TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Märt Sessman

Asjade internet WeMos näitel

Bakalaureusetöö (9 EAP)

Juhendajad: Anne Villems
Alo Peets
Taavi Duvin

Tartu 2017

Asjade internet WeMos näitel

Lühikokkuvõte:

Käesoleva bakalaureusetöö käigus valmib tark taimekasvatuse süsteem eesmärgiga uurida WeMos arendusplaadi võimekust ja sellise arendusplaadi kasutusvõimalusi. Bakalaureusetöö koosneb neljast peatükist. Esimeses peatükis räägitakse asjade internetist üldiselt. Teises peatükis tutvustatakse WeMos arendusplaadi omadusi. Kolmandas näidatakse, kuidas arendusplaati interneti kaudu juhtida. Neljandas tutvustatakse valmiva projekti komponente ning näidatakse nende kasutamist. Samuti näidatakse, kuidas loodud projekti automatiseerida.

Võtmesõnad:

Asjade internet, ESP8266, WeMos, robootika

CERCS: P170 (Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine)

Internet of Things based on the example of WeMos

Abstract:

The purpose of this bachelor thesis is to explore the capabilities of WeMos development board by creating a smart gardening system. The bachelor thesis consists of four chapters. The first chapter gives an overview of Internet of Things. The second chapter introduces WeMos development board and describes its specifications. The third chapter shows how to use WeMos as an Internet of Things device. The fourth chapter describes the components used in smart gardening system and shows how to use them. The fourth chapter also explains how to automate the system created.

Keywords:

Internet of Things, ESP8266, WeMos, robotics

CERCS: P170 (Computer science, numerical analysis, systems, control)

Sisukord

Sissejuhatus

Asjade internet on teema, millest räägitakse palju juba viimased paar aastat. Järjest rohkem puutuvad inimesed sellega kokku enda igapäevaelus: näiteks autod ja paljud kodumasinad on juba internetiühendusega ning püüavad ühel või teisel viisil inimesele kasulikud olla. Kortereid, maju, isegi terveid linnu muudetakse nutikateks värkvõrgu abil. Otsitakse pidevalt viise igapäevaelu efektiivsemaks ja mugavamaks muutmiseks.

Käesolev bakalaureusetöö on rakenduslik töö, mille käigus valmib keerukas targa taimekasvatuse süsteem kasutades WeMos nimelist arendusplaati. Töö eesmärk on valmiva projekti käigus uurida kui võimekas on üks odav, umbes 7€ maksev [1] asjade interneti arendusplaat, millised on sellise arendusplaadi rakendused ning kas see suudab ka reaalselt inimestele igapäevaselt kasulik olla.

Bakalaureusetöö koosneb neljas peatükist. Esimeses peatükis antakse lühiülevaade värkvõrgu ajaloost, tutvustatakse värkvõrgu kasutusvaldkondi ning kirjeldatakse värkvõrgu lahenduste isetegemise võimalusi. Teises peatükis tutvustatakse WeMos arendusplaati, kirjeldatakse tehnilisi omadusi ning juhatatakse arendusplaadi seadistamisega sisse bakalaureusetöö praktiline osa. Kolmandas peatükis näidatakse, kuidas saata käske WeMos arendusplaati interneti kaudu. Neljandas peatükis kirjeldatakse loodavat targa taimekasvatuse süsteemi ja selle komponente, samuti näidatakse, kuidas targa taimekasvatuse süsteemi programmeerida. Neljanda peatüki viimases punktis tehakse kokkuvõte tehtud töö tulemustest.

1. Asjade internet ehk värkvõrk

Järgnevas peatükis tehakse lugejale kokkuvõte asjade interneti ajaloost, kirjeldatakse värkvõrgu olemust, tulevikuperpektiivi ja tuuakse näiteid rakendusvaldkondadest.

1.1. Sissejuhatus asjade internetti

Asjade internet ehk värkvõrk (*Internet of Things*) on mõiste, millele on raske anda ühest definitsiooni, kuna tehnoloogia areng on kiire ja mõiste areneb koos tehnoloogiaga. Siiski on asjade interneti idee erinevates definitsioonides sama. Küberfüüsikalise süsteemitehnika terminibaasis on värkvõrk defineeritud järgmiselt: "Asjade internet (ka värkvõrk) on füüsiliste seadmete võrgustik, millesse on paigaldatud elektroonika, tarkvara, sensorid, täiturseadmed ning interneti ligipääs. Eelnimetatu võimaldab seadmetel andmeid koguda ja vahetada [2]."

Asjade interneti mõiste tekkimisest kirjutab näiteks Gérald Santucci artiklis "The Internet of Things: Between the Revolution of the Internet and the Metamorphosis of Objects" [3], millel põhineb ka käesolev lõik. Asjade interneti idee sai alguse ligi kakskümmend aastat tagasi Massachusettsi Tehnoloogiainstituudist, kus aastal 1999 loodi Auto-ID keskus. Algselt tegeles see keskus mitmete identifitseerimise tehnoloogiatega, kuid 2003 kandus põhirõhk RFID ehk raadiosagedustuvastuse edasiarendamisele. Auto-ID arendustöö eesmärk oli võimaldada arvutitel automaatselt tuvastada ja jälgida füüsiliste objektide asukohta, et ettevõtete logistikat odavamaks ja tõhusamaks muuta. 2003. aastal loodi Auto-ID uurimislaborid, mille eesmärk on edasi arendada võrgustikku, mis seoks füüsilised objektid interneti abil arvutitega. Sellest ajast peale on mõiste pidevalt arenenud ning RFID abil objektide tuvastamine ja jälgimine on vaid üks paljudest värkvõrgu rakendustest.

Asjade internet on hetkel üks kiiremini arenevaid tehnoloogiavaldkondi, sest vahendid muutuvad järjest odavamaks ja inimestele kättesaadavamaks. Ajakirja Business Insider värkvõrgu teemaline aruanne [4] ennustab aastaks 2021 asjade interneti seadmete koguarvu kasvu 6,6 miljardilt 22,5 miljardini. Juba praegu on turul palju nutikaid internetiühendusega seadmeid mida võib igaüks omale soetada, et igapäevaelu mugavamaks teha. Tihtipeale on need nišitooted aga kallid ning ei pruugi omavahel ühilduda. Samuti ei pruugi valmisseadmete

funktsionaalsus olla piisavalt paindlik. Lahenduseks on võimalus ise oma seadmed ehitada, andes võimaluse ka tehnoloogiavaldkonnas end harida.

1.2. Värkvõrgu rakendusvaldkonnad

Asjade internetti on inimese kasuks võimalik rakendada väga erinevatel viisidel. Käesolevas alapunktis tuuakse välja põhilised valdkonnad, kus tänapäeval asjade internetti rakendatakse.

1.2.1. Äritegevuses

Äritegevuses on asjade interneti kasutamine hüppeliselt kasvanud. Aastal 2016 investeerisid ettevõtted asjade internetti ligi 18% rohkem kui aasta varem ja sarnast investeeringute mahu kasvu oodatakse ka käesolevaks aastaks [5]. Järjest enam leitakse viise, kuidas värkvõrgu abil hoida ettevõtluses raha kokku ja seeläbi suurendada ettevõtete tulu. Ettevõtted kasutavad juba praegu äritegevuses asjade internetti järgmiselt:

- Toodete kasutamise jälgimine tootearenduse tõhustamiseks
- Tootmisliini automaatika jälgimine nõrkade kohtade avastamiseks
- Logistika tõhustamine saadetiste jälgimise ja sobiva marsruudi valimise abil
- Energiatarbe jälgimine ja seadmete automaatne väljalülitamine ressursi säästmiseks
- Põllumajanduses saagi kasvu jälgimine ja tõhusamaks muutmine
- Karjakasvatuses loomade tervise jälgimine haiguspuhangute ennetamiseks

Eelnev loetelu on vaid osa asjade interneti rakendustest, mis äritegevusele kaasa aitavad.

1.2.2. Linnapildis

Asjade internet leiab kasutust ka linnades. Näiteks nutikas tänavavalgustus, ühistransport, targad majad. Värkvõrgu eesmärk linnas on muuta inimeste elu mugavamaks, turvalisemaks ja säästa linnaressursse.

Näiteid värkvõrgu rakendustest linnades ei pea kaugelt otsima. Eestis tegutseb targa linna klaster Smart City Lab, mis keskendub linnaelu eri valdkondi hõlmavate uudsete lahenduste loomisele, arendamisele ja ka ekspordile [6]. Smart City Lab on loonud näiteks ühistranspordi

mugavdamiseks ühiskaardi süsteemi, samuti ühise platvormi parkimise korraldamiseks ning mitme uudse lahenduse arendustöö alles käib. 2016. Aasta veebruaris käivitati rahvusvaheline koostööprojekt SmartEnCity, milles osaleb Eesti linnadest Tartu [7]. Selle projekti eesmärk on viia ellu targa ja säästva linnakeskkonna terviklahendus. Tartus viiakse projekt ellu kesklinna piirkonnas ja projekti raames on planeeritud täita näiteks järgnevad eesmärgid [8]:

- Umbes 23 korterelamu renoveerimine ja kaasajastamine
- Nutikodu platvorm
- Tark ja energiasäästlik tänavavalgustus liikumis- ja müraandureid kasutades

Projekti kestvus on planeeritud kauemaks kui viis aastat, aga põhitegevused peaksid jääma esimesele 2-3 aastale.

1.2.3. Tervishoius

Asjade internetis nähakse potentsiaali muuta inimeste tervise jälgimist tõhusamaks. Asjade interneti abil saaks arst patsiendiga kohtumata jälgida patsiendi tervisenäitajaid nagu näiteks vererõhk, kehatemperatuur, veresuhkru tase. Targad seadmed muudavad andmete sisestamise patsiendiportaali lihtsaks ja kiireks ning selliselt hoitakse kokku aega ja raha. Massidesse on läinud ka kantavad seadmed nagu nutivõrud ja nutikellad. Nendega saavad inimesed iseenda tervise kohta rohkem teada ja jälgida näiteks oma unerütmi, vaadata pulssi, lugeda samme ja kulutatud kaloreid.

1.2.4. Inimeste kodudes

Enim asjade interneti seadmeid on kasutusel just inimeste kodudes, muutes igapäevaelu mugavamaks. Värkvõrgu lahenduste alla kodus kuuluvad näiteks:

- valvesüsteemid,
- nutikas valgustus,
- targad koduseadmed,
- energiasäästlik ja tark elektrivõrk,
- automatiseeritud taimede kasvatamine,
- maja temperatuuri ja ventilatsiooni juhtimine,
- targad uksed, väravad

Tulevikule mõeldes oleks mõistlik iga eramaja projekteerida targa maja lahendustega. Kuigi seadmeid, mis aitavad kodu targaks teha, on palju, on nende hinnaklass siiski päris kõrge ja paljudele kättesaamatu. Hiina odavamatel lahendustel samas on sageli probleeme kvaliteediga või teiste seadmetega ühilduvusega.

1.3. Asjade internet isetegijatele

Asjade internetiga tutvumiseks ei pea alati ostma kalleid seadmeid, võimalus on ka seadmed ise luua. Kui praegu võib igaüks minna tehnikapoodi ja osta omale sobiv arendusplaat, millega asjade interneti lahendusi looma hakata, siis veidi enam kui kümme aastat tagasi sellist võimalust ei olnud.

1.3.1. Tehnoloogia muutub kättesaadavamaks

Aastal 2005 pandi alus isetegemise laiemale levikule, kui loodi Arduino, avatud riistvara ja tarkvaraplatvorm, mis tegi riistvara programmeerimise palju lihtsamaks, odavamaks ja kättesaadavamaks kõigile [9]. Arduino arendusplaatide külge on võimalik ühendada andureid, mootoreid, tulesid, ekraane, nuppe ja palju muud. Arduino programmeerimisega on lihtne algust teha, sest selle kohta on internetis palju materjali ja näiteid. Ka Arduino enda tarkvaraarendusplatvormiga *Arduino IDE* tuleb kaasa palju näiteprogramme. Arduino ametlikud arendusplaadid on ise üsna kallid, aga avatud platvorm on võimaldanud luua ka odavamaid kloone ja derivaate.

1.3.2. Võimalusi oma seadmete internetiga ühendamiseks

Arendusplaadi internetiga ühendamiseks on variante mitu:

- WiFi,
- mobiilside,
- sinihammas,
- kaabliga internet

Nendest kaabliga ühendus on kõige kindlam, samas puudub mobiilsus ja seadme ümberpaigutamine on juhtme tõttu raskendatud. Sinihamba tehnoloogia on energiasäästlik, kuid nõuab vastuvõtjat, mis ei tohi olla saatjast eriti kaugel [10]. Mobiilside nõuab SIM kaarti ja kulutab andmesidet, mis on lisakulu kasutajale. Seda eriti siis, kui seadmeid on rohkem. Samas on mobiilside sobilik lahendustele, mis peavad olema mobiilsed või asuvad kusagil hoonestusest eemal ja kus muud varianti ei ole. WiFi ühendus on kodus ilmselt parim, kui juba on WiFi võrk olemas ja ei kavatseta seadmeid koduse WiFi levialast välja viia. WiFi ühendusega ei kaasne lisatasusid ja säilib mobiilsus WiFi leviala piires, kodustes tingimustes asjade interneti loomisel rohkemat üldjuhul tarvis ei ole. Käesolev töö keskendubki WiFi toega arendusplaadiga asjade interneti lahenduse loomisele.

1.3.3. Asjade interneti loomine muutub odavamaks

Aastal 2014 hakkas Hiina ettevõte Espressif tootma mikrokontrollerit ESP8266, millel on standardile vastav WiFi tugi. Esimest korda tutvustati seda mikrokontrollerit inglisekeelsele publikule 2014. aasta augustis Hackaday portaalis [11]. Kui ESP8266 turule tuli, oli see mitu korda odavam hinna poolest järgmisest analoogsest turul olevast WiFi kontrollerist, makstes vaid ligi viis dollarit [11, 12]. Esialgu ei teatud päris hästi, mida ESP8266 võimaldab teha, sest dokumentatsioon oli hiina keeles. Huviliste kiire töö tulemusel oli juba sama aasta oktoobriks valmis jõutud ESP8266 arendustarkvaraga, mis tegi senisest WiFi kontrollerist täielikult programmeeritava mikrokontrolleri [13]. Kui seni kasutati ESP8266 mikrokontrollerit teistele arendusplaatidele, nagu näiteks Arduino Uno, WiFi toe lisamiseks, siis arendustarkvara muutis ESP8266 teistest arendusplaatidest täielikult sõltumatuks.

Ei läinud kaua, kui Hiina turule hakkasid tekkima odavad ESP8266 mikrokontrolleril põhinevad arendusplaadid, mis tegid mikrokontrolleri kasutamise veelgi lihtsamaks. Enam ei pidanud kasutaja mõtlema sellele, kuidas mikrokontrollerit arvutiga ühendada või kuidas kõige paremini ligi pääseda mikrokontrolleri sisenditele ja väljunditele. Kõigest viie dollari eest oli võimalik hakata looma asjade interneti lahendusi. Suur huvi ESP8266 vastu on aidanud kaasa arendustarkvara arengule ja praegu on võimalik mikrokontrollerit programmeerida mitmes programmeerimiskeeles mitme erineva arendustarkvaraga. Inimene, kes soovib ise asjade internetiga tutvust teha ja oma kätega midagi teha, võikski alustada ESP8266 mikrokontrolleriga arendusplaadist.

1.3.4. Asjade interneti arendusplaatide valik

Huvilisel on võimalik valida kümnete erinevate ESP8266 mikrokontrolleril põhinevate arendusplaatide vahel. Kõige populaarsemad ESP8266 arendusplaadid on järgnevad [14]:

- NodeMCU,
- Adafruit Feather HUZZAH,
- Knewron SmartWIFI,
- SparkFun Thing,
- WeMos D1 Mini,
- Wio Link.

Loetelus olevad arendusplaadid ei ole kahjuks kõik Eestis kättesaadavad või on välismaalt tellimise puhul saatmiskulud ebamõistlikult kõrged. Näiteks Adafruit Feather HUZZAH arendusplaati on võimalik tellida Adafruit veebipoest 17 dollari eest, kuid saatmiskulud Eestisse on ligi 50 dollarit [15]. Samuti on raske leida Knewron SmartWIFI arendusplaati.
Eestist on võimalik osta järgnevaid arendusplaate:

- NodeMCU, hind umbes 17 eurot [16],
- SparkFun Thing, hind umbes 16 eurot [17],
- Wio Link, hind umbes 24 eurot [18],
- WeMos D1 Mini, hind umbes 15 eurot [19]

Kellel on aega oodata ja soovib veelgi enam raha kokku hoida, siis tasub vaadata Hiina e-poodide ja seal müüdavate veelgi odavamate kloonide ja derivaatide poole. Hiina e-poodidest on võimalik tellida nendest arendusplaatidest kaht: NodeMCU ja WeMos D1 Mini. Näiteks Aliexpress e-poes maksavad nii WeMos D1 Mini kui ka NodeMCU arendusplaadid kõigest viis eurot [20, 21], mis on umbes kolm korda madalam hind kui Eestist ostes. Tuleb arvestada kohaletoimetamise ajaga, mis algab kahest nädalast ja võib ulatuda isegi kahe kuuni. Autori senise kogemuse põhjal on keskmine aeg tellimuse esitamisest kättesaamiseni umbes kolm nädalat.

NodeMCU ja WeMos D1 Mini erinevused on peamiselt sisend-väljundviikude arvus ja arendusplaadi enda füüsilises suuruses. NodeMCU on mõõtmetelt suurem ja suurema viikude arvuga. Erinevalt NodeMCU arendusplaadist, mida toodetakse vaid üht mudelit, ei ole WeMos D1 Mini ettevõtte WeMos ainus ESP8266 mikrokontrolleriga arendusplaat. WeMos kasuks

räägib ka laiendusplaatide olemasolu WeMos D1 Mini arendusplaadile [22]. Laiendusplaadid teevad kindlate riistvaralahenduste loomise oluliselt kiiremaks ja lihtsamaks, võimaldades laiendusplaadi ühendamisel arendusplaadiga lisada funktsionaalsust ja tehes riistvara modulaarseks. Kasutajal pole vaja hakata ühendama juhtmeid ja looma elektriskeeme, piisab laiendusplaadi ühendamisest arendusplaadiga ja vaja on vaid programmeerida.

Järgnevas punktis tutvustatakse WeMos arendusplaate lähemalt ning kirjeldatakse WeMos arendusplaatide omadusi.

2. WeMos arendusplaatide tutvustus

Järgnev peatükk annab ülevaate WeMos Electronics poolt toodetud arendusplaatidest. Peatükis kirjeldatakse arendusplaatide omadusi ja nende võimekust, tutvustatakse erinevaid WeMos mudeleid ja nendele loodud laiendusplaate ning näidatakse, kuidas alustada WeMos arendusplaadi programmeerimist.

2.1. WeMos arendusplaatide lühitutvustus

WeMos Electronics on Hiina ettevõte, mis toodab erinevaid arendusplaate ja nendele laiendusplaate. WeMos arendusplaatide valikusse kuuluvad D1, D1 Mini, D1 Mini Pro ja WeMos XI [23]. Lisaks on valikus laiendusplaadid arendusplaatidele WeMos D1 Mini ja D1 Mini Pro funktsionaalsuse lisamiseks [22].

WeMos D1, D1 Mini ja D1 Mini Pro on omadustelt üsna sarnased. Mainitud kolm arendusplaati on sisseehitatud WiFi tõttu sobilikud asjade interneti lahenduste loomiseks. WeMos arendusplaatide mikrokontrolleriks on Espressif ESP8266EX [24-26], mille tähtsamateks omadusteks asjade interneti kontekstis on sügava une režiim energia säästmiseks ja suur temperatuuritaluvus (-40°C kuni +125°C) [27]. WeMos arendusplaatide tehnilisi andmeid ja omadusi kirjeldatakse täpsemalt järgnevas punktis.

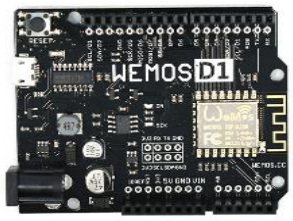
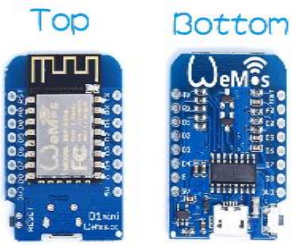
2.2. WeMos omadused ja tehnilised andmed

Tehniliste andmete poolest on WeMos D1, D1 Mini ja D1 Mini Pro väga sarnased. WeMos arendusplaatide ühised omadused on järgnevad [24-26]:

- Taktsagedus: 80MHZ/160MHZ
- Tööpinge: 3,3V
- 11 digitaalset sisend-väljundviiku
- Üks analoogsisend

Arendusplaatide erinevused ja pildid on välja toodud tabelis 1.

Tabel 1. WeMos arendusplaatide erinevused [24-26].

Arendusplaat	Tehnilised andmed ja omadused	Joonis
WeMos D1 R2	Arduino UNOga ühilduv Välkmälu: 4MB Plaadi pikkus: 68,6 mm Plaadi laius: 53,4 mm Kaal: 25 g Välistoite pesa 9-24 volti USB draiver: CH340G Hind: 6 eurot [1]	 Joonis 1. D1 R2 [26]
WeMos D1 Mini	Välkmälu: 4MB Plaadi pikkus: 34,2 mm Plaadi laius: 25,6 mm Kaal: 10 g USB draiver: CH340G Hind: 3,7 eurot [20]	 Joonis 2. D1 Mini [24]
WeMos D1 Mini Pro	Välkmälu: 16MB Plaadi pikkus: 34,2 mm Plaadi laius: 25,6 mm Kaal: 10 g Välise WIFI antenni ühenduspesa Sisseehitatud keraamiline antenn USB draiver: CP2104 Hind: 4,7 eurot [28]	 Joonis 3. D1 Mini Pro [25]

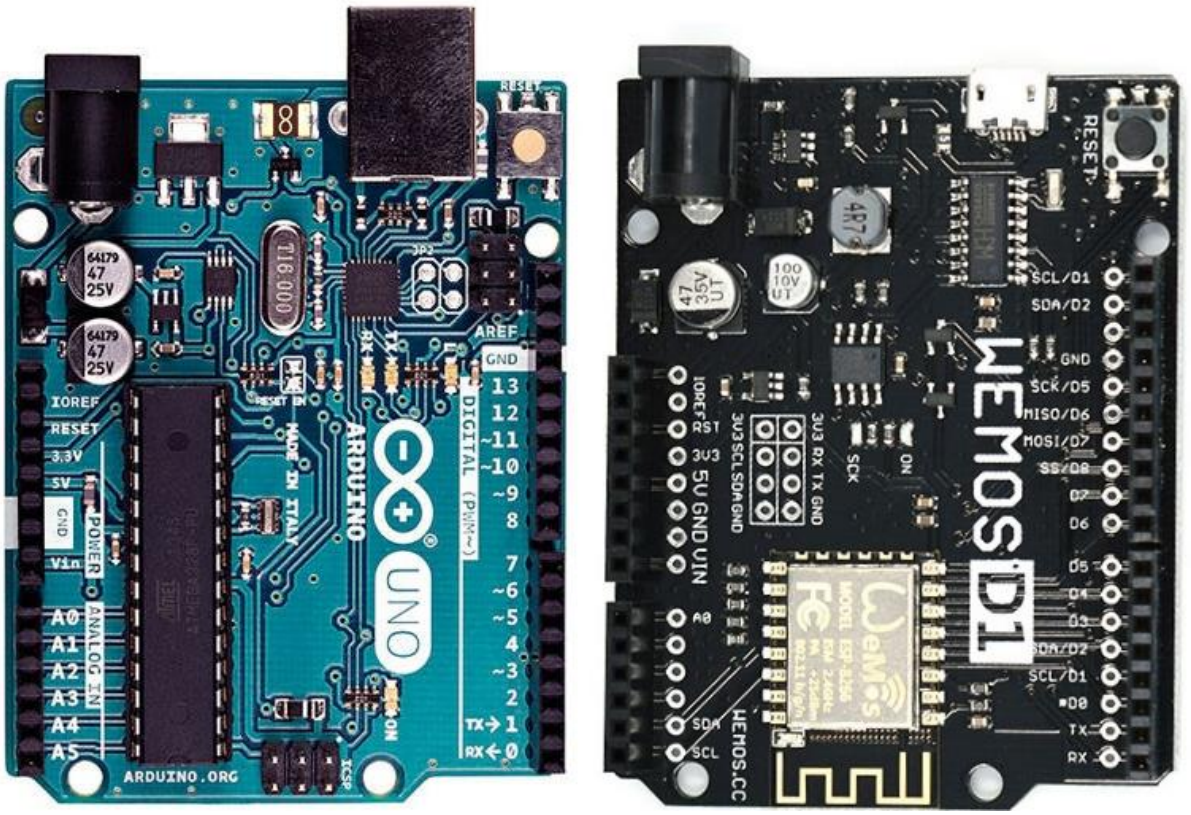
Tuleb tähele panna, et kuigi WeMos D1 on Arduino UNO-ga ühilduv, on neil siiski mõned olulised erinevused, mis ka järgnevas välja on toodud.

Esiteks on Arduino UNO analoogsisendviike kuus, WeMos arendusplaadil aga kõigest üks. Kui tekib vajadus ühendada ühe arendusplaadi külge rohkem kui üks analoogandur, on võimalik

kasutada multiplekserit, mis võimaldab mitme analooganduri signaali suunata ühte sisendisse [29].

Teiseks on WeMos arendusplaadi analoogsisendviigul maksimaalne sisendpinge 3,3 volti. Kui seda teadmist ei arvesta ja 5V analoogandurit kasutada, võib kergesti arendusplaadi ülepingega ära lõhkuda. Sellise anduri kasutamiseks tuleb kasutada pingemuundurit, mis muudab anduri pingevahemiku WeMos arendusplaadile sobivaks [30].

Kolmandaks on erinev ka digitaalsete sisend-väljundviikude paigutus ja arv, millega tuleb arvestada, kui tekib soov Arduino UNO tarbeks kirjutatud programme WeMos arendusplaadile paigaldada. Sellisel juhul tuleb programmis viikude numbrid viia vastavusse WeMos arendusplaadi viikude numbritega. Viikude numbrite vastavused leiab tabelist 2 ning Arduino UNO ja WeMos D1 viikude paigutust plaatidel võib näha joonisel 4.



Joonis 4. Arduino UNO ja WeMos D1.

Tabel 2. Arduino UNO R3 ja WeMos D1 R2 sisendväljundviikude vastavused [26, 31, 32].

Arduino UNO R3				**WeMos D1 R2**		
Nimetus	Tähis plaadil	Funktsionaalsus		Nimetus	Tähis plaadil	Funktsionaalsus
SCL		I2C: SCL	→	GPIO5	SCL/D1	I2C: SCL
SDA		I2C: SDA	→	GPIO4	SDA/D2	I2C: SDA
GPIO13	13	SPI: SCK	→	GPIO14	SCK/D5	SCK
GPIO12	12	SPI: MISO	→	GPIO12	MISO/D6	MISO
GPIO11	~11	SPI: MOSI	→	GPIO13	MOSI/D7	MOSI
GPIO10	~10	SPI: SS	→	GPIO15	SS/D8	SS
GPIO9	~9		→	GPIO13	D7	
GPIO8	8		→	GPIO12	D6	
GPIO7	7		→	GPIO14	D5	WeMos LED
GPIO6	~6		→	GPIO2	D4	ESP8266 LED
GPIO5	~5		→	GPIO0	D3	
GPIO4	4		→	GPIO4	SDA/D2	I2C: SDA
GPIO3	~3		→	GPIO5	SDA/D1	I2C: SCL
GPIO2	2		→	GPIO16	D0	
GPIO1	TX→1	TX	→	GPIO1	TX	TX0
GPIO0	RX←0	RX	→	GPIO3	RX	RX0

WeMos Mini ja Mini Pro jaoks on olemas järgnevad ametlikud laiendusplaadid, mille abil saab arendusplaadile lihtsalt funktsionaalsust lisada [22]:

- RGB LED laiendusplaat,

- MicroSD mälukaardi laiendusplaat,
- Releega laiendusplaat,
- Laiendusplaat prototüüpimiseks,
- Laiendusplaat OLED ekraaniga suurusega 64x64 pikslit,
- Mootori kontrollimiseks mõeldud laiendusplaat,
- Temperatuuri ja õhuniiskuse anduriga laiendusplaadid,
- Liitiumaku laiendusplaat.

WeMos D1 R2 jaoks ametlikke laiendusplaate tehtud pole, küll aga võivad sobida mõned Arduino UNO laiendusplaadid. Sellisel juhul tuleb kindlasti jälgida, mis pingevahemikus laiendusplaat töötab ja milliseid sisend-väljund viike kasutab ning vastavalt tabelile 2 tuleb ka laiendusplaadi püsivaras viikude numbrite muudatused sisse viia. Järgmises alapunktis antakse ülevaade WeMos arendusplaatide rakendamise võimalustest reaalses elus.

2.3. Arendusplaadi kasutusvaldkonnad

Mikrokontrolleri madal taktsagedus ja välkmälu suurus seavad omad piirangud arendusplaatidel realiseeritavate tarkvaraliste lahenduste keerukusele. Näiteks ei ole WeMos sobilik keerukamate tarkvaraprojektide jaoks nagu meediatöötlus või tehisintellekti rakendamine. Väga edukalt saab WeMos hakkama lihtsamate riistvaraliste ülesannetega nagu andurite informatsiooni edastamine internetti või internetist saadud info peale reageerimine ja näiteks valgustuse juhtimine või vahelduvvooluseadmete kontrollimine.

Kasutades WeMos arendusplaati on võimalik luua väga palju erinevaid asjade interneti seadmeid. Üle maailma on loodud juba praegu nii WeMos kui mitmeid teisi ESP8266 mikrokontrolleril põhinevaid arendusplaate kasutades tohutult erinevaid projekte. Näiteks veebiportaalis Hackaday.IO leiab näiteks selliseid põnevaid projekte nagu:

- ligipääsu kontroll kasutades NFC-d ehk lähiväljasidet [33],
- suitsuanduri ühendamine internetiga kasutades MQTT protokolli [34],
- tark veekeetja [35]

Milline WeMos arendusplaadi mudel endale valida, sõltub loodava tehnilise lahenduse iseloomust. Kui on oluline, et riistvara oleks kompaktne ja võtaks vähe ruumi, tuleks pigem

vaadata mõõtmetelt väiksemate D1 Mini ja D1 Mini Pro poole. Näiteks koduseadmeid juhtivad lahendused peaks olema võimalikult kompaktsed ning võimalusel koduseadme enda sisse peidetud, et arendusplaat kodu visuaalset üldpilti ei rikuks. Mini Pro on küll kallim, kuid välise WiFi antenni lisamise võimalus võib osutuda määravaks, kui arendusplaat asub ruuterist piisavalt kaugel, et ilma antennita enam stabiilset internetiühendust saavutada pole võimalik.

Käesolevas töös on valitud WeMos arendusplaadi võimalusi demonstreerivaks projektiks nutikas taimekasvatuse süsteem. Kuna taimekasvatuse süsteem on autori hinnangul keerukas ja koosneb mitmetest seadmetest ning digitaal- ja analooganduritest, kerkivad esile punktis 2.2 välja toodud WeMos arendusplaadi puudused võrreldes Arduino UNO arendusplaadiga ning tekib võimalus neile mõistlik lahendus leida. Projektil on hariduslik eesmärk, tutvustades lugejale asjade interneti maailma ning riistvara- ja tarkvaralahenduste loomist. Mitmed targa taimekasvatuse projektis kasutatud lahendused on universaalsed ning kasutatavad ka teistes värkvõrgu projektides. Lisaks teeb loodav projekt autori enda kodus kasvuhoonetaimede ettekasvatamise mugavamaks ning efektiivsemaks. Loodavas projektis on arendusplaadiks valitud WeMos D1 Mini selle kompaktsuse tõttu.

Järgmises punktis näidatakse, kuidas WeMos arendusplaati arvutiga ühendada ja seda Arduino tarkvaraarenduskeskkonda kasutades programmeerida.

2.4. Arendusplaadi seadistamine

WeMos arendusplaate on võimalik programmeerida mitmel viisil [26, 36]:

- kasutades Arduino tarkvaraarenduskeskkonda, edaspidi Arduino IDE (*Integrated Development Environment* ehk integreeritud arenduskeskkond),
- NodeMcu tarkvaraarenduskomplekti abil,
- MicroPython püsivara abil programmeerimiskeeles Python.

Arduino on avatud lähtekoodiga arendusplatvorm, mis põhineb lihtsasti kasutataval riistvaral ja tarkvaral [37]. NodeMcu on avatud lähtekoodiga püsivara ja tarkvaraarenduskomplekt, mis võimaldab seda toetavatele arendusplaatidele luua asjade interneti lahendusi programmeerimiskeeles Lua [38]. MicroPython on programmeerimiskeele *Python* 3 implementatsioon erinevatele mikrokontrolleritele ja kuna MicroPython toetab ESP8266

mikrokontrollerit, on võimalik ka WeMos arendusplaate *Python*'is programmeerida [36]. Käesolevas töös keskendutakse Arduino IDE abil programmeerimisele.

Järgnev osa on praktiline ja kirjeldab WeMos D1 kasutamiseks seadistamist kuni esimese töötava programmi loomise ja käivitamiseni.

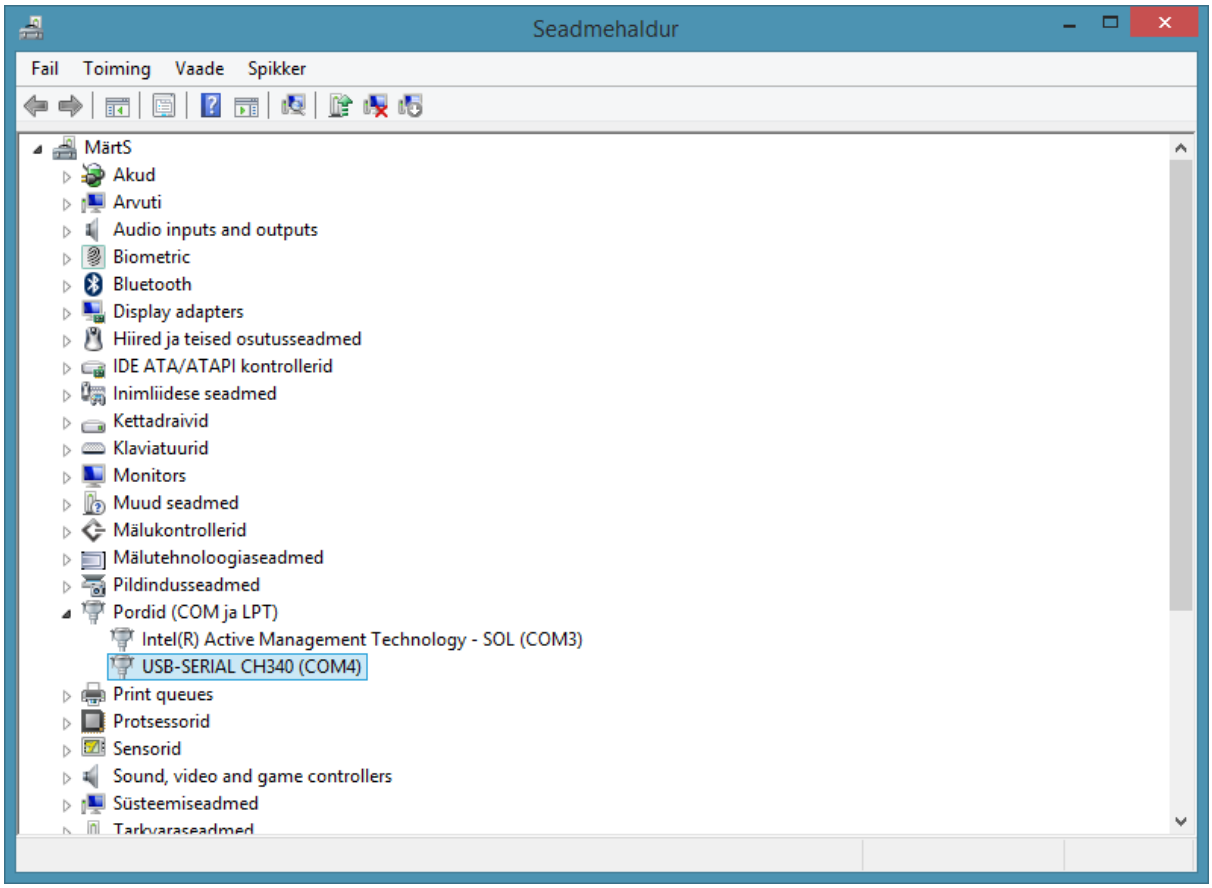
2.4.1. Arvutiga ühendamine

Alustuseks tuleb WeMos arendusplaat arvutiga ühendada. Selleks on olemas arendusplaadil Micro USB pesa. Ühendamiseks sobib ka näiteks enamasti kodus leiduv nutitelefoni USB juhe, aga kindlasti tuleb jälgida, et juhe ei oleks mõeldud ainult seadmete laadimiseks vaid et sel oleks ka andmevahetuse võimekus. Kui arendusplaat on arvutiga ühendatud, peaks arvuti seadme üles leidma. Kui seda ei juhtu, tuleb paigaldada vastav *draiver*.

2.4.2. Draiverite kontrollimine

WeMos D1 R2 ja D1 Mini kasutavad USB ühenduseks kontrollerit CH340G ja tuleb otsida sellele sobiv draiver, WeMos D1 Mini Pro jaoks läheb tarvis CP2104 draiverit [24-26]. CP2104 on juba olemas Microsofti draiverite andmebaasis ning paigaldatakse arendusplaadi ühendamisel arvutiga automaatselt. Paigaldamisel kindlasti jälgida operatsioonisüsteemi sobivust. Pärast draiveri paigaldust vajadusel taaskäivitada arvuti ja siis peaks juba arvuti seadme tuvastama. Järgnev juhend on koostatud Windows 8.1 operatsioonisüsteemi põhjal.

Kui arendusplaat on arvutiga ühendatud, tuleb avada Seadmehaldur (*Device manager*) ja kontrollida, kas arendusplaat tuvastatakse õigesti, avades Seadmehalduris Pordid (*Ports*) alammenüü ja kontrollides, kas sinna on tekkinud uus seade (joonis 5). Edaspidi läheb vaja ka COM Port numbrit. COM Port võib ka muutuda, kui vahetada USB pesa. Kui arendusplaat tuvastatakse, on draiverite paigaldus õnnestunud ja võib asuda Arduino IDE paigaldamise juurde.

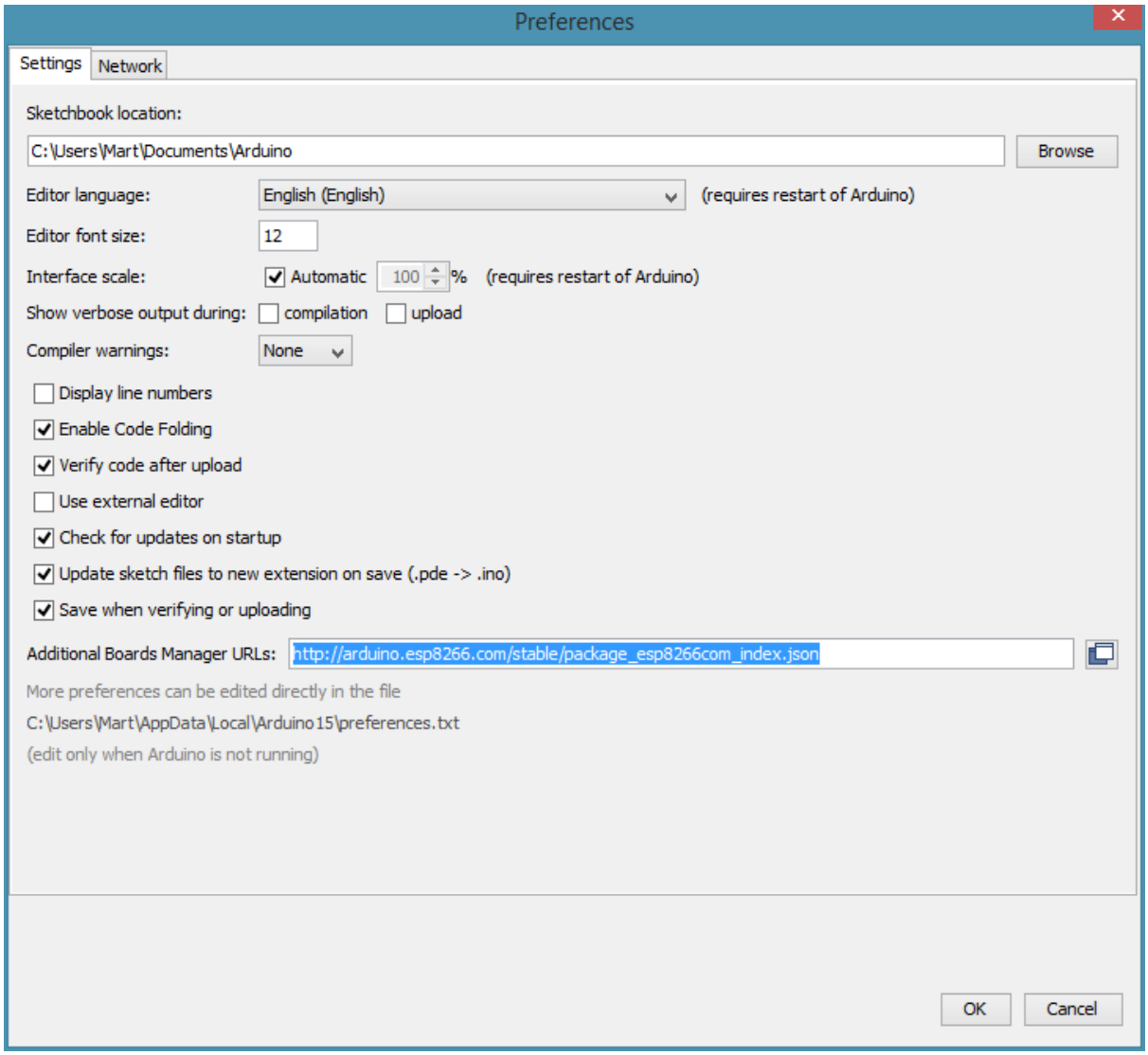


Joonis 5. Seadmehaldur.

Järgnevalt paigaldatakse tarkvara arendusplaadi programmeerimiseks.

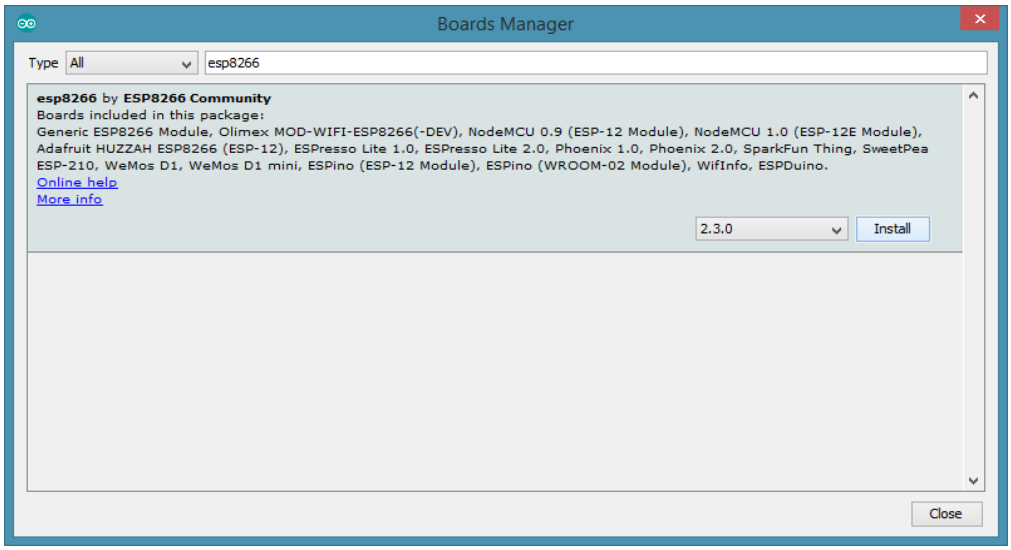
2.4.3. Arduino IDE paigaldamine ja seadistamine

Arduino IDE leiab Arduino ametlikult veebilehelt *www.arduino.cc*. Tuleks otsida menüüribalt nupp *"Download"* ja laadida alla oma operatioonisüsteemile vastav pakk, näiteks operatsioonisüsteemi *Windows* puhul *"Windows Installer"*. Kui Arduino IDE on paigaldatud, tuleks see avada. Arduino IDE ei ole algselt seadistatud ESP8266 mikrokontrolleris põhinevate arendusplaatidega töötama. Seadistamiseks tuleb menüüst valida *"File"* ja avanenud menüüst *"Preferences"*. Avanenud aknas tuleb leida rida *"Additional Board Manager URLs"* ja sinna lahtrisse sisestada järgnev aadress: *http://arduino.esp8266.com/stable/package_esp8266com_index.json (*joonis 6).



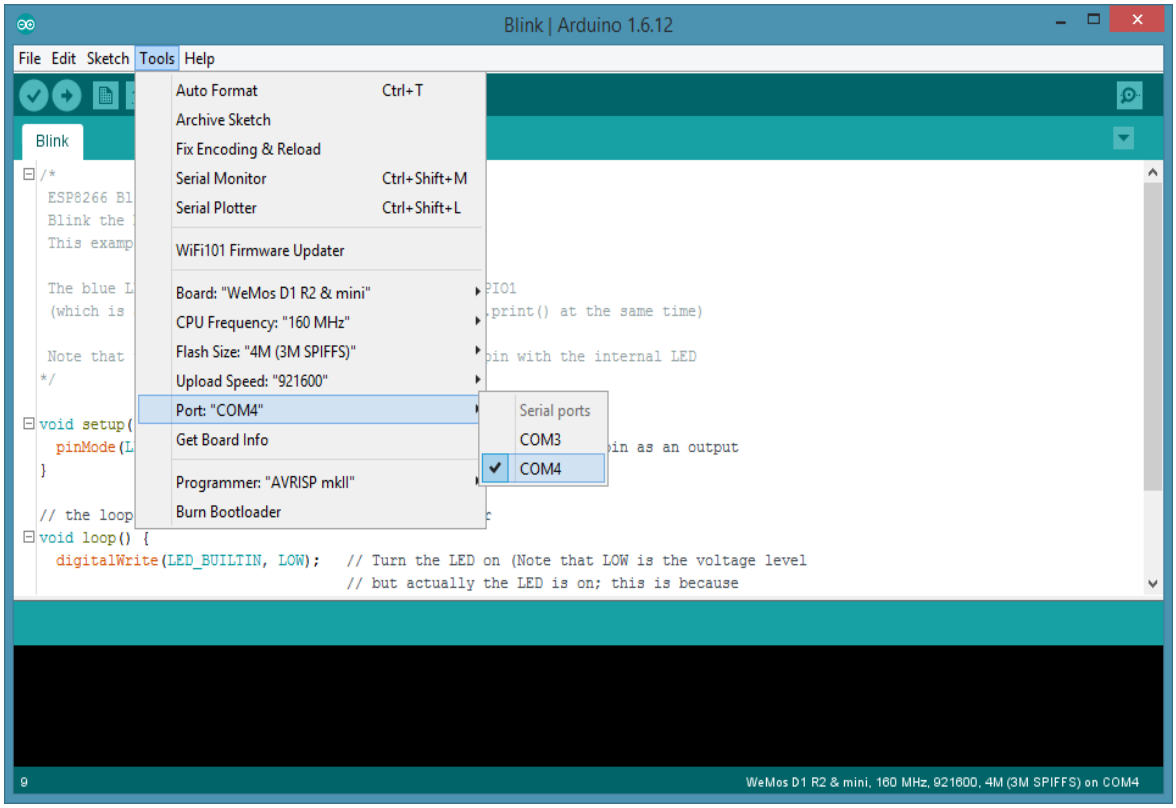
Joonis 6. Arduino IDE "*Preferences*" aken.

Pärast nupule OK vajutamist tuleb menüüribalt avada "*Tools*" → "*Board*" → **"***Boards Manager…"* ja avanenud aknas otsinguribale trükkida "esp8266", klikkida ette tulnud kirjele ja vajutada nupule "*Install*" (joonis 7). Nüüd paigaldatakse Arduino IDE-sse pakk erinevate ESP8266 arendusplaatide konfiguratsioonidega ning ESP8266 mikrokontrolleri funktsioonide kasutamiseks vajalike teekidega.



Joonis 7. Arduino IDE *"Boards Manager"* aken.

Järgnevaks sammuks tuleb valida taas menüüribalt "*Tools*" → *"Board"* ja seejärel otsida menüüst kirje "WeMos D1 R2 & Mini" ja vajutada sellele. Nüüd on "*Tools*" menüüs näha, et õige plaat on aktiveeritud, tuleb vaid valida õige pordi ehk pesa number (joonis 8), mis eelnevalt seadmehalduris kindlaks sai tehtud. "*Tools*" menüüs saab muuta ka teisi arendusplaadi parameetreid nagu näiteks selle taktsagedus.



Joonis 8. "*Tools*" menüü ja õige pordi numbri valimine.

Kui kõik eelnev on tehtud, on aeg arendusplaadi testimiseks esimese programmiga, mida kirjeldab järgmine punkt.

2.4.4. Arendusplaadi programmeerimine

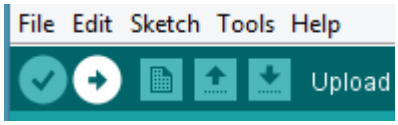
Arduino IDE-ga tuleb kaasa palju näidisprogramme, mida võib alati enda projektides kasutada. Neist lihtsaim, millega testida arendusplaadi tööd, on programm nimega *Blink*, mille ülesanne ei ole mitte midagi muud kui plaadil oleva LED tule vilgutamine.

Selle jaoks tuleb menüüribal valida *"File"* → *"Examples"* → *"ESP8266"* → *"Blink"*. Avaneb uus aken programmi koodiga. Arduino programmeerimiskeel on implementeeritud keeles C/C++ [39] ja koodi põhistruktuur koosneb kahest funktsioonist, `setup()` ja `loop()` [40, 41]. Esimene meetod käivitatakse vaid korra, arendusplaadi alglaadimisel. Seejärel jooksutatakse funktsiooni `loop()` sees olevat koodi tsüklis kuni seadme töö lõppemiseni. Täpsemalt on programmi koodi kirjeldatud joonisel 9.

```
/*
ESP8266 Blink by Simon Peter
Blink the blue LED on the ESP-01 module
This example code is in the public domain

See programm kasutab arendusplaatide pakis defineeritud muutujat LED_BUILTIN
arendusplaadile sisseehitatud LED viigu määramiseks.
*/
void setup() {
 pinMode(LED_BUILTIN, OUTPUT);     // Määrame LED_BUILTIN viigu väljundiks
}

// loop funktsioon töötab tsüklis arendusplaadi töö katkemiseni
void loop() {
 digitalWrite(LED_BUILTIN, LOW);   // Määrame LED tule viigu signaaliks LOW ehk 0
                                   // LED tuli süttib, sest ESP mikrokontrolleri
                                   // LED tulel on LOW aktiivne seis
 delay(1000);                      // Programm ootab 1000 millisekundit
 digitalWrite(LED_BUILTIN, HIGH);  // Määrame signaaliks HIGH ja LED tuli kustub
 delay(2000);                      // Programm ootab kaks sekundit
}
```

Joonis 9. Programm "*Blink*".

Programmi laadimiseks arendusplaadile tuleb vajutada nupuribal ümmargusele nupule, millel on kujutatud nool paremale (joonis 10). Seejärel programm kompileeritakse, laaditakse arendusplaadile ja käivitatakse. Kui arendusplaadil hakkab sinine tuli vilkuma, on arendusplaadi seadistamine ja testimine edukalt lõppenud.



Joonis 10. Nupuriba, programmi laadimise nupp aktiivne.

Järgmises punktis näidatakse, kuidas arendusplaat ühendada parooliga kaitstud WiFi võrguga.

2.5. Internetiühenduse loomine arendusplaadil

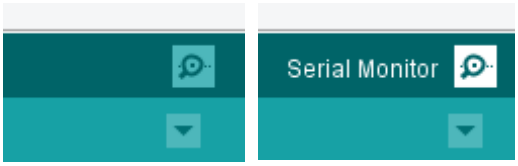
WeMos arendusplaadi abil asjade interneti lahenduste loomiseks tuleks arendusplaat ühendada internetti. Selleks on varasemalt paigaldatud ESP8266 mikrokontrollerite pakis teek nimega *"ESP8266WiFi"*, mille saab programmi importida Arduino IDE-s valides menüüribalt *"Sketch"* → *"Include Library"* → *"ESP8266WiFi"* või trükkides käsitsi programmi koodi algusesse rea "`#include <ESP8266WiFi.h>`". Seejärel tuleb defineerida muutujad WiFi võrgu nime ning parooli jaoks. Näidiskoodi leiab näiteks Arduino IDE menüüribalt valides *"File"*→*"Examples"*→*"ESP8266WiFI"*→*"WiFiClient"*. Joonisel 11 on programmi kood autoripoolsete muudatustega: lisatud on eesti keelsed kommentaarid ning eemaldatud read, mida edaspidi vaja ei lähe. Seda programmi hakatakse järgnevates punktides täiendama erinevate projektis vajaminevate funktsionaalsustega.

```
#include <ESP8266WiFi.h>

const char* ssid     = „wifi-ruuteri-SSID“;
const char* password = „wifi-parool“;

void setup() {
 Serial.begin(115200); // Avab jadapordi andmevahetuskiirusega 115200
 delay(10);
```

```
 // println() ja print() funktsioonid kirjutavad jadaporti andmeid
 // inimesele loetaval kujul, println lisab automaatselt reavahetuse märgi
 // andmete lõppu
 Serial.println();
 Serial.print(„Connecting to „);
 Serial.println(ssid);

 /*
   Määrame WiFi režiimiks wifi WIFI_STA ehk klientrežiim
   Algseadistuses on ESP8266 nii klient kui ka ligipääsupunkt
   ja see võib tekitada WiFi võrgus probleeme teiste seadmetega
 */
 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);

 // kontrollime, kas WiFi ühendus on loodud, kui ei ole, ootame pool
 // sekundit ning kontrollime uuesti
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(„.“);
 }
 // WiFi staatus on ühendatud, siis kuvame Serial Monitori
 // arendusplaadile omistatud IP aadressi
 Serial.println(„“);
 Serial.println(„WiFi connected“);
 Serial.println(„IP address: „);
 Serial.println(WiFi.localIP());
}

void loop() {
}
```

Joonis 11. Näidisprogramm *“WiFiClient”*.

Samuti on joonisel 11 kommentaarides kirjeldatud jadapordi avamist ning sellesse kirjutamist. Jadaporti kasutab arendusplaat näiteks arvutiga suhtlemiseks. Jadapordi kaudu suhtluse jälgimiseks on Arduino IDE-s tööriist nimega *“Serial Monitor”*, mille saab avada klahvide kombinatsiooniga *“Control”* + *“Shift”* + *“M”*, valides menüüribalt *“Tools”* → *“Serial Monitor”* või klikkides programmi üleval paremas sevas asuvale luubi ikooniga nupule (joonis 12).



Joonis 12. *"Serial Monitor"* ikoon (vasakul) ning sama ikoon aktiivsena (paremal).

Lisaks tüüpilisele parooliga kaitstud internetivõrgule on ESP8266 mikrokontroller uusima tarkvaraarenduskomplekti (SDK versioon 2.0.0) kasutades võimeline ühenduma ka WPA2 *Enterprise* WiFi võrku, mis vajab autentimiseks lisaks paroolile veel ka kasutajanime. Selline WiFi võrk on näiteks ka Eduroam. ESP8266 Arduino IDE pakk ei sisalda veel uusimat SDK-d. Kasutades mitteametlikku „*Nightly*" versiooni ESP8266 Arduino pakist ning muutes kuueteistkümnendkoodis kompileeritud faili „lipwpa2.a" õnnestus siiski ühendada WeMos arendusplaat Eduroam WiFi võrguga. Abi sai autor ESP8266 Arduino GitHub repositooriumist [42]. Paraku võib ühendamiseks kuluda palju aega, sest mingil põhjusel võib arendusplaat enne edukat WiFi võrguga ühendumist kümneid kordi „*Exception* 3" veaga kokku joosta. Kokkujooksmine võib olla ka tingitud „*Nightly*" versiooni ebastabiilsusest. Bakalaureusetöö kirjutamise hetkel veel ametlikku stabiilset versiooni SDK versiooni 2.0.0 sisaldavast ESP8266 Arduino pakist valminud ei olnud.

Kui arendusplaat on ühendatud internetti, tuleks seda ka kuidagi juhtida. Järgmises peatükis kirjeldatakse, kuidas seda interneti kaudu teha.

3. Asjade internet WeMos arendusplaadiga

Sageli arvatakse, et asjade internet töötab selliselt, et ühendatakse mingi seade internetivõrku ja kõik juba toimib. Tegelikult on seadme internetiga ühendamine kõigest üks väike osa toimivast värkvõrgust. Asjade interneti juurde kuulub ka internetti ühendatud asjade haldamine, jälgimine ja juhtimine üle võrgu ning seadmete omavaheline suhtlus. Käesolev peatükk tutvustab enamlevinud vahendeid, mis oluliselt lihtsustavad ja kiirendavad asjade interneti lahenduste loomist ning hiljem isikliku värkvõrgu haldamist. Peatükis kirjeldatakse põhilisi protokolle sõnumite edastamiseks ja vastuvõtmiseks ning selgitatakse, kuidas värkvõrgu seadmeid juhtida saab.

3.1. Sõnumivahetusprotokolli tutvustus

Internetiga ühendatud seadmeid saab andmeid edastama ja vastu võtma panna mitut moodi. Enim levinud viisid on kasutades WebSocket või MQTT protokolli [43].

WebSocket on võrgu transpordikihi protokoll, mis töötab üle TCP ja võimaldab reaalajas kahepoolset suhtlust kliendi ja serveri vahel [44]. WebSocket protokolli puhul tuleb samas andmete edastamise viis ehk rakenduskihi protokoll ise kirjutada [43].

MQTT (MQ *Telemetry Transport*) protokoll on välja töötatud IBM poolt ja selle eelisteks WebSocket protokolli ees on juba välja töötatud ja standardiseeritud sõnumivahetus seadmete ja serveri vahel kasutades *Publish-Subscribe* sõnumivahetuse mustrit [45]. *Publish-Subscribe* mustri [45] põhimõte on järgmine: on olemas kliendid, kes saavad kuulata kindlaid kanaleid (*Subscribe*) ja kindlatesse kanalitesse sõnumeid saata (*Publish*). Server töötab sõnumivahendajana, võttes vastu saadetud sõnumeid ja edastades neid sobivasse kanalisse, kust siis teised seadmed sõnumi kätte saavad. MQTT üheks heaks omaduseks on teenusekvaliteedi (QoS) määramise võimalus. Sõnumi saatja saab määrata sõnumile teenusekvaliteedi ning ka vastuvõtja saab määrata, millise kvaliteediga ta sõnumeid soovib saada. Vastuvõtja saab alati sõnumi kätte enda määratud kvaliteediga olenemata sellest, mis kvaliteediastmega sõnum välja saadeti.

Kvaliteediastmeid on kolm [45, 47]:

- QoS 0 – madalaima kvaliteediga, sõnum saadetakse ainult üks kord ja selle kohalejõudmine ei ole garanteeritud ja sõnum võib minna kaduma, kui näiteks kliendi ühendus serveriga katkeb;
- QoS 1 – sõnum saadetakse vähemalt üks kord, kuid võib juhtuda, et saadetakse ka mitu korda sama sõnumit ja seega võib sõnumi vastuvõtja sama sõnumit korduvalt töödelda.
- QoS 2 – kõrgeim teenusekvaliteet, sõnum saadetakse kliendile täpselt üks kord, ei rohkem ega vähem. Sõnumi saatja saab kinnituse, kui sõnum on vastu võetud ning seejärel annab sõnumi saatja vastuvõtjale loa sõnumit töödelda. Sellega garanteeritakse, et sõnum jõuab kindlasti kohale, aga seda ei töödelda rohkem kui üks kord.

Ka käesolevas töös kasutatakse sõnumivahetuseks seadmete ja serveri vahel MQTT protokolli. Järgmises alapunktis tutvustatakse erinevaid variante värkvõrgu seadmete haldamiseks ja juhtimiseks.

3.2. Värkvõrgu seadmete juhtimine

Internetti ühendatud asjadest on kasu juhul, kui asjade omanik neid ka kergesti hallata saab. Seadmete lihtsamaks haldamiseks on loodud mitmesuguseid süsteeme ja platvorme. On platvorme, mis on mõeldud kommertslahenduste tarbeks ning eesmärgiga juhtida keerukaid süsteeme. Üheks selliseks on näiteks Cumulocity asjade interneti platvorm, millele toetub ka ettevõte Telia asjade interneti ökosüsteem ning mida kasutatakse targa linna projektis SmartEnCity [48]. Samuti on olemas vabavaralisi avatud lähtekoodiga platvorme, mida tihti kasutatakse targa kodu lahenduste loomisel. Levinumad taolised avatud lähtekoodiga platvormid on Home Assistant [49] ning openHAB [50]. Eelpool mainitud süsteemid on keerukad, vajavad seadistamiseks aega ning oskusi, samas on need võimekad ja toetavad väga palju erinevaid asjade interneti seadmeid. Soovides kasutada ainult mõnda WeMos arendusplaati asjade internetiga tutvumiseks, võib keerukate platvormide kasutamine olla ebavajalik ning seadistamine ülemäära tülikas.

Asjade interneti levimisega massidesse on vajadus lihtsate ja tavakasutajale arusaadavate süsteemide järele aina kasvamas. Otsides võimalusi värkvõrgu seadmete mugavaks haldamiseks leidis autor teenuse Adafruit IO, mis just seda probleemi püüab lahendada:

Adafruit IO soovib muuta värkvõrgu seadmete haldamise interneti kaudu inimestele lihtsaks ja mugavaks. Järgmises punktis kirjeldatakse teenust Adafruit IO lähemalt.

3.2.1. Adafruit IO

Adafruit IO on ettevõtte Adafruit poolt loodud veebiteenus. Adafruit teeb tööd asjade interneti inimestele lähemale toomiseks, tootes mitmesuguseid seadmeid arendusplaatidele ning koostades põhjalikke juhendeid erinevate mikrokontrollerite kasutamiseks [51]. Adafruit on seega suurepärane keskkond mitmesuguste riistvaralahenduste loomise õppimiseks.

Adafruit IO toetab MQTT protokolli ja HTTP päringuid, seega kui soov on asjade internetiga tutvust teha kasutades vaid WeMos või teisi ESP8266 mikrokontrolleril põhinevaid arendusplaate, sobib Adafruit IO selleks ideaalselt. Kuna tegemist on veebiteenusega, on sel ka omad piirangud, mida kasutaja kontrollida ei saa [52]:

- maksimaalne MQTT kanalite arv 10,
- maksimaalne juhtpaneelide arv 10,
- andmeid saab ühte kanalisse saata kuni 125 korda minuti jooksul,
- andmeid hoitakse Adafruit IO serveris 30 päeva

Adafruit IO puhul ei pea kasutaja väga aega kulutama süsteemi seadistamisele. Süsteem on kohe kasutamiseks valmis, seega saab kasutaja pigem keskenduda värkvõrgu lahendustele endile kui süsteemile, mis neid lahendusi juhtima hakkab.

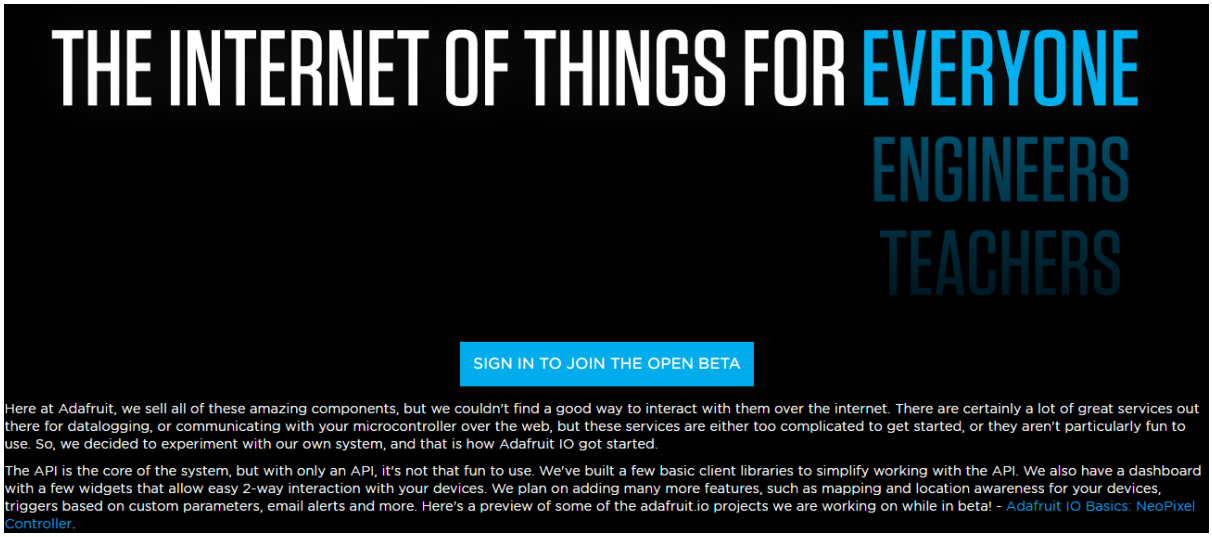
Analüüsides erinevaid võimalusi arendusplaadi juhtimiseks läbi interneti osutus ka loodava projekti jaoks otstarbekas kasutada Adafruit IO platvormi. Selliselt on võimalik keskenduda arendusplaadi programmeerimisele ja riistvaralahenduse loomisele värkvõrgu platvormi seadistamise pealt aega kokku hoides. Adafruit IO juhendid on põhjalikud ning süsteemi kasutamine ei tohiks valmistada kellelegi raskusi. Järgnevas punktis tutvustatakse platvormi lähemalt.

3.3. Värkvõrgu platvormi Adafruit IO seadistamine

Käesolevas punktis tehakse tutvust Adafruit IO platvormiga, kirjeldatakse selle olulisemaid osasid ning näidatakse, kuidas ühendada arendusplaat Adafruit IO platvormiga kasutades MQTT protokolli.

3.3.1. Kasutaja loomine

Esmalt tuleb navigeerida veebilehele *io.adafruit.com*. Avanenud leheküljel on inglise keelne lühitutvustus ning sinine nupp *"Sign in to join the open beta"* (joonis 13). Vajutades sellele nupule avaneb uus vaade, kus on vorm leheküljele sisenemiseks ning ka nupp *"Sign up"*, millele tuleks järgmisena vajutada.

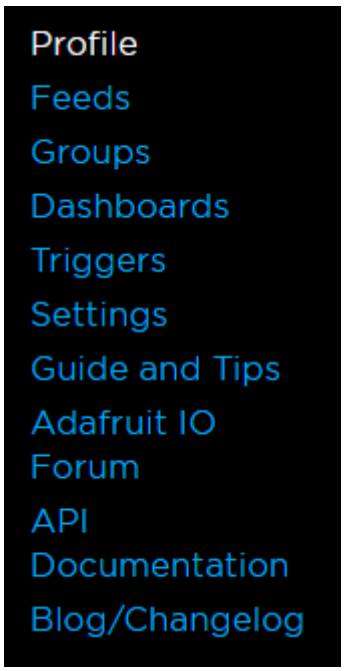


Joonis 13. Adafruit IO avaleht.

Avaneb registreerimisvorm, kuhu tuleb sisestada enda andmed ja seejärel vajutada nupule *"Create account"*. Seejärel toimub automaatne sisselogimine ja suunatakse edasi äsjaloodud kasutaja juhtpaneelide leheküljele. Järgnevas punktis tutvustatakse Adafruit IO süsteemi juurde kuuluvaid osasid.

3.3.2. Süsteemi osade tutvustus

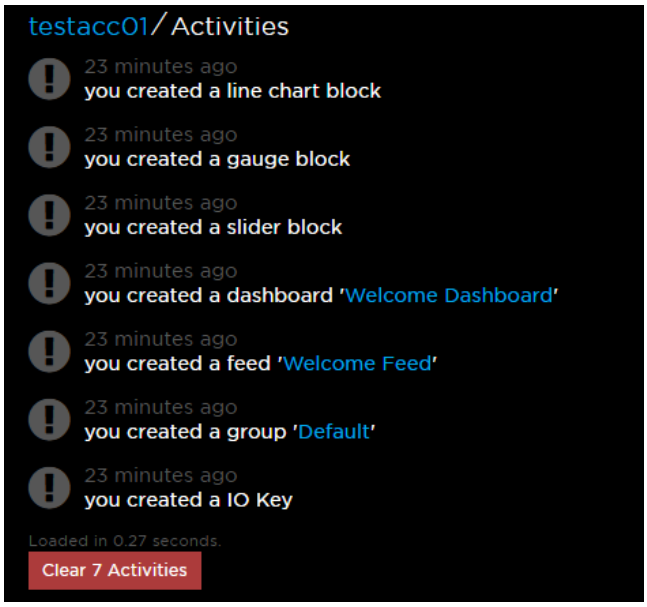
Adafruit IO süsteem koosneb mitmest osast, mis kõik on ligipääsetavad menüüriba kaudu (joonis 14). Kasutaja jaoks kõige olulisemaid osasid tutvustavad järgmised punktid.



Joonis 14. Adafruit IO menüü.

3.3.2.1. "*Profile*"

Menüüriba esimene link on nimega *"Profile"*. Vajutades sellele avaneb lehekülg nimega *"Activities"*, mis kuvab kasutaja kõiki tegevusi süsteemis (joonis 15). Esimesed seitse tegevust on loodud süsteemi poolt süsteemi demonstreerimise eesmärgil.

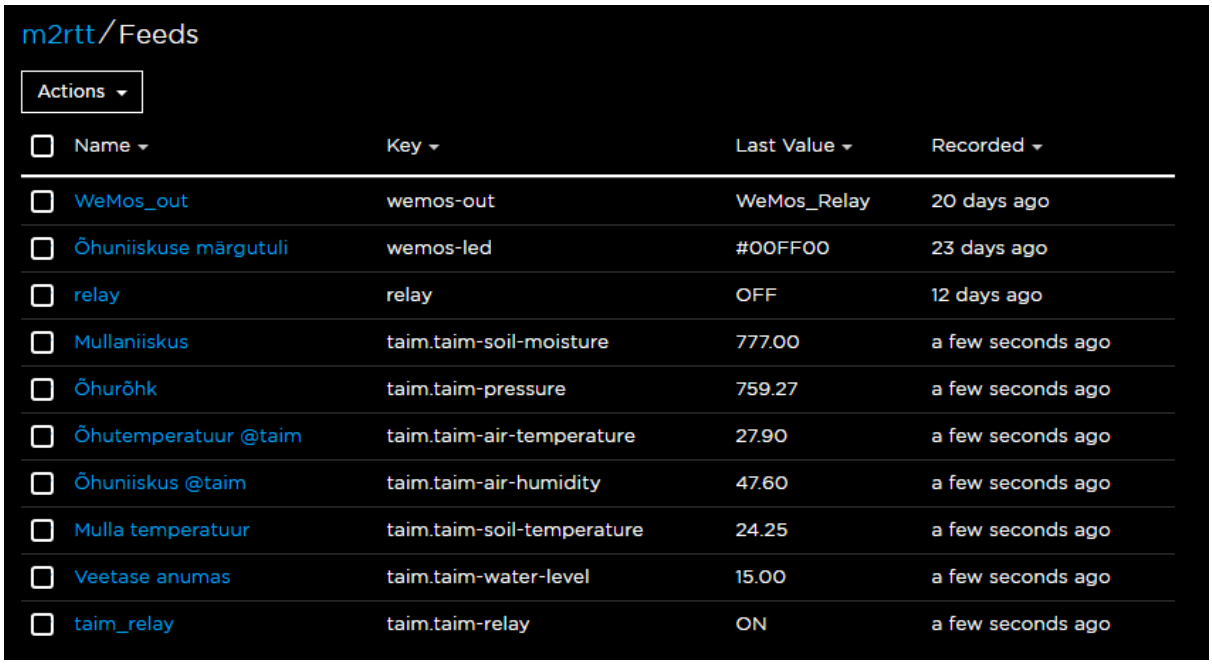


Joonis 15. Lehekülg *"Activities"*.

Kasutajal on võimalus ka tühjendada tegevuste ajalugu vajutades nupule *"Clear x Activities"*, kus "x" on kasutaja tegevuste koguarv süsteemis.

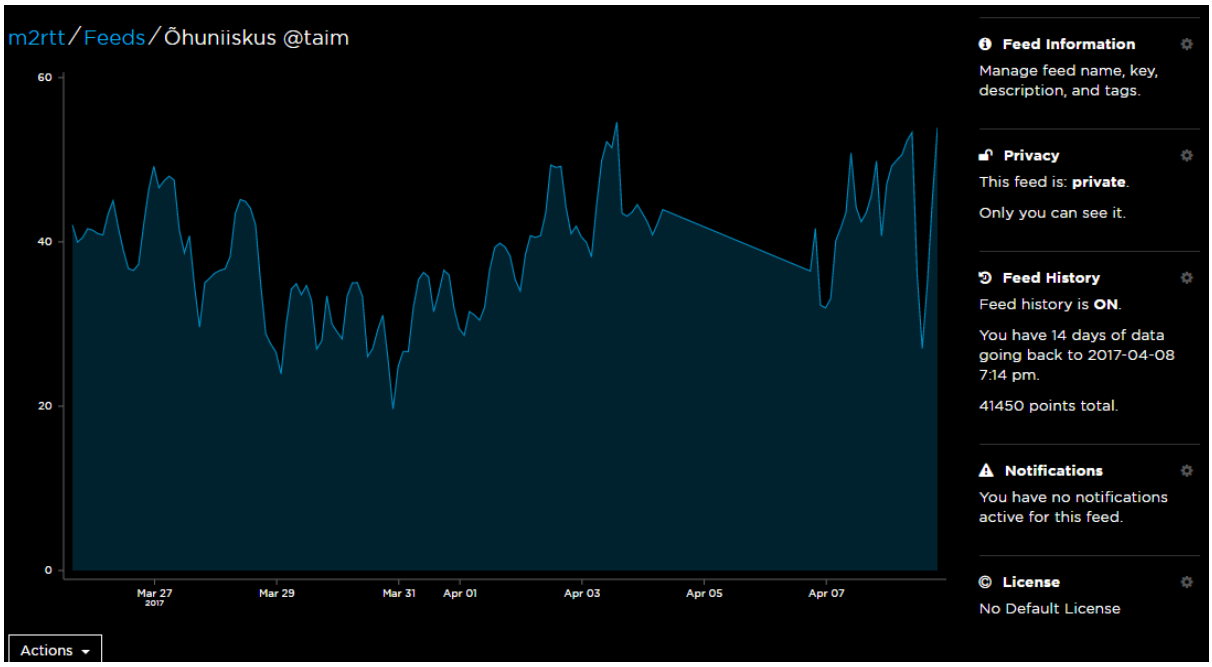
3.3.2.2. "*Feeds*"

Teine link menüüs on nimega "Feeds". Klikkides sellele avaneb lehekülg, kus kuvatakse ühes sorteeritavas tabelis kõiki kasutaja loodud MQTT kanaleid (joonis 16). Iga kanali kohta on kirjas selle nimi, võti, viimane väärtus ja millal see väärtus kanalisse saadeti.



Joonis 16. Lehekülg *"Feeds"*.

Kanaleid on võimalik märgistada vajutades kastile kanali nime ees. Klikkides nupule *"Actions"*, avaneb rippmenüü valikutega kanalite lisamiseks (*"Create a New Feed"*), märgistatud kanalite muutmiseks (*"Edit Selected Feeds"*) ja kustutamiseks (*"Delete Selected Feeds"*). Menüüs on ka valik valitud kanalitele graafikute koostamiseks (*"Graph Selected Feeds"*). Klikkides kanali nimele, avaneb uus lehekülg valitud kanali täpsema ülevaatega (joonis 17).



Joonis 17. Konkreetse kanali vaade.

Leheküljel kuvatakse kanalisse saabunud andmeid ja ka graafikut. Graafiku all ja tabeli kujul andmete peal asub nupp *"Actions"*, millele klikkides avaneb rippmenüü valikutega andmete käsitsi lisamiseks (*"Add Data"*), tabelis märgistatud andmete eemaldamiseks (*"Remove Selected Data"*) ning andmete allalaadimiseks JSON või CSV formaadis.

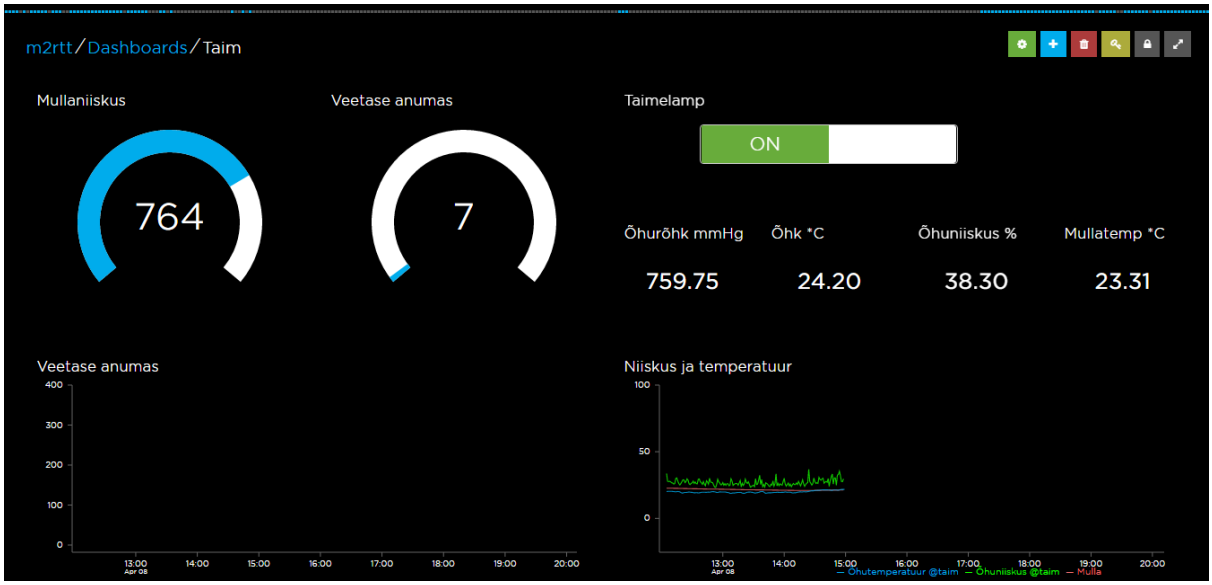
Lehe paremas osas asuvad erinevad kanali seaded, igaühe juures on hammasratas, millele klikkides saab vastavat seadet muuta. Näiteks *"Feed information"* juures saab muuta kanali nime, võtit ja kirjeldust, *"Privacy"* tähendab privaatsusseadeid. Muutes kanali avalikuks, pääseb sellele ligi iga huviline. Veel on võimalik sisse ja välja lülitada kanali ajalugu, lülitada sisse teavitused selle kohta, kui kanalisse pole andmeid saabunud ning määrata ka kanali litsents.

3.3.2.3. "*Groups*"

Lehekülg *"Groups"* on mõeldud kanalite gruppide haldamiseks. Grupid on kasulikud, kui kasutajal on mitu seadet süsteemiga ühenduses ja igal seadmel omad kanalid. Gruppide lehel on nupp *"Action"*, mille abil sarnaselt lehekülje *"Feeds"* samanimelise nupule saab lisada, muuta ja kustutada gruppe. Grupp nimega *"Default"* sisaldab kõiki selliseid kanaleid, millel pole grupp määratud. Vajutades grupi nimele avaneb leheküljele *"Feeds"* sarnane vaade, kus kuvatakse kõiki gruppi kuuluvaid kanaleid. Ka *"Action"* nupu valikud on samad, lisandunud on vaid valikud *"Add Feed to Group"* olemasoleva kanali lisamiseks gruppi ning *"Remove Selected Feeds from Group"* valitud kanalite grupist eemaldamiseks.

3.3.2.4. "*Dashboards*"

Leheküljel *"Dashboard"* kuvatakse kasutaja loodud juhtpaneele. Juhtpaneelid on mõeldud seadmete juhtimiseks ning andmete visualiseerimiseks. Nupu *"Action"* funktsionaalsus leheküljel on sama nagu lehekülgedel *"Feeds"* ja *"Groups"*. Juhtpaneele võib luua mitu, igal paneelil oma konkreetne eesmärk. Selliselt saab mugavalt seadmeid hallata ja ei teki probleemi, kus ühel paneelil on liiga palju elemente. Klikkides konkreetsele juhtpaneelile avaneb valitud juhtpaneeli vaade (joonis 18). Paneelil kuvatakse elemente, mis on mõeldud kas andmete kuvamiseks või saatmiseks elemendile määratud kanalisse.



Joonis 18. Konkreetse juhtpaneeli vaade.

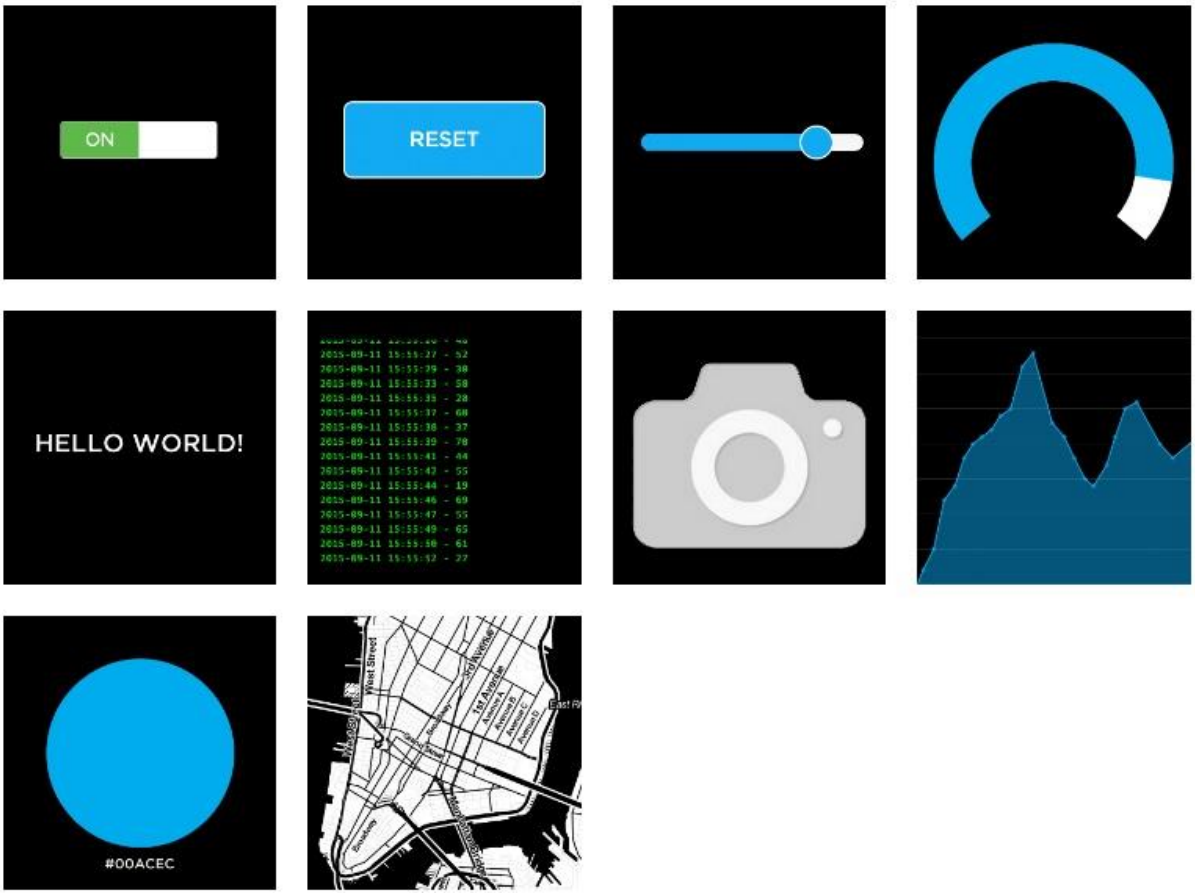
Juhtpaneeli paremas servas on nupud juhtpaneeli kontrollimiseks. Alustades paremalt:

- hall nooltega nupp juhtpaneeli täisekraanile viimiseks,
- hall lukuga nupp juhtpaneeli privaatsuse muutmiseks,
- kollane nupp Adafruit IO salajase võtme kuvamiseks,
- punane nupp juhtpaneeli kustutamiseks,
- sinine nupp juhtpaneelile uue elemendi lisamiseks,
- roheline nupp juhtpaneelil olevate elementide muutmiseks.

Adafruit IO salajane võti on vajalik selleks, et teised seadmed Adafruit IO-ga ühendada saaksid. Juhtpaneelile on võimalik lisada erinevaid elemente. Elemendi lisamise nupule klikkides avaneb elementide valik (joonis 19). Alustades ülevalt vasakult on elemendid järgnevad:

- kahe võimaliku olekuga nupp,
- lihtne nupp kindlasse kanalisse kindla väärtuse saatmiseks,
- liugur väärtuse valimiseks määratud vahemikust,
- näidiku stiilis element väärtuse kuvamiseks,
- tekstielement, mida saab panna kuvama staatilist teksti või väärtusi määratud kanalist,
- voog, mis kuvab kuni viite kanalisse saabunud andmeid,
- pildielement base64 kodeeringus pildi kuvamiseks,
- joondiagramm kuni viie kanali andmete graafikuna kuvamiseks,
- värvivalija, mis saadab kanalisse ja kuvab kuueteistkümnendsüsteemi vormingus värve,

- kaart kanalist saabunud asukohaandmete kuvamiseks



Joonis 19. Juhtpaneelile valitavad elemendid.

Elementide muutmiseks ja ümberpaigutamiseks tuleb klikkida rohelisele nupule. Seejärel saab element ümber paigutada, muuta elementide suurust, seadeid ja elemente kustutada.

3.3.2.5. "*Triggers*"

Lehekülg *"Triggers"* on mõeldud päästikprotsesside loomiseks ja haldamiseks. *"Action"* nupule klikkides avaneb menüü päästikute lisamiseks, muutmiseks ja eemaldamiseks. Võimalik on lisada kaht tüüpi päästikprotsesse: reaktiivsed protsessid ja plaanitud protsessid.

Reaktiivsed protsessid reageerivad määratud kanali väärtuse muutusele ja võivad reageerida kolmel viisil: saadavad e-kirja Adafruit IO seadetes määratud e-posti aadressile, saadavad *webhook*-päringu kindlale aadressile või saadavad sõnumi määratud väärtusega valitud Adafruit IO MQTT kanalisse. Näiteks võib teha päästikprotsessi, mis saadab kasutajale e-kirja,

kui toatemperatuur on liiga madalale langenud. Selliselt saab ka ühe seadme andmete abil juhtida teisi seadmeid. Näiteks kui valgusanduriga seade teatab, et on juba hämar, saadab Adafruit IO kindlasse kanalisse teate, mille peale paneb valgustust kontrolliv seade toas tuled põlema.
Plaanitud protsessid on mõeldud kindla ajavahemiku järel kasutaja e-posti aadressile teate saatmiseks määratud kanali väärtusega.

3.3.2.6. Süsteemi muud osad

Leheküljel *"Settings"* saab hallata Adafruit IO konto seadeid. Leheküljel näeb enda Adafruit IO privaatvõtit, saab ühendada konto veebiteenuse IFTTT (*If This Then That*) kontoga, muuta konto andmeid nagu ees- ja perenimi, parool ja palju muud. Kasutaja saab alla laadida kõik konto andmed, sealjuures ka andmed gruppide ja kanalite kohta. Soovi korral on võimalik ka kasutajakonto kustutada. IFTTT teenust kirjeldatakse lähemalt viiendas peatükis.

Ülejäänud linkidelt Adafruit IO menüüs leiab kasutusjuhendi, dokumentatsiooni Adafruit IO kasutamise kohta, foorumi abi otsimiseks ja süsteemi muudatuste logi. Kasutusjuhendit ja dokumentatsiooni tasub kindlasti lugeda, kuna need on põhjalikud ja sisaldavad ohtralt näiteid.

3.4. Arendusplaadi värkvõrgu platvormiga ühendamine

Teises peatükis näidati, kuidas arendusplaat internetiga ühendada. Käesolevas punktis näidatakse, kuidas programmeerida arendusplaat MQTT protokolli kasutama ning kuidas arendusplaat Adafruit IO MQTT serveriga suhtlema panna.

3.4.1. Arendusplaadi ühendamine MQTT serveriga

MQTT serveriga suhtlemiseks on olemas vastavad teegid. Neist levinuim on teek nimega *PubSubClient.* Eelnevalt tuleks teek Arduino IDE-s installeerida valides menüüribal *"Sketch" → "Include Library" → "Manage Libraries…"*.
Avanenud akna otsinguribale tuleb trükkida *"PubSubClient",* klõpsata leitud teegile ning seejärel nupule *"Install".* Seejärel tuleb teek lisada lisada programmi koodi varem kirjeldatud viisil. Kasutades teegiga kaasa tulnud näiteprogrammi *"mqtt_esp8266"* tuleks täiendada

internetiühenduse loomise programmi. Kõigepealt tuleb defineerida uued muutujad MQTT serveriga ühenduse loomiseks (joonis 20).

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid     = „wifi-ruuteri-SSID“;
const char* password = „wifi-parool“;

const char* mqtt_server = „io.adafruit.com“;
const char* mqtt_user = „Adafruit IO kasutajanimi“;
const char* mqtt_password = „AIO-võti“;
const int port = 8883; // 8883 on turvalise SSL-ühenduse jaoks

const char* clientID = „WeMos_seade“; // iga seadme ID peab olema unikaalne

// Adafruit IO serveri sõrmejälg turvalise ühenduse loomiseks
const char* fingerprint = „26 96 1C 2A 51 07 FD 15 80 96 93 AE F7 32 CE B9 0D 01 55 C4“;

// subscribe kanal on sõnumite vastuvõtmiseks
const char* subscribeKanal = „kasutajanimi/feeds/esimene-kanal“;
// publish kanali kaudu saadab arendusplaat sõnumeid
const char* publishKanal = „kasutajanimi/feeds/teine-kanal“;

char msg[50]; // muutuja, mida kasutatakse sõnumi teksti koostamisel
// turvalise WiFi ühenduse klient ja MQTT klient
WiFiClientSecure espClient;
PubSubClient client(espClient);
```

Joonis 20. Programmi täiendused koos kommentaaridega.

Muutuja “`mqtt_password`” väärtuseks on Adafruit IO võti, mida kirjeldati eelnevalt punktis 3.3.2.4. Adafruit IO serveri sertifikaadi sõrmejälg on leitav *Adafruit*’i blogipostitusest pealkirjaga *“IoT Security: Connecting Your ESP8266 to Adafruit IO with SSL/TLS”* [53], mis kirjeldab turvalise ühenduse loomist Adafruit IO süsteemiga. Adafruit IO kanali nime formaat on range ja kui seda ei järgi, ei õnnestu kanaliga ühendust saada. Kanali nime formaat on järgmine: “kasutajanimi/feeds/kanalinimi”. Kasutajanimi on Adafruit IO kasutajanimi, „*feeds*“ märgib, et tegu on kanalitega ja “kanalinimi” on soovitud kanali nimi või võti. Mõistlik oleks kasutada võtit, sest see on igal kanalil unikaalne ning võib sisaldada ainult ASCII kodeeringus väiketähti, numbreid ja sidekriipsu [54].

Sarnaselt näitekoodile on vaja lisada funktsioon `callback()`, mis tegeleb saabunud MQTT sõnumitega ning funktsioon `reconnect()`, mis tegeleb MQTT ühenduse haldamisega (joonis 21).

```
// mõeldud saabuvate sõnumitega tegelemiseks
void callback(char* topic, byte* payload, unsigned int length) {
 // Teisendame sisse tulnud baidijärjendi sõneks
 payload[length] = '\0'; // Lisame lõpumärgi, mis tähistaks järjendi lõppu
 String message = (char*)payload;
 // teisendame tüübi char järjendi sõneks
 String teema = (char*)topic;
 // Serial Monitori kaudu saame lugeda, mis kanalisse mis sisuga sõnum saabus
 Serial.print(„Message arrived on topic: [„);
 Serial.print(topic);
 Serial.print(„], „);
 Serial.println(message);
 // järgnevalt tegeleme saabunud sõnumitega
 // kuulatavaid kanaleid võib olla mitu, siis saame iga kanali puhul
 // eraldi tegutseda
 if(teema == subscribeKanal){
   // Siia lisatakse see, mida kuidas programm käitub,
   // kui kanalisse saabub sõnum
 }
}

void reconnect() {
 // Kui klient ei ole ühendatud, proovime ühendada
 while (!client.connected()) {
   Serial.print(„Attempting MQTT connection...“);
   // püüame ühendust luua
   if (client.connect(clientID, mqtt_user, mqtt_password)) {
     Serial.println(„connected“);
     // kui ühendus on loodud, hakkame enda kanalit kuulama.
     client.subscribe(subscribeKanal, 1);
   } else {
     // kui ühenduse loomine ebaõnnestus, kuvame Serial Monitori põhjuse
     Serial.print(„failed, rc=“);
     Serial.print(client.state());
     Serial.println(„ try again in 5 seconds“);
     // Viie sekundi pärast proovime uuesti
```

```
    delay(5000);
   }
 }
}
```

Joonis 21. Funktsioonid `callback()` ja `reconnect()`.

Turvalise ühenduse loomiseks on vaja Adafruit IO süsteemi poolset kinnitust sertifikaadi sõrmejälje õigsuse kohta. Selle jaoks tuleb koodi lisada funktsioon `verifyFingerprint()` (joonis 22), mis on samuti leitav eelpool mainitud blogipostitusest turvalise ühenduse kohta.

```
void verifyFingerprint() {
 // kuvame Serial Monitori, kuhu ühendust luuakse
 Serial.print(„Connecting to „);
 Serial.println(mqtt_server);
 // kui ühendust ei suuda luua, katkestame töö.
 if (! espClient.connect(mqtt_server, port)) {
   Serial.println(„Connection failed. Halting execution.“);
   // lõpmatu tsükkel, kus midagi ei tehta,
   // põhjustab arendusplaadi taaskäivitamise
   while(1);
 }
 // üritame sõrmejälge serveriga kinnitada
 if (espClient.verify(fingerprint, mqtt_server)) {
   Serial.println(„Connection secure.“);
 } else {
   // ebaõnnestunud katse korral ei ole ühendus turvaline
   Serial.println(„Connection insecure! Halting execution.“);
   while(1);
 }
}
```

Joonis 22. Funktsioon `verifyFingerprint()` sõrmejälje kinnitamiseks.

Täiendada tuleb ka `setup()` funktsiooni kolme reaga, mille leiab jooniselt 23. Need kolm rida tuleks lisada funktsiooni lõppu pärast WiFi võrguga ühendamise koodi ehk rea `Serial.println(WiFi.localIP())` alla.

```
verifyFingerprint(); // funktsioon sõrmejälje kinnitamiseks
client.setServer(mqtt_server, port); // määrame MQTT serveri
// callback tegeleb kanalisse saabunud sõnumitega
client.setCallback(callback);
```

Joonis 23. Funktsiooni `setup()` lisatavad read.

Programm on nüüd valmis suhtlema Adafruit IO-ga. Järgmises peatükis tutvustatakse lähemalt loodavat projekti ning kirjeldatakse projekti riistvaralist ning tarkvaralist poolt.

4. Tark taimekasvatus WeMos arendusplaadiga

Taimede kasvatamine võib tunduda lihtne, kuid taimede kiireks ja efektiivseks kasvuks peab olema hoolas ning teadma kasvatatava taime kasvutingimusi. Põhilised taimekasvutegurid on järgnevad [55]:

- valgus,
- soojus,
- vesi,
- õhk,
- toitained

Loetelus mainitud tegureid tuleb jälgida pidevalt ning vastavalt taime eripäradele ka kasvutingimusi muuta, mida saab teha näiteks järgmistel viisidel:

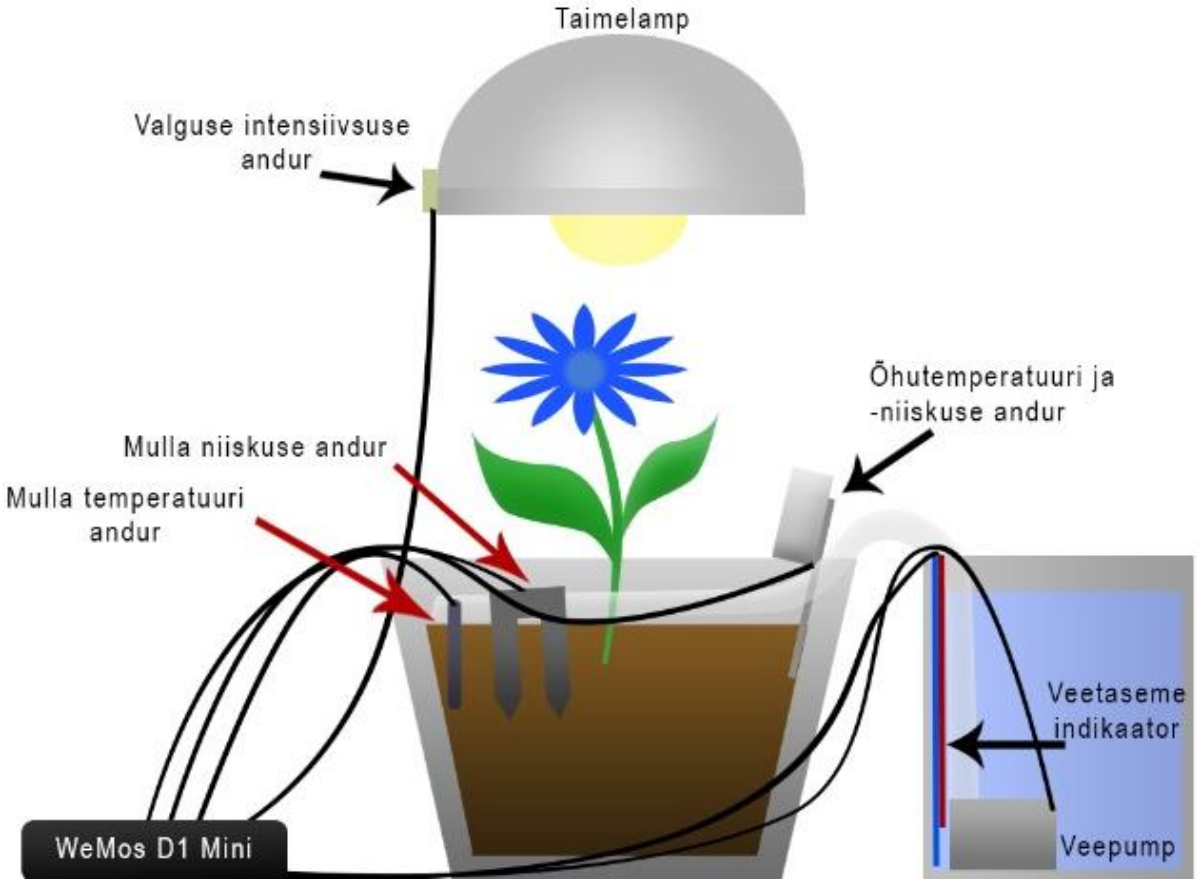
- valguse kogust, kasutades hämaral ajal taimelampi,
- temperatuuri, paigutades taime sobiva temperatuuriga kohta oma elamises,
- vee hulka, kastes taime regulaarselt ja sobivas koguses,
- toitaineid, väetades taimi

Selleks, et teada, kas taim vajab kastmist või temperatuuri muutust, tuleks jälgida vastavaid parameetreid nagu mulla temperatuur ja niiskus ning õhutemperatuur ja –niiskus. Kogenud taimekasvatajad juba enam-vähem teavad, kuidas tegutseda, et taimed hästi kasvaks. Kui tegemist on hobiaedniku või õpilasega, kes pole varem kokku puutunud taimekasvatusega, kuid soovib näiteks ühiselamu aknalaual värskeid maitsetaimi kasvatada, võib pimesi ja sageli hooletu eksperimenteerimine lõppeda taimede hävimisega. Üks lahendus on osta Eesti idufirma Click and Grow tark siseaed, mis kasvatab taimi ise, hoolitsedes nii valguse, niiskuse kui ka toitainete eest [56]. Teine variant on sarnane süsteem ise luua. Ise millegi sellise loomine on kindlasti hariduslikult kasulikum ja õpetlikum.

4.1. Projekti kirjeldus

Projekti käigus valmib seade, mis mõõdab mulla niiskust ja temperatuuri, õhuniiskust ja õhutemperatuuri, valguse kogust ning edastab need Adafruit IO süsteemi. Lisaks andmete

kogumisele suudab taimekasvatuse süsteem lülitada taimelampi ning omab automaatse kastmise süsteemi, mida saab ka juhtida Adafruit IO juhtpaneelist. Loodava projekti illustratsioon on joonisel 24.



Joonis 24. Loodava targa taimekasvatuse illustratsioon.

Süsteem kasutab ka IFTTT teenuse võimekust ning suudab edastada teateid omanikule näiteks liiga madalast veetasemest veepumba mahutis. Järgnevas punktis tutvustatakse komponente, mida sellise süsteemi loomiseks läheb vaja.

4.2. Vajaminevad komponendid

Loodavas projektis kasutatakse WeMos arendusplaadiga mitmeid andureid, seadmeid kui ka seadmete töölepanemiseks vajalikke lisakomponente. Komponente leiab ka Eestist, kuid kui projekti valmimisega ei ole väga kiire, soovitab autor raha kokkuhoiu mõttes pigem tellida need Hiinast, näiteks e-poest Aliexpress või Banggood. Tabelis 3 on toodud vajaminevad seadmed ning komponendid koos e-poe Aliexpress hinnaga ning hinnaga Eesti poodides:

Tabel 3. Projektis kasutatavad komponendid ja nende hinnad.

Nimetus	Hind Eestis	Hind Hiina e-poes
WeMos D1 Mini või Mini Pro	Kuni 18€ [59]	~7€ [28]
400 punktiga maketeerimislaud 2tk	12€ [57]	~3€ [58]
8 kanaliga multiplekser	~2,5€ [60]	~1,5€ [61]
Kahesuunaline loogika pingemuundur	~2,8€ [63]	~0,6€ [62]
WeMos relee laiendusplaat	9€ [65]	~3,6€ [64]
Mullaniiskuse andur	6€ [67]	~0,7€ [66]
Õhutemperatuuri ja –niiskuse andur DHT-22/AM2302 kolme viiguga	~18€ [69]	~4€ [68]
Veekindlas korpuses temperatuuriandur DS18b20	4,5€ [71]	~1€ [70]
Uputatav veepump tööpingega 3-5V	Kirjutamise hetkel ei leidnud	~4,5€ [72]
Voolik veepumba jaoks, näiteks Cristallo Extra AL mõõtudega 6/9mm	0,6€/m [73]	-
Ühendusjuhtmed (isa-isa tüüpi ja isa-ema tüüpi)	12€ [75, 76]	~2,3€ [74]
Maketeerimislaua ühendustraadid	7€ [78]	~1,6€ [77]
Erinevate takistite komplekt	6€ (150tk) [80]	~1,5€ (300tk) [79]
Transistor 2N2222	0,1€ [81]	~0,01€
Fototakisti	1€ [82]	~0,05€

Eelnevas tabelis kõik hinnad kokku liites tuleb projekti maksumus veidi üle 32€, kui tellida komponendid Hiinast. Eestist ostes läheb projekt maksma ligi 100€. Transistor ja fototakisti on otstarbekas pigem osta Eestist, seetõttu ei pidanud autor vajalikuks ka tabelis hinnale viidata. Hiina e-poed müüvad neid suurtes kogustes ja komplekti hind koos saatmisega tuleks umbes sama, mis Eestist üksikuna ostes. WeMos D1 Mini, multiplekser ja pingemuundur saabuvad

Hiinast selliselt, et viigud on plaadi külge jootmata jäetud. Sellisel juhul tuleb ise viigud plaadi külge joota.

4.3. Arendusplaadi puuduste kõrvaldamine

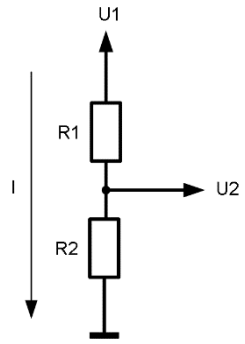
Punktis 2.2. võrreldi WeMos D1 arendusplaati Arduino UNO arendusplaadiga ning toodi välja, et WeMos D1 arendusplaadil on vaid üks analoogsisend ning sisendviigu maksimaalne pinge on 3,3 volti. Järgnevates punktides näidatakse, kuidas kasutada pingejagurit ja pingemuundurit, et WeMos arendusplaadiga ka 5V analoogandureid kasutada Samuti näidatakse, kuidas kasutada multiplekserit, et arendusplaadi külge rohkem kui üht analoogandurit ühendada.

4.3.1. Kõrgema pingega analoogandurite kasutamine

Soovides kasutada 5V analoogandurit WeMos arendusplaadiga, tuleb anduri väljundpinge muuta arendusplaadile sobivaks. Kõrgema pinge madalamaks tegemiseks võib kasutada näiteks pingejagurit. Pingejagur on Ohmi seadusel põhinev kahest takistist koosnev elektriahel, mille väljundpinge sõltub takistite takistuse suhtest (joonis 25) [83]. Pingejaguri valem on järgmine:

$$U_2 = U_1 * \frac{R_2}{R_1 + R_2}$$

Selles valemis U_2 on väljundpinge ehk juhe, mis läheb arendusplaadi analoogsisendisse, U_1 on sisendpinge ehk juhe, mis on ühendatud analooganduri külge ning R_1 ja R_2 on takistid.



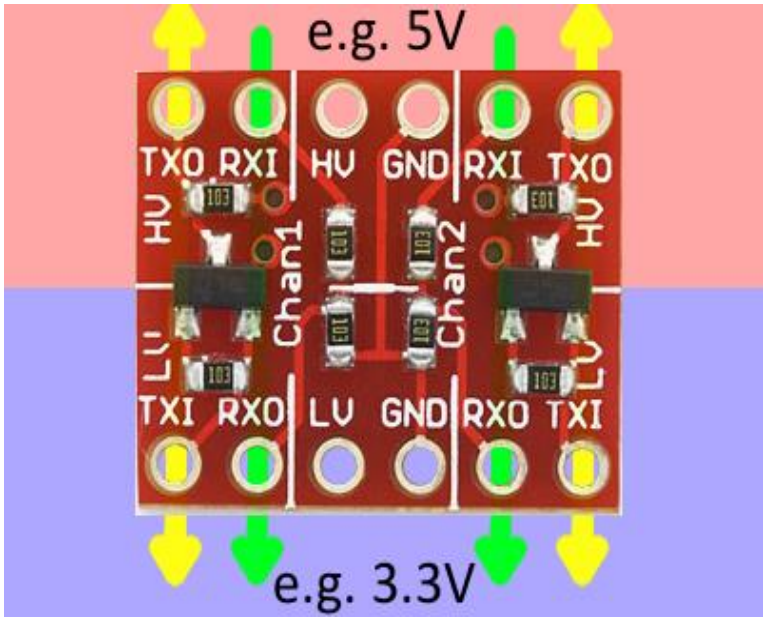
Joonis 25. Pingejaguri skeem [83].

Pingejagurit kasutades 5 voldi 3,3 voldiks muutmiseks sobiksid näiteks takistiteks R_1 = 10kΩ ja R_2 = 20kΩ.

Pinge muutmiseks ei pea ise pingejaguri ahelat kokku panema, vaid võib kasutada ka loogika pingemuundurit. Loogika pingemuundur on seade, mille ülesanne on muuta pinget. Pingemuundurid on sageli kahesuunalised ehk suudavad ka pinget kõrgemaks muuta. Näiteks osutub madalama pinge muutmine kõrgemaks vajalikuks, kui tahta WeMos arendusplaadiga juhtida näiteks adresseeritavat LED valgusriba ehk sellist riba, kus iga valgusdioodi värv on eraldi muudetav. Levinud WS2812 LED-i optimaalne sisendpinge on 5V [84] ja WeMos arendusplaadi 3,3V tööpingest jääb väheks.

Pingemuundureid on erinevaid ja seega tuleb iga toote puhul eraldi jälgida, kuidas selle kasutamine käib. Kuigi käesolevas töös valmivas lahenduses pinget kõrgemaks muuta ei ole tarvis, on autor siiski otsustanud kasutada pinge muutmiseks pingejaguri asemel kahesuunalist loogika pingemuundurit.

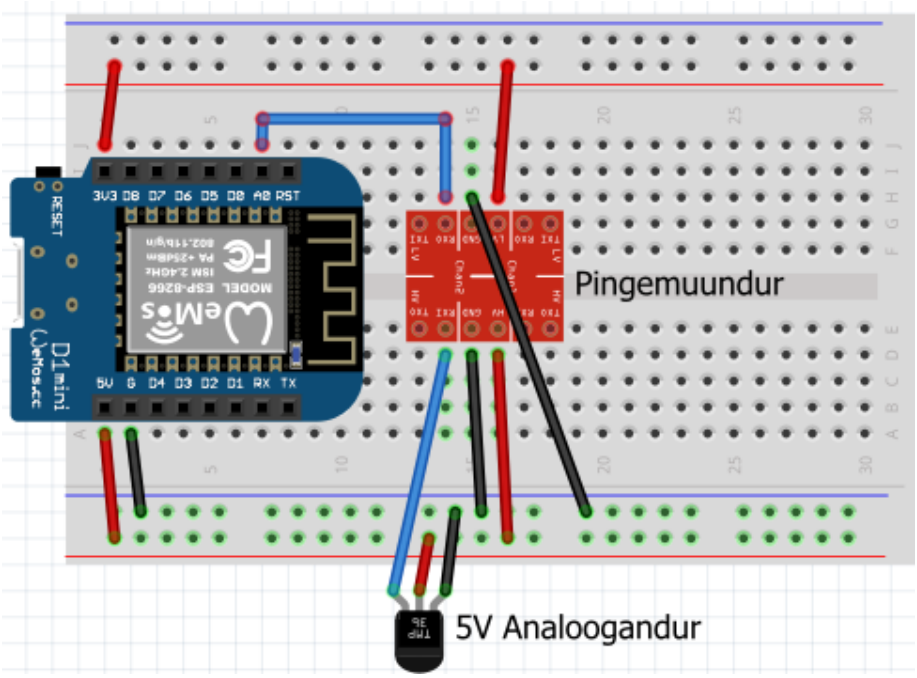
Töös kasutatav pingemuundur on odav Sparkfun nelja kanaliga kahesuunalise loogika pingemuunduri koopia, kus kaks kanalit on kahesuunalised ning kaks kanalit ühesuunalised, muutes ainult kõrgemat pinget madalamaks [30]. Joonisel 26 on pingemuundur pealtvaates ning joonisele on märgitud ühesuunalised kanalid rohelise noolega ning kahesuunalised kanalid kollase noolega.



Joonis 26. Sparkfun nelja kanaliga kahesuunaline pingemuundur [30].

Antud Sparkfun pingemuunduri HV tähisega viigu külge tuleb ühendada juhe arendusplaadi 5V viigust ning LV tähisega viigu külge juhe arendusplaadi 3V3 viigust. Samuti tuleb mõlemad

GND tähisega viigud ühendada arendusplaadi GND viigu ehk maanduse külge. Seejärel võib ühendada 5V analoogseadme näiteks RXI viigu külge ning omakorda ühendada juhtmega sama kanali RXO viik arendusplaadi analoogsisendiga. Joonisel 27 on kirjeldatud viisil ühendatud 5V analoogandur WeMos D1 Mini arendusplaadiga kasutades Sparkfun pingemuundurit.

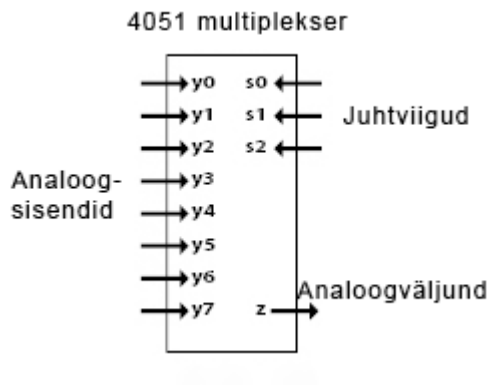


Joonis 27. Analoogandurі ühendamine kasutades pingemuundurit.

Järgmises punktis tutvustatakse multiplekseri tööpõhimõtet ning näidatakse, kuidas selle abil saab kasutada WeMos arendusplaadil mitut analoogandurit.

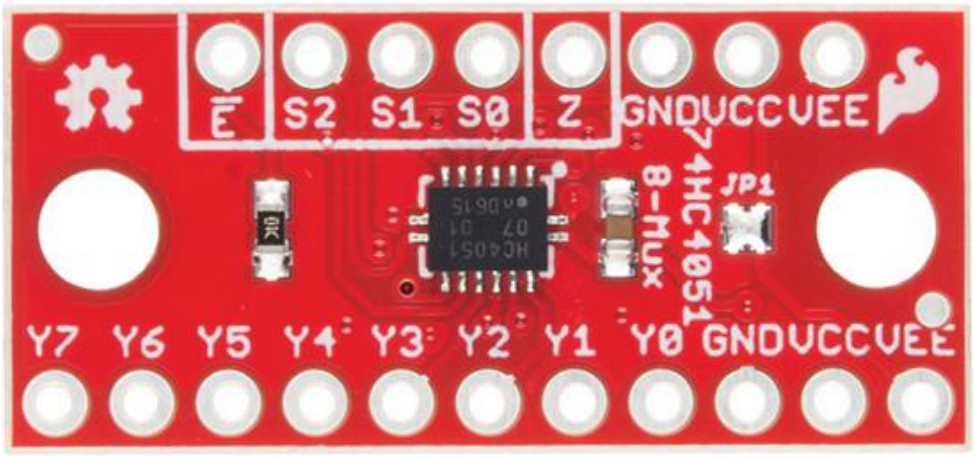
4.3.2. Multiplekseri kasutamine analoogandurite arvu suurendamiseks

Aeg-ajalt võib tekkida vajadus ühe arendusplaadi külge ühendada mitu analoogandurit. Ka käesoleva töö käigus valmivas lahenduses on kasutatud analoogandureid rohkem kui üks. Analoogsisendite arvu on võimalik suurendada kasutades multiplekserit. Multiplekser on seade, millel on mitu sisendit ja üks väljund ning juhtsignaali abil toimub juhtsignaali poolt määratud sisendi suunamine väljundisse. Käesolevas töös on kasutatud 8-kanaliga analoogmultiplekserit 4051 (joonis 28). Sellel multiplekseril on kaheksa analoogsisendit (tähised Y0-Y7 joonisel 28) ning üks väljund (tähis Z joonisel 28) ehk analoogseadmete ja –andurite arvu on võimalik suurendada kuni kaheksani.



Joonis 28. Multiplekser 4051 [85].

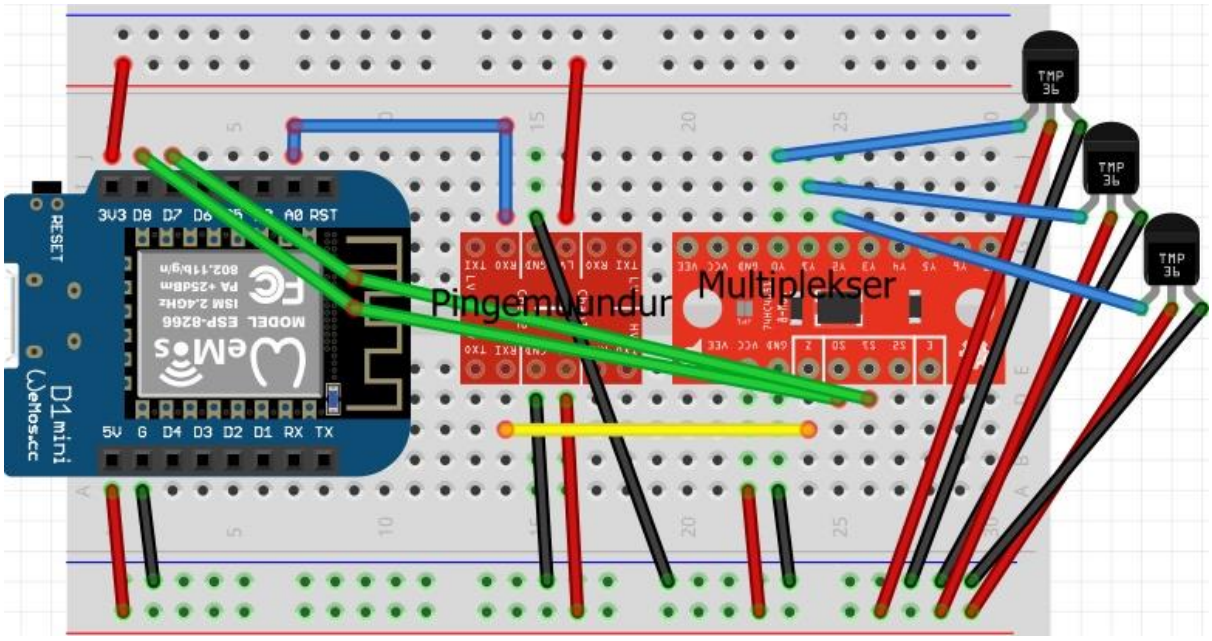
Multiplekser 4051 kasutab sobiva analoogsisendi valimiseks kolme juhtsignaali viiku S0, S1 ja S2, mis on ka joonisel 22 märgitud. Iga viik on ühendatud WeMos arendusplaadi digitaalväljundi viigu külge. Igale juhtsignaali viigule vastab oma number (S0=1, S1=2, S2=4) ning see number edastatakse multiplekserisse, kui arendusplaadi digitaalväljundist saadetakse vastavasse juhtsignaali viiku kõrge signaal ehk 1 [85], programmis on selleks meetod `digitalWrite(väljundviik, 1)`. Kolme signaali numbrid liites saame selle sisendi, mille multiplekser väljundisse suunab [85]. Näiteks kui soovitakse saada signaali analoogandurilt, mis on ühendatud viigu Y3 külge, siis tuleb määrata S0 ja S1 signaal kõrgeks, sest 1+2+0=3. Multiplekser loeb juhtsignaali viikude väärtuseid kui kahendsüsteemis arvu. Näiteks viigu Y3 väärtuse lugemiseks tuleb saata viikudesse S0 ja S1 signaal 1, sest kahendsüsteemis 011 on kümnendsüsteemis 3. Käesolevas töös on kasutatud Sparkfun multiplekserit (joonis 29).



Joonis 29. Sparkfun 4051 8 kanaliga multiplekser [86].

Kõik kolm juhtsignaali viiku ei pea olema arendusplaadiga ühendatud. Vaid kahe analoogseadme kasutamiseks piisab, kui ühendatakse arendusplaadiga vaid S1. Selliselt saab valida sisendiks viike Y0 või Y1. Kasutades kaht juhtsignaali viiku on võimalik juba valida nelja analoogseadme vahel ja paljudel juhtudel on see täiesti piisav. Selliselt üleliigsete juhtsignaali viikude ühendamata jätmisel jäävad arendusplaadil digitaalsisend-väljundviigud hõivamata ning neid saab kasutada teiste digitaalsignaali kasutavate seadmete jaoks.

Kuna paljud analoogandurid edastavad signaali vahemikus 0-5V, on mõistlik ühendada multiplekser jadamisi pingemuunduriga. Joonisel 30 on täiendatud joonist 27 multiplekseri ühendamisega pingemuunduri külge. Kuna planeeritav analoogandurite arv ei ületa nelja, on ühendatud arendusplaadiga vaid juhtsignaali viigud S1 ja S2. Samuti on joonisel näidatud kolme anduri näitel, kuidas ühendada andureid multiplekseri külge.



Joonis 30. Multiplekser ja pingemuundur ühendatud WeMos D1 Mini arendusplaadiga.

Joonisel 30 on jäetud ühendamata multiplekseri viigud Ē ning VEE. Viik Ē on mõeldud selleks, et multiplekserit sisse ja välja lülitada ning kui loodavas lahenduses see vajalik pole, võib viik ühendamata jääda [86]. Viik VEE on negatiivse pinge jaoks ja Sparkfun multiplekseril on see juba ühendatud maandusega (GND), seega eraldi ühendamist ei vaja [86].

Programmi koodis tuleb määrata viigud D7 ja D8 digitaalväljunditeks, kus D7 on ühendatud multiplekseri viigu S0 ning D8 multiplekseri viigu S1 külge nagu näidatud joonisel 30.

Kasutades funktsiooni `digitalWrite()` tuleks seejärel soovitud multiplekseri viigu lugemiseks viikude D7 ja D8 signaali väärtuseks määrata kas 1 või 0 sõltuvalt sellest, millise multiplekseri viigu väärtust soovitakse lugeda. Et ei peaks hakkama mõtlema, mis väärtuste kombinatsioone kasutama peaks, on Arduino programmeerimiskeeles olemas funktsioon `bitRead()`, mis teisendab etteantud arvu kahendsüsteemi ning tagastab kahendsüsteemis arvu biti väärtuse soovitud kohal [87]. Joonisel 31 on programm, mis loeb väärtuseid multiplekseri viikudelt Y0, Y1 ja Y2 ning väljastab need *Serial Monitor*'i.

```
#define S0            D7 // Multiplekseri viik S0
#define S1            D8 // Multiplekseri viik S1

int val = 0; // analoogviigu väärtuse muutuja

void setup() {
 Serial.begin (9600);
 // multiplekseri juhtsignaali viigud määrame väljundiks
 pinMode(S0, OUTPUT);
 pinMode(S1, OUTPUT);
}

void loop() {
 val = readMultiplexerValue(0); // loeme väärtuse multiplekseri viigult Y0
 Serial.println(val);           // ning väljastame selle Serial Monitori
 delay(1000);                   // ootame sekund
 val = readMultiplexerValue(1); // loeme väärtuse multiplekseri viigult Y1
 Serial.println(val);
 delay(1000);
 val = readMultiplexerValue(2); // loeme väärtuse multiplekseri viigult Y2
 Serial.println(val);
 delay(1000);
}
/*
* Funktsioon valitud multiplekseri viigu väärtuse leidmiseks
* Sisend: Viigu number, mille väärtust soovime
* Väljund: loetud täisarvuline analoogviigu väärtus
*/
int readMultiplexerValue(int pinNumber) {
 // määrame S0 viigu väärtuseks soovitud viigu biti väärtuse kohal 0
 digitalWrite(S0, bitRead(pinNumber,0));
 // ning viigu S1 väärtuseks soovitud viigu biti väärtuse kohal 1
```

```
 digitalWrite(S1, bitRead(pinNumber,1));
 int result = analogRead(A0); // loeme analoogviigu väärtuse
 return result; // tagastame selle
}
```

Joonis 31. Multiplekseri programm.

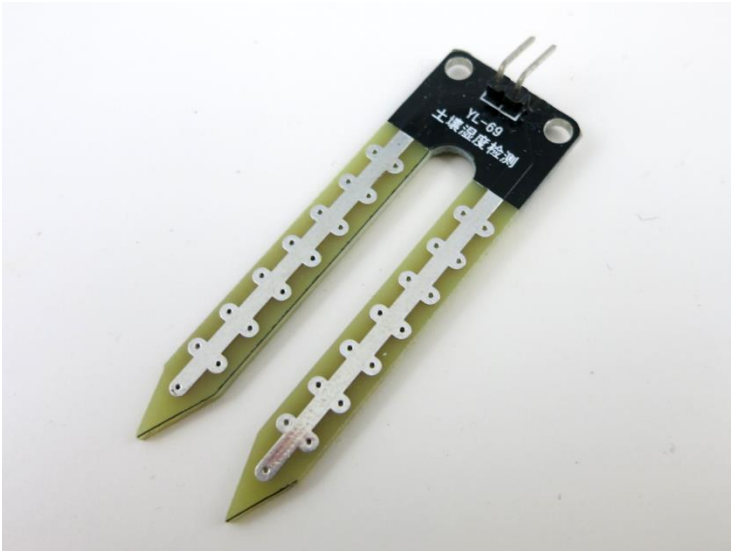
Järgmises punktis kirjeldatakse andureid ning seadmeid, mida kasutatakse loodavas automaatse taimekasvatuse projektis.

4.4. Projektis kasutatavad andurid ja seadmed

Käesolevas punktis kirjeldatakse projektis kasutatud andureid ja seadmeid ning selgitatakse nende tööpõhimõtet.

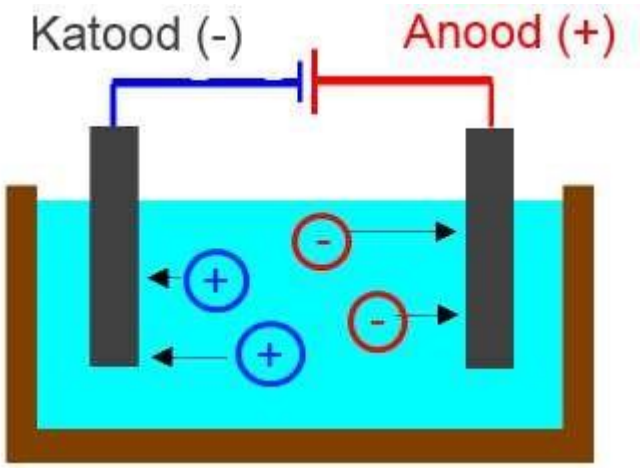
4.4.1. Mulla niiskuse andur

Mulla niiskuse mõõtmiseks kasutatakse loodavas projektis odavat analoogandurit, mis mõõdab mullas sisalduvat vett kasutades mulla elektritakistust. Valitud andur koosneb kahest osast. Anduri mullatakistust mõõtvaks osaks on kaheharuline hark (joonis 32), mis paigutatakse mulda.



Joonis 32. Mulla niiskuse anduri mulda käiv osa.

Hargi harusid nimetatakse elektroodideks ja neid läbib alalisvool ühelt elektroodilt teisele. Mida niiskem on muld, seda väiksem on mulla elektritakistus. Alalisvoolu korral osutub probleemiks elektri ühesuunaline liikumine, mis põhjustab elektrolüüsi ning selle tagajärjel toimub ühe elektroodi – anoodi – oksüdeerumine [88]. Joonisel 33 on näha elektronide liikumist elektroodide vahel vees.



Joonis 33. Elektroodid vees, anood loovutab elektrone ehk oksüdeerub [88].

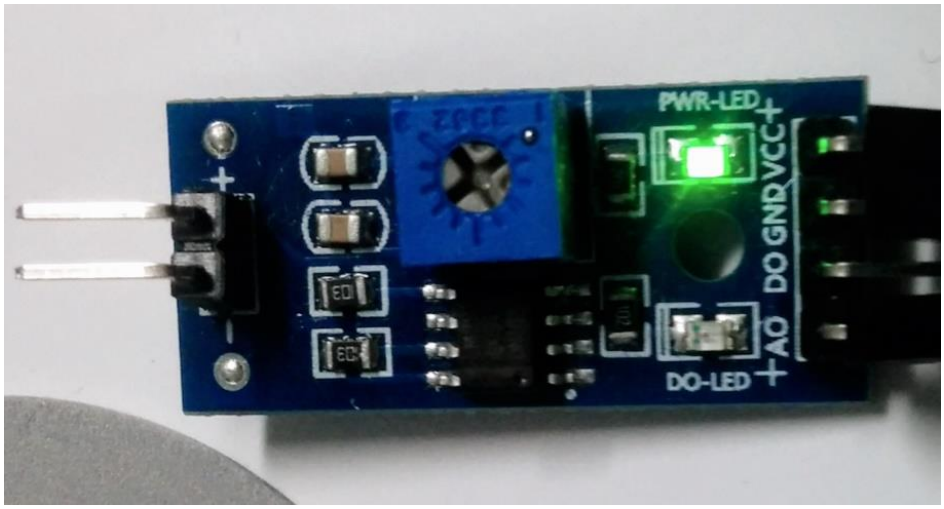
Oksüdeerumise käigus metall aeglaselt hävineb. Hiina odavate andurite elektroodid on väga õhukesest metallist ning ei kesta kaua. Joonisel 34 on näha kõigest kaks nädalat mullas olnud harki. Aeg-ajalt tuleks tõsta anduri hargi juhtmed ümber, et elektrivoolu suunda elektroodide vahel muuta. Selliselt kuluvad elektroodid ühtlasemalt. Samuti võiks elektroode aeg-ajalt puhastada, kuna oksiidikiht mõjutab elektrijuhtivust.

Joonis 34. Oksüdeerunud elektroodid.

Autor soovitab asendada Hiina anduri elektroodide asemel kasutada näiteks kaht naela või polti (joonis 35), sest need on paksemast metallist ning kestavad seega oluliselt kauem.

Joonis 35. Autori meisterdatud polt elektroodid.

Anduri loogikaplaadil (joonis 36) on neli viiku: sisendpinge (VCC), maandus (GND), digitaalväljund (DO) ja analoogväljund (AO). Sisendpinge võib olla nii 5V kui ka 3,3V. Digitaalväljund väljastab loogilise madala signaali, kui mulla niiskus on piirtasemest allpool ning loogilise kõrge signaali, kui niiskus on piiri ületanud. Piirtaseme reguleerimiseks on sinine nelinurkne potentsiomeeter. Plaadil on ka kaks LED-i. Valgusdiood tähisega PWR-LED põleb, kui plaadil on toide sees (näha ka joonisel 36) ning DO-LED, mis läheb piirtaseme ületamisel põlema.



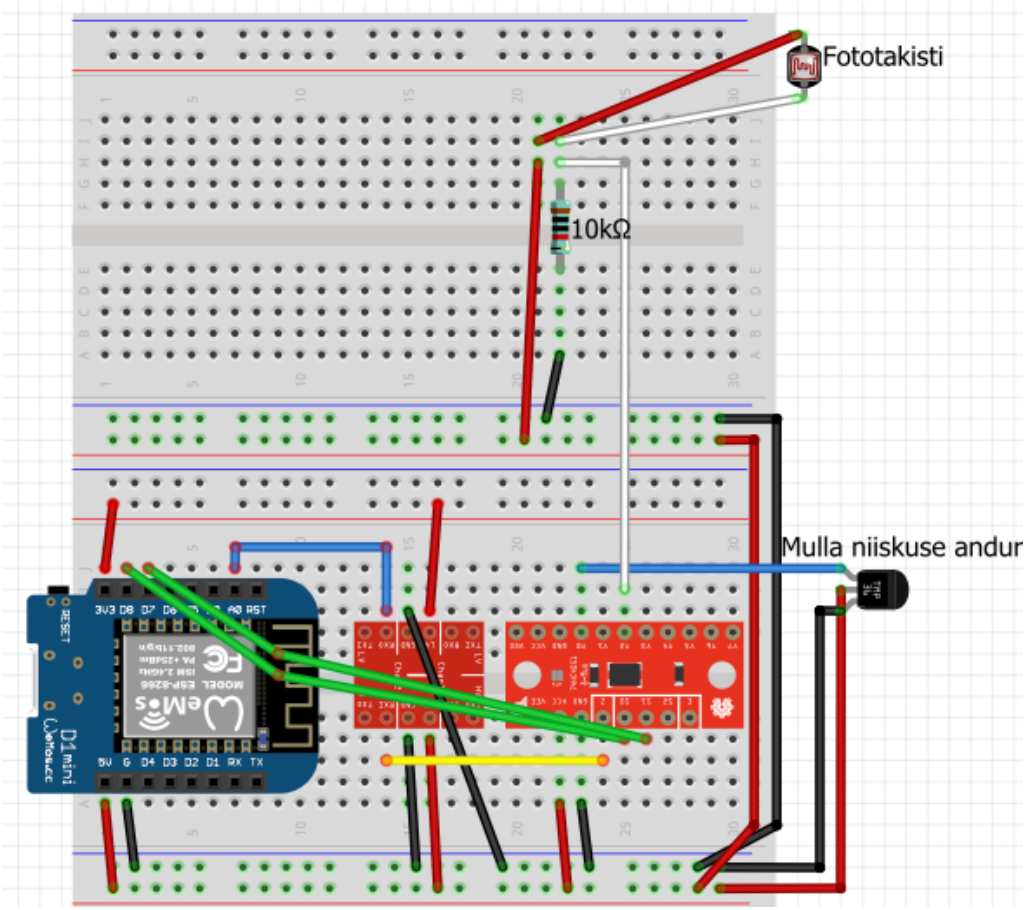
Joonis 36. Mulla niiskuse anduri loogikaplaat.

Käesolevas projektis kasutatakse anduri analoogväljundit täpse niiskuse näidu saamiseks ning ühendada võib selle näiteks multiplekseri Y0 viigu külge. Andur väljastab näiduks numbreid vahemikus 0 – 1023, kusjuures mida suurem on mulla veesisaldus, seda väiksem on number.

Mulla tegeliku niiskuse leidmiseks tuleks andur eelnevalt kalibreerida. Seda võib teha järgnevalt: kõigepealt tuleb vaadata, milline on anduri näit täiesti kuivas mullas ning seejärel milline on anduri näit märjas mullas [89]. Mulla niiskusest kalibreerimisel sõltub ka hilisem näidu täpsus, seega tasub kuiva mulla näidu võtmise eel mulda kuivatada kõrge temperatuuri juures ja aeg-ajalt segades, et võimalikult palju niiskust mullast eemaldada. Kui anduri elektroodid vahetada näiteks naelte vastu, tuleks andur uuesti kalibreerida. Samuti tuleks seda teha, kui andurit kasutatakse uue mullatüübi niiskuse mõõtmiseks [89].

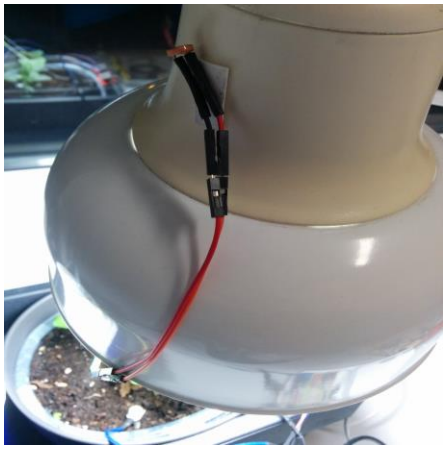
4.4.2. Fototakisti kasutamine valguse intensiivsuse mõõtmiseks

Fototakisti on muutuva takistusega seade, mille takistus sõltub seadmele peale langeva valguse intensiivsusest: mida intensiivsem on valgus, seda madalam on takistus [90]. Fototakisti kasutamiseks tuleb koostada punktis 4.3.1. kirjeldatud pingejaguri elektriahel (joonis 25), kus R1 on asendatud fototakistiga, sest fototakisti enda takistust otseselt arendusplaadiga mõõta ei saa vaid saab mõõta takistusest sõltuvat pinget [90]. Fototakisti skeemi koostamine maketeerimislaual on näidatud joonisel 37.



Joonis 37. Fototakisti ühendamine.

Joonisel on juba näha ka eelnevas punktis kirjeldatud mulla niiskuse anduri ühendamine ning juurde on lisatud teine maketeerimislaud. Autor soovitab fototakisti ühendada maketeerimislauaga kasutades pikemaid juhtmeid, et takisti oleks võimalik näiteks lambi külge panna (joonis 38). Selliselt saab mõõta keskkonna valguse intensiivsust nii, et lambi tekitatud valgus anduri näitu ei mõjuta.



Joonis 38. Fototakisti kahepoolse teibiga lambi külge kinnitatud.

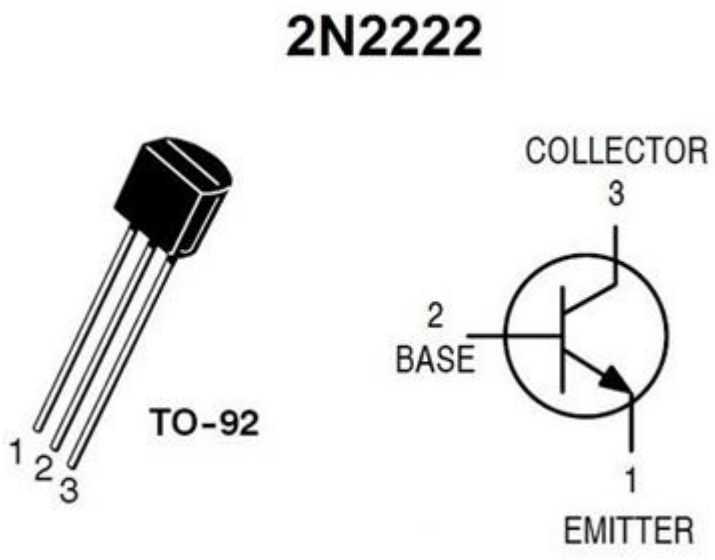
Järgmises punktis kirjeldatakse veepumba kasutamist ning veepumba mahutis oleva vee taseme indikaatori valmistamist.

4.4.3. Veepump ja veetaseme indikaator

Projektis kasutatav veepump (joonis 39) [72] on alalisvoolumootoriga pump, mis töötab pingevahemikus 3 - 5V ning vajab töötamiseks voolutugevust 100 milliamprit ja seetõttu töötab pump WeMos arendusplaadiga vajamata välist toiteallikat.

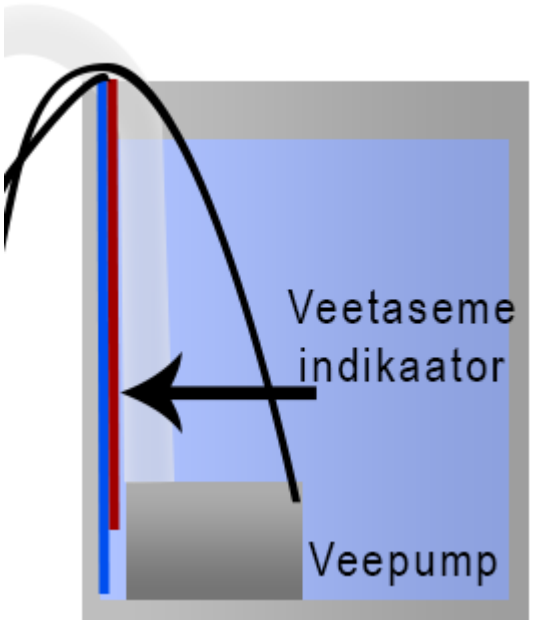
Joonis 39. Veepump [72].

Veepump suudab pumbata 1,2 – 1,6 liitrit vett minutis ja on uputatav ehk see paigutatakse veeanuma põhja. Veepumba mootor töötab, kui juhtmed on ühendatud arendusplaadi maandusega ja 3V3 või 5V pinge viiguga. Arendusplaadiga pumba juhtimiseks tuleb lisada mootori vooluahelasse lüliti. Selliseks otstarbeks sobib tavaline madala võimsusega elektriahela jaoks mõeldud NPN-tüüpi bipolaartransistor [91]. Näiteks transistor 2N2222, mida ka loodavas projektis kasutatakse. Transistoril on kolm viiku: emitter, baas ja kollektor. Baas ühendatakse arendusplaadi digitaalväljundiga, emitter on ühendatud maandusega ning kollektor veepumbaga. Baasi ja arendusplaadi digitaalväljundi vahele tuleb ühendada lisaks 1kΩ takisti, et piirata transistori baasile antavat voolutugevust ja sellega kaitsta transistorit [91]. Peab jälgima, et transistori ühendamisel ei ajaks segi kollektorit ja emitterit. Transistori 2N2222 viikude paigutus on näidatud joonisel 40.



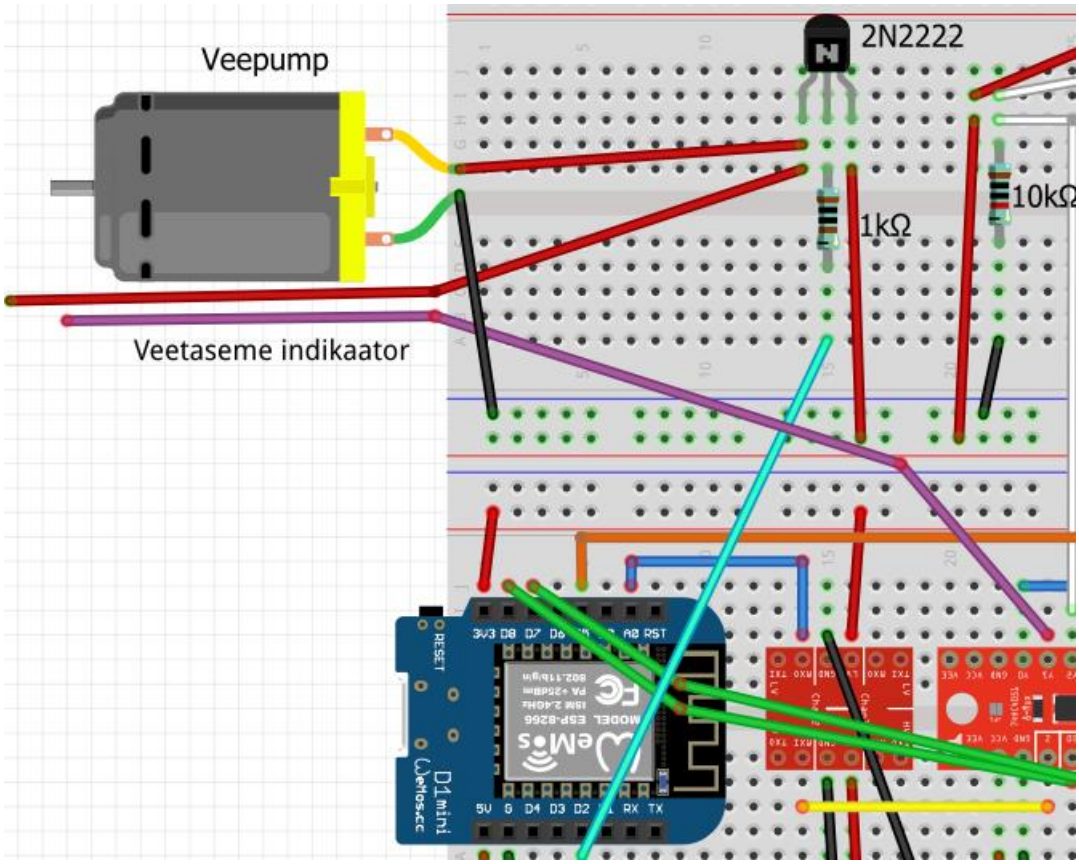
Joonis 40. Transistor 2N2222 viikude paigutus: 1 – emitter, 2 – baas, 3 – kollektor [92].

Lisaks veepumbale tuleks lisada ka pumba veemahutisse veetaseme indikaator, mis annaks märku madalast veetasemest ning ei laseks pumbal kuivalt töötada. Veetaseme indikaatorina võib kasutada kaht isoleeritud juhet, mis on paigutatud veeanumasse. Ühe juhtme ots on veeanuma põhjas ja teine soovitud minimaalse taseme juures. Indikaator töötab vee elektrijuhtivuse põhimõttel: kui veetase on piisavalt kõrge, on mõlemad juhtme otsad vees ning vooluahel suletud. Kui veetase langeb nii madalale, et üks juhtme ots enam vette ei ulatu, vooluring katkeb. Joonisel 41 on kujutatud indikaatori juhtmeid ning veepumpa veenõus.



Joonis 41. Pump ja indikaatori juhtmed veenõus.

Indikaatorit võib kasutada kui analoogandurit ning multiplekseriga selle näitu lugeda. Et elektrolüüsi võimalikult vähe toimuks, on mõistlik ühendada veetaseme indikaator veepumba vooluringiga selliselt, et indikaator kontrolliks veetaset ainult siis, kui pump sisse lülitatakse. Joonisel 42 on näidatud indikaatori ja veepumba ühendamine arendusplaadiga maketeerimislaual. Joonisel on näha, et transistori baas on ühendatud juhtmega digitaalviigu D2 külge ning veetaset loeb multiplekseri viik Y1.



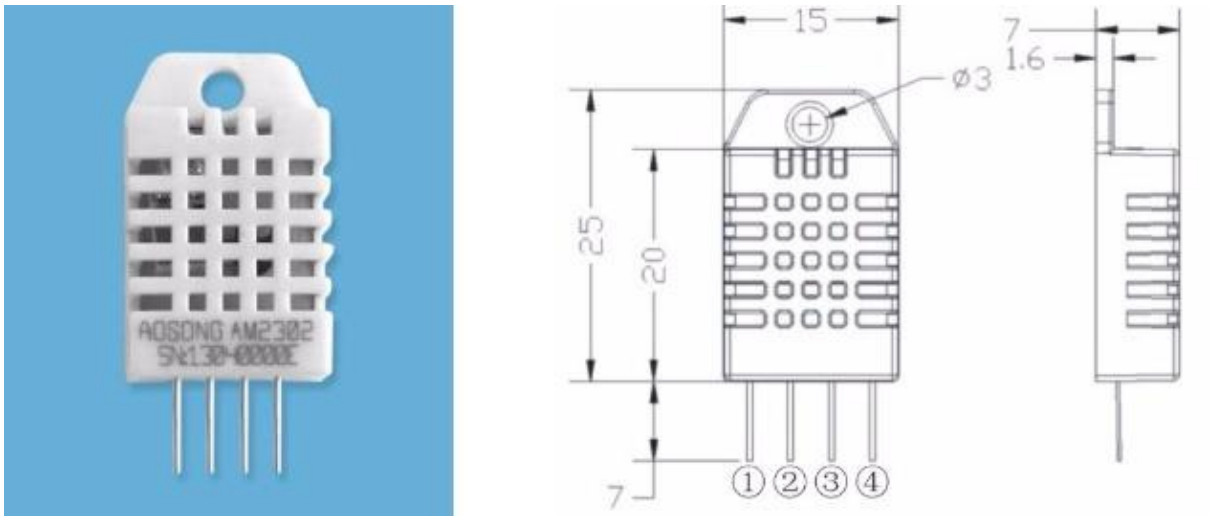
Joonis 42. Indikaatori ja veepumba skeem.

Järgmises punktis ühendatakse plaadi külge andur õhuniiskuse ja –temperatuuri mõõtmiseks.

4.4.4. Õhuniiskuse- ja temperatuuriandur

Andureid, mis mõõdavad õhuniiskust ja –temperatuuri, on erinevaid. Andurid erinevad nii täpsuse, suuruse kui ka näiteks arendusplaadiga suhtlemise viisi poolest. Käesolevas töös kasutatakse õhuniiskuse ja –temperatuuri mõõtmiseks üsna levinud digitaalandurit DHT-22,

teise nimega AM2302 (joonis 43). Andur kasutab arendusplaadiga suhtlemiseks *single-bus* andmesidet ehk ühe juhtme abil suudab andur nii andmeid vastu võtta kui arendusplaadile saata [93]. DHT-22 edastab andmeid niiskuse või temperatuuri kohta ainult siis, kui arendusplaat seda küsib.



Joonis 43. Andur DHT-22 (AM2302). Paremal mõõtmed millimeetrites [93].

Anduril on neli väljaviiku, millest joonisel 43 vasakult kolmas ei tee midagi [93]. Sageli müüakse seda andurit kolme viiguga trükkplaadil, millele on lisatud *pull-up* takisti toite ja andmete siini vahele. *Pull-up* takisti ülesanne *single-bus* seadmete juures on eemaldada ühenduse oleku määramatus hetkel kui andmeid ei vahetata, viies ühendus anduri ja arendusplaadi vahel kindlasse, antud juhul kõrgesse olekusse [93]. Takisti toitejuhtme ja andmete siini vahel muudab oleku andmete siini kõrgeks. Nii teab andur, et ühendus on vaba ning on valmis uut signaali vastu võtma [93]. DHT-22 üsna aeglane andur, suutes andmeid edastada kahesekundiliste intervallidega [93].

DHT-22 on kontrollitud keskkonnas kalibreeritud ja testitud ning suudab mõõta temperatuuri ja niiskust üsna suure täpsusega: temperatuuri täpsus on kuni ±0.5℃ ja niiskuse mõõtetäpsus ±2% [93]. Temperatuurivahemik, mida andur suudab mõõta, on -40 - +80℃.

Arendusplaadi programmeerimiseks DHT-22 andurit kasutama tuleb eelnevalt Arduino IDE-s paigaldada kaks teeki: *"Adafruit Unified Sensor"* ja *"DHT sensor library"*. Kui teegid on paigaldatud, on kõige lihtsam viis anduri testimiseks avadada näidisprogramm nimega *DHTtester*, valides menüüribalt *"File"* → *"Examples"* → *"DHT sensor library"* →

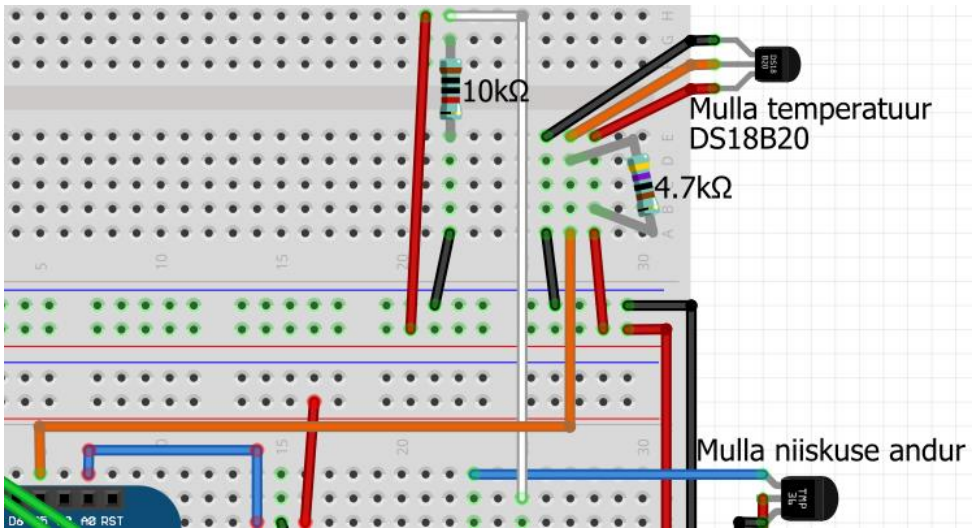
“DHTtester”. Näidisprogramm on põhjalike inglise keelsete kommentaaridega ning tutvustab anduri ja selle teegi erinevaid funktsioone.

4.4.5. Temperatuuriandur mulla temperatuuri mõõtmiseks

Mulla temperatuuri mõõtmiseks kasutatakse projektis digitaaltermomeetrit DS18b20, mida müüakse ka veekindlas metallist korpuses, sobides ideaalselt mulla sisse paigaldamiseks (joonis 44).

Joonis 44. DS18b20 veekindlas korpuses [71].

Sarnaselt andurile DHT-22 kasutab ka DS18b20 arendusplaadiga suhtlemiseks *single-bus* ühendust [94]. Andur DS18b20 suudab mõõta temperatuuri vahemikus -55 – +125℃ ning selle täpsus vahemikus -10 – +80℃ on ±0,5℃ [94]. Igal DS18b20 anduril on unikaalne seerianumber ja seetõttu saab vaid üht arendusplaadi ühe digitaalviiku kasutades juhtida mitut andurit [94]. Nagu DHT-22 puhul, kasutatakse ka DS18b20 ühendamiseks *pull-up* takistit. Selle takistus peab olema 4,7kΩ ning takisti ühendatakse samuti toitepinge ning andmete juhtme vahele [94]. Joonisel 45 on näidatud, kuidas on ühendatud DS18b20 andur loodavas projektis.

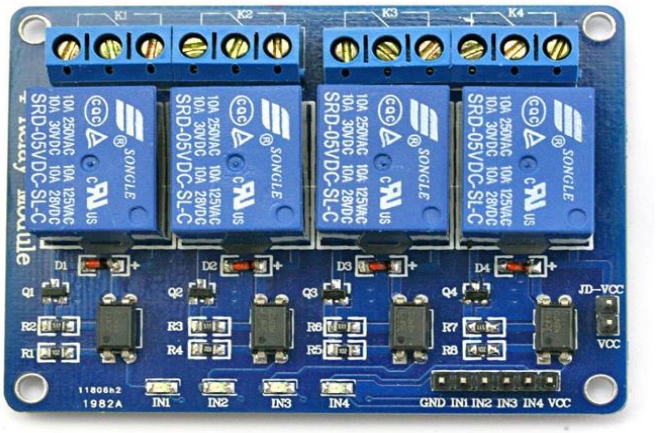


Joonis 45. DS18b20 ühendamine arendusplaadiga maketeerimislaual.

DS18b20 anduri kasutamiseks tuleb installeerida eelnevalt kaks teeki: *"OneWire"* ja *"DallasTemperature"*. Kui teegid on installeeritud, leiab näiteprogramme anduri kasutamiseks *"DallasTemperature"* näidete alt näiteks nimedega *Simple* ja *Single*.

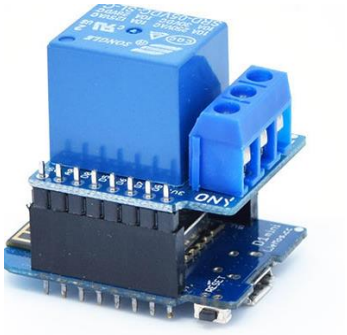
4.4.6. Relee vahelduvvooluseadmete lülitamiseks

Relee on seade, mis toimib nagu elektrooniline lüliti ning sobib hästi suure vooluga ahelate lülitamiseks [95]. Relee sobib seega hästi näiteks vooluvõrguseadmete lülitamiseks arendusplaadi abil. Loodavas projektis kasutatakse releed selleks, et juhtida vooluvõrku ühendatavat taimelampi. Arendusplaatide jaoks toodetakse mooduleid releedega. Releede arv moodulil võib olla erinev, näiteks joonisel 46 on näha nelja releega moodulit.



Joonis 46. Nelja releega moodul [96].

Releemooduli valikul tuleb olla tähelepanelik. Paljud releemoodulid vajavad lülituseks juhtsignaalilt 5V pinget, WeMos arendusplaadi digitaalväljundi pinge on aga 3,3V. Võib juhtuda, et selline moodul hakkab tööle, kuid kindlam on hankida moodul, mis vajab töötamiseks 3,3V pinget. Kui piisab vaid ühest releest, siis selle jaoks on ideaalne spetsiaalselt WeMos D1 Mini ja Mini Pro tarbeks loodud relee laiendusplaat (joonis 47).

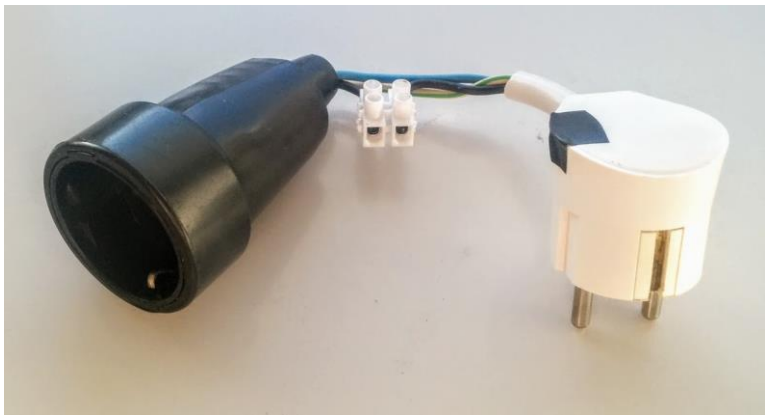


Joonis 47. Relee laiendusplaat WeMos D1 Mini külge ühendatud [97].

WeMos relee laiendusplaat kasutab digitaalviiku D1. Sellega tuleb arvestada, kui plaanitakse kasutada mõnda I^2C protokolli kasutavat seadet nagu ekraan või mõned digitaalandurid, sest WeMos arendusplaadi viigud D1 ja D2 on mõeldud ka vastavalt I^2C protokolli taktsignaali (SCL) ning andmesignaali (SDA) jaoks [26]. Näiteks ei saa I^2C protokolli kasutavat WeMos OLED ekraani laiendusplaati ja relee laiendusplaati samaaegselt kasutada. Relee tuleks sellisel juhul ümber ühendada mõne teise digitaalviigu külge.

Relee ühendamiseks vooluvõrguga on relee küljes kolm ühendusklemmi. Keskmine klemm on ühine ning äärmised määravad vastavalt tähisele ära, kas relee on normaalolekus suletud (NC) või avatud (NO). Normaalolekus avatud režiim tähendab seda, et lüliti on avatud, kui relee sisendsignaal on madal ning suletud, kui releele saadetakse arendusplaadilt kõrge signaal. Normaalolekus suletud režiim toimib vastupidiselt.

Relee juhtmega ühendamist võib lahendada mitmel viisil. Autor tegi relee ühendamiseks lühikese vahejuhtme (joonis 48), mis koosneb maandusklemmidega pistikust ja pistikupesast, lühikesest kolme soonega elektrijuhtmest ning klemmliistust relee juhtmete ühendamiseks.



Joonis 48. Juhe relee ühendamiseks.

Tüüpiline elektrijuhe koosneb kolmest soonest. Kollase roheliste triipudega juhe on maandusjuhe, sinine on nulljuhe ning pruun faasijuhe. Kahe soonega juhtmetel puudub maandusjuhe.

Vooluvõrgu lülitid, ka releed tuleks alati ühendada faasijuhtme vahele [98]. Ka joonisel 48 näidatud juhtme seinakontakti ühendamisel tuleb veenduda, milline pistikupesa auk on faasijuhtme jaoks ja pistik õiget pidi pessa panna. Kõige lihtsam on faasijuhet kontrollida pingeindikaator kruvikeerajaga, mis annab kontaktil faasijuhtmega sellest tulukesega märku [99]. Releed ei tohi ühendada seinakontakti enne, kui ollakse veendunud iseenda ohutuses. Kindlasti tuleks vaadata, et ühtegi lahtist juhtmeotsa või isoleerimata elektrijuhet kusagil ei oleks. Kui relee on ühendatud vooluvõrguga, tuleks vältida ohutuse mõttes relee korpuse puutumist.

Relee lülitamise programmeerimine on lihtne: kasutades WeMos relee laiendusplaati, toimub relee lülitamine funktsiooniga `digitalWrite(D1, HIGH)` või `digitalWrite(D1, LOW)`. Järgmises punktis näidatakse, kuidas programmeerida arendusplaati, et ühendatud anduritelt saadud andmeid Adafruit IO süsteemi saata ning pumpa ja releed lülitada.

4.5. Andurite ja seadmete programmeerimine

Käesolevas punktis näidatakse, kuidas arendusplaati programmeerida kirjeldatud andureid ja seadmeid kasutama ning kuidas seadistada Adafruit IO juhtpaneeli. Järgnevates punktides täiendatakse varem loodud programmi, mis ühendab arendusplaadi Adafruit IO-ga.

4.5.1. Analoogandurid

Analoogandurite lugemise näidisprogramm on kirjeldatud juba varem punktis 4.3.2. Näidisprogrammi read, välja arvatud funktsiooni `loop()` sisu, tuleks nüüd kopeerida loodava projekti programmi. Samuti on mõistlik programmis defineerida muutujad andurite viikude väärtustega (joonis 49), et edaspidi multiplekseri funktsiooni kasutamine segadust ei tekitaks.

```
int soilMUXPin = 0; // mulla niiskus Y0
int waterMUXPin = 1; // veetase Y1
int lightMUXPin = 2; // valgus Y2
```

Joonis 49. Analoogandurite jaoks defineeritud muutujad.

Siis saab `readMultiplexerValue()` funktsiooni parameetriks anda numbri asemel vastava muutuja nime, näiteks `readMultiplexerValue(soilMUXPin)` mulla niiskuse mõõtmiseks. Selliselt on programm arusaadavam. Samuti kui peaks multiplekseril andureid ümber tõstma, ei pea muutma numbrit igas funktsiooni väljakutses vaid piisab, kui muuta ära muutuja väärtus.

4.5.2. Õhuniiskuse- ja temperatuuriandur

Alustuseks tuleks importida programmi teek nimega *"DHT sensor library"*. Seejärel tuleb defineerida joonise 50 eeskujul anduri digitaalviik, anduri objekt ning abimuutujad.

```
// DHT-22
#define DHTPIN      D4 // DHT-22 on viigul D4

DHT dht(DHTPIN, DHT22); // anduri objekt, tüübiks DHT22
float previousTemp = 0; // temperatuuri salvestamiseks
float previousHumidity = 0; // õhuniiskuse salvestamiseks
```

Joonis 50. DHT-22 jaoks defineeritavad muutujad.

Funktsioonis `setup()` tuleb määrata anduri viik sisendiks reaga `pinMode(DHTPIN, INPUT)` ja seejärel luua funktsioon või funktsioonid, mis tagastaks kas õhuniiskuse või –temperatuuri. Autor kirjutas selle jaoks kaks funktsiooni: üks, mis tagastab õhutemperatuuri ning teine, mis tagastab õhuniiskuse (joonis 51).

```
float readAirTemperature() {
 // anduri funktsioon readTemperature() tagastab ujukomaarvu (float).
 float temp = dht.readTemperature();
 // isnan funktsioon tagastab tõeväärtuse „true“, kui tegemist pole numbriga
 if(isnan(temp)) {
   // kui temperatuurinäitu ei õnnestu saada, kasutame viimast salvestatud temperatuuri
   temp = previousTemp;
 } else {
   previousTemp = temp; // salvestame uue temperatuuri
 }
 Serial.print(„Õhutemperatuur: „);
 Serial.print(temp);
 Serial.println(„ *C“);
 return temp; // tagastame temperatuuri näidu
}
float readAirHumidity() {
 float humidity = dht.readHumidity();
 if(isnan(humidity)) {
   humidity = previousHumidity;
 } else {
   previousHumidity = humidity;
 }
 Serial.print(„Õhuniiskus: „);
 Serial.print(humidity);
 Serial.println(„ %“);
 return humidity;
}
```

Joonis 51. Funktsioonid õhutemperatuuri ja –niiskuse lugemiseks.

Eraldi loetakse näite sellepärast, et need tuleb hiljem ka Adafruit IO süsteemi eraldi saata.

4.5.3. Mulla temperatuuri andur

Mulla temperatuuri anduri kasutamiseks tuleb eelnevalt importida punktis 4.4.5. välja toodud teegid *“OneWire”* ja *“DallasTemperature”*. Seejärel, kasutades teegi *“DallasTemperature”* näiteprogrammi, tuleb enda programmi lisada kõigepealt joonisel 52 näidatud read.

```
// DS18b20
#define ONE_WIRE_BUS  D3  // DS18B20 viik
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

Joonis 52. DS18b20 anduri kasutamiseks vajalikud read.

Temperatuurianduri kasutamiseks tuleb anda `DallasTemperature` objektile alustamise käsk ehk funktsiooni `setup()` tuleb lisada rida `sensors.begin()`. Seejärel võib näitekoodi eeskujul teha uue funktsiooni, mis tagastaks anduri temperatuuri näidu (joonis 53).

```
float readSoilTemperature() {
 sensors.requestTemperatures(); // Küsime anduritelt temperatuure
 Serial.print(„Mullatemperatuur: „);
 float temp = sensors.getTempCByIndex(0); // valime esimeselt andurilt temperatuuri
 Serial.println(temp);
 return temp; // tagastame temperatuuri
}
```

Joonis 53. Funktsioon mulla temperatuuri näidu saamiseks.

Järgmises punktis näidatakse, kuidas eelpool kirjeldatud funktsioone kasutades andurite näite Adafruit IO süsteemi saata.

4.5.4. Andurite ühendamine Adafruit IO-ga

Andurite näitude lugemisel ja Adafruit IO süsteemi edastamisel tuleb arvestada sellega, et süsteem suudab andmeid vastu võtta piiratud kiirusel. Iga anduri ja seadme jaoks tuleks luua eraldi kanal ja anda kanalile tähenduslik nimi (joonis 54).

```
// subscribe kanal on sõnumite vastuvõtmiseks
const char* subscribeRelay = „m2rtt/feeds/taim-relay“; // relee
const char* subscribePump = „m2rtt/feeds/taim-water-pump“; // veepump
// publish kanali kaudu saadab arendusplaat sõnumeid
const char* publishWater = „m2rtt/feeds/taim-water-level“; // veetase
const char* publishSoilMoisture = „m2rtt/feeds/taim-soil-moisture“; // mulla niiskus
const char* publishSoilTemp = „m2rtt/feeds/taim-soil-temperature“; // mulla temp
const char* publishTemp = „m2rtt/feeds/taim-air-temperature“; // õhutemp
```

```
const char* publishHumidity = „m2rtt/feeds/taim-air-humidity“; // õhuniiskus
const char* publishPump = „m2rtt/feeds/taim-water-pump“; // veepump
const char* publishLightLevel = „m2rtt/feeds/taim-light-level“; // valgus
```

Joonis 54. Kanalite nimed, “m2rtt” on autori kasutajanimi Adafruit IO-s.

Andurite näidu lugemiseks võib teha eraldi funktsiooni, mis loeb andurite näitusid kindlas järjekorras ning edastab neid õigetesse kanalitesse. Näide, kuidas selline funktsioon võiks välja näha, on koos vajalike muutujatega joonisel 55.

```
Unsigned long sensorsPreviousMillis = 0; // aeg, mil viimati andurit loeti
const long sensorsInterval = 4000;    // anduri lugemise intervall
int count = 0; // alustame andurite lugemist 0-st
int totalCount = 4; // kokku loeme pidevalt viit näitu, null kaasaarvatud

void readSensors() {
 // hetkeaeg
 unsigned long currentMillis = millis();
 // kui piisavalt aega on möödas ehk hetkeaeg – viimane aeg >= lugemise intervall
 // siis loeme count muutuja numbrile vastava anduri näidu
 if(currentMillis – sensorsPreviousMillis >= sensorsInterval) {
   // määrame hetkeaja viimaseks lugemise ajaks
   sensorsPreviousMillis = currentMillis;
   int analog; // analoogandurite näitude muutujad
   float digital; // digitaalandurite näidud
   switch(count) {
     // esimesena loeme mullaniiskuse näidu
     case 0:
       analog = readMultiplexerValue(soilMUXPin);
       // 1023 – näit, sest anduri näit on ümberpööratud:
       // niiskema mulla puhul väiksem number
       dtostrf(1023-analog, 2, 2, msg); // kopeerime numbrilise väärtuse msg muutujasse
       client.publish(publishSoilMoisture, msg); // edastame väärtuse õigesse kanalisse
       break; // break viib switch() funktsioonist välja
     case 1: // valguse lugemine
       analog = readMultiplexerValue(lightMUXPin);
       dtostrf(analog, 2, 2, msg);
       client.publish(publishLightLevel, msg);
       break;
     case 2: // õhutemperatuuri lugemine
       digital = readAirTemperature();
```

```
        dtostrf(digital, 2, 2, msg);
        client.publish(publishTemp, msg);
        break;
      case 3: // õhuniiskus
        digital = readAirHumidity();
        dtostrf(digital, 2, 2, msg);
        client.publish(publishHumidity, msg);
        break;
      case 4: // mulla temperatuur
        digital = readSoilTemperature();
        dtostrf(digital, 2, 2, msg);
        client.publish(publishSoilTemp, msg);
        break;
    }
    // kui loetava anduri number oli väiksem kui andurite koguarv,
    // suurendame numbrit, et lugeda järgmise anduri näitu
    if(count < totalCount) {
      count++;
    } else {
      // vastasel juhul alustame otsast peale
      count = 0;
    }
  }
}
```

Joonis 55. Funktsioon andurite näitude lugemiseks ja edastamiseks anduri kanalisse.

Seejärel tuleb lisada loop() funktsiooni rida `readSensors()` andurite pidevaks lugemiseks.

4.5.5. Veepump ja veetaseme andur

Veepumba jaoks tuleb defineerida viik, mille külge pump on ühendatud ning määrata see funktsioonis `setup()` väljundiks käsuga `pinMode(PUMPPIN, OUTPUT)`. Pumba kasutamise jaoks on autor teinud eraldi funktsiooni `handleWaterPump()` (joonis 56).

```
// veepump
#define PUMPPIN       D5
bool pumpActivated = false; // algkäivitusel pump ei tööta
unsigned long pumpPreviousMillis = 0;
const long pumpInterval = 4000; // intervall, mitu sekundit pump järjest töötab
```

```
// piirid pumba jaoks, kui Adafruit IO süsteem peaks alt vedama
int waterLevelTreshold = 50; // umbes 60 on siis kui on vees, <10 siis kui ei ole
int soilMoistureMax = 900; // maksimaalne mulla niiskus. Üle selle, siis pump ei tööta

void handleWaterPump() {
 // kontrollime, et veetase oleks piisav ning muld ei oleks juba liiga niiske
 if (readMultiplexerValue(waterMUXPin) < waterLevelTreshold || 1023 –
readMultiplexerValue(soilMUXPin) >= soilMoistureMax) {
   // veetase on liiga madal, siis saadame veetaseme kanalisse sõnumi vastava tekstiga
   if(readMultiplexerValue(waterMUXPin) < waterLevelTreshold) {
     client.publish(publishWater, „LOW“);
     Serial.println(„Water level LOW“);
   } else {
     // kui veetase on normis, saadame vastava teate
     // ja järelikult on muld liiga niiske
     client.publish(publishWater, „NORMAL“);
     Serial.println(„Soil too moist“);
   }
   // lülitame pumba välja
   digitalWrite(PUMPPIN, LOW);
   pumpActivated = false;
   Serial.println(„Turning pump off: limits exceeded“);
   delay(200);
   // saadame pumba kanalisse sõnumi OFF
   client.publish(publishPump, „OFF“);

 }
 unsigned long currentMillis = millis();
 // kui pump töötanud piisavalt kaua, lülitame selle välja
 if(currentMillis – pumpPreviousMillis >= pumpInterval) {
   pumpPreviousMillis = currentMillis;
   pumpActivated = false;
   if(readMultiplexerValue(waterMUXPin) >= waterLevelTreshold) {
     client.publish(publishWater, „NORMAL“);
   }
   digitalWrite(PUMPPIN, LOW);
   Serial.println(„Turning pump off: time up“);
   client.publish(publishPump, „OFF“);
 }
}
```

Joonis 56. Funktsioon veepumba jaoks.

Eelmisel joonisel toodud funktsiooni väljakutsumine toimub funktsioonis `loop()` siis, kui pump on aktiivne ehk muutuja `pumpActivated` on tõene (joonis 57).

```
if (pumpActivated) {
   handleWaterPump();
 }
```

Joonis 57. Pumba funktsiooni väljakutsumine funktsioonis `loop()`.

Arendusplaadi programm on peaaegu valmis, jäänud on vaid kanalitesse saabunud sõnumite haldamine funktsiooniga `callback()`, mida kirjeldati punktis 3.4.1. Kõigepealt on vaja programm panna kuulama varem defineeritud kanaleid. Selle jaoks täiendame funktsiooni `reconnect()`, asendades käsu `client.subscribe(subscribeKanal, 1)` käskudega `client.subscribe(subscribePump, 1)` ja `client.subscribe(subscribeRelay, 1)`. Seejärel täiendame funktsiooni `callback()` joonisel 58 näidatud viisil.

```
// relee kanal
 if(teema == subscribeRelay){
     // sõnumi ON puhul lülitame relee sisse
     if (message == „ON“) {
       Serial.println(„relay on“);
       digitalWrite(RELAYPIN, HIGH);
     }
     // sõnumi OFF puhul välja
     if (message == „OFF“) {
       Serial.println(„relay off“);
       digitalWrite(RELAYPIN, LOW);
     }
 }
 if(teema == subscribePump){
     if (message == „ON“) {
       Serial.println(„pump on“);
       // pumba töö alustamise aeg
       pumpPreviousMillis = millis();
       // pump tööle ja aktiivseks
       digitalWrite(PUMPPIN, HIGH);
       pumpActivated = true;
     }
     if (message == „OFF“) {
```

```
      Serial.println(„pump off“);
      // lülitame pumba välja ja teeme mitteaktiivseks
      digitalWrite(PUMPPIN, LOW);
      pumpActivated = false;
    }
}
```

Joonis 58. Funktsiooni `callback()` lisatavad read.

Defineerimata on veel relee viik. Selle jaoks tuleb lisada programmi algusesse rida

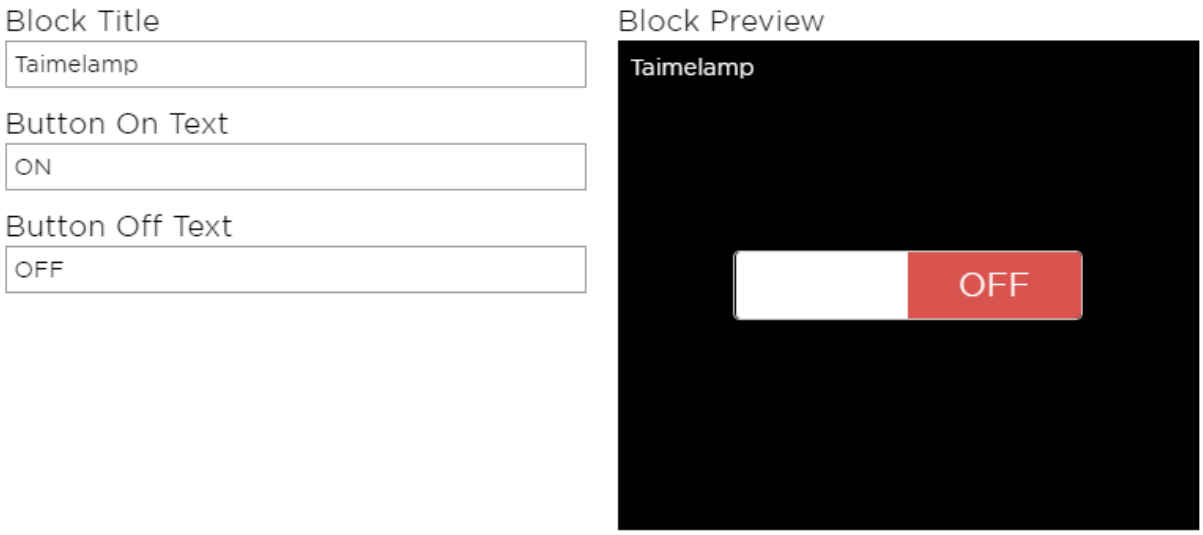
```
#define RELAYPIN            D1
```

Programm on nüüd valmis ning kui andurid ja seadmed on ühendatud maketeerimislaual, võib programmi arendusplaadile laadida. Projekti skeemi leiab lisast 1 ning valminud programmi kood on lisas 2. Järgmiseks on vaja seadistada Adafruit IO süsteem.

4.6. Adafruit IO seadistamine

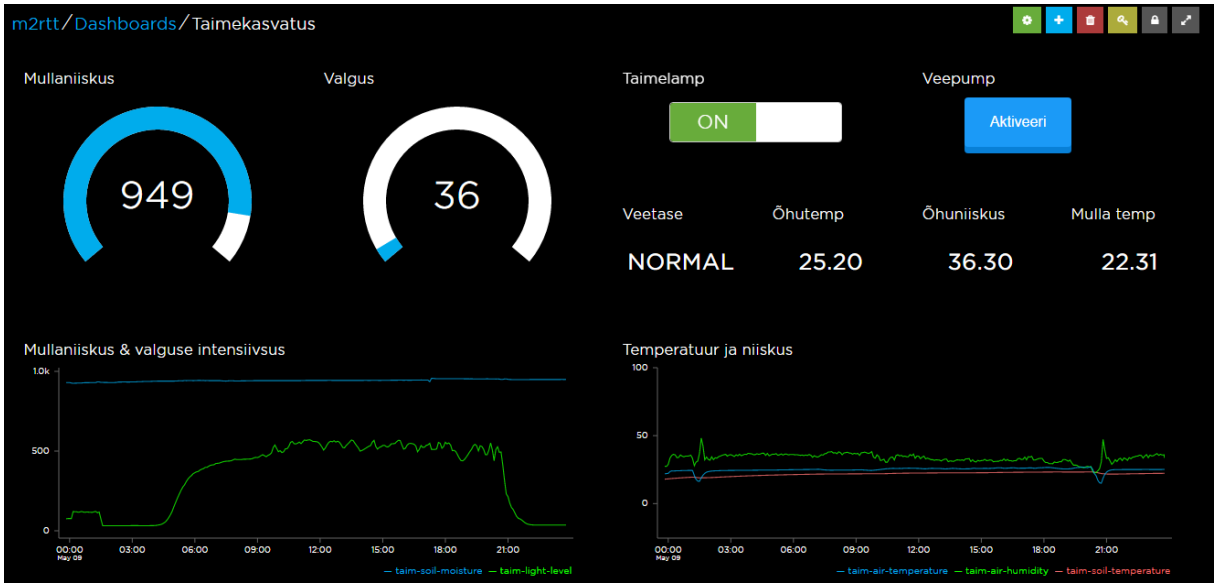
Järgnevalt tuleb Adafruit IO süsteemis lisada uued kanalid, mille võtmed vastavad programmis kirjeldatud võtmetele. Taimekasvatuse juhtimiseks on mõistlik teha uus juhtpaneel ning sinna lisada endale meelepärased elemendid andurite andmete kuvamiseks.

Pumba ja relee lülitamiseks sobib näiteks kahe režiimiga nupp. Nupu lisamisel tuleb arvestada sellega, et “Button On Text” ja Button Off Text” on sõnumid, mida nupule määratud kanalisse edastatakse. Seega kui arendusplaadi programmis kontrollitakse, kas sõnum on “ON” või “OFF” nagu programmi koodis (joonis 58), tuleks ka sama tekst nupule juhtpaneelis lisada (joonis 59).



Joonis 59. Nupu lisamine juhtpaneelile.

Üks võimalik juhtpaneeli elementide paigutus on joonisel 60. Selline paigutus on ka autori juhtpaneelil.



Joonis 60. Targa taimekasvatuse juhtpaneeli näide.

Kui juhtpaneel on seadistatud ja arendusplaat töötab, peaks hakkama Adafruit IO süsteemi laekuma andmeid. Kui seda ei juhtu, tuleks kõigepealt kontrollida, mida Arduino IDE *Serial Monitor* kuvab informatsiooni ühenduse loomise kohta. Kui ühendus on loodud, tuleks vaadata üle kanalite nimed ja veenduda nende korrektsuses. Ainult juhtpaneeli kasutades annab süsteem küll hea ülevaate taime elust ning võimaldab nuppude abil kasta taime või valgustust lülitada,

kuid käsitsi seda teha võib olla tüütu. Järgmises punktia näidatakse, kuidas seda saaks automatiseerida.

4.7. Värkvõrgu ja internetiteenuste automatiseerimine

Värkvõrgu seadmete haldamine juhtpaneeli kaudu on mugav, kuid seadmete kontrollimine toimub käsitsi. Mõnikord võib tekkida vajadus seadmete töö automatiseerimise järele. Näiteks soovitakse pimeduse saabudes kodus tuled automaatselt põlema panna või toa temperatuuri järgi küttesüsteemi reguleerida. Kui selliseid funktsioone kontrollib üks seade, on võimalik automatiseerimine seadme programmi sisse kirjutada, kuid see ei ole alati parim variant, sest automaatika muutmiseks tuleb seadme töö ajutiselt katkestada ja seade ümber programmeerida. Kui mainitud funktsioonid on eraldi seadmete täita, näiteks üks seade mõõdab pidevalt toatemperatuuri ja teine seade kontrollib valgustust, siis saab nende tööd automatiseerida vaid läbi serveri, kuhu mõlemad seadmed on ühendatud. Juhtpaneelidel on üldjuhul olemas mingisugune võimalus seadmete tööd automatiseerida. Adafruit IO süsteemis on selleks funktsioon “Triggers”, mis on ka punktis 3.3.2.5. kirjeldatud.

4.7.1. Värkvõrgu automatiseerimiseks mõeldud veebiteenused

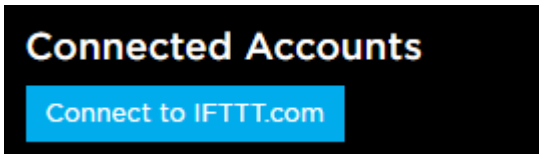
Automatiseerimise lihtsustamiseks ja mitmekesistamiseks on loodud veebiteenuseid, mille ülesanne on erinevaid veebirakendusi ja teenuseid omavahel ühendada. Selliste automatiseerimiseks mõeldud veebiteenuste peamine idee on reageerida ühe teenuse muutusele ja selle põhjal käivitada mingi funktsioon teises teenuses. Näiteks kui kasutajale saabub e-kiri kindlalt e-posti aadressilt, lisatakse selle kohta kalendrisse märge. Võimalusi ja kombinatsioone on sadu. Hetkel populaarsemad automatiseerimise veebiteenused on IFTTT (*If This Then That*), Zapier ja Microsoft Flow [100].

Nii IFTTT kui ka Zapier toetavad ka Adafruit IO-d, Microsoft Flow aga mitte. IFTTT on rohkem suunatud tavakasutajale, sisaldades rohkelt sotsiaalvõrgustike teenuseid, näiteks Twitter, Instagram, Facebook. Samuti leiab sealt mitmeid asjade interneti seadmete teenuseid nagu Philips HUE valgustus või Samsung konditsioneerid ja külmkapid. Zapier teenused on pigem suunatud ettevõtetele ja asjade interneti seadmete teenuseid sealt ei leia. Seega tavakasutajale ja asjade interneti automatiseerimiseks on autori hinnangul mõistlik valik

IFTTT. Järgnevalt näidatakse, kuidas kasutada IFTTT-i, et saata telefonisse teade, kui veepumba anumas on veetase liiga madal.

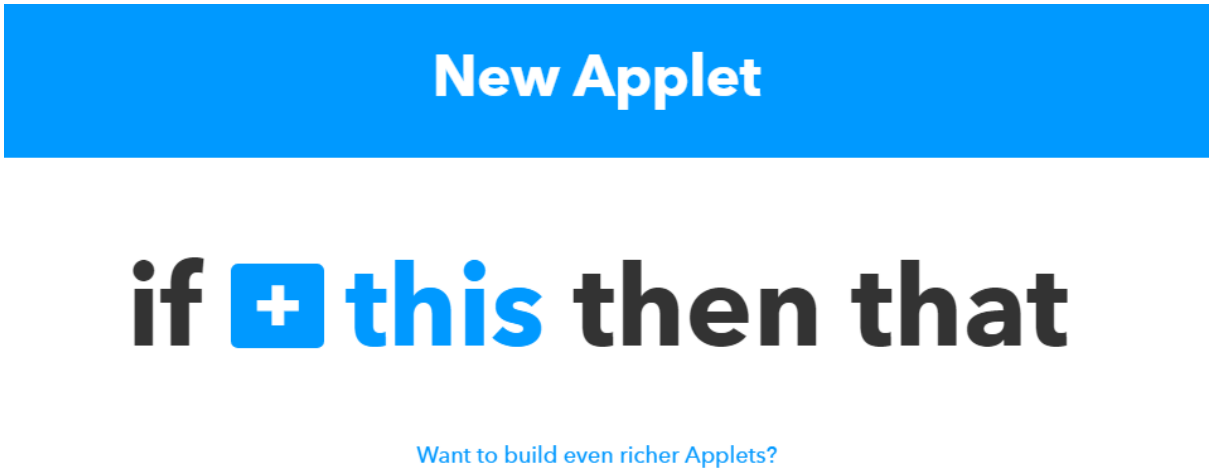
4.7.2. IFTTT kasutamise juhend veetaseme teavituse näitel

Alustuseks tuleb luua veebiteenuse IFTTT kasutajakonto. Seejärel tuleb siduda Adafruit IO konto IFTTT teenusega. Selle jaoks on Adafruit IO süsteemis leheküljel *"Settings"* nupp *"Connect to IFTTT.com"* (joonis 61).



Joonis 61. Nupp IFTTT teenusega ühendamiseks.

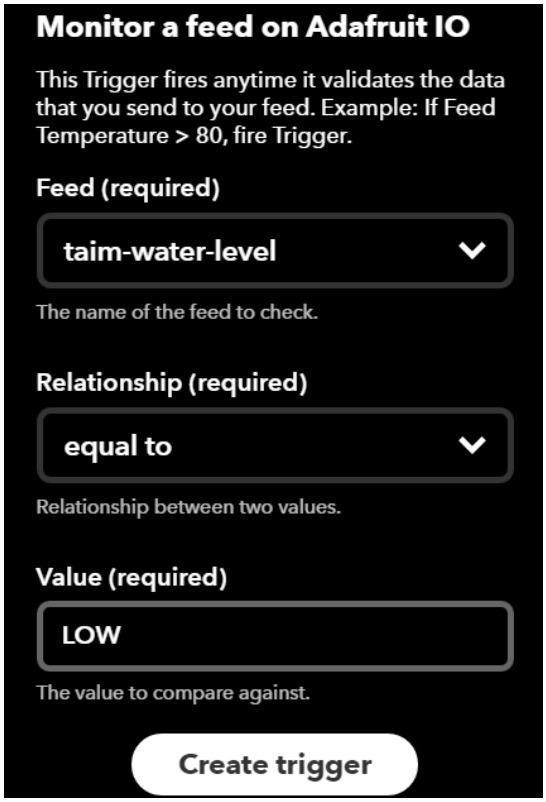
Järgmiseks tuleb IFTTT veebilehel luua uus aplett (ingl. k. *"applet"*) minnes IFTTT alamlehele *"My Applets"* ning vajutades nupule *"New Applet"*. Avaneb lehekülg, kus esmalt tuleb valida teenus, mis põhjustab apleti käivitamise (joonis 62).



Joonis 62. Lehekülg *"New Applet"*.

Klõpsates sõnale *"this"* avaneb teenuste valik. Valikust tuleb otsida Adafruit ning sellele vajutada. Adafruit pakub IFTTT teenuses kaht valikut: reageerimine kindale muutusele

Adafruit IO kanalis, näiteks kui õhuniiskus on üle 60%, või reageerimine iga kord, kui kanalisse andmeid laekub. Hetkel on mõistlik valida esimene pealkirjaga *"Monitor a feed on Adafruit IO"*. Järgmisel sammul tuleb valida soovitud kanal, antud juhul veetaseme kanal. Samuti tuleb määrata märk kahe väärtuse vahel (*"relationship"*): kas võrratus- või võrdusmärk. Kuna veetaseme kanali väärtus on sõne, saab ainult vaadata, kas kaks väärtust võrduvad omavahel. Sobiv valik on hetkel *"equal to"*. *"Value"* lahtrisse tuleks kirjutada "LOW" ilma jutumärkideta (joonis 63).



Joonis 63. IFTTT apletti loomine.

Iga kord kui veetaseme kanalisse saadetakse väärtus "LOW", käivitub loodav aplett. Pärast *"Create Trigger"* nupule vajutamist saab valida uue teenuse. Võimalik on näiteks saata e-kiri või postitada sotsiaalmeediasse. Teenus *"Notifications"* on mõeldud selleks, et saata nutiseadmesse teateid. Teenuse kasutamiseks on vajalik IFTTT rakenduse olemasolu nutiseadmes. Teate sisu saab muuta endale meelepäraseks ning vajutades *"Create Action"* nupule ongi aplett valmis. Enam ei pea ise veetaset juhtpaneelist jälgima vaid nutiseadmesse saabub automaatselt teade, kui vett juurde on vaja lisada.

Kahjuks on IFTTT funktsionaalsus üsna piiratud, näiteks ei saa teha keerulisema ülesehitusega apleteid, kus saaks kontrollida mitut tingimust enne teenuse käivitamist. Tekib ka probleem ühe apleti korduvas järjestikus käivitamises. Näiteks tehakse aplett, mis saadab Twitteris säutsu, kui muld on kuiv. Apleti looja võiks eeldada, et saadetakse üks säuts, aga tegelikult käivitub aplett iga kord, kui kanalisse saabuv mulla niiskuse number on alla apletis määratud taseme. Selliselt võib kogemata kasutaja säutsuda sama asja iga kord, kui mulla niiskuse kanalisse uusi andmeid saadetakse.

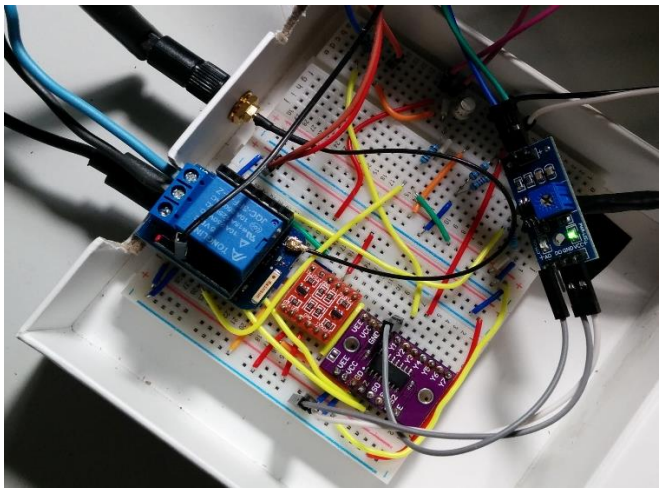
Töö käigus lahendust otsides leidis autor teenuse Apilio.IO, mis kasutades IFTTT teenust *"Maker Channel"* võimaldab luua keerukamaid automatiseerimisi [101]. IFTTT ja Apilio abil tegi autor valgustuse automaatseks: tuli süttib, kui taim piisavalt valgust ei saa. Samas arvestatakse kellaajaga ning öösel lamp ei põle. Automaatne on ka kastmine: kui muld on liiga kuiv, lülitatakse pump sisse. Täpsema juhendi kirjeldatud automaatse valgustuse loomisest kasutades IFTTT ja Apilio teenuseid leiab soovi korral lisast 4. Järgmises punktis tehakse kokkuvõte valminud töö tulemustest.

4.8. Töö tulemused

Töö käigus valmis tark taimekasvatuse süsteem, mis mõõdab taimedele olulisi näitajaid nagu õhutemperatuur ja –niiskus, mulla temperatuur ja niiskus ning valguse intensiivsus. Relee võimaldab juhtida taimelampi ning veepumba abil on võimalik taime kasta. Käesolev töö on esimene eestikeelne põhjalik ESP8266 mikrokontrolleril põhineva arendusplaadi tutvustus.

Ka loodud tark taimekasvatus on uudne. Taolist kodus järgitehtavat mõistlikus hinnaklassis süsteemi pole autori teada varem tehtud. Portaalist Hackaday.IO leiab küll taimekasvatusega seotud projekte, kuid ükski neist pole nii mitmekülgne. Näiteks leiab portaalist juhendi kastmissüsteemi loomise kohta, kuid puudub näiteks valgustuse reguleerimine [102]. Keerukamad projektid koosnevad juba oluliselt rohkematest komponentidest ning lisaks ESP8266 mikrokontrollerile sisaldavad ka teistsuguseid arendusplaate [103].

Järgnevalt mõned joonised autori loodud süsteemist. Joonisel 61 on näidatud projekti maketeerimislaudadele ühendatud skeem.



Joonis 61. Autori projekt maketeerimislaual.

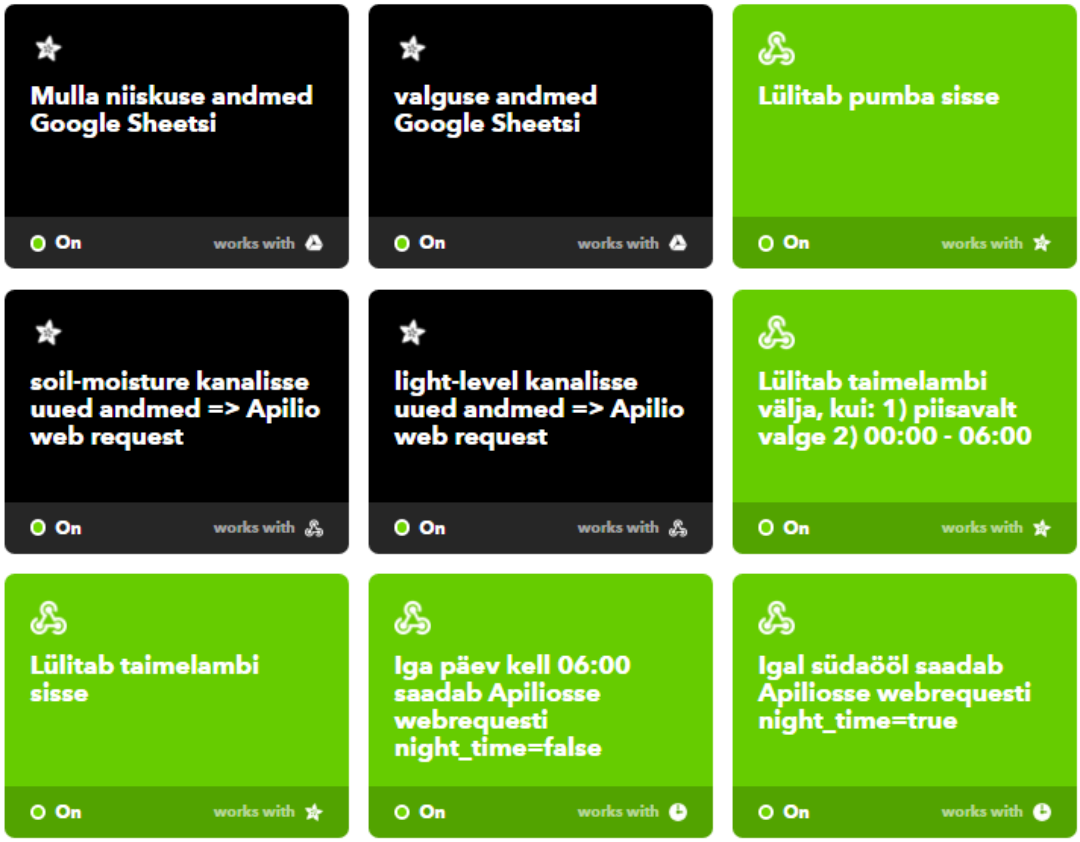
Joonisel 62 on näha projekti ülejäänud osa: potitaim anduritega ning veenõu.

Joonis 62. Potitaim ja veeanum. Taime valgustab sobiva pirniga laualamp.

Joonisel 63 on täpsemalt näha taime pott. Potis on ise meisterdatud mullaniiskuse anduri elektroodid. Samuti musta juhtmega temperatuurandur ning pildi paremas servas õhuniiskuse ja –temperatuuriandur.

Joonis 63. Taime pott anduritega.

Adafruit IO abil saab jälgida seadmest tulnud andmeid ning nuppude abil lülitada taimelampi ja veepumpa. Samuti on IFTTT abil süsteemi automatiseeritud (joonis 64).



Joonis 64. Valik autori IFTTT apleteid.

IFTTT teenust on kasutatud, et salvestada andurite andmeid Google Sheets arvutustabelisse. Autor kasutas ka veebiteenust Apilio.IO ning IFTTT ja Apilio abiga on automatiseeritud kastmine, kui mulla niiskus langeb alla kindla taseme. Samuti on taimelamp automaatne. Lamp põleb, kui on liiga hämar, samas vahemikus südaööst hommikul kella kuueni ei põle.

Autori kogemusel on IFTTT paraku kohati aeglane: mõnikord on viivitus andmete laekumise Adafruit IO süsteemi ja IFTTT apleti käivitamise vahel ligi pool tundi. Taimekasvatuse puhul see ei häiri, kuid kui peaks tahtma ajakriitilisi sündmusi automatiseerida, võib tekkida probleeme. Üldiselt on süsteem töökindel ning suuremaid probleeme töö käigus ei esinenud.

Autori hinnangul on WeMos arendusplaat oma suuruse ning hinna kohta vägagi võimekas ning võiks leida rakendust ka osana suurematest projektidest nagu näiteks SmartEnCity.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua tark taimekasvatuse süsteem kasutades WeMos arendusplaati ning praktilise töö käigus uurida, kui võimekas on selline väike ja odav arendusplaat. Autori hinnangul on WeMos arendusplaat oma suuruse ning hinna kohta vägagi võimekas ning võiks leida rakendust lisaks kodusele kasutusele ka osana suurematest projektidest.

Töös anti ülevaade asjade internetist, tutvustati WeMos arendusplaati ja näidati, kuidas seda kasutada. Valmis juhend WeMos arendusplaadi seadistamiseks ning kasutamiseks. Praktilise töö käigus näidati, kuidas kompenseerida mõningaid arendusplaadi puuduseid. Töö käigus valmis süsteem, mis suudab taimi kasta, lülitada valgustust ning edastada taimekasvu juures olulisi parameetreid nagu temperatuur, niiskus ja valgus. Bakalaureusetöö sisaldab projekti skeeme ja selgitustega programmi koodi, võimaldades lugejal soovi korral sarnase süsteemi ise kodus valmis teha.

Põhjalik ülevaade anti asjade interneti veebiplatvormist Adafruit IO, millega on lihtne asjade internetiga tutvust teha. Samuti valmis juhend automatiseerimise teenuse IFTTT kasutamiseks. Valminud bakalaureusetööst on kasu kõigile, kes soovivad tutvust teha asjade internetiga, kuid kel ei ole võimalik sellele palju raha kulutada. Bakalaureusetööd saab kasutada näiteks erinevates elektroonika õpitubades. Praktilise töö käigus veendus autor, et värkvõrk ei ole kallis ja keeruline tulevikutehnoloogia vaid on kõigile kättesaadav ja praktiline juba praegu.

Samas on tehnoloogial veel arenemisruumi. Puudub universaalne platvorm, mis rahuldaks ka nõudlikuma kasutaja vajadused ja oleks sealjuures kõigile vabalt kättesaadav. Keerukama lahenduse loomine nõuab leidlikkust ja mitme süsteemi kooskasutamist. Samuti on tasuta teenuste reageerimiskiirus kohati aeglane.

Bakalaureusetöö valmimise käigus tutvus autor esmakordselt riistvara programmeerimisega ning asjade interneti seadmete loomisega. Teema oli autori jaoks põnev ning suurendas huvi asjade interneti ja riistvara programmeerimise vastu.

Kasutatud kirjandus

[1] Aliexpress, „WeMos D1 R2 V2.1.0 WiFi uno based ESP8266 for arduino nodemcu Compatible“ https://www.aliexpress.com/store/product/WeMos-D1-WiFi-uno-based-ESP8266-for-arduino-Compatible/1331105_32455782552.html (24.04.2017)

[2] Küberfüüsikalise süsteemitehnika terminibaas, „Värkvõrk“. http://term.eki.ee/termbase/view/6811586/et/en/?initial=V#/concept/view/1112650191/ (18.10.2016)

[3] Gérald Santucci, “The Internet of Things: Between the Revolution of the Internet and the Metamorphosis of Objects”, 2010 http://cordis.europa.eu/fp7/ict/enet/documents/publications/iot-between-the-internet-revolution.pdf

[4] Business Insider, „THE INTERNET OF THINGS 2017 REPORT: How the IoT is improving lives to transform the world“, 2017 http://www.businessinsider.com/the-internet-of-things-2017-report-2017-1 (14.01.2017)

[5] IDC, „Internet of Things Spending Forecast to Grow 17.9% in 2016 Led by Manufacturing, Transportation, and Utilities Investments, According to New IDC Spending Guide“, 2017 http://www.idc.com/getdoc.jsp?containerId=prUS42209117 (15.01.2017)

[6] Smart City Lab, „Klastrist“ http://smartcitylab.eu/klastrist (20.01.2017)

[7] Tark Tartu, „Projekt SmartEnCity“ http://tarktartu.ee/avaleht/ulevaade/ (20.01.2017)

[8] Tark Tartu, „Kavandatud tegevused“ http://tarktartu.ee/avaleht/kavandatud-tegevused/ (20.01.2017)

[9] Attendly, „Arduino is making the world a better place for charities and non-profits“ http://www.attendly.com/arduino-is-making-the-world-a-better-place-for-charities-and-non-profits/ 26.01.2017)

[10] Bluetooth, „How it works“ https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works (27.01.2017)

[11] Hackaday, „New chip alert: the ESP8266 wifi 80õdule (it’s $5)“ http://hackaday.com/2014/08/26/new-chip-alert-the-esp8266-wifi-module-its-5/ (30.01.2017)

[12] 43oh, „Comparing IoT Wi-Fi Modules – ESP8266 vs CC3000 vs RN131 vs HDG204“ http://43oh.com/2015/06/comparing-iot-wi-fi-modules-esp8266-vs-cc3000-vs-rn131-vs-hdg204/ (30.01.2017)

[13] Hackaday, „An SDK for the ESP8266 WIFI chip“ http://hackaday.com/2014/10/25/an-sdk-for-the-esp8266-wifi-chip/ (30.01.2017)

[14] Losant, „Topp 6 ESP8266 modules for IoT projects“ https://www.losant.com/blog/topp-6-esp8266-modules (30.01.2017)

[15] Adafruit, „Adafruit Feather Huzzah“ https://www.adafruit.com/product/2821 (16.03.2017)

[16] ITT Group, „NodeMcu (v3) Lua WIFI arendusplaat“ http://www.ittgroup.ee/et/mikrokontrollerite-arendusplaadid/799-nodemcu-v3-lua-wifi-arendusplaat-esp8266.html (16.03.2017)

[17] Yeint, „SparkFun ESP8266 Thing“ http://yeint.ee/it-multimeedia/kaablid-ja-tarvikud-1/arvutikaablid/wifi/sparkfun-esp8266-thing (16.03.2017)

[18] ITT Group, „Wio Link“ http://www.ittgroup.ee/et/pood/747-wio-link.html (16.03.2017)

[19] ITT Group, „WeMos D1 Mini“ http://www.ittgroup.ee/et/asjade-internet/980-wemos-d1-mini-esp8266-12s.html (16.03.2017)

[20] Aliexpress, „D1 mini V2“ https://www.aliexpress.com/store/product/D1-mini-Mini-NodeMcu-4M-bytes-Lua-WIFI-Internet-of-Things-development-board-based-ESP8266/1331105_32529101036.html (20.04.2017)

[21] Aliexpress, „NodeMCU V3“ https://www.aliexpress.com/item/NodeMcu-Lua-V3-ESP8266-CH340G-Wireless-ESP8266-Serial-Port-Internet-Dev-Kit-Development-Board-Module-Electronics/32730366755.html (16.03.2017)

[22] WeMos Electonics, „D1 Mini Shields“ https://www.wemos.cc/D1-mini-Shields (20.04.2017)

[23] WeMos Electronics, „WeMos Products“ https://www.wemos.cc/product (20.04.2017)

[24] WeMos Electronics, „D1 Mini“ https://www.wemos.cc/product/d1-mini.html (20.04.2017)

[25] WeMos Electronics, „D1 Mini Pro“ https://www.wemos.cc/product/d1-mini-pro.html (20.04.2017)

[26] WeMos Electronics, „D1“ https://www.wemos.cc/product/d1.html (20.04.2017)

[27] Espressif, „ESP8266 Overview“ https://espressif.com/en/products/hardware/esp8266ex/overview (07.02.2017)

[28] Aliexpress, „WEMOS D1 mini Pro“ https://www.aliexpress.com/store/product/WEMOS-D1-mini-Pro-16M-bytes-external-antenna-connector-ESP8266-WIFI-Internet-of-Things-development-board/1331105_32724692514.html (20.04.2017)

[29] EdgeFX Kits, „What is Multiplexer and Types of Multiplexing Techniques“ http://efxkits.com/blog/what-is-multiplexer-and-types/ (08.02.2017)

[30] SparkFun, „Using the Logic Level Converter“ https://learn.sparkfun.com/tutorials/using-the-logic-level-converter (12.04.2017)

[31] Instructables, „Programming the ESP8266 WeMos-D1R2 using Arduino Software/IDE“ http://www.instructables.com/id/Programming-the-WeMos-Using-Arduino-SoftwareIDE/ (10.02.2017)

[32] Libraries.io, “Arduino-Pinout, Arduino Uno R3“ https://libraries.io/github/Bouni/Arduino-Pinout (10.02.2017)

[33] Hackaday, „CarontePass: Open Access Control“ https://hackaday.io/project/10498-carontepass-open-access-control (02.04.2017)

[34] Hackaday, „Hacking a smoke detector“ https://hackaday.io/project/19207-hacking-a-smoke-detector (02.04.2017)

[35] Hackaday, „IoTea kettle“ https://hackaday.io/project/20217-iotea-kettle (02.04.2017)

[36] MicroPython, „General information about the ESP8266 port“ https://docs.micropython.org/en/latest/esp8266/esp8266/general.html (12.02.2017)

[37] Arduino Guide, „What is Arduino?“ https://www.arduino.cc/en/Guide/Introduction (12.02.2017)

[38] NodeMcu Documentation http://nodemcu.readthedocs.io/en/dev/en/ (12.02.2017)

[39] Arduino Reference, „Arduino/Processing Language Comparison“ https://www.arduino.cc/en/Reference/Comparison (14.02.2017)

[40] Arduino Reference, „setup()“ https://www.arduino.cc/en/reference/setup (14.02.2017)

[41] Arduino Reference, „loop()“ https://www.arduino.cc/en/reference/loop (14.02.2017)

[42] GitHub ESP8266 Arduino repositoorium, „WPA2-enterprise + PEAP“ https://github.com/esp8266/Arduino/issues/1032 (08.05.2017)

[43] Systembash, „MQTT vs WebSocket vs HTTP/2 : The Best IoT Messaging Protocol?“ https://systembash.com/mqtt-vs-websockets-vs-http2-the-best-iot-messaging-protocol/ (25.02.2017)

[44] WebSocket, „About WebSocket“ https://www.websocket.org/aboutwebsocket.html (25.02.2017)

[45] IBM, „What is MQTT“ https://www.ibm.com/developerworks/mydeveloperworks/blogs/aimsupport/entry/what_is_mqtt_and_how_does_it_work_with_websphere_mq?lang=en (25.02.2017)

[46] IBM, „Publish/subscribe messaging“ https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.pro.doc/q004870_.htm (25.02.2017)

[47] Kenneth Peeples, “Internet of Things MQTT Quality of Service Levels“, 2015 http://www.ossmentor.com/2015/04/internet-of-things-mqtt-quality-of.html (25.02.2017)

[48] Telia, „Nutikas linn“ https://www.telia.ee/ari/muud-teenused/iot (27.02.2017)

[49] Home Assistant veebilehekülg https://home-assistant.io/ (27.02.2017)

[50] OpenHAB, „About openHAB“ http://docs.openhab.org/introduction.html (27.02.2017)

[51] Adafruit, „About us“ https://www.adafruit.com/about (27.02.2017)

[52] Adafruit IO, „Data policies“ https://learn.adafruit.com/adafruit-io?view=all#data-policies (27.02.2017)

[53] Adafruit IO, „IoT Security: Connecting Your ESP8266 to Adafruit IO with SSL/TLS“, 2016 https://io.adafruit.com/blog/security/2016/07/05/adafruit-io-security-esp8266/ (14.03.2017)

[54] Adafruit IO, „Naming things in Adafruit IO“ https://learn.adafruit.com/naming-things-in-adafruit-io (03.03.2017)

[55] Otsus Andro, „Maaviljelus“, 2004 https://luua.kovtp.ee/documents/105873/1600210/Maaviljelus.pdf (11.03.2017)

[56] Click and Grow veebilehekülg https://www.clickandgrow.com/ (11.03.2017)

[57] ITT Group, „Maketeerimislaud 400“ http://www.ittgroup.ee/et/prototuupimine-ja-toide/250-maketeerimislaud-400.html (20.04.2017)

[58] Aliexpress, „2 pcs Breadboard 400 Point“ https://www.aliexpress.com/item/2-pcs-Mini-Solderless-Prototype-Experiment-Test-DIY-Breadboard-400-Point-Tie-points-85-x-55/1651988755.html (20.04.2017)

[59] ITT Group, „WeMos D1 Mini Pro“ http://www.ittgroup.ee/et/asjade-internet/981-wemos-d1-mini-pro-esp8266.html (20.04.2017)

[60] YEInternational, „SparkFun Multiplexer Breakout - 8 Channel“ http://yeint.ee/elektroonika-1/arendusvahendid/iot-asjade-internet/sparkfun-multiplexer-breakout-8-channel-74hc4051 (20.04.2017)

[61] Aliexpress, „8 channel analog multiplexer module“ https://www.aliexpress.com/item/CFSUNBIRD-74HC4051-8-Channel-Mux-CJMCU-4051-8-channel-analog-multiplexer-module/32796368290.html (20.04.2017)

[62] Aliexpress, „3.3V 5V 2 Channel Logic Level Converter“ https://www.aliexpress.com/item/3-3V-5V-2-Channel-Logic-Level-Converter-TTL-Logic-Level-Conversion-Bidirectional-Mutual-Convert-Compatible/1803892611.html (20.04.2017)

[63] Oomipood, „Loogika muundur 5V-3.3V mõlemasuunaline“ https://www.oomipood.ee/product/lvl_conv_loogika_muundur_5v_3_3v (20.04.2017)

[64] Aliexpress, „Relay shield for WeMos D1 Mini“ https://www.aliexpress.com/store/product/Relay-Shield-for-WeMos-D1-mini-button/1331105_32596395175.html (20.04.2017)

[65] Oomipood, „Releeplaat WeMos D1“ https://www.oomipood.ee/product/wemos_relay_releeplaat_wemos_d1 (20.04.2017)

[66] Aliexpress, „Soil moisture sensor“ https://www.aliexpress.com/item/lot-soil-the-hygrometer-detection-module-soil-moisture-sensor-Robot-smart-car-For-UNO-R3/32568274107.html (20.04.2017)

[67] Oomipood, „Pinnase hügromeeter veeandur“ https://www.oomipood.ee/product/yl_69_pinnase_hugromeeter_veeandur (20.04.2017)

[68] Aliexpress, „DHT22 Digital temperature and humidity sensor module“ https://www.aliexpress.com/item/AM2302-DHT22-Digital-Temperature-and-Humidity-Sensor-Module-for-Arduino/32598962945.html (20.04.2017)

[69] YEInternational, „DHT22 temperature humidity sensor“ http://yeint.ee/elektroonika-1/arendusvahendid/andurid-1/temperatuuri-ja-niiskusandurid/am2302-wired-dht22-temperature-humidity-sensor (20.04.2017)

[70] Aliexpress, „Waterproof DS18b20 temperature probe“ https://www.aliexpress.com/item/Stainless-Steel-Package-1-Meters-Waterproof-

DS18b20-Temperature-Probe-Temperature-Sensor-18B20/32359386131.html (20.04.2017)

[71] ITT Group, „Veekindel temperatuuriandur DS18b20“ http://www.ittgroup.ee/et/andurid-ja-nende-tarvikud/403-veekindel-temperatuuriandur-ds18b20.html (20.04.2017)

[72] Aliexpress, „Submersible water pump DC 3V-5V“ https://www.aliexpress.com/item/Mute-Submersible-Pump-Water-Punp-DC-3V-5V-For-PC-Cooling-Water-Circulation-DIY/32717325894.html (20.04.2017)

[73] Bauhof, „Voolik Cristallo Extra AL 6/9mm“ http://www.bauhof.ee/voolik-cristallo-extra-al-6-9mm.html (20.04.2017)

[74] Aliexpress, 120tk Dupont maketeerimislaua juhtmed https://www.aliexpress.com/item/Free-shipping-Dupont-line-120pcs-10cm-male-to-male-male-to-female-and-female-to-female/32347691887.html (20.04.2017)

[75] Oomipood, Maketeerimislaua juhtmed 40tk isa-isa https://www.oomipood.ee/product/brdb_jump_10_mm_40_maketeerimislaua_juhtmed_40tk_isa_isa_10cm_2_54mm (20.04.2017)

[76] Oomipood, Maketeerimislaua juhtmed 40tk isa-ema https://www.oomipood.ee/product/brdb_jump_10_mf_40_maketeerimislaua_juhtmed_40tk_isa_ema_10cm_2_54mm (20.04.2017)

[77] Aliexpress, „140pcs Breadboard jumper wires“ https://www.aliexpress.com/item/Free-Shipping-140pcs-in-one-package-convenient-New-Solderless-Flexible-Breadboard-Jumper-wires-Cables-HOT-Sale/2044172287.html (20.04.2017)

[78] Oomipood, „Maketeerimislaua ühendustraatide komplekt 140tk“ https://www.oomipood.ee/product?product_id=12587 (20.04.2017)

[79] Aliexpress, „Metal film resistor assorted kit“ https://www.aliexpress.com/item/Free-Shipping-1-6W-1-8W-Resistor-Metal-Film-Resistor-Assorted-Kit-Sample-bag-4-7K/32361951063.html (20.04.2017)

[80] ITT Group, takistite komplekt 150tk http://www.ittgroup.ee/et/prototuupimine-ja-toide/955-takistite-komplekt-14w-150tk.html (20.04.2017)

[81] Oomipood, „Transistor 2N2222a“ https://www.oomipood.ee/product/2n2222a_pl_2n2222a_si_n_75v_0_8a_0_5w_to92_ksp2222abu (20.04.2017)

[82] Oomipood, fototakisti 7mm https://www.oomipood.ee/product/ldr09_fototakisti_7mm_20_50k_ldr (20.04.2017)

[83] Roboticlab, „Pingejagur“ http://home.roboticlab.eu/et/electronics/voltage_divider (15.03.2017)

[84] Sparkfun, „WS2812 Intelligent control LED integrated light source“ http://cdn.sparkfun.com/datasheets/Components/LED/WS2812.pdf (17.03.2017)

[85] Arduino Playground, „Analog Multiplexer/Demultiplexer – 4051“ http://playground.arduino.cc/Learning/4051 (17.03.2017)

[86] Sparkfun, „Multiplexer Breakout Hookup Guide“ https://learn.sparkfun.com/tutorials/multiplexer-breakout-hookup-guide (22.03.2017)

[87] Arduino Reference, bitRead() http://www.arduino.org/learning/reference/bitread (18.03.2017)

[88] Taskutark, „Elektrolüüs“ https://www.taskutark.ee/m/elektroluus/ (20.03.2017)

[89] Edaphic Scientific, „Soil moisture sensor calibration“ http://www.edaphic.com.au/soil-water-compendium/soil-moisture-sensor-calibration/ (20.03.2017)

[90] Roboticlab, Fototakisti http://home.roboticlab.eu/et/examples/sensor/photoresistor (20.03.2017)

[91] SparkFun, „Transistors, Applications I: Switches“ https://learn.sparkfun.com/tutorials/transistors/applications-i-switches (25.03.2017)

[92] Circuit Diagramz, „All about 2n2222 transistor and its Circuit diagrams“ http://circuit-diagramz.com/%E2%80%8B2n2222-transistor-circuit-diagrams/ (25.03.2017)

[93] Aosong, „Temperature and humidity module AM2302 Product Manual“ http://akizukidenshi.com/download/ds/aosong/AM2302.pdf (28.03.2017)

[94] MaximIntegrated, „DS18b20 Programmable Resolution 1-Wire Digital Thermometer https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf (29.03.2017)

[95] Järvamaa kutsehariduskeskuse õppematerjalid, Relee http://jkhk.ee/media/Oppematerjalid/elektrotehnika/relee.html (30.03.2017)

[96] Nelja releega plaat http://www.ittgroup.ee/et/muundurid/812-releemoodul-4x10a.html (30.03.2017)

[97] WeMos Electronics, Relay Shield https://www.wemos.cc/product/relay-shield.html (30.03.2017)

[98] Acme HowTo, „How an Electric Light Switch Works“ http://www.acmehowto.com/electrical/switch-works.php (30.03.2017)

[99] Jain Arpit, „Insight - How electrical Line Tester screwdriver works“, 2012 https://www.engineersgarage.com/insight/how-electrical-line-tester-screwdriver-works (30.03.2017)

[100] Lifehacker, „Automation Showdown: IFTTT vs Zapier vs Microsoft Flow“ http://lifehacker.com/automation-showdown-ifttt-vs-zapier-vs-microsoft-flow-1782584748 (12.04.2017)

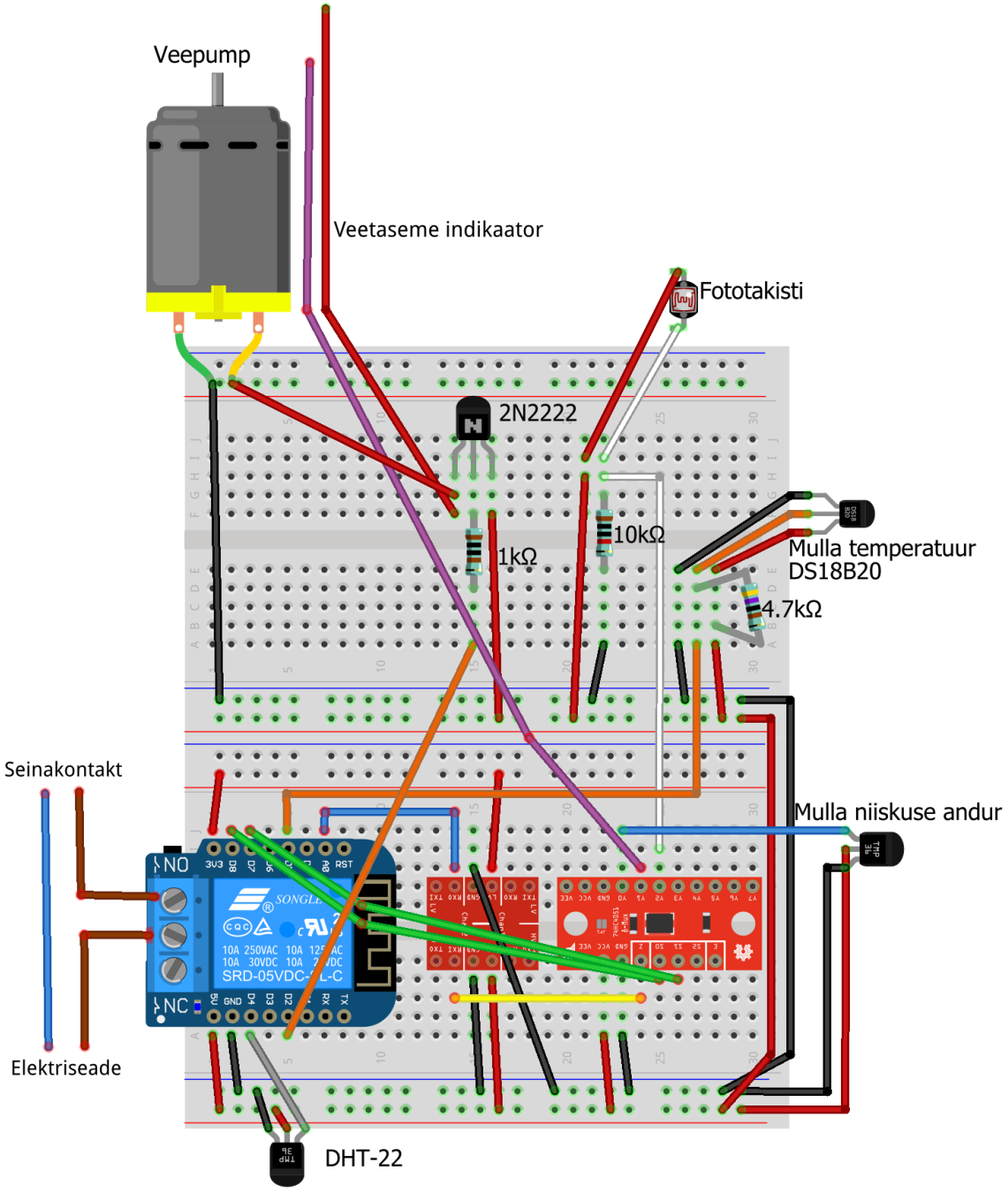
[101] Apilio.IO veebilehekülg http://www.apilio.io/ (04.05.2017)

[102] Hackaday, „ESP8266 Tutorial: Build An Automatic Plant Waterer“ https://hackaday.io/project/21758-esp8266-tutorial-build-an-automatic-plant-waterer (09.05.2017)

[103] Hackaday, „HydroPWNics“ https://hackaday.io/project/2964-hydropwnics (09.05.2017)

Lisad

Lisa 1. Targa taimekasvatuse projekti skeem



Lisa 2. Projekti programm

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <OneWire.h>
#include <DallasTemperature.h>

const char* ssid     = "wifi-SSID";
const char* password = "wifi_parool";

const char* mqtt_server = "io.adafruit.com";
const char* mqtt_user = "kasutajanimi";
const char* mqtt_password = "AIO-võti";
const int port = 8883; // 8883 on turvalise SSL-ühenduse jaoks

const char* clientID = "WeMos_seade"; // iga seadme ID peab olema unikaalne

// Adafruit IO serveri sõrmejälg turvalise ühenduse loomiseks
const char* fingerprint = "26 96 1C 2A 51 07 FD 15 80 96 93 AE F7 32 CE B9 0D 01 55 C4";

// subscribe kanal on sõnumite vastuvõtmiseks
const char* subscribeRelay = "kasutajanimi/feeds/taim-relay"; // relee
const char* subscribePump = "kasutajanimi/feeds/taim-water-pump"; // veepump
// publish kanali kaudu saadab arendusplaat sõnumeid
const char* publishWater = "kasutajanimi/feeds/taim-water-level"; // veetase
const char* publishSoilMoisture = "kasutajanimi/feeds/taim-soil-moisture"; // mulla niiskus
const char* publishSoilTemp = "kasutajanimi/feeds/taim-soil-temperature"; // mulla temp
const char* publishTemp = "kasutajanimi/feeds/taim-air-temperature"; // õhutemp
const char* publishHumidity = "kasutajanimi/feeds/taim-air-humidity"; // õhuniiskus
const char* publishPump = "kasutajanimi/feeds/taim-water-pump"; // veepump
const char* publishLightLevel = "kasutajanimi/feeds/taim-light-level"; // valgus

char msg[50]; // muutuja, mida kasutatakse sõnumi teksti koostamisel
// turvalise WiFi ühenduse klient ja MQTT klient
WiFiClientSecure espClient;
PubSubClient client(espClient);

// multiplekseri viigud
#define S0            D7 // Multiplekseri viik S0
#define S1            D8 // Multiplekseri viik S1

// analoogandurite viigud
int soilMUXPin = 0; // mulla niiskus Y0
int waterMUXPin = 1; // veetase Y1
int lightMUXPin = 2; // valgus Y2

// DHT-22
#define DHTPIN      D4 // DHT-22 on viigul D4
DHT dht(DHTPIN, DHT22);
float previousTemp = 0; // temperatuuri salvestamiseks
float previousHumidity = 0; // õhuniiskuse salvestamiseks

// DS18b20
#define ONE_WIRE_BUS  D5  // DS18B20 viik
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

unsigned long sensorsPreviousMillis = 0; // aeg, mil viimati andurit loeti
const long sensorsInterval = 4000;    // anduri lugemise intervall
int count = 0; // alustame andurite lugemist 0-st
```

```
int totalCount = 4; // kokku loeme pidevalt viit näitu, null kaasaarvatud

// veepump
#define PUMPPIN       D2 // pump on viigul D2
bool pumpActivated = false; // algkäivitusel pump ei tööta
unsigned long pumpPreviousMillis = 0;
// pumba tööaeg määrata vastavalt vajadusele
const long pumpInterval = 4000; // intervall, mitu sekundit pump järjest töötab
// piirid pumba jaoks, kui Adafruit IO süsteem peaks alt vedama
// veetaseme anduri näit tuleb piiri määramiseks eelnevalt mõõta
int waterLevelTreshold = 50; // umbes 60 on siis kui on vees, <10 siis kui ei ole
int soilMoistureMax = 900; // maksimaalne mulla niiskus. Üle selle, siis pump ei tööta

// relee viik
#define RELAYPIN    D1

void setup() {
 Serial.begin(115200); // Avab jadapordi andmevahetuskiirusega 115200

   // multiplekseri juhtsignaali viigud määrame väljundiks
 pinMode(S0, OUTPUT);
 pinMode(S1, OUTPUT);
 pinMode(DHTPIN, INPUT);
 pinMode(PUMPPIN, OUTPUT);
 digitalWrite(PUMPPIN, LOW);
 sensors.begin();
 delay(10);

 // println() ja print() funktsioonid kirjutavad jadaporti andmeid
 // inimesele loetaval kujul, println lisab automaatselt reavahetuse märgi
 // andmete lõppu
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);

 /*
   Määrame WiFi režiimiks wifi WIFI_STA ehk klientrežiim
   Algseadistuses on ESP8266 nii klient kui ka ligipääsupunkt
   ja see võib tekitada WiFi võrgus probleeme teiste seadmetega
 */
 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);

 // kontrollime, kas WiFi ühendus on loodud, kui ei ole, ootame pool
 // sekundit ning kontrollime uuesti
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 // WiFi staatus on ühendatud, siis kuvame Serial Monitori
 // arendusplaadile omistatud IP aadressi
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());

 verifyFingerprint(); // funktsioon sõrmejälje kinnitamiseks
 client.setServer(mqtt_server, port); // määrame MQTT serveri
 // callback tegeleb kanalisse saabunud sõnumitega
 client.setCallback(callback);
}


void loop() {
 // kontrollime, kas MQTT serveriga on ühendus
```

```
 if (!client.connected()) {
   reconnect(); // kui ei ole, ühendame uuesti
 }
 // loop funktsioon on vajalik MQTT kliendi töötamiseks
 readSensors();
 if (pumpActivated) {
   handleWaterPump();
 }
 client.loop();
}
void handleWaterPump() {
 // kontrollime, et veetase oleks piisav ning muld ei oleks juba liiga niiske
 if (readMultiplexerValue(waterMUXPin) < waterLevelTreshold || 1023 -
readMultiplexerValue(soilMUXPin) >= soilMoistureMax) {
   // veetase on liiga madal, siis saadame veetaseme kanalisse sõnumi vastava tekstiga
   if(readMultiplexerValue(waterMUXPin) < waterLevelTreshold) {
     client.publish(publishWater, "LOW");
     Serial.println("Water level LOW");
   } else {
     // kui veetase on normis, saadame vastava teate
     // ja järelikult on muld liiga niiske
     client.publish(publishWater, "NORMAL");
     Serial.println("Soil too moist");
   }
   // lülitame pumba välja
   digitalWrite(PUMPPIN, LOW);
   pumpActivated = false;
   Serial.println("Turning pump off: limits exceeded");
   delay(200);
   // saadame pumba kanalisse sõnumi OFF
   client.publish(publishPump, "OFF");

 }
 unsigned long currentMillis = millis();
 // kui pump töötanud piisavalt kaua, lülitame selle välja
 if(currentMillis - pumpPreviousMillis >= pumpInterval) {
   pumpPreviousMillis = currentMillis;
   pumpActivated = false;
   if(readMultiplexerValue(waterMUXPin) >= waterLevelTreshold) {
     client.publish(publishWater, "NORMAL");
   }
   digitalWrite(PUMPPIN, LOW);
   Serial.println("Turning pump off: time up");
   client.publish(publishPump, "OFF");
 }
}
void readSensors() {
 // hetkeaeg
 unsigned long currentMillis = millis();
 // kui piisavalt aega on möödas ehk hetkeaeg - viimane aeg >= lugemise intervall
 // siis loeme count muutuja numbrile vastava anduri näidu
 if(currentMillis - sensorsPreviousMillis >= sensorsInterval) {
   // määrame hetkeaja viimaseks lugemise ajaks
   sensorsPreviousMillis = currentMillis;
   int analog; // analoogandurite näitude muutujad
   float digital; // digitaalandurite näidud
   switch(count) {
     // esimesena loeme mullaniiskuse näidu
     case 0:
       analog = readMultiplexerValue(soilMUXPin);
       // 1023 - analog, sest anduri näit on ümberpööratud:
       // niiskema mulla puhul väiksem number
       dtostrf(1023-analog, 2, 2, msg); // kopeerime numbrilise väärtuse msg muutujasse
       client.publish(publishSoilMoisture, msg); // edastame väärtuse õigesse kanalisse
       break; // break viib switch() funktsioonist välja
```

```
      case 1: // valguse lugemine
        analog = readMultiplexerValue(lightMUXPin);
        dtostrf(analog, 2, 2, msg);
        client.publish(publishLightLevel, msg);
        break;
      case 2: // õhutemperatuuri lugemine
        digital = readAirTemperature();
        dtostrf(digital, 2, 2, msg);
        client.publish(publishTemp, msg);
        break;
      case 3: // õhuniiskus
        digital = readAirHumidity();
        dtostrf(digital, 2, 2, msg);
        client.publish(publishHumidity, msg);
        break;
      case 4: // mulla temperatuur
        digital = readSoilTemperature();
        dtostrf(digital, 2, 2, msg);
        client.publish(publishSoilTemp, msg);
        break;
    }
    // kui loetava anduri number oli väiksem kui andurite koguarv,
    // suurendame numbrit, et lugeda järgmise anduri näitu
    if(count < totalCount) {
      count++;
    } else {
      // vastasel juhul alustame otsast peale
      count = 0;
    }
  }
}
/*
* Funktsioon valitud multiplekseri viigu väärtuse leidmiseks
* Sisend: Viigu number, mille väärtust soovime
* Väljund: loetud täisarvuline analoogviigu väärtus
*/
int readMultiplexerValue(int pinNumber) {
 // määrame S0 viigu väärtuseks soovitud viigu biti väärtuse kohal 0
 digitalWrite(S0, bitRead(pinNumber,0));
 // ning viigu S1 väärtuseks soovitud viigu biti väärtuse kohal 1
 digitalWrite(S1, bitRead(pinNumber,1));
 int result = analogRead(A0); // loeme analoogviigu väärtuse
 Serial.print(pinNumber);
 Serial.print(": MUX val: ");
 Serial.println(result);
 return result; // tagastame selle
}
float readSoilTemperature() {
 sensors.requestTemperatures(); // Küsime anduritelt temperatuure
 Serial.print("Mullatemperatuur: ");
 float temp = sensors.getTempCByIndex(0); // valime esimeselt andurilt temperatuuri
 Serial.println(temp);
 return temp; // tagastame temperatuuri
}
float readAirTemperature() {
 // anduri funktsioon readTemperature() tagastab ujukomaarvu (float).
 float temp = dht.readTemperature();
 // isnan funktsioon tagastab tõeväärtuse "true", kui tegemist pole numbriga
 if(isnan(temp)) {
   // kui temperatuurinäitu ei õnnestu saada, kasutame viimast salvestatud temperatuuri
   temp = previousTemp;
 } else {
   // salvestame uue temperatuuri
   previousTemp = temp;
 }
```

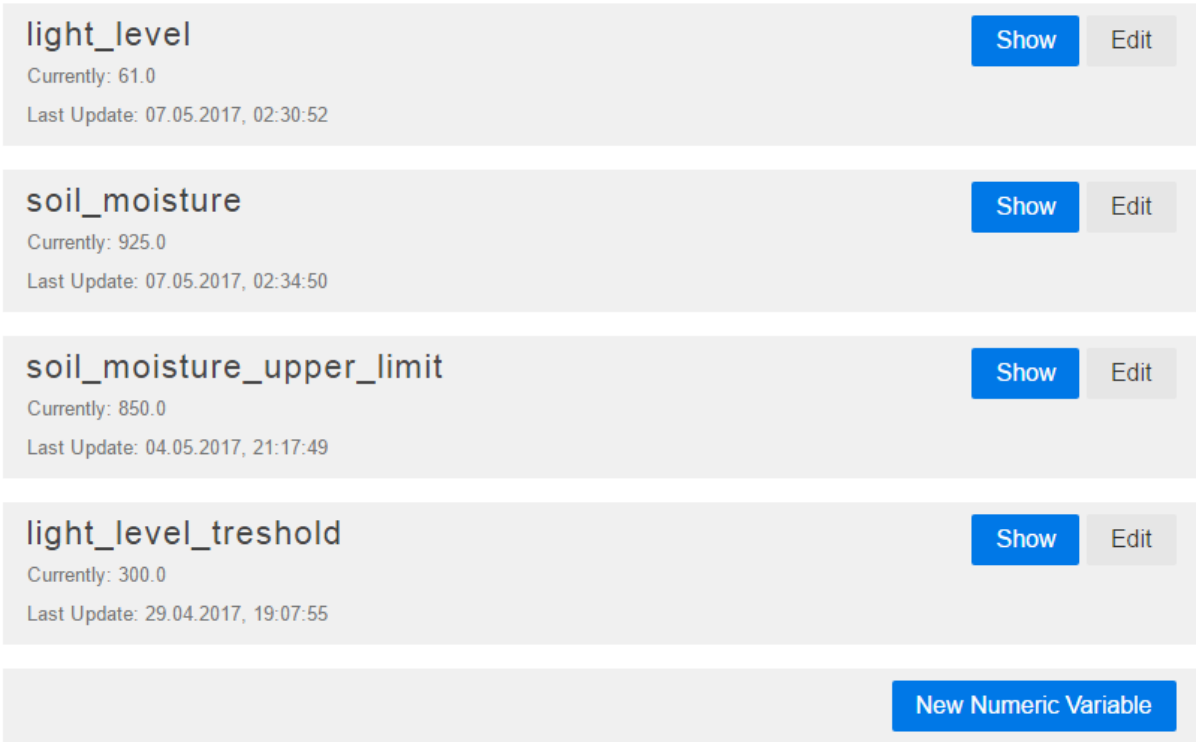
```
 Serial.print("Õhutemperatuur: ");
 Serial.print(temp);
 Serial.println(" *C");
 return temp; // tagastame temperatuuri näidu
}
float readAirHumidity() {
 float humidity = dht.readHumidity();
 if(isnan(humidity)) {
   humidity = previousHumidity;
 } else {
   previousHumidity = humidity;
 }
 Serial.print("Õhuniiskus: ");
 Serial.print(humidity);
 Serial.println(" %");
 return humidity;
}

// mõeldud saabuvate sõnumitega tegelemiseks
void callback(char* topic, byte* payload, unsigned int length) {
 // Teisendame sisse tulnud baidijärjendi sõneks
 payload[length] = '\0'; // Lisame lõpumärgi, mis tähistaks järjendi lõppu
 String message = (char*)payload;
 // teisendame tüübi char järjendi sõneks
 String teema = (char*)topic;
 // Serial Monitori kaudu saame lugeda, mis kanalisse mis sisuga sõnum saabus
 Serial.print("Message arrived on topic: [");
 Serial.print(topic);
 Serial.print("], ");
 Serial.println(message);

 // relee kanal
 if(teema == subscribeRelay){
     // sõnumi ON puhul lülitame relee sisse
     if (message == "ON") {
       Serial.println("relay on");
       digitalWrite(RELAYPIN, HIGH);
     }
     // sõnumi OFF puhul välja
     if (message == "OFF") {
       Serial.println("relay off");
       digitalWrite(RELAYPIN, LOW);
     }
 }
 if(teema == subscribePump){
     if (message == "ON") {
       Serial.println("pump on");
       // pumba töö alustamise aeg
       pumpPreviousMillis = millis();
       // pump tööle ja aktiivseks
       digitalWrite(PUMPPIN, HIGH);
       pumpActivated = true;
     }
     if (message == "OFF") {
       Serial.println("pump off");
       // lülitame pumba välja ja teeme mitteaktiivseks
       digitalWrite(PUMPPIN, LOW);
       pumpActivated = false;
     }
 }
}

void reconnect() {
 // Kui klient ei ole ühendatud, proovime ühendada
 while (!client.connected()) {
```

```
   Serial.print("Attempting MQTT connection...");
   // püüame ühendust luua
   if (client.connect(clientID, mqtt_user, mqtt_password)) {
     Serial.println("connected");
     // kui ühendus on loodud, hakkame enda kanalit kuulama.
     client.subscribe(subscribePump, 1);
     client.subscribe(subscribeRelay, 1);
   } else {
     // kui ühenduse loomine ebaõnnestus, kuvame Serial Monitori põhjuse
     Serial.print("failed, rc=");
     Serial.print(client.state());
     Serial.println(" try again in 5 seconds");
     // Viie sekundi pärast proovime uuesti
     delay(5000);
   }
 }
}

void verifyFingerprint() {
 // kuvame Serial Monitori, kuhu ühendust luuakse
 Serial.print("Connecting to ");
 Serial.println(mqtt_server);
 // kui ühendust ei suuda luua, katkestame töö.
 if (! espClient.connect(mqtt_server, port)) {
   Serial.println("Connection failed. Halting execution.");
   // lõpmatu tsükkel, kus midagi ei tehta,
   // põhjustab arendusplaadi taaskäivitamise
   while(1);
 }
 // üritame sõrmejälge serveriga kinnitada
 if (espClient.verify(fingerprint, mqtt_server)) {
   Serial.println("Connection secure.");
 } else {
   // ebaõnnestunud katse korral ei ole ühendus turvaline
   Serial.println("Connection insecure! Halting execution.");
   while(1);
 }
}
```

Lisa 3. IFTTT funktsionaalsuse laiendamine Apilio abil

Järgnevalt tehakse läbi näide, kuidas automaatselt lülitada taimelampi. Esmalt tuleb minna Apilio.IO veebilehele ning registreerida kasutaja. Kui kasutaja on loodud, tuleb Apilio teenusesse sisse logida. Avaneb lehekülg, kus on ees nelja sammuga juhend Apilio teenuse seadistamise ja kasutamise kohta. Kui inglise keele oskus on vähemalt rahuldaval tasemel, ei tohiks juhendi järgi tegutsemine raskusi valmistada. Autor teeb siinkohal eelduse, et lugeja saab esmase seadistamisega ise hakkama.

Kui Apilio ja IFTTT teenus *Maker Webhooks* on juhendi järgi seadistatud, tuleb luua Apilios kaks numbrilise väärtusega muutujat leheküljel *"Numeric Variables"*. Üks muutuja, olgu selle nimeks *"light_level"*, on mõeldud anduri näidu hoidmiseks ning teine muutuja, olgu selle nimeks *"light_level_treshold"*, on muutumatu ning mõeldud piirväärtuseks (joonis 65). Kui muutuja *"light_level"* väärtus langeb alla piiri, läheb taimelamp põlema.



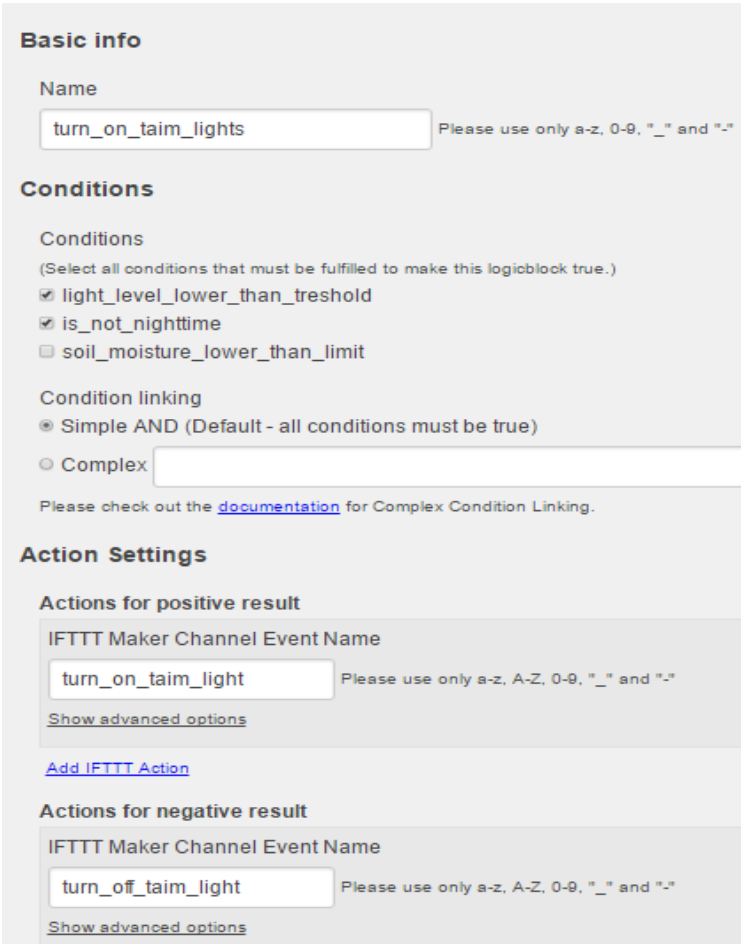
Joonis 65. Muutujad leheküljel *"Numeric Variables"*.

Lisada tuleks ka üks tõeväärtusega muutuja leheküljel *"Boolean Variables"*. Muutuja nimeks olgu *"night_time"* mille väärtuseks võib panna *"False"*, kui muutuja luuakse päevasel ajal.

Seejärel tuleks lisada kaks tingimust leheküljel *"Conditions"*: üks, mis oleks tõene, kui valguse näit on madalam kui määratud piir ja teine, mis oleks tõene, kui ei ole öine aeg (joonis 66).

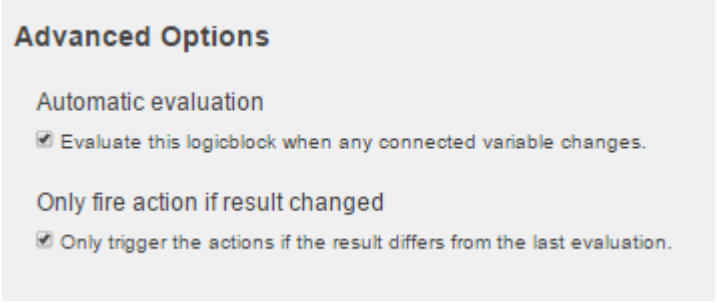
Joonis 66. Valguse taseme tingimus (vasakul) ja öise aja tingimus (paremal).

Seejärel tuleb teha tingimuslause leheküljel *"Logic Blocks"*, mis käivitab IFTTT Maker Webhooks teenuses tulede sisselülitamise sündmuse, kui mõlemad loodud tingimused on tõesed ehk ei ole öine aeg ning valguse tase on alla piiri (joonis 67). Kui vähemalt üks tingimustest on väär, käivitatakse tulede väljalülitamise sündmus.



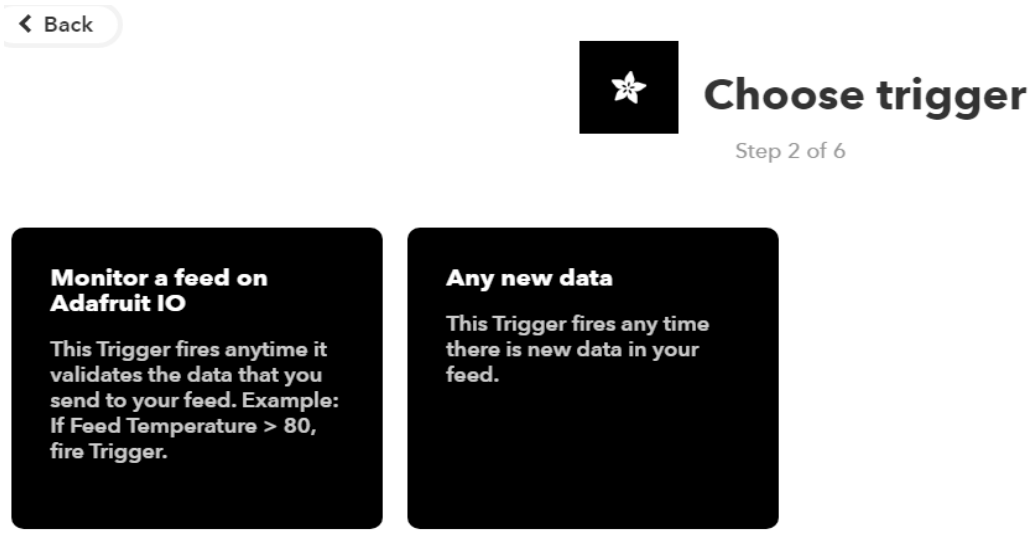
Joonis 67. Tingimuslause taimelambi lülitamiseks.

Veel tuleks tingimuslause loomise leheküljе all kaks linnukest märgistada, nagu on näidatud joonisel 68. Esimene neist, *"Automatic evaluation"* kontrolli tingimuslause kehtivus iga kord, kui mõne muutuja väärtus muutub, näiteks kui valguse kanalisse saabub uusi andmeid. Teise märgistamisel käivitatakse *Maker Webhooks* sündmus ainult siis, kui tingimuslause väärtus erineb lause viimasest väärtusest. Näiteks käivitatakse tulede sisselülitamine ainult siis, kui valguse tase on alla piiri ja öine aeg saab läbi. Samas ei lülitata tulesid korduvalt sisse, kui need on juba tingimuslause viimase kontrolli tulemusel sisse lülitatud.



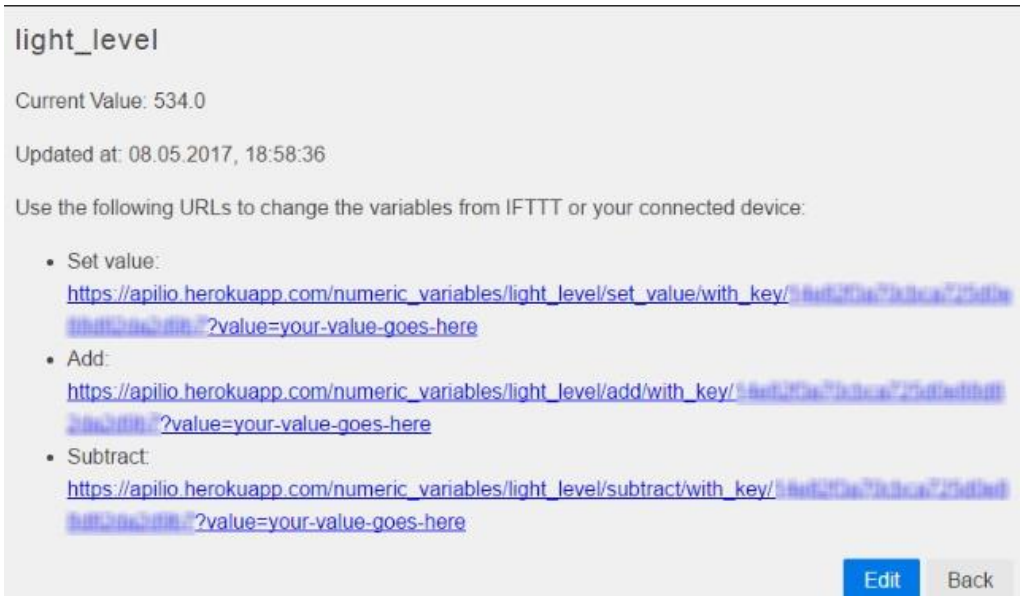
Joonis 68. Tingimuslause lisavõimalused.

Pärast tingimuslause loomist tuleb minna IFTTT veebilehele ning luua viis uut apletti. Esimene aplett käivitub, kui taime valguse kanalisse saabub uusi andmeid ning edastab saabunud andmed *Maker Webhooks* teenuse abil Apiliosse. Apleti "This" teenus on seega Adafruit. *"Choose Trigger"* sammu juures tuleb valida *"Any new data"* (joonis 69).



Joonis 69. Apleti loomise *"Choose Trigger"* samm.

Kanaliks on see, kuhu saadetakse informatsiooni valguse intensiivsuse kohta. Apleti "That" teenuseks tuleb valida Maker Webhooks ning *"Make a web request"*. Nõutud URL aadressi leiab Apiliost lehelt *"Numeric Variables"*, vajutades muutuja *"light_level"* juures olevale nupule *"Show"*. Aadress tuleb võtta "Set value" punkti alt (joonis 70).

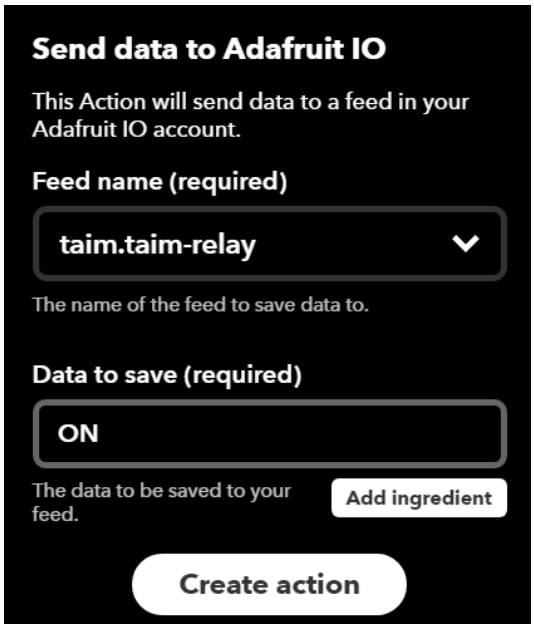


Joonis 70. Apilio muutuja informatsiooni leht.

Aadressis oleva võrdusmärgile järgneva teksti asemele tuleb kirjutada apleti loomise URL lahtris "{{Value}}" nagu on näidatud joonisel 71. Apleti käivitamisel on "{{Value}}" komponendi asemel kanalisse saabunud numbriline väärtus.

Joonis 71. Apleti loomine, URL lahter.

Järgmised kaks apletti on sarnased. Üks lülitab taimelambi sisse, teine lülitab välja. Mõlemal apletil on *"This"* teenuseks Maker Webhooks. Päästiku (*"Trigger"*) loomisel tuleb *"Event Name"* määrata selle järgi, milline on see nimetus Apilio tingimuslauses. Joonisel 67 on nimedeks määratud "turn_on_taim_light" ja "turn_off_taim_light". Esimese nime puhul on eesmärgiks tuli põlema panna. *"That"* teenuseks tuleb valida Adafruit ja kanaliks relee kanal ning kanalisse saadetavaks väärtuseks "ON" (joonis 72) või "OFF" vastavalt apleti eesmärgile.



Joonis 72. Õige kanali ja saadetava väärtuse valimine.

Viimased kaks apletti on kellaaja järgi Apilio muutuja tõeväärtuse muutmise jaoks. Selleks tuleb *"This"* teenuseks valida *"Date & Time"*, mis iga päev näiteks kell kuus hommikul määraks Apilios *"night_time"* muutuja väärtuse vääraks ning iga päev südaööl sama muutuja väärtuse tõeseks. *"That"* teenus on taaskord Maker Webhooks ja õiged aadressid leiab Apiliost *"night_time"* muutuja juurest vajutades nupule *"Show"*. Aplett, mis kell kuus käivitub, võiks kasutada URL-i *"Set false"* punkti alt ehk aadressi, mis määrab muutuja väärtuse vääraks. Südaööl tuleks siis määrata muutuja tõeseks ehk kasutada *"Set true"* punkti juures olevat URLi.

Kui Apilio on seadistatud ning IFTTT apletid loodud, on süsteem valmis ning võimeline automaatselt tulesid lülitama, arvestades nii valguse intensiivsusega kui kellaajaga. Selliselt saab määrata väga täpselt, kui palju kasvatatav taim ööpäevas valgust saab.

Lisa 4. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Märt Sessman**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Asjade internet WeMos näitel“, mille juhendajad on Anne Villems, Alo Peets ja Taavi Duvin,
   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **11.05.2017**