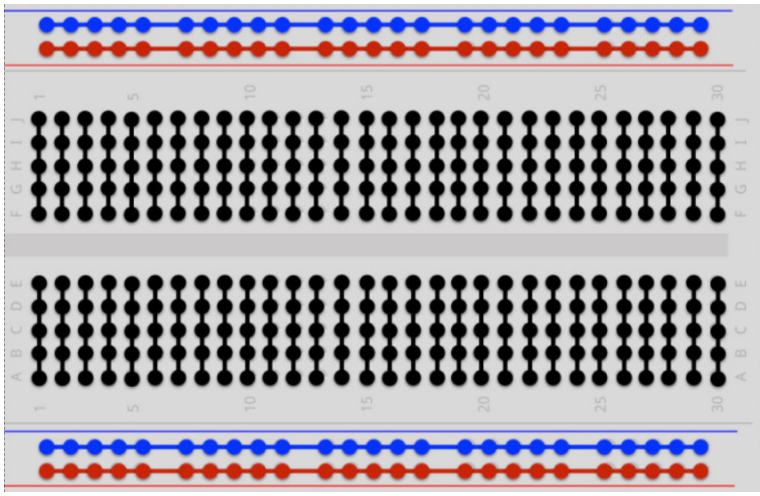


Figure 1. Breadboardi ühendused

3. **LED** (*Light Emitting Diode*}) ehk valgusdiood on pooljuht, mis eritab valgust, kui teda läbib vool. LEDidel on 2 viiku, mida nimetatakse elektroodideks: anood (+) ja katood (-). LED'i viike saab eristada pikkuse järgi, nimelt on anoodi jalg lühem. Kuna LED on diood, siis on tema pinge-voolu karakteristik mittelineaarne. See tähendab, et pinge väike tõus tekitab suure muutuse voolu läbilaskevõimes. Kuna LEDid vajavad võrdlemisi vähe voolu (enamasti 10 kuni 25 mA) ning enamus GPIO-viike on võimelised välja andma rohkem, on oluline ühendada LEDiga jadamisi ka takisti, mis voolu piirab. Selle takistuse saab välja arvutada valemiga

$R = \frac{V_toide - V_led}{I_led}$

kuid käesolva aine raames kasutame lihtsalt olemasolevaid takisteid (tavaliselt ca. 150 oomi).

7-segmendiline valgusdiood on number 8 kujuliselt paigutatud LEDidega seade, mille LEDid on tavalised LEDid. Erinevus on selles, et kõik elemendid jagavad ühist katoodi või anoodi. Praktikumis võib kohata mõlemat, tüübi saab kindlaks teha katseeksitusmeetodil või multimeetriga. Kontrollküsimus iseendale, kuidas seda testida võimalikult väheste veakohtadega?

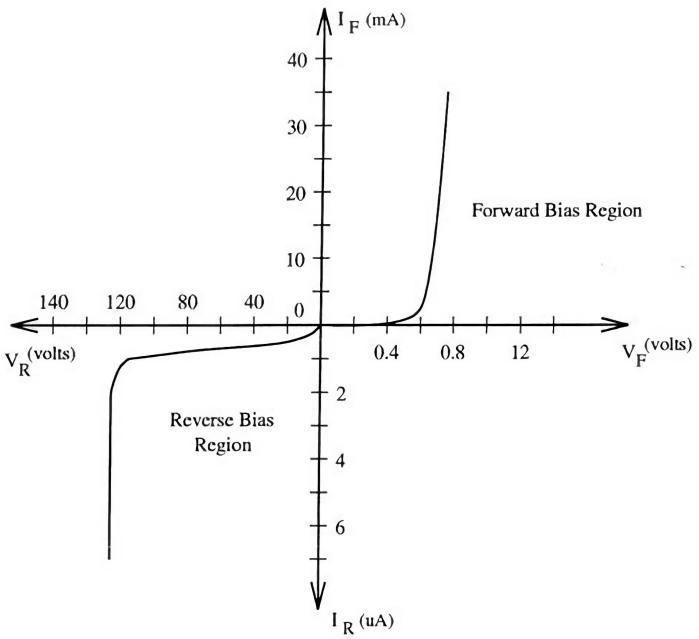


Figure 5.1: Diode Characteristics

Figure 2. Dioodi pinge-voolu karakteristik

4. **Nupp** on elektriskeemi element, millega on võimalik vooluringi ajutiselt või püsivalt kokku ühendada või katkestada. Seda kasutades tuleb silmas pidada, kuidas nupuvajutust tuvastatakse. Oluline on, et sellel ajal, kui nupp ei ole alla vajutatud, vooluring ei toimi. Kui nupp korraks alla vajutada, siis tekitatakse hetkeks vooluring ja nupuvajutust tuvastav vooluringi osa saab selle kohta signaali. Kui aga üritada näiteks multimeetriga mõõta pinget kahe punkti vahel, mis ei ole üksteisega ühendatud, siis on näha, et mõõteseadme näit pole stabiilne. See on põhjustatud keskkonna poolt tekitatud signaalidest, nt. staatiline elekter ja vahelduvvool, mida ühendamata juhe antennina vastu võtab. Sama efekt leiab aset, kui üritada ükskõik millise GPIO-sisendiga nupuvajutusi tuvastada. Selle vältimiseks tuleks kasutada pull-down või pull-up takistit. `https://grantwinney.com/using-pullup-and-pulldown-resistors-on-the-raspberry-pi/#a-simple-circuit`

5. **Pull-down(up) takisti** on oma olemuselt tavaline elektroonikas kasutatav takisti, mis ühendab signaali läbi iseenda vastavalt kas 0-potentsiaaliga (GND) või kõrge potensiaaliga (VCC). Seda üldist ideed kasutatakse paljudes skeemides, näiteks nupu ühendamisel GPIO-viigu külge. Nagu enne mainitud, ei saa jätta mõõtmist läbi viivat viiku (see viik, mis peaks kindlaks tegema, kas nuppu vajutati) ujuvasse (mitte ühendatud/*floating*/High-Z) olekusse. Selleks ühendame ta pull-down takistiga maa (GND) külge, mis tähendab, et nupu vaikeolek on madal. Nüüd, kui nupu teine pool on ühendatud VCC külge (RasPi puhul tuleb tähele panna, et kasutate GPIO viikudele turvalist pinget või takistit), siis gümnaasiumi füüsikatundi meelde tuletades näed, et nupule vajutades tekkib GPIO-viigul positiivne pinge, kui vool liigub läbi takisti.

RasPi'l on olemas ka sisseehitatud tarkvaraliselt aktiveeritavad pull-down ja -up takistid.

Python 3 RPi.GPIO moodulis sisseehitatud pull-down/pull-up taktistite aktiveerimine:

```
GPIO.setup(viik, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(viik, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

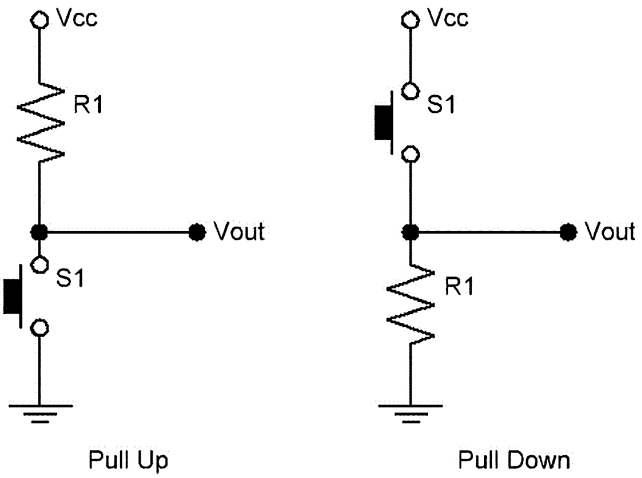


Figure 3. Pull-up ja pull-down takisti näide (Vout siinkohal GPIO-viik)

6. **git**: neil, kes pole varem versioonikontrolliga kokku puutunud, on võimalik tutvuda gitiga siin: `https://learngitbranching.js.org/`

1.4 Ülesanded

1. Ülesanne: git-i repositooriumi loomine

Kaesolevas aines kasutame oma failide hoiustamiseks `http://gitlab.ut.ee` keskkonda.

GitLab-i keskkonda saab siseneda Tartu Ülikooli autentimisega:

Sign in with Shibboleth

HTTPS-kloonimise kasutamiseks on vaja GitLabi kasutajale lisada parool. Selleks liikuda *Preferences*(*User Settings*) → *Password* ja lisada uus parool.

Samuti on mõistlik üle vaadata oma kasutajanimi, mida on vaja oma repositooriumi kloonimiseks, selle leiab *User Settings* → *Account* menüüst.

Repositooriumi loomiseks vali *Create blank project* ja selle nimeks pane *LOTI.05.060_eesnimi_perenimi*.

Visibility level vali *Private*. Selleks, et juhendajad saaksid sinu koodi kontrollida, jaga oma loodud repositoorium juhendajatega vähemalt õigustel *Reporter*.

Juhendajate emailid:

`heiki.kasemagi@ut.ee`
`kaarel.rhede@ut.ee`
`lauri.raudla@ut.ee`
`kristjan.laugesaar@ut.ee`
`jako.aimsalu@ut.ee`
`aleksandr.makarov@ut.ee`
`priit.rooden@ut.ee`
`herman.klas.ratas@ut.ee`

Oma repositooriumi kohalikku arvutisse saamiseks saab kasutada *git clone <aadress>* käsku. Aadressi saab kopeerida GitLabi repositooriumis **Clone** → **Clone with HTTPS** kastist.

Praktikumide jooksul tuleks kõik oma koodid hoiustada giti **organiseeritud** repositooriumisse (kausta).

Iga kord, kui mõne ülesande tehtud saad või praktikum lõppema hakkab, on vajalik teha *git commit*, et oma tehtud töö salvestada ja praktikumi lõpus *git push*, et see pilve laadida.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb juhendajale ette näidata oma giti repositoorium ning teha sinna esimene *commit* ja *push* loodud organiseerimissüsteemiga.

2. Ülesanne: Raspberry Pi seadistamine

Käesolevas ülesandes valmistad endale ette edaspidiseks kasutamiseks microSD kaardi, millele installeerid Raspberry Pi OS'i. Selle kohta leiab kõige ajakohasema juhendi *Raspberry Pi Foundation'i* lehelt `https://www.raspberrypi.org/software/`.

Operatsioonisüsteemiks vali **Raspberry Pi OS (Other)** alt **Raspberry Pi OS Lite(32-bit)**. Väga hiljuti tuli välja ka Raspberry OS 64-bitine versioon, kuid hetkel me seda ei kasuta, et vältida väga uue keskkonnaga kaasnevaid probleeme.

RasPi imageri kasutades luba SSH ja muuda *hostname*. Selleks on vaja kasutada *RasPi imageri* täpsemate seadete menüüd, millele saab ligi klahvikombinatsiooniga **CTRL + SHIFT + X**.

Advanced options menüüs tuleb:

1. seada oma RasPile uus *hostname*, et arvutivõrke skaneerivates rakendustes oleks *hostname* lihtsamini leitav;

2. lubada SSH ja seada sellele parool;
3. panna linnuke *Configure wifi* juurde, ning valida riigiks EE ja sisesta parool ja SSID
4. salvestada SSH parool giti, et vältida unustamist.

Enamasti saab *Advanced options* menüüst seadistada ka WiFiga ühendamist, kuid hetkel teadmata põhjusel ei tööta *advanced optionsis* sisestatud parooli kirjutamine WiFi konfiguratsioonifaili, seega muudame seda käsitsi peale OSi installeerimist. See oleks vajalik ka graafilist kasutajaliidest kasutades, kuna RasPi OSil puudub võimalus läbi graafilise kasutajaliidese(GUI) peidetud SSIDga WiFi võrkudesse ühendumist seadistada.

Kui OS-i installeerumine on lõppenud, **ära võta** SD-kaarti arvutist välja, sest veel on vaja seadistada praktikumis kasutusel olev WiFi, milleks laboris 140 on peidetud WiFi võrk ***robootika***.

Raspi Wifi konfiguratsioonifaili muutmiseks tuleb leida mälukaardi *rootfs* partitsioon, sealt liikuda kausta *etc/wpa_supplicant* ning muuta faili *wpa_supplicant.conf*. Selleks tegevuseks on vaja *root* õigusi.

Ära kopeeri terminali või terminalirakendustesse teksti, kirjuta see käsitsi ja TAB klahvi abiga. Kopeerimine põhjustab nähtamatute karakterite jõudmist teksti, mis teeb süsteemifailid ja käsud katki.

Kui terminalis saab konfiguratsioonifaili muuta käsuga

```
sudo nano /media/kasutaja/rootfs/etc/wpa_supplicant/wpa_supplicant.conf
```

Selle faili sisu peaks olema[source, wpa_supplicant.conf]

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=EE
network={
    scan_ssid=1
    ssid="robootika"
    psk="DigiLaboriArvutiKlass"
    key_mgmt=WPA-PSK
    }
```

Käsk lahtiseletatult:

1. sudo - *superuser do* ehk täida järgmine käsk admini (unixis *root*) õigustes;

2. nano - terminalis teksti redigeerimise rakendus;

3. /media/kasutaja/rootfs/etc/wpa_supplicant/wpa_supplicant.conf - absoluutne failitee failini *wpa_supplicant.cont* ehk teed alustatakse juurkataloogist (”/” kõige ees).

Nüüd võib SD-kaardi arvutist välja võtta ja sellega RasPi tööle panna.

Järgmiseks on vaja SSHd kasutades oma RasPi-sse sisse saada. Selleks peab esmalt kontrollima, et arvuti on ühendatud RasPi-ga samasse võrku. Siis tuleb leida oma RasPi IP. Selleks on palju erinevaid võimalusi, näiteks terminalirakendus *arp-scan*, mis seab vastavusse kõik võrgus olevad IP-aadressid vastavate seadmete MAC-aadressidega, mis on kõigil seadmetel unikaalsed. Oma RasPi *arp-scan* väljundis teistest eristamiseks on laboris olevatele RasPidele nende MAC-aadressid peale kleebitud. Kui sinu RasPi peal on oleval kleepsul on MAC aadressi ees kirjas ETH, siis erineb see WiFi võrgus olevast 1 võrra(kuueteistkümnendsüsteemis).

```
sudo arp-scan --localnet
ssh *RasPi kasutajanimi*@*RasPi IP*
```

TIP: Leia mida Linuxi terminalis ”|” teeb, ja kuidas kasutada käsku grep, ning mida see teeb.

Kui sa oled selle punktini jõudmise ajaks oma RasPi parooli ära unustanud, siis (siin) on juhend, kuidas seda muuta.
Arvestus: Käesoleva ülesande arvestuse saamiseks peab:

1. olema giti üles pandud RasPi parool

2. juhendajale demonstreerima SSH ühendust enda RasPiga;

3. RasPilt giti repositooriumisse tegema commiti lab1_task2, selle raames on vaja installeerida git.

Enne edasi liikumist tuleks uuendada tarkvarapakettide andmebaasi, et see sisaldaks kõige värskemaid versioone olemasolevatest lisadest. See on vajalik selleks, et oleks võimalik repositooriumis olevat tarkvara installeerida, sest kaardile tehtud installatsioon on üsna minimaalne. Seda saab teha käsuga

```
sudo apt update
```

3. Ülesanne: GPIO alused

Üks lihtsamaid ja mugavamaid variante kaugelt tarkvara arendamiseks on kasutada tasuta tekstiredaktorit *Visual Studio Code*, millele on üle SSH toimiv kaughalduse laiendus. Selle jaoks tuleb oma arvutis alla laadida tekstiredaktor ise, seejärel lisapakett Remote -SSH. *VSCodes* saab klahviga **F1** ligi otsingumenüüle. Leides sealt *SSH: Connect to a Host*, saab kasutada terminalist tuttavat *ssh*-käsku, Pärast ühendamist on võimalik otse arenduskeskkonnast RasPi

failisüsteemi sirvida, uusi faile luua, koodi kirjutada ja ka RasPi käsurida kasutada.

GPIO (General Purpose Input/Output) on üldotstarbeline port, millega saab suhelda väliste elektroonikaseadmetega ja neid kontorollida, nt. LEDid, servod (mootorid), erinevad andurid. GPIO-viikude puhul on oluline jälgida, millisel pingel ühendatav seade töötab, sest kõrgemat pinget rakendades on kahjustused tõenäolised. RasPi ja hiljem kasutusele tuleva ESP32 arendusplaadi GPIO viigud töötavad 3.3-voldisel pingel, kuid mujal võite kokku puutuda ka teistel pingetel töötavate GPIO-viikudega. Näiteks Arduino Nano ja Uno töötavad pingel 5 V. See ei tähenda, et neid seadmeid kokku ühendada ei saaks, lihtsalt on vajalik kasutada mingit pinget tõlkivat skeemi (*level-shifter*), et hoida pinge mõlema seadme jaoks õigel tasemel ilma signaali ja seadmeid rikkumata.

RasPi'l kasutame GPIO viikude manipuleerimiseks Pythoni moodulit *RPi.GPIO* (Dokumentatsioon). See peaks olema Raspberry Pi OS'is koheselt olemas, kuid kui mingil põhjusel seda ei ole, siis saab selle installeerida käsuga:

```
sudo apt install python3-rpi.gpio
```

See moodul kasutab kahte erinevat viikude nummerdamise süsteemi: BCM ja BOARD. Kusjuures ei ole mitte mingisugust vahet, kumba nummerdamissüsteemi kasutada, kuid peab meeles pidama, et need ei ole ühilduvad. BOARD-süsteemis on viikude numbriteks nende füüsiliste asukohtade loendid RasPi plaadil, seevastu BCM kasutab kindlaksmääratud GPIOxx nummerdust.

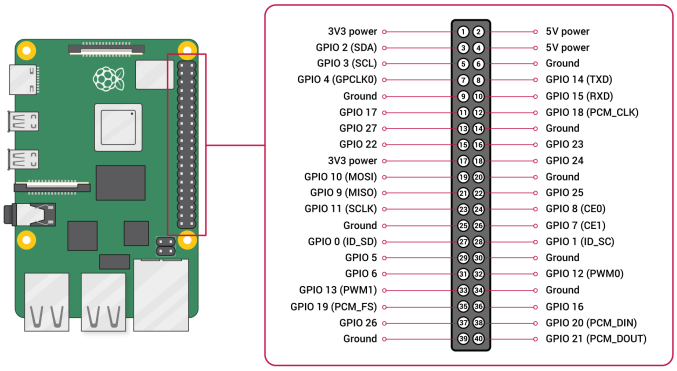


Figure 4. RasPi *pinout*

Näide, kuidas BOARD-viigu 12 külge anoodiga ühendatud LED 5 sekundiks põlema panna

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

def LED_example():
```

```
    led_pin = 12
    # voib olla ka list nt led_pin = [12, 16, 18]
    GPIO.setup(led_pin, GPIO.OUT, initial=GPIO.LOW)
    # GPIO.LOW  = False = 0
    GPIO.output(led_pin, GPIO.HIGH)
    # GPIO.HIGH = True  = 1

    start = time.time()
    while time.time() < start + 5:
        pass

    GPIO.output(led_pin, GPIO.LOW)
    GPIO.cleanup()

if __name__ == "__main__":
    LED_example()
```

Arvestus: Käesoleva ülesande arvestuse saamiseks peab:

1. selgitama juhendajale:
   (a) kuidas ülaltoodud koodis viivitus saavutatakse ning kas-miks-kuidas on see viis parem kui time.sleep(n);
   (b) miks on hea tava kasutada konstruktsiooni

   ```
   if __name__ == "__main__":
       #code body
   ```

   (c) mida teeb rida *GPIO.setwarnings(False)*
   (d) mida teeb rida #!/usr/bin/env python3
2. demonstreerima juhendajale kahte valgusdioodi vilkumas vastandfaasis sagedusel 1 Hz;
3. tegema giti repositooriumisse tegema _commit_i lab1_task3

4. Ülesanne: GPIO nupp

Selle ülesande lahendamiseks soovitame tutvuda olekumasina (*finite-state machine*) ja sisenditega RasPil.

Koodi mall

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

def lab1_task4():
    #setup

    try:
        while True:
            #main loop

    except KeyboardInterrupt:
        print("\n keyboard interrupt")
        GPIO.cleanup()

if __name__ == "__main__":
    lab1_task4()
```

Arvestus: Käesoleva ülesande arvestatud saamiseks peab:

1. selgitama juhendajale, mis on olekumasin ja kuidas seda sellise probleemi lahendamisel rakendada saab;
2. demonstreerima juhendajale programmi, mis suudab:
   (a) kuvada 7-segmendilisel LEDil numbreid 0-st 9-ni sammuga 1. 9-ni jõudes alustatakse uuesti 0-st ja kogu tsükkel kordub uuesti;
   (b) esimene nupulevajutus muudab numbrite kuvamise järjekorra vastupidiseks - suuremast väiksemani, 0-ni jõudes alustatakse 9-st ja kogu tsükkel kordub taas. Peale nupulevajutust on esimeseks numbriks jooksev number nupulevajutamise hetkel. Samm on 1;
   (c) teisel nupulevajutamisel hakatakse numbreid kuvama juhuslikult vahemikus 0..9;
   (d) kolmas nupulevajutus taastab esialgse olukorra nagu punktis 1 alates jooksvast numbrist.

Eelpool kirjeldatud skeem on korratav ilma programmi taaskäivitamata ja nupuvajutuse tuvastamisel on viivis väiksem kui 0.01s.

Soovitav on ülesandele läheneda ositi ehk näiteks kõigepeale panna tööle 7-segmendiline LED, seejärel nupp ja lõpuks töötavad osad kokku kombineerida. Ka koodi kirjutamisele läheneda samal viisil, et kõigepealt saada tööle numbrite kuvamine kui selline, siis nende kasvavalt loendamine, siis kahaneval, juhuslik ja lõpuks kõik kolm kokku panna. Niimodi on koodist ja ühendustest lihtsam vead üles leida.

C Socketite praktikum

2 Socketite praktikum

2.1 Ülevaade

See praktikum tutvustab *socketeid* pythonis ning SSH-protokolli võimalusi. Juhend sisaldab endas ülesannete kirjeldusi koos pseudokoodide ja vihjetega. Töö sujuvuseks on välja toodud ka probleemseid kohti.

2.2 Vajalikud vahendid

1. Internetiga arvuti
2. Raspberry Pi
3. microSD kaart
4. Maketeerimislaud (breadboard)
5. 10-segmendiline valgusdiood ja takisti
6. 7-segmendiline valgusdiood

2.3 Praktikumis esmakordselt kasutusele tulevate teemade tutvustused

1. **IP (*Internet Protocol*)** defineerib, kuidas rakendused saavad luua andmevahetuseks kanaleid üle IP-d kasutava võrgu. See defineerib, kuidas adresseerida ja ruutida (route) iga pakett õigesse sihtpunkti. Igal sisenemisel uude võrku suunab võrgulüüs (gateway) paketi temas sisalduva aadressi poole edasi. IP-st on kasutusel 2 erinevat versiooni - IPv4 ja IPv6.

   IPv4-võrgus on igal seadmel 32-bitine unikaalne IP aadress, mis tähendab et ühte võrku saab ühendada kuni 2^{32}=4,294,967,296 seadet. See on tänapäeval probleem, sest võrku ühendatud seadmete arv on plahvatuslikult kasvanud. Osaliselt selle tõttu hakati laialdaselt kasutama võrguaadresside tõlkimist või maskeerimist ehk NAT-i (*Network Address*

Translation) , mis võimaldab terve võrgu ühe IP aadressi taha ehitada, kuid sellega kaasnevad ka probleemid. Näiteks on NAT-i konfigureerimine ajamahukas, võib muuta liiklust aeglasemaks ning takistada kahel seadmel otseühenduse loomist.

IPv6 arendati välja kõigi nende ja muude puuduste likvideerimiseks. IPv6 kasutab 128-bitiseid aadresse, mis teeb umbes $3.4*10^{38}$ aadressi. IPv6-aadresse esitatakse kolmekümne kahe kuueteistkümnendsüsteemis numbrina, näiteks 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Kuna nii pikkade aadresside kasutamine ei ole alati vajalik ja mugav, siis kasutavad enamus alamvõrke IPv4, samuti meie siin praktikumides.

2. **TCP (*Transmission Control Protocol*)** määrab, kuidas andmed pakettideks tükeldatakse ja sihtkohas tagasi kokku pannakse. See on usaldusväärne, stabiilne, vigade kontrolliga ja andmete järjestatud edastusega andmevahetusprotokoll, mis kasutab IP-d. TCP puhul on oluline, et see protokoll kontrollib, kas kõik andmed on kohale jõudnud ning probleemide korral saadab kadunud osa uuesti. TCP on kasulik ühenduseks, kus iga pakett on tähtis.

3. **UDP (*User Datagram Protocol*)** kasutab samuti IP-d, kuid erinevalt TCP-st ei kontrolli ta mingil viisil, kas andmed on üldse kuskile jõudnud, samuti puudub järjestuse ja korduvate andmete kontroll. UDP on kasulik näiteks voogesituseks, kus on oluline, et enamus andmeid jõuaks paljude klientide juurde. On oluline tähele panna, et UDP puhul ei ole sellist suunatud ühendust nagu TCP-s, vaid rakendatakse ideed, et pakettide saatmine/vastuvõtt toimub kõigilt.

4. **Socket ehk pesa** on tarkvaraline algus- või lõpp-punkt võrku ühendatud seadmel, millelt saadetakse või millega võetakse andmeid vastu. Iga programm, mis kuidagi võrguga suhtleb, kasutab selleks socketeid. Näiteks uue veebibrauseri akna avamine tekitab socketi, mis ühendub sisestatud aadressile vastavas serveris oleva socketiga, mille järel saab toimuda andmete vahetus. On oluline ära mainida, et socketid ei kontrolli osapoolte autentsust ega taga liikuva info konfidentsiaalsust, neid on võimalik pealt kuulata ja on võimalik esineda teise osapoolena, et saata (vale)infot. Selleks, et selline ühendus turvaliseks teha, on vaja üle socketi liikuvad andmed krüpteerida, milleks on olemas protokollid TLS (*Transport Layer Security*) ja SSL (*Secure Socets Layer*).

   Kunagi ei tohi luua faili nimega *socket.py*, sest täpselt selle nimega teek on juba olemas. Lisaks sellele ei tohiks kunagi luua programmifaile, mis ükskõik millise operatsioonisüsteemis kasutatava faili või skriptiga nime jagab, samuti ei ole soovitatav neid kasutada muutujanimedena.

5. **Klient-server mudel** on hajutatud struktuur, kus klient või kliendid kasutavad serveri resursse. Tavaliselt ootab server, et klient temaga ühenduse algataks, seejärel saab klient kasutada serveri ressursse.

6. **Port** on ühenduse algus-/lõpp-punkt IP *stackis*. *Stack* on kommunikatiooniprotokollide (nagu TCP/IP) tarkvaraline teostus. Igal IP aadressil on 2^{16}=65536 porti, millest osade kasutus on standardiseeritud (siit).

7. **Lõim (*thread*)** on teistest programmiosadest eraldiseisvalt ja sõltumatult täidetav programmi osa.

8. **OOP (*Object Oriented Programming*)** ehk objektorienteeritud programmerimisviis on paradigma, kus programm koosneb kas osaliselt või täielikult osadest, mida nimetatakse objektideks. Objekt koosneb andmetest ja koodist (protseduurid ehk meetodid), mida saab neile (või teistele) andmetele rakendada. Näiteks ei ole Pythonis sõne (*String*) teglikult lihtsalt baitide massiiv, nagu see on näiteks C-keeles, vaid objekt, millest tuleneb võimalus kasutada konkreetsel *String*-tüüpi objektil tema meetodeid, näiteks *.lower()*, mis muudab juba olemasolevat objekti, mitte ei kirjuta seda üle uuega. Seda on mõistlik meeles pidada, kuna järgnevate ülesannete lahendamisel puutute sellega kokku.

2.4 Ülesanded

1. Ülesanne: Socketi klientprogramm

Serverisse ühe sõnumi saatva ja seejärel ühe vastuse võtva klientprogrammi sammud on:

1. Luua socketi objekt	1. socket.socket()
2. Ühenduda serveri külge	2. .connect()
3. Saata sõnum	3. .sendall()
4. Võtta vastu sõnum	4. .recv()
5. Socket sulgeda	5. .close()

Siin ei ole välja toodud nendele käskudele vajalikke parameetreid, mille kohta leiab infot Pythoni Socketi teegi dokumentatsioonist. Tuleb tähele panna, et vaatad oma *Pythoni* versioonile vastavat dokumentatsiooni (ülemine vasak nurk).

Näide veahaldusest socketi loomisel

```
import socket, sys

try:
    s = socket.socket()
except socket.error as msg:
    print('failed to create socket with error code: ' + str(msg.args[0])
    sys.exit()
```

Ülesande eesmärgiks on vaja luua programm, mis:

1. haldab vigu korrektselt ning teavitab kasutajat, kui viga tekkis;
2. võtab käsurealt sisendina ühenduse parameertid (IP aadress ja pordi number);
3. saadab serverisse käsurealt sisseloetud sõnumeid, peale iga sõnumi saatmist väljastab saadud vastuse;
4. töötab, kuni käsurealt sisestatakse tühik, seejärel programm sulgeb socketi ja lõpetab töö.

Arvestus: Käesoleva ülesande arvestuse saab koos järgmise ülesandega.

2. Ülesanne: Socketi serverprogramm

Eelnevalt kirjeldatud kliendiga sõnumi vahetava socketserveri sammud on:

1. Luua socketi objekt	1. socket.socket()
2. Siduda socket hosti ja pordi külge	2. .bind()
3. Oodata saabuvaid ühendusi	3. .listen()
4. Võtta vastu saabuv ühendus	4. .accept()
5. Võtta vastu sõnum	5. .recv()
6. Saata vastus	6. .sendall()
7. Socket sulgeda	7. .close()

Ülesande eesmärgiks on vaja luua programm, mis:

1. teavitab kasutajat informatiivselt tekkinud vigadest ning haldab neid korrektselt;

2. võtab käsurealt sisendina serveri parameertid (IP aadress ja pordi number) ja loob nendega serveri;
3. väljastab käsureale klientidelt tulnud sõnumid;
4. vastab klientide sõnumitele neid tagasi kajades lisada ”Echo from server: ”;
5. võimaldab korraga ühenduda mitmel kliendil;
6. lõpetab töö kontrollitult **Ctrl+C** katkestuse peale.

Näide, kuidas luua ja käivitada funktsioon teises threadis

```
import threading, time

def thread_example(output):
    while(True):
        print(output)
        time.sleep(2)

example_output = "hello world"
threading.Thread(target=thread_example,
    args=(example_output,)).start()
```

Arvestus: Käesoleva (ja eelmise) ülesande arvestuse saamiseks tuleb:

1. seletada juhendajale, mis eesmägil on järgnevas koodis võrdusmärkide järel olevad parameetrid ning mida need tähendavad ja mis valikud veel olemas on. Selle kohta tuleks oma programmi sees kommentaar kirjutada.

   ```
   class socket.socket(family=AF_INET, type=SOCK_STREAM,
   proto=0, fileno=None)
   ```

2. seletada juhendajale, kuidas oma koodis veahalduse lahendasid;
3. demonstreerida juhendajale omavahel ühendatud socketserverit ja -klienti.

   Mõlemad programmid peavad täitma ülaltoodud eesmärke;
4. seletada juhendajale, mida teeb *str.encode()* ja *str.decode()*;
5. saama aru, mis on socketid ja kuidas need töötavad.

3. Ülesanne: SSH tunnel

Siiani oleme socketeid kasutanud ühe seadme sees suhtlemiseks nii, et meie host on *localhost*, kuid enamasti paikneb host mõnes teises seadmes. Siis tuleb mängu tulemüür, mis on kohtvõrku ja ülejäänud internetti eraldav turvaseade, mis kaitseb kohtvõrku autoriseerimata ligipääsu ning pahavara eest. Pikemalt saab lugeda siit.

Selleks, et oma socketserver välismaailmale nähtavaks teha, saab selle üles seada oma arvuti IP aadressil, kasutades porti, mis on läbi tulemüüri lubatud (avatud). Sellisel juhul saavad sellele pordile ligi kõik, kes sinna pöörduda oskavad ning nagu enne räägitud, ei ole loodud ühendus privaatne. See annab potentsiaalselt igale ründajale ligipääsu kummaskis otsas töötavale programmile ja nende vahel liikuvale infole, mida on omakorda võimalik ära kasutada. Sellest tulenevad turvariskid on näiteks programmeerija tehtud vead, mis võimaldavad ründajal ligi pääseda programmi privaatsetele osadele või andmetele, krüpteerimata andmed, mida on võimalik pealt kuulata või nõrgad paroolid eeldusel, et enne suhtluse alustamist klient peab ennast autentima.

Selle vältimiseks on võimalik kasutada SSH tunneleid (sageli öeldakse selle kohta ka *port forwarding*) ning see töötab ideel, et oma seadme üks port mapitakse mõne teise seadme valitud pordi külge. Peale tunneli loomist on tunneli ühe otsa poole pöördudes võimalik suhelda tunneli teise otsaga, kusjuures andmed nende vahel liiguvad krüpteeritult ehk ühendus on turvaline. Siinkohal tuleb silmas pidada ja mitte segamini ajada, et SSH tunnel on turvaline keskond teistele teenustele, nt. HTTP, aga samas võib SSH ise ka olla teenus, mis tunnelit kasutab.

3.1 Ülesanne: SSH lokaalpordi tunneldamine

```
ssh -L local_port:destination_ip:remote_port ssh_server_ip
```

Lokaalpordi tunneldamine on põhimõtteliselt teise masina pordi enda masina pordile mappimine.

Näide kuidas tunneldada ut.ee port 80 localhosti porti 8080

```
ssh -L 8080:ut.ee:80 localhost
```

Arvestus: Käesoleva ülesande arvestuse saamiseks peab:

1. demonstereerima juhendajale ülesandes 2.1 ja 2.2 loodud serveri-kliendi suhtlust nii, et server töötab võrgus kättesaadava IP aadressi pordil, kuid teises arvutis töötav kilent pöördub localhosti pordi poole;
    - seejuures localhosti port erineb serveri originaalsest pordist.

3.2 Ülesanne: SSH kaugpordi tunneldamine

```
ssh -R remote_port:localhost:local_port ssh_server_ip
```

Kaugpordi tunneldamine on põhimõtteliselt enda masina (lokaalne) pordi teise masina (kaug) pordile suunamine. See teeb võimalikus enda masinas olevale serverile ligi saada läbi teise masina pordi, isegi kui enda arvutil on see sisenev port tulemüüri poolt keelatud. Sisuliselt tähendab see seda, et enda arvutist lükatakse teise arvutisse tunnel, mis sealt kohe tagasi suunatakse ja liiklus hakkab kulgema samuti selles suunas - kaugarvuti poolt lokaalsesse. Tuleb silmas pidada, et kaugpordi tunneldamine tuleb alati algatada lokaalarvutis ja tulemüüris peab lokaalarvuti väljuv port olema avatud.

Arvestus: Käesoleva ülesande arvestuse saamiseks peab:

1. demonstreerima juhendajale ülesandes 2.1 ja 2.2 loodud serveri-kliendi suhtlust nii, et server töötab localhosti pordil ning teises arvutis asuv klient pöördub vastavalt enda localhostil oleva pordi poole;
    - seejuures pordid, mida server ja klient kasutavad, on erinevad.

4. Ülesanne: Kontroll üle socketserveri

Ülesande eesmärk on koos tööle panna kogu seni kirjutatud kood. Selleks peab server jooksma RasPi peal ning praktikumis 1 loodud skeemid tuleb uuesti kokku ühendada. Käesolevas ülesandes tuleks *importida* 1. praktikumis loodud koodifailid. Kui nendes olev kood ei ole korrektselt funktsioonidesse jagatud ja/või ei oma kogu funktsionaalsust, mida on siin vaja, tuleks seda vastavates failides teha.

Arvestus: Käesoleva ülesande arvestuse saamiseks peab:

1. valmima programm, mis võimaldab serverisse ühenduvatel klientidel (korraga rohkem kui 1) kontrollida 1. praktikumis loodud programme;
    - klient saab kohe peale ühendamist teada, milliseid käske on võimalik kasutada, samuti peaks see samm toimuma peale alamprogrammi valimist;
    - kliendid saavad kontrollida ühte neist programmidest korraga;
    - kaks klienti ei saa sama programmi üheaegselt kontrollida - rakendub kontroll, kas alamprogramm pole kasutuses;
    - klient peab saama kontrollida mõlemat programmi ühe sessiooni ajal, kus enne teise üle kontrolli võtmist on esimese kontroll loovutatud;
    - ülesandes 1.3 loodud alamprogrammi kasutaval kliendil peab olema võimalik:

    - küsida alamprogrammi olekut (vilgub/ei vilgu);
    - muuta vilkumise sagedust;
    - lülitada vilkumist eradi sisse ja välja;
    - loovutada kontroll selle alamprogrammi üle;
  - ülesandes 1.4 loodud alamprogrammi kasutaval kliendil peab olema võimalik:
    - küsida alamprogrammi olekut (kuvab numbreid/ei kuva);
    - muuta numbrite loendamise kiirust;
    - valida üles, alla ja suvalise loendamise vahel;
    - loovutada kontroll selle alamprogrammi üle;

2. oma koodi koos juhendajaga läbi käima ja tekkivatele küsimustele vastama;

3. saama aru, kuidas ja mida loodud programm teeb.

D UART ja ESP32 praktikum

3 UART ja ESP32 praktikum

3.1 Ülevaade

Käesolev praktikum tutvustab järgmistes praktikumides kasutatavat ESP32 arendusplaati, Arduino keskkonda, milles programmeerimine hakkab toimuma ning USART suhtlusprotokolli.

3.2 Vajalikud vahendid

1. Internetiga arvuti
2. Raspberry Pi
3. microSD kaart
4. ESP32 arendusplaat

3.3 Praktikumis esmakordselt kasutusele tulevate teemade tutvustused

1. **ESP32** on sisseehitatud WiFi (802.11 b/g/n) ja Bluetooth(v4.2 BR/EDR ja BLE) võimekusega mikrokontroller, mis on tänu sellele ja oma madalale hinnale IoT lahendustes laialdaselt kasutusel. Peab ära mainima, et ESP32 suudab kasutada ainult 2.4 GHz sagedust. ESP32'l on võimekus suhelda üle kõigi laialdaselt kasutusel oleva üle juhtmete toimetavate protokollide (USART, SPI, I2C, I2S, CAN, Ethernet, PWM). Veel üks väga kasulik omadus selle mikrokontrolleri juures on võimalus peaaegu kõiki GPIO viike peaaegu kõikide perifeeriaseadmetega suhtlemiseks kasutada. Nimelt on protsessori sisse ehitatud maatriks, mis võimaldab tarkvaraliselt kontrollida, millise viigu külge sisemised perifeeriaseadmed ühendatakse. Sisemised perifeeriaseadmed on näiteks UART, SPI ja I2C. Pikem nimekiri saadaval olevatest seadmetest ja nende kasutamisest on siin.

2. **Arduino IDE** on avatud lähtekoodiga arenduskeskkond, mis algselt loodi Arduino projekti raames. See sisaldab endas nii tekstiredaktorit koodi kirjutamiseks kui ka tarkvara, mis koodi mikrokontrollerile üles laeb ning jadapordi kuulamise võimekust, et töötava mikrokontrolleriga suhelda. Arduino IDE ei oma võimekust aktiivselt koodi kirjutamisele

kaasa aidata (autocomplete), koodi sammhaaval täita, jooksutamise ajal registrite sisu vaadata, koodi jooksutamist pausile panna ega riistvara simuleerida. Nende ja teiste puuduste tõttu on see tööriist kasutust leidnud peamiselt isetegijate hulgas. Meie kasutame selle kursuse raames Arduino IDE't kuna seal on olemas ja lihtsasti leitavad-kasutatavad teegid, milles on juba valmis kirjutatud registri-taseme loogika, mis ei ole selle aine teemas.

Arduino keel on modifitseeritud versioon C++ist, mis on natuke lihtsam algajatele, kuid on problemaatiline suuremate projektide juures, kuna ei võimalda projektide jagamist osadeks. Oluline käesoleva kursuse jaoks on, et Arduino keeles on saadaval suur kogus draivereid, mida me kasutame ja mille eest võib tänada selle ümber olevat suurt kommuuni.

3. **ESP-IDF** on ESP32 ametlik arendusraamistik. See sisaldab endas näiteks kompilaatorit, mis meie koodist masinkoodi teeb ning protsessi, kuidas ESP32 peale koodi laetakse. Meie ei kasuta seda tööriista otse, vaid läbi Arduino IDE mooduli, mis võimaldab seda ESP32 arenduseks kasutada. Muuhulgas on ESP-IDF'i repositooriumis olemas suur hulk näidiskoode, kuidas erinevaid perifeeriaseadmeid väga madalal tasemel liidestada jne. Need on aga kõik C keeles, mida me jällegi siin ei käsitle.

4. **USART (Universal Synchronous Asynchronous Receiver Transmitter)** ehk universaalne jadavärat on väga paindlik jadaliides. Jadaliideseks kutsutakse ühendust, kus andmeid saadetakse kommunikatsioonikanalis bitikaupa, osapooled vastavalt võtavad ühes otsas saadetava info bittideks lahti ning teises otsas panevad seda kokku tagasi. Seda protokolli saab kasutada sünkroonselt (ühine kellasignaal) ja asünkroonselt (mõlemad osapooled peavad olema konfigureeritud kasutama sama taktsagedusega kella). Sünkroonses režiimis töötav USART on ühilduv SPI protokolliga. Samuti võib USART töötada nii täisdupleks (saab samal ajal andmeid saata ja vastu võtta) kui ka poolduplekssrežiimis (ei ole võimalik saata ja vastu võtta samaaegselt). USART võib kasutada ka vookontrollisignaale DTR (*Data Terminal Ready*), DSR(*Data Terminal Set*), CTS (*Clear to Send*) ja RTS(*Ready to Send*). Viimased 2 ei olnud algselt mõledud voo kontrollimiseks, vaid pool-dupleks suhtluse konrolliks.

   Käesolevas praktikumis kasutame me asünkroonset režiimi ilma vookontrollita ja täisdupleks režiimis. Sellist konfiguratsiooni kutsutakse *UART*iks. Selline ühendus vajab ühist referentsi ja andmeviikude ühendamist. UARTi andmeid edastavad viigud kannavad nimetusi Tx(*transmit* saatmine) ja Rx(*recieve* vastuvõtt). Ühendamisel tuleks mõelda, kuidas need viigud loogiliselt peaksid töötama.

5. **ADC (Analog Digital Converter)** ehk analoog-digitaalmuundur on seade, mis seab sisendpingele vastavusse diskreetse arvu (kindlas vahemikus olev täisarv). ADC töötamise mee-

todeid on mitmeid, kuid kõik töötavad põhimõttel, et kindla aja tagant tehakse mõõtmine. Seda on oluline meeles pidada, kuna ADC-sid saab eristada kahe parameetri järgi: esiteks resolutsioon, nt. 12 bitti ehk 2^{12}=4096 erinevat võimalikku väärtust ESP32 sisemisel ADC-l; teiseks eristatakse ADCsid sageduse järgi, millega mõõtmisi teostatakse, nt. 1 kHz (vahel ka ksample/s ehk kilo-mõõtmist sekundis). ESP32 plaadil on saavutatud mõõtesagedus 21.3 kHz, kuid näiteks klassis olevate ostsilloskoopide sees leiduv ADC on võimeline töötama 1 GHz sagedusel, kuid saab teha ainult 8 biti laiuseid mõõtmisi. On oluline ära mainida, et mõõteala laius ei korreleeru tingimata mõõdetava pingevahemikuga, kuna pinget on võimalik skaleerida.

6. **Pingejagur** on elektriskeem, mis koosneb kahest takistist. See töötab Ohmi seaduse põhimõtetel **U=RI** ehk kui vool ahelas ei muutu, siis on pinge takisti otstel proportsionaalne takisti väärtusega. See skeem ei ole mõeldud kasutuseks toiteallikana, kuna kolmanda takisti lisamine skeemi muudaks pinget. Selle pärast kasutatakse seda skeemi pinge skaleerimiseks väiksemasse piirkonda, kuna mõõteseadmete sisetakistust loetakse lõpmatuks (klassis olevatel multimeetritel reaalselt 10 megaoomi).

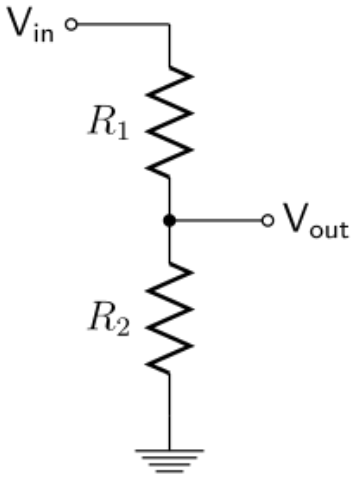


Figure 1. Pingejaguri skeem

3.4 Ülesanded

1. Ülesanne: ESP32 blink

Ülesande eesmärk on korda seada edaspidi kasutatavad tööriistad. Selleks on esiteks vaja lisada Arduino IDE plaatide hulka ESP32 perekond.

Pikem inglisekeelne juhend selle protsessi jaoks on olemas siin. Samast kohast on võimalik leida ka näiteid paljude muude ESP32 võimaluste kohta, mis tulevad teemaks järgmistes praktikumides. Samuti on näitekoodid nii selle kui ka teiste arendsuplaatide korral saadaval Arduino IDE vastavast menüüst.

Selleks, et Arduino IDE oskaks ESP32 protsessorit programmeerida, on vaja et IDE oleks sellega tuttav. Esiteks on vaja lisada **File → Preferences → Additional Boards Manager URLs** hulka

```
https://dl.espressif.com/dl/package_esp32_index.json
```

Seejärel saab Tools → Board: → Board Manager menüüst otsida ESP32 ja sellest installeerida kõige uuem versioon.

Nüüd saab Arduino IDEs valida Tools → Board: menüü alt ESP32 Arduno plaatide perekond, täpsemalt **ESP32 Dev Module**, mis on üldine ESP32 arendusplaat ning töötab kõikide ESP32 Wroom32 moodulitega.

Selleks, et ESP32 moodulile koodi peale laadida, on vaja see kõigepealt saada *bootloader*'isse, milleks on tavaliselt plaatidel **RESET** ja **EN** või **BOOT** nupud. Bootloaderisse jõudmiseks tuleks hoida all **BOOT** nuppu ja hetkeks vajutata **RESET**. Seda tuleks teha ainult siis, kui USB-UART osa plaadist ei oska bootloader'isse siseneda UART'i vookontrollisignaalide järgi. ja **Arvestus**: Käesoleva ülesande arvestuse jaoks on vaja demonstreerida juhendajale ESP32 arendusplaadile edukalt üles laetud programmi, mis vilgutab morse koodis "SOS". Morsekoodi ajastus peab vastama standardile. Ühe ühiku pikkuseks valida 100 ms.

TIP : Kui saad üleslaadimisel veateate, et Pythoni interpretaatorit ei leitud, siis on probleem keskkonnamuutujate nimes. Nimelt kasutatakse Pythoni nimetamiseks nii "python" kui "python3", ESP-IDF otsib nime ainult "python", mistõttu tekkib probleem, kui süsteem teab interpretaatorit nimega "python3". Üks lahendus on luua sümboolne link "python" "python3"'le. Seda saab teha käsuga

```
sudo ln -s /usr/bin/python3 /usr/bin/python
```

2. Ülesanne: RasPi ja ESP32 ühendamine omavahel UART-i kaudu

Ülesande eesmärk on tutvuda UART protokolliga ja kirjutada programmid selle kasutamiseks.

RasPis UARTi ehk jadavärati (*serial port*) kasutamiseks on vaja see enne *raspi-config*'is lubada. Selleks tuleb raspi-config menüüs minna *Interface Options → Serial Port*, *login shell* tuleks ära keelata ning *serial hardware port* lubada. Seejärel saab RasPi kasutada Rx ja Tx GPIO viike UART ühenduse jaoks. Seriali port on saadaval seadmena **/dev/serial0** või **/dev/ttyS0**.

Selleks, et seriali üle Pythoni kasutada, on vaja installeerida Pythoni teek, mis registritaseme loogika meie eest peidetult ära teeb, seda saab teha käsuga:

```
sudo apt install python3-serial
```

Enne ESP32 mooduli ja RasPi ühendamist tuleks mõelda, milliste viikude vahel tuleb ühendused teha ja kuidas.

ESP32 moodulil on kokku 3 UART perifeeriaseadet. UART0 on kasutuses mooduli programmeerimiseks ja ühtlasi Arduino IDEs **Serial monitorina**. Raspiga ühendamiseks soovitame kasutada Serial2, ehk kolmandat UARTi, kuna selle jaoks kasutatavad viikudel ei ole vaikimisi teisi otstarbeid ja need on mugavalt arendusplaadil välja toodud.

ESP32 seriali lugemiseks soovitame kasutada käsku .readStringUntil()

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. näidata juhendajale ESP32 programmi, mis võimaldab üle Arduino IDE Serial monitori suhelda RasPis jooksva Pythoni programmiga üle jadaühenduse. Programmid peavad sarnaselt eelmises praktikumis loodud socketitele võimaldama kahepoolset suhtlust. Mõlemad pooled peavad oma terminalis saadud sisendi saatma üle jadaühenduse teisele seadmele.
2. Pythoni programm ei tohiks blokeeruda input käsu peale (see on võimalik lahendada lõimedega);
3. selgitada juhendajale, mis juhtub, kui vestluse osapooled kasutavad erinevat taktsagedust.

3. Ülesanne: ADC

ESP32 ei võimalda erinevalt osadest teistest mikrokontrolleritest oma ADCle välist referentspinget anda. On võimalik sisemise referentspinge kalibreerimine, kuid ilma selle sammuta on ADC absoluutne täpsus piiratud. ESP võimaldab mõõtevahemikku muuta mõne sisseehitatud sammu vahel, mis võimaldab täpselt mõõta väiksemaid relatiivseid muutusi. Selle kohta saab täpsemalt lugeda ESP32 ADC dokumentatsioonist

Pythoniga Serial0 kasutamiseks peame teadma, et ESP32 plaat kasutab RTS ja DTR signaale protsessori taaskäivitamiseks ja koodi üleslaadimiseks alglaadurisse sisestamiseks. Sel põhjusel peame Serial0 liidesega pythoni kasutamisel muutma Seriali objekti, et väljastada õiged RTS ja DTR signaalid.

RTS ja DTR signaalide seadistamine

```
ser = serial.Serial("serial path", baudrate)
ser.rts = 0
ser.dtr = 0
```

Pythoniga graafikute loomiseks on olemas mitmeid teeke. Teegi võib vabalt valida. Soovitame kasutada teeki *pyqtgraph*, koos **pyqt5**'ga.

PyQtGraph jooksvalt muutuva graafiku näide

```
#!/usr/bin/python

import numpy as np
import pyqtgraph as pg
from pyqtgraph.Qt import QtGui, QtCore

data_plot = None
# create global variable to hold plot object
data_set = np.zeros(500)
# create numpy array with 500 elements

pts = np.linspace(0, 500, 500)
def on_update():
    global data_plot
    global data_set

    data_set[0] = np.int32(np.random.randint(0, 100))
    data_set = np.roll(data_set, -1)
    data_plot.setData(pts, data_set)         # data axises


def main():
    global data_plot
    win = pg.GraphicsLayoutWidget(show=True, title="Example graph")

    data_plot = win.addPlot()
    # greate plot object
    data_plot = data_plot.plot(pen='y')
    # add a data line to plot

    timer = QtCore.QTimer()
    # create a timer object
    timer.timeout.connect(on_update)
    # set timer to call function on_update
    timer.start(200)
```

```
    # start timer with 200ms interval between each function call
    QtGui.QApplication.instance().exec_()
    # start showing the graph


if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass
```

Arvestus: Käesoleva ülesande arvestuse saamiseks peab:

1. demonstreerima juhendajale ESP32 programmi, mis saadab üle jadaühenduse ADCga mõõdetud pinget potentsiomeetri pealt 0.1-sekundilise intervalliga;
2. demonstreerima juhendajale Pythoni programmi, mis kuulab vastavat jadaühendust ja lisab saadud andmed jooksvalt graafikule;
3. programmi töö lõppedes kirjutama kõik saadud andmed faili;
4. graafiku y-telje skaala olema voltides;
5. graafiku x-telje skaala olema sekundites;
6. selgitama juhendajale mida teeb np.roll() käsk;
7. selgitama juhendajale kuidas järjestikusel lähenemisel põhinev (**SAR - Successive-Approximation**) digitaal-analoogmuundur idee poolest töötab;
8. näitama juhendajale, kuidas pinge mõõtepiirkonda koodis muuta.

E ESP32 arendusplaadi DEV32 perifeeriaseadmete praktikum

4 ESP32 arendusplaadi DEV32 perifeeriaseadmete praktikum

4.1 Ülevaade

Siin praktikumis saate iseseisvalt tutvuda erinevate sisend-väljund seadmetega, mis praktikumi arendusplaadil DEV32 saadaval on.

Vajalikud vahendid

1. Internetiga arvuti
2. DEV32 arendusplaat
3. Mini-USB/USB-A kaabel arendusplaadi ühendamiseks arvutiga

1.3 Praktikumis esmakordselt kasutusele tulevate teemade tutvustused

1. **ESP32 arendusplaat DEV32** on spetsiaalselt käesoleva kursuse jaoks disainitud arendusplaat, mis võimaldab mugavalt ühendada kõiki praktikumides kasutusel olevaid mooduleid. Arendusplaadil on potentsiomeeter ja enkooder koos nupuga kui lihtsamad sisendseadmed ning indivduaalselt kontrollitavad RGB LED-id ja läbi nihkeregistrite ühendatud kaks 7-segmendist LED-i.
2. **Individuaalselt adresseeritav LED WS2812B** on RGB ehk mitmevärviline LED, millega saab kuvada 2^{8+8+8} erinevat tooni. Nimelt on võimalik sellele LEDile anda ette 8-bitised väärtused, määramaks punase, rohelise ja sinise tooni erksust. Neid LED-e ei kontrollita mitte vooluga, nagu varasemates praktikumides, vaid üle jadaühenduse, mis siinkohal on küll ainult pooldupleks ehk ühepoolne. Samuti on tegemist individuaalselt adresseeritavate LED-idega. See saavutatakse niimoodi, et iga LED võtab saabuva signaali algusest 24 bitti infot ära ning saadab ülejäänu edasi. Selliselt saab järjest ühendada teoreetiliselt lõpmata palju LED-e. Rohkem infot nende kohta on andmelehes.

3. **Triger (*flip-flop*)** on elektroonikalülitus, mis hoiab 1 bitti. Trigereid on mitut tüüpi. Kõige lihtsam on SR-triger, mis koosneb kahest transistorist, millel on 2 sisendit: S (*set*) ja R (*reset*), 2 väljundit: Q ja !Q ehk väljund ja tema vastand. Kui sisendile S (*set*) saata tõusev front, siis lülitatakse väljund Q kõrgeks ja kui saata sisendile R (*reset*) tõusev front, lülitatakse väljund madalaks. Vaata animatsiooni näiteks siit.

   Järgneva teema juures on vajalik aru saada natuke keerulisemast D-trigerist (*D-flip-flop*). Tema sisendid on D (*data*) ja CLK (*clock*) ning väljund Q. Iga kord, kui CLK-sisendil esineb tõusev front, salvestatakse D hetkeolek väljundisse.

4. **Nihkeregister (*shift register*)** on D-trigeritest koosnev ahel, kus iga trigeri väljund läheb järgmise sisendisse. Neid saab kasutada näiteks jadaühenduse dekodeerimiseks. Selliselt töötab näiteks eelmises praktikumis kasutusel olnud UART. UART-i korral on CLK-sisendise genereeritud seadistus määratud boodide järgi ning D on ühendatud Rx- ehk vastuvõtva viiguga. Vastavalt seadistuses määratud sõnumi pikkusele saab nihkeregistrist just samapalju bitte paralleelselt kätte. DEV32 arendusplaadil on 2 8-bitist nihkeregistrit ühendatud jadamisi, moodustades 16-bitise nihkeregistri. Selle väljunditesse on ühendatud kaks 7-segmendilist LEDi, mis on juba eelmistest praktikumidest tuttavad. Rohkem infot meie arendusplaadil olevate nihkeregistrite kohta nende andmelehes

5. ***Quadrature* enkooder** on astmeline dekooder, mis lülitab kahte väljundit keerates väljundeid maa ja toite vahel. Keeramise suuna määrab see, kumb väljunditest enne muutub. See on kasulik diskreetsete sisendite saamiseks kasutajalt, näiteks menüüs kerimine. DEV32-plaadil oleval enkooderil on ka sisseehitatud nupp ehk seda on võimalik vajutada.

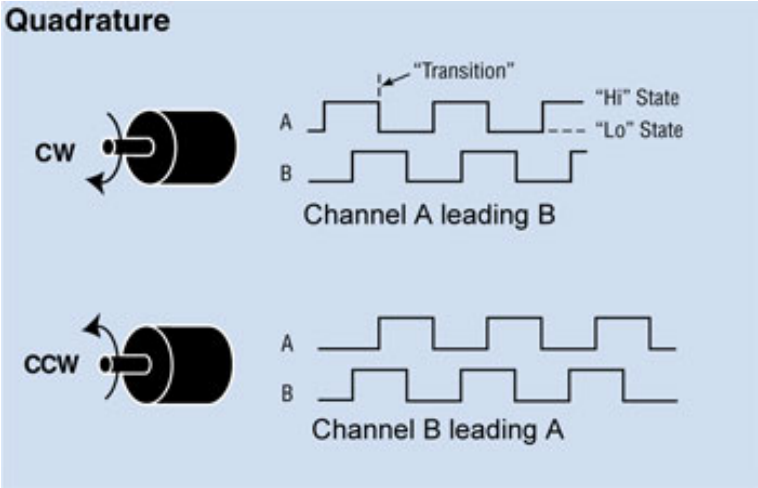


Figure 1. *Quadrature* enkooderi tööpõhimõte

4.2 Ülesanded

1. Ülesanne: Enkooderilt sisendi saamine

Siin ülesandes tuleks kasutada katkestusi ja-või ESP32 pulsilugemis perifeeriaseadet, et enkooderi lugemine ise implementeerida. Selle jaoks on olemas ja teegid, kuid siin ülesandes ei ole nende kasutamine lubatud.

Näide katkestuse teenindusrutiini defineerimisest

```
void IRAM_ATTR isr() {
    //code to be executed when interrupt has occurred
}
```

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. lugeda arendusplaadil olevat enkooderit ja hoida selle väärtus muutujas;
2. iga klikk päripäeva peaks muutuja väärtust ühe võrra suurendama ja iga klikk vastupäeva seda väärtust ühe võrra vähendama;
3. väärtus printida ainult iga muutumise järel jadaliidese monitorile;
4. enkooderi nupule vajutamisega nullida väärtus.

2. Ülesanne: 7-segmendiline LED läbi nihkeregistrite

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. kuvada arendusplaadil olevatel 7-segmendilistel LED-idel arve:
    a. Saada kasutajalt kuni kahekohaline arv (0 kuni 99) sisendiks, ehk vahemikus 0 kuni 9.9 peab kuvama ka esimest komakohta
    b. Kuva väärtus 7-segmendilistel LEDidel

3. Ülesanne: WS2812B individuaalselt adresseeritavate LEDide kontrollimine

Selle ülesande jaoks võime eeldada, et meie ADC töötab vahemikus 0V kuni 3.3V, kuigi me andmelehest teame, et reaalsuses on meie mõõtevahemik kuskil 0.15V kuni 2.45V ümbruses.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. teha programm, mis teisendab potentsiomeetri väljundpinge juhitavatel RGB LED-idel värviskaalasse:
    a. 0 V - esimene LED põleb siniselt;

b. pinge tõustes muutub värv sujuvalt punakamaks;

c. 3.3/4 = 0.825 V - esimene LED põleb punaselt;

d. kui esimene LED on maksimaalse punase väärtuse saavutanud, läheb järgmine LED täiesti siniselt põlema ja pinge kasvamisel jätkub sujuv värvi muutus nagu esimese puhul;

e. 3.3 V - kõik 4 LEDi põlevad punaselt.

F Bluetooth ja AT käsud praktikum

5 Bluetooth ja AT käsud praktikum

5.1 Ülevaade

Siin praktikumis hakkame kasutama erinevaid *Bluetooth* seadmeid. Tutvume ka AT-käskudega, mis on kasutusel modemite ja mitmete teiste juhtmevaba ühendust kasutavate moodulite konfigureerimiseks eelmisest praktikumist tuttava UART-ühendusega.

5.2 Vajalikud vahendid

1. Internetiga arvuti
2. Raspberry Pi
3. microSD kaart
4. DEV32 arendusplaat
5. HC-05v3 Bluetooth moodul
6. Androidiseade

5.3 Praktikumis esmakordselt kasutusele tulevate teemade tutvustused

1. ***Bluetooth*** on lühimaa **traadita side standard**, mis alates versioonist 4.0 jaguneb omakorda kaheks standardiks. Klassikaline Bluetooth(BR/EDR) on kasutusel juhtmevabaks *ad hoc* suhtluseks üksikseadmete vahel ning võimaldab teha ka juhtmevabu jadaühendusi. See on tagasiühilduv füüsiline protokoll, mis ei määra ära, kuidas üle loodud ühenduse andmeid saadetakse, toetab kiiruseid kuni 3 Mb/s. *Bluetoothi* suhtluses kutsuti ajalooliselt osapooli isandaks(master) ja orjaks (slave), kus isand alustab ühendust, seda terminoloogiat on Erinevad topoloogiad, mida klassikalise *Bluetoothiga* kasutada saab, on üksiku kliendiga suhtlemine, mitme kliendiga suhtlemine ja hajusvõrk (*scatternet*). *Bluetooth* ühenduste puhul kutsutakse osapooli *masteriks* ja *slaveks*, kus *master* loob ühenduse ja kontrollib seda.

Bluetooth BLE (varem *Wibree*) on loodud väikeste andmekoguste energiaefektiivseks vahendamiseks. Seda peegeldavad ka temaga loodavad topoloogiad. Lisaks kahe seadme vahelisele suhtlusele toetab BLE laialipaiskamist (*Broadcast*), kus seade saab kõigile kuulavatele seadmetele andmeid saata, ja võrku (*mesh*), kus paljud seadmed jagavad omavahel informatsiooni. BLE on klassikalisest *Bluetoothist* täiesti eraldiseisev ja kasutab oma kuhja (*stack*). BLE on leidnud väga palju kasutust IoT-seadmetes, eriti akutoitel töötavates tänu oma madalale energiatarbimisele. BLE-d kasutatakse tihti koos GATT-protokolliga, kus kogu suhtlus koosneb karakteristikutest (standardiseeritud karakteristikud koos unikaalsete ID-koodide ja otstarvetega), mis seda seadet iseloomustavad ja mida kilendid saavad vaadata või muuta.

Kogu *Bluetooth* suhtlus käib vahemikus 2.402 kuni 2.480 GHz ning on jagatud 40 (BR ja EDR) või 79 (BLE) kanaliks. *Bluetoothi* standardid on pidevas arenduses *Bluetooth SIG* organisatsiooni poolt. Viimane spetsifikatsioon *Bluetooth* 5.3 anti välja 2021 aasta keskel.

2. **AT-käsk** on modemi või mõne muu seadme konfigureerimiseks kasutatav käsk. Enamasti edastatakse AT-käske üle jadaühenduse. Need käsud on tavaliselt seadme-spetsiifilised. Käsu formaat on **AT+*käsk=uus väärtus*** seadistuse muutmiseks või **AT+*käsk*?** seadistuse küsimiseks. AT-käskude kohta on igal seadmel olemas andmeleht, näiteks siin on käesolevas praktikumis kasutatava *Bluetooth* mooduli oma.

5.4 Ülesanded

1. RasPi ja telefoni ühendamine üle *Bluetoothi*

Siin ja järgmises ülesandes on vaja Androidiseadet, millel on Bluetooth. Kui sul sellist endal ei ole, siis klassis on saadaval üks tahvelarvuti. Küsi seda juhendajalt. Selles ja järgmises ülesandes on *master* sinu telefon.

Esiteks on vaja Raspik muuta faili */etc/systemd/system/dbus-org.bluez.service*:

- leida rida algusega *ExecStart=* ja lisada selle lõppu *-C*
- sellele reale tuleb teha uus järgnev rida, millele kirjutada *ExecStartPost=/usr/bin/sdptool add SP*
- peale selle faili muutmist on vajalik teha RasPile restart.

Raspil nagu enamikel muudel tänapäevastel nutiseadmetel, on võimekus ühenduda teiste Bluetooth seadmetega. Selle jaoks on Linuxis terminalirakendus *bluetoothctl*, mida ka meie siin kasutame. See rakendus on vajalik bluetoothi seadmete leidmiseks, paaritamiseks ja usaldamiseks. Selle rakenduse käskude kohta saab infot kui peale rakenduse käivitamist sisestada

käsk *help*.

Peale seadme paaritamist ja usaldamist on võimalik RasPil avada Bluetoothi ühendus. Seda saab teha käsuga *rfcomm watch hci0*. See terminaliaken tuleks jätta suhtluse ajaks lahti. Teises terminaliaknas on nüüd võimalik alustada üle loodud ühenduse suhtlust. Selleks kasutada kolmanda praktikumi 2. ülesandes loodud jadaühenduse suhtlusprogrammi. Leia ise, mis on *Bluetoothi* ühenduse jadapordi nimi.

Telefonis soovitame kasutada ”Serial Bluetooth Terminal” äppi

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. paaritada Androidiseade oma RasPiga, kasutades *bluetoothctl* -i;
2. näidata juhendajale suhtlust Androiditelefonis oleva *Bluetooth* serialäpi ja oma RasPi programmi vahel.

2. ESP32 slave ja telefoni ühendamine üle klassikalise *Bluetoothi*

ESP32 on palju odavam ja kättesaadavam alternatiiv RasPi-le paljudes IoT ja targa kodu seadmetes, millel on sisseehitatud *Bluetooth* moodul.

TIP: Vaata Arduino IDE-s olevaid ESP32 näitekoode.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. paaritada Androidiseade oma ESP32-mooduliga. Ühendamisel tuleb kasutada autentimist (kontrollida, et telefonis ja ESP32-ss näidatakse sama koodi ja see kinnitada);
2. näidata juhendajale suhtlust Androiditelefonis oleva *Bluetooth* serialäpi ja ESP32-plaadi vahel.
3. telefoniäpist peab olema võimalik küsida ADC pinget, ESP32 peab sellele vastama hetkepingega voltides.

3. HC-05v3 *Bluetooth* mooduli ühendamine ESP32 Bluetooth slavega

Protsessorivälised juhtmevaba ühendust pakkuvad seadmed nt. GSM ja *Bluetooth* moodulid kasutavad põhiprotsessoriga suhtlemiseks tavaliselt jadaühendust. Moodulid on disainitud töötama mitmes režiimis, milleks on suhtlus ja seadistus. Kui suhtlusrežiimis on andmete töötlemine võrdlemisi lihtne, kuna mooduli töö on neid lihtsalt vahendada, siis seadistuse osa on keerulisem, sest UART võimaldab kirjutada ainult UART-i enda sisendregistrisse ehk otse üle jadaühenduse ei saa seadistusregistreid muuta. Seega on vaja kuidagi dekodeerida seadistamise režiimis kätte

saadud sõnumid, et nende põhjal seadistust määravaid registreid mooduli siseselt muuta. AT-käsud on täpselt selleks otstarbeks loodud. Nii on moodulite põhivaras defineeritud kindlad baidijadad, millega saab mooduli seadeid küsida või muuta.

Käesolevas ülesandes kasutame eelmises ülesande ESP32 *slave* programmi koos lisadega ja seadistame HC-05 *Bluetooth* mooduli olema *master* rollis ehk ühendust alustama. Siin ülesandes ei ole vajalik SSP osa.

Funktsioon, mis prindib välja ESP32 *bluetoothi* slave MAC aadressi

```
#include "esp_bt_device.h"
#include "esp_bt_main.h"
void printDeviceAddress() {
  bt_start();
  esp_bluedroid_init();
  esp_bluedroid_enable();
  const uint8_t* point = esp_bt_dev_get_address();

  for (int i = 0; i < 6; i++) {
    char str[3];
    sprintf(str, "%02X", (int)point[i]);
    Serial.print(str);

    if (i < 5){
      Serial.print(":");
    }
  }
}
```

Kogu info olemasolevate AT-käskude kohta leiab siit.

Moodulite konfigureerimine AT-käskudega toimub UART-i kiirusel 38400 boodi ning AT-režiimi läheb moodul siis, kui HC-05 mooduli VCC-viigu toite seamisel ajal on mooduli EN-viik tõmmatud kõrgeks. Kuna meie DEV32-plaatidel on HC-05 mooduli viik VCC ühendatud otse 5 V külge, siis saab seda teha mooduli koodis määratud ajal külge ühendamisega, et soovi korral AT-režiimi minna. Peale seadistuse muutmise lõpetamist saab AT-režiimist väljuda, kui tõmmata EN-viik tagasi madalaks ning kasutada AT-käsku, et moodulile taaskäivitus teha. Vihje: HC05 mooduli suhtlusrežiimi UARTi vaikekiirus ei ole 38400 boodi.

Funktsioon, millega HC05 moodulile AT-käske saata

```
void atcommand(const String _atcommand){
```

```
  Serial.print("== ");
  Serial.print(_atcommand);
  Serial2.print(_atcommand);
  delay(1000);
    while(Serial2.available())
    Serial.write(Serial2.read());
};
```

Mooduli hetkerežiimi saab eristada visuaalselt, vaadates mooduli LEDi:

1. aeglane vilkumine - AT-režiim;
2. kiire vilkumine - otsib ühendust;
3. perioodiliselt 2-3 kiiret vilgatust - ühendatud.

Kui mooduli LED näitab, et ollakse AT-režiimis, siis tuleks proovida, kas mooduliga õnnestub AT-režiimis andmeid vahetada. Seda saab näitekoodi kasutades teha käsuga

```
atcommand("AT\r\n");
```

Peale mooduli ühendamist ei pruugi sellele käsule koheselt vastust saada, seega tuleks seda saata vastavalt vajadusele 2-3 korda. Sellele käsule peaks moodul alati vastama ”OK”, juures olevad küsimärgid funktsionaalsust ei sega.

Hetkel peaks moodulit konfigureerides kindlasti tegema läbi järgnevad sammud:

Kontrollida ühendust *Bluetooth* mooduliga AT režiimis	AT
Mooduli tehaseseadete taastamine	
Kõigi paaritatud seadmete eemaldamine mooduli mälust	AT+RMAAD
Mooduli nime seadistamine	
Mooduli rolli muutmine *masteriks*	
Lubada ühendumine ainult kindla *slave* rollis oleva mooduliga (leia oma ESP32 mooduli BT MAC aadress)	AT+CMODE
Seada mooduli tüüp (Peaks olema vaikeväärtus kuid osadel moodulitel ei ole)	AT+CLASS=0
Paaritada moodul selle *slave* mooduli aadressiga	
Siduda oma moodul selle *slave* mooduli aadressiga	AT+BIND
Linkida oma mooduli selle *slave* mooduli aadressiga	AT+LINK
Tarkvaraliselt taaskäivitada moodul suhtlusrežiimi	
Saadaval olevate seadmete küsimine režiimis	AT+INQ

Tühjadele lahtritele vastavad käsud leiad andmelehest.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. kirjutada kood, mis konfigureerib DEV32-plaadile lisatud HC-05 mooduli *master* režiimi ning ühendab selle jadasuhtluseks eelmises ülesandes tehtud *slavega*;

2. moodulit ja teist DEV32-plaati kasutades luua võimekus kahe seadme vaheliseks suhtlemiseks analoogselt UART-i praktikumi teisele ülesandele (mõlemad pooled saavad saata ja vastu võtta samaaegselt);

3. kui kumbki osapool saab sõnumi, mida annab kuvada plaadil olevatel 7-segmendilistel LED-idel, siis saadud sõnum sinna ka kuvada (1 või 2 numbrit koos või ilma punktideta).

G I2C praktikum

6 I2C praktikum

6.1 Ülevaade

Käesolevas praktikumis hakkame kasutama erinevaid *I2C* seadmeid.

6.2 Vajalikud vahendid

1. DEV32 arendusplaat
2. LCD (16x2)
3. OLED (128x32 või 128x64)
4. Time of Flight sensor (edaspidi ToF) VL53l0x

6.3 Praktikumis esmakordselt kasutusele tulevate teemade tutvustused

1. **I2C (Inter-Integrated Circuit)** on sünkroonne pool-dupleks jadaliides, mis kasutab suhtlemiseks pakettkommutatsiooni (*packet switching*). Atmeli protsessorites kutsutakse seda liidest nimega *TWI* ja Arduino IDEs *Wire*. Seda kasutatakse väikest läbilaskevõimet (enamasti maksimaalselt 400 kbit/s) vajavate perifeeriaseadmetega suhtlemiseks üle lühikeste liinide. I2C kasutab kahte kahesuunalist avatud kollektori/paisuga liini: SDA andmete jaoks ja SCL taktsignaali jaoks, mis on tavaolekus *pullup* takistiga kõrgeks tõmmatud.

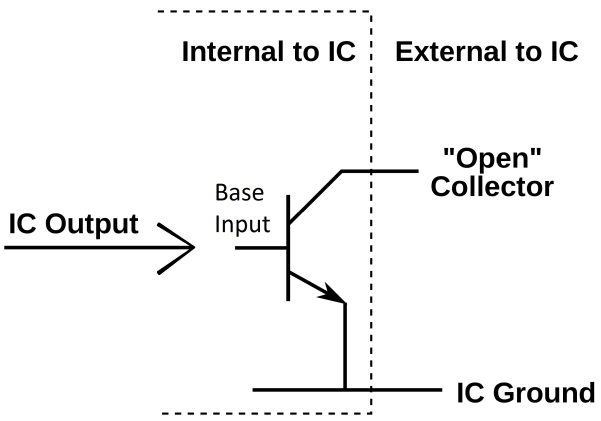


Figure 1. Avatud kollektoriga transistor iga I2C seadme sees

Sellise skeemiga saab iga I2C siini külge ühendatud seade kontrollida SDA- ja SCL-liine. Kuna I2C võimaldab ühendada mitut seadet sama andmesiini külge, siis on vaja kasutada seadmete adresseerimist. Ühe siini külge ühendatavate seadmete hulka piirab kasutada olev aadressiruum, mis on enamasti 7 bitti lai ja siini kogumahtuvus, mis peab jääma alla 400 pF. Suhtluse osapooled on kontroller ja sõlm, mida nimetatakse ka ülemaks (*master*) ja alluvaks (*slave*). Kontroller genereerib takti ja alustab sõlmedega suhtlust, saates START-signaali, mis tähendab, et SDA viiakse madalaks sel ajal, kui SCL on kõrge. Seejärel saadab kontroller selle sõlme aadressi, millega ta vestlust tahab alustada ja READ/WRITE-biti. Peale igat sõnumit ootab kontroller kliendilt vastust saadetud info kohta, ehk ACK (sõlm tõmbab SDA madalaks) või NACK (ei tõmba madalaks), aadressile oodatakse vastust ACK näitamaks, et selline seade on siinil olemas. Seejärel jätkub vestlus vastavalt sellele, kas kontroller soovis lugeda või kirjutada. Kui soov on kirjutada, siis võib seda teha seni, kuni kõik soovitud andmed saavad saadetud või sõlm saadab NACK-signaali. Samuti on lugemisega - sõlm jätkab järjest andmete kirjutamist, kuni kontroller vastab ACK-sigaaliga, esimese NACK'i peale saatmine lõpetatakse. Suhtlus ühe seadmega kestab kuni STOP-singaalini, milleks on tõusev front SDA-siinil sel ajal kui SCL on kõrge. Sellest parema ülevaate saamiseks koos näidetega on soovitav vaadata DEV32-plaadil oleva RTC andmelehe I2C peatükki alates lk. 16.

2. **DS3231M RTC (Real-Time Clock)** ehk **reaalajakell** on seade, mis mõõdab väga täpselt aega. DEV32-plaadil on DS3231 reaalajakell, mis on ühendatud I2C-siini külge. See suudab lugeda aega täpsusega kuni ±0.432 sekundit 24 h kohta ja hoida infot sekundite, minutite, tundide ja kuupäeva kohta, kaasa arvatud aastad ja liigaastad. Samuti sisaldab ta endas temperatuuriandurit (andmeleht).

3. **LCD (Liquid-Crystal Display)** ehk **vedelkristallekraan** koosneb läbipaistvate elektroodide vahel olevast vedelkristallist ja kahest polariseerivast filtrist, mis on omavahel risti. Kui vedelkristall on pingestamata olekus, pööratakse pärast esimest filtrit valguse polarisatsioonitasandit 90 kraadi, kui see läbib vedelkristalli, misjärel saab valgus ka teist

filtrit läbida. Kristalli pingestades muutub selle strkutuur pingega proportsionaalselt, hakates valguse polarisatsioonitasandit üha vähem pöörama, kuni polarisatsioonitasand enam kristalli läbides ei muutu. Siis neeldub neeldub kogu valgus teises filtris ja läbi ei tule enam midagi. LCD ei tooda ise valgust. Kasutatakse taustavalgust, milleks on enamasti LEDid.

Praktikumis olev LCD kasutab paralleelliidest, kuid on saadaval erinevaid mooduleid, mis muudavad LCD paralleelliidese kas I2C või SPI jadaliideseks. Meil on kasutusel I2C moodul, mis kasutab PCF8574 kiipi.

4. **Time of Flight (ToF)** ehk kaugussensor mõõdab kaugust, kasutades laserit ja mõõtes aega, mis kulub, et valgus mõõdetavalt pinnalt tagasi sensorisse jõuaks. ToF-sensorid on peamine komponent LIDAR-süsteemides, kus tehakse mõõtmisi paljudes ruumisuundades. Teades mõõtmiste järjekorda ja suunda, saab punktidest konstrueerida mudeli. Praktikumis kasutusel olev sensor suudab mõõta kuni 2 m pikkusi kaugusi, kasutades 940 nm lainepikkusega laserit.

6.4 Ülesanded

1. Ülesanne: I2C seadme aadressi leidmine

Esimene samm iga I2C seadmega suhtlemise juures on selle aadressi leidmine, milleks on mitu võimalust. Kui seade või moodul on ühendatud protsessori külge, on võimalik kasutada programmi, mis leiab kõik siini külge ühendatud I2C seadmed. Kui tegemist on laialdaselt kasutatava seadme või mooduliga, siis võib selle aadressi leida andmelehtdelt või mujalt internetivarusest. Siiski tuleks eelistada seadme andmelehte. Otsides näiteks internetist MPU6050 güroskoobi I2C aadressi, saadakse vastuseks 0x68. Samas leiab meie DEV32 arendusplaadil I2C skänner, et aadressil 0x68 on seade, isegi kui MPU6050 ei ole ühendatud. Kui güroskoop ühendada siis on tema aadressiks hoopis 0x69. Seda põhjustab plaadil olev RTC, mille I2C aadressi ei ole võimalik muuta, kuid MPU6050 moodulil on I2C aadress kontrollitav ühe viigu olekut muutes.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb:

1. näidata juhendajale programmi, mis *setup()* osas leiab kõik ühendatud I2C seadmete aadressid ja väljastab need;
2. programm peaks olema võimalikult lihtsalt teistesse programmidesse integreeritav.

2. Ülesanne: DS3231M registrilugemine

Erinevalt UART ühendusest ei kasuta I2C eraldi sisend-väljundregistreid. I2C muudab registreid ja loeb otse registritest, millele seade ligi pääseb, kus hoitakse seadistust või väärtusi.

See võimaldab I2C siini kasutavatel seadmelel vältida vajadust keeruka püsivara (*firmware*) järgi, et seadistust muuta näiteks AT-käske kasutades.

Paljude I2C siini kasutavate seadmete jaoks on olemas teegid, mis teevad kogu töö peidetult ära. Neid kasutad hilisemates ülesannetes, kuid käesolevas ülesandes tuleks andmelehes toodud näidete järgi alustada andmevahetust ja lugeda DS3231M RTC-lt temperatuuriregistrite sisu. Suhtluseks kasutada *Wire.h* teeki. Selle jaoks soovitame tugevalt lugeda Arduino *Wire.h* funktsioonide dokumentatsiooni. Siin ei tohi kasutada ühtegi teeki, mis I2C vestluse peidetult ära teeb.

Selleks vajalikud käsud on:

1. beginTransmission()
2. write()
3. requestFrom()
4. read()
5. endTransmission()

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb näidata juhendajale:

1. DS3231M andmelehest, mis on selle I2C aadress ja kust selle leiab;
2. DS3231M andmelehest, mis registrist saab lugeda välja kiibi temperatuuri;
3. programm, mis loeb registritest temperatuuri ja väljastab selle *Serial Monitorile* Celsiuse skaalal.

TIP: Info saamiseks kasutada internetti. Käesolevas ülesandes ei või kasutada teeki ennast, kuid selle lähtekoodi uurimisest võib palju kasu olla.

3. Ülesanne: LCD ekraan ja ToF sensor

Käesoleva ülesande korral lisada seadmed eelmise ülesande seadistusele, et kõik seadmed töötaks üheaegselt koos.

Käesolevas ülesandes on soovitatav LCD ja VL53l0x ToF sensoriga suhtlemiseks kasutada vabalt valitud teeke.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb näidata juhendajale:

1. programm, mis mõõdab ToF sensoriga perioodiliselt kaugust ja kuvab selle LCD ekraanil koos ühikuga;

2. ekraan ei tohi vilkuda;
3. mõõtmised peavad olema selliselt keskmistatud, et väljund oleks stabiilne;
4. suunata DS323M-ist loetavad temperatuuriväärtused *Serial Monitori* asemel LCD-le.

4. Ülesanne: OLED ekraan

Käesoleva ülesande korral lisada seadmed kahe eelmise ülesande seadistusele, et kõik seadmed töötaks üheaegselt koos.

Käesolevas ülesandes on soovitatav OLED ekraaniga suhtlemiseks kasutada vabalt valitud teeki.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb näidata juhendajale:

1. programm, mis loeb sisendi *Serial Monitorist* ja kuvab saadud sõnumi OLED ekraanil;
2. programm peab oskama kuvada mitmerealisi sõnumeid selliselt, et rea lõpus ei tohi olla üleliigseid sümboleid.

H WiFi praktikum

7 WiFi praktikum

7.1 Ülevaade

Selles praktikumis kasutame WiFit andmete edastamiseks ESP32 plaatide vahel, kasutades teisest praktiikumist tuttavaid socketeid. Lisaks kasutatame uusi andureid.

7.2 Vajalikud vahendid

1. DEV32 arendusplaat
2. LCD ekraan
3. Ultrasonic sensor
4. PIR sensor
5. DS18B20 temperatuurisensor
6. Potentsiomeeter

7.3 Praktikumis esmakordselt kasutusele tulevate teemade tutvustused

1. **WiFi** on juhtmevaba ühenduse standard, mis kasutab peamiselt kahte sagedusvahemikku - 2.401 GHz kuni 2.495GHz ja 5.160GHz kuni 7.125GHz, kuid *IEEE 802.11* spetsifikatsiooni järgi, on võimalik kasutada ka <1GHz sagedusi, sagedusvahemikku 3.657GHz kuni 3.692GHz ja 57.24GHz kuni 70.20GHz. Need vahemikud ja nende jaotus kanaliteks erineb piirkonniti, samuti erineb piirkonniti nendel sagedustel lubatud edstusvõimsus, kuid enamsti on lubatud kuni 25mW. WiFi on stabiilne ja väga laialdaselt kasutatav ühendus. Tuleb tähele panna, et ESP32 kasutab ainult 2.4GHz sagedust, kuna 5GHz vajab teistsugust antenni.
2. **Ultrahelisensor (*ultrasonic sensor*)** ehk sonar on kauguse mõõtmise sensor. Tööpõhimõtteks on väljasaadetud ultrahelilaine tagasipeegeldumisaja mõõtmise põhjal kauguse

arvutamine peegldava pinnani. Selle sensori puudus tuleb esile piiratud ruumis, kus võib tekkida kaja kahe seina vahel, mis mõõtmise ära rikub. Rohkem infot on siin

3. **PIR-sensor (*Pyroelectric "Passive" InfraRed*)** on liikumist tuvastav sensor. See on infrapunasensor, mis põhineb ümbritsevast keskkonnast soojemate kehade soojuskiirguse detekteerimisel oma vaateväljas, kui need liiguvad. Sensor ise on enamasti peidus poolläbipaistva soojuskiirgust läbilaskva läätse taga, mis fokuseerib suurema vaatevälja sensori siledale pinnale. Rohkem infot siin

4. **DS18B20** temperatuurisensor on digitaalne sensor, millega suhtlus käib üle **1-Wire** siini. DS18B20 mõõdab temperatuuri kuni 9 kuni 12 biti resulutsiooniga. See töötab toitevahemikus 3V kuni 5.5V ja **vajab 4.7kOomist pullup takistit oma andmesiinil, see takisti tuleks ühendada 3.3V toite külge.** Rohkem infot andmelehest.

7.4 Ülesanded

1. Ülesanne: WiFi ühendus ja kaugserveriga suhtlemine

Käesolevas ülesandes tuleb ühendada DEV32-plaadi WiFi-moodul olemasolevasse WiFi võrku, nt *robootika*. Peale WiFi-võrku ühendamist kasutada loodud internetiühendust, et küsida NTP-ajaserverilt jooksvat aega ja sellega initsialiseerida eelmises praktikumis kasutatud RTC. Siin on RTC-teegi kasutamine lubatud.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb näidata juhendajale programmi, mis:

1. ühendub käivitudes automaatselt WiFi võrku;
2. kui võrguühendus saavutatakse, siis küsib NTP ajaserverilt hetkeaja;
3. saadud ajaga initsialiseerib RTC;
4. töötamise käigus kasutab edasi RTC-st saadud aega, et olla sõltumatu internetiühendusest;
5. näitab kasutajale aega vabalt valitud visualiseerimisliidesel.

2. Ülesanne: Socket klient DEV32-l

Käesolevas ülesandes tuleb ühendada DEV32-plaadi WiFi moodul olemasolevasse WiFi võrku, nt *robootika*. Peale WiFi-võrku ühendumist peab moodul looma ühenduse socket-serveriga ning saatma sinna infot.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb näidata juhendajale:

1. programm, mis ühendub käivitades automaatselt WiFi võrku;

2. ühendub 2. praktikumis loodud socket-serveriga;
3. saadab 10 Hz sagedusega ultrahelisensoriga tehtud mõõtmised perioodiliselt serverisse;
4. server annab kasutajale märku, kas ultrahelisensor leidis takistuse või mitte.

3. Soft-AP ja mitme kliendiga socket-server DEV32-l

Käesolevas ülesandes tuleb kasutada DEV32-plaadi ESP32-mooduli võimekust olla internetiruuteriks. Näiteks, kui kasutada WiFi-baasil toimivat targa kodu lahendust, ei pruugi olla optimaalne, et kõik IoT-seadmed on ühendatud internetiga, kuna see võib kaasa tuua turvariske. Selle vältimiseks on võimalik ESP32-moodulit konfigureerida WiFi-ruuteriks. Käesolevas ülesandes kasutab ESP32 WiFi-moodul soft-AP(*Access Point*) režiimi, mis tähendab, et kogu võrgul puudub ligipääs laiemasse internetti. ESP32 jaoks on olemas ka NATi kasutav WiFi ulatuse pikendaja kood, mis annab kõigile alamvõrku ühendunud klientidele võimaluse ruuteriks oleva mooduli kaudu ülejäänud maailmaga suhelda, kuid see ei ole antud juhul kohustuslik (soovi korral muidugi võib).

Koos pääsupunktiga on siin ülesandes on vaja luua socket-server. ESP32 platformil ei ole võimalik kasutada lõimesid samamoodi nagu Python võimaldab, kuigi on olemas FREERTOS operatsioonisüsteem, millega saab teha ESP32-l Pythonile analoogseid lõimesid, kuid siin ülesandes ei ole vaja seda kasutada (soovi korral muidugi võib). Mitme kliendiga on võimalik suhelda mitte paralleelselt, vaid kordamööda, kasutades näiteks tsükleid. Sellisel juhul tuleks oma programmi kirjutades mõelda blokeeruvate funktsioonide ja maksimaalse klientide arvu peale.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb näidata juhendajale:

1. programm, mis loob oma WiFi võrgu;
2. loob socket-serveri, millesse saab ühenduse luua vähemalt 3 klienti;
3. kuvab viimaseid kahte saadud sõnumit LCD ekraanil nii, et uuem sõnum on ülemisel real;
4. klientideks tuleks võib kasutada 2. praktikumis loodud pythoni programmi või teisi DEV32-plaate;
5. üks klientidest peab edastama DS18B20 temperatuurisensori näitu;
6. seletada juhendajale, DS18B20 sensori kohta:
    a. kasutada saab sellega suhtlda üle ainult kahe juhtme mõlemat pidi
    b. millised piirangud andmevahetusele on
    c. kuidas saab sensor toite

d. millised nõuded ühendamisel on

e. mitu sensorit saab ühe siini külge ühendada, kui mitu siis kuidas neid omavahel eristatakse

4. Ülesanne: Tark lamp

Käesolevas ülesandes tuleks edasi arendada eelmistes ülesannetes loodud koode, et luua midagi targa lambi laadset, kasutades RGB LEDi, mida saab kontrollida üle WiFi.

Arvestus: Käesoleva ülesande arvestuse saamiseks tuleb näidata juhendajale:

1. serveri programm, mis loob oma WiFi võrgu ja kuvab lambi olekut, kasutades RBG LEDi;

2. 3 klienti, mis kõik võivad serveriga ühenduda;

    a. 1. kliendi programm, mis võimaldab potentsiomeetriga lambi eredust muuta ja enkooderiga lambi värvide vahel valida, enkooderi nupu vajutus lülitab lampi püsivalt sisse-välja;

        i. Seleta juhendajale, miks me ei saa selle ülesande jaoks kasutada arendusplaadil olevat potentsiomeetrit, ESP32 ADC andmelehe alusel.

    b. 2. kliendi programm, mis kasutab vabalt valitud kaugussensorit ja toimib puutevaba lülitina, kui mõõdetakse kaugust vahemikus 5-20 cm.

    c. 3. kliendi programm, mis kasutab PIR-sensorit ja paneb lambi põlema siis, kui parasjagu on liikumine tuvastatud. Liikumise signaali katkemisel põleb lamp veel 30s, kui just mõnest muust lülitist ei tule lambi väljalülitamise signaali.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Kaarel Rhede

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

   Aine ”Nutilahendute praktikum” värskendamine

   mille juhendaja on Heiki Kasemägi

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni..

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Kaarel Rhede
18.05.2022