jamk.fi

Kosteuden ja lämpötilan mittauslaitteen suunnittelu ja toteutus

Timo Riekko

Opinnäytetyö
Marraskuu 2020
Tekniikan ala
Insinööri (AMK), tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Riekko, Timo	Opinnäytetyö, AMK	11 2020
	Sivumäärä 45	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Kosteuden ja lämpötilan mittauslaitteen suunnittelu ja toteutus

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Jari Hautamäki, Matti Mieskolainen

Toimeksiantaja(t)
eÄlytelli, Olli Väänänen

Tiivistelmä

Työn tavoitteena oli suunnitella ja toteuttaa mittalaite, joka tulisi eÄlytelli-hankkeen käyttöön. Yhtenä hankkeen osa-alueena on tutkia sekajätekasan itsesyttymisen ennaltaehkäisyä. Mittalaitteen tarkoitus olisi mitata sekajätekasasta lämpötilan ja kosteuden arvo, jotta voidaan nähdä, nouseeko kasan lämpötila liian korkeaksi ja aiheuttaisi itsesyttymisen riskin.

Mittalaitteen täytyi pystyä lähettämään mitatut arvot pilvipalveluun ja sen täytyi olla vähävirtainen. Arvojen lähettämiseksi ainoa ratkaisu oli LoRa-verkko, jota työssä tutkitaan ja käytetään.

Laitteeseen valittiin komponentit ja suunniteltiin fyysinen rakenne, sekä ohjelmoitiin laite toimimaan oikein. Näiden jälkeen voitiin rakentaa laite. Tiedonsiirtoa varten tutustuttiin LoRa-verkkoon ja toteutettiin ratkaisu, jossa arvot siirtyivät laitteelta pilveen. Tuloksena syntyi toimiva Internet of Things (IoT)-ympäristö, jossa mittalaite lähetti dataa LoRa-verkon ylitse ThingsBoard-palveluun, josta asiakas pystyi käydä katsomassa mitattuja arvoja.

LoRaa voidaan hyödyntää IoT-projekteissa ja se osoittautui hyvin käteväksi ja hyödylliseksi tekniikaksi kuuluvuuden ja vähäisen virrankulutuksen vuoksi. Komponenttien valinnalla oli iso merkitys järjestelmän toimivuuteen ja niitä vaihtamalla voitaisiin jatkokehittää mittalaitetta.

Avainsanat (asiasanat)
IoT, Esineiden Internet, LoRa, LoRaWAN, langaton tiedonsiirto

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Riekko, Timo	Bachelor's thesis	11 2020
		Language of publication:
	Number of pages 45	Permission for web publication: x

Title of publication

Designing and creating a temperature and moisture measuring device

Degree programme

Information and Communications Technology

Supervisor(s)

Jari Hautamäki, Matti Mieskolainen

Assigned by

eÄlytelli, Olli Väänänen

Abstract

The objective of the thesis was to design and create a device, which would be used by eÄlytelli project. One of the project's purposes is to research ways to prevent mixed waste pile self-ignition. The device would measure temperature and moisture of the pile to see if there is a risk of self-ignition.

The device had to send the measurements to the cloud and its power consumption had to be low. The only way to send the measurements was to use a LoRa-network which was researched and implemented in this thesis.

After choosing components, designing the device and programming it to work, it was time to start building it. To send the measurements to cloud it was necessary to research LoRa and use it with the device. As a result, there was a working Internet of Things (IoT) environment which gathers data and sends them to ThingsBoard service by using LoRa-network. A customer was able to read the measurements from ThingsBoard.

LoRa can be used in IoT systems and it turned out to be very handy and useful because of it has long range and low power consumption. The components had to be chosen carefully because they had big impact on the IoT system and by changing components it is possible to further develop the device.

Keywords/tags (subjects)

IoT, Internet of Things, LoRa, LoRaWAN, Wireless communication

Miscellaneous (Confidential information)

Sisältö

1 Johdanto 5

1.1 Tavoitteet 5

1.2 Toimeksiantaja 5

1.3 Tutkimusmenetelmät 5

1.4 Aiemmat toteutukset 6

2 Työn teoria 7

2.1 IoT 7

2.2 LoRa 8

2.2.1 Modulaatio 9

2.3 LoRaWAN 11

2.3.1 Päätelaitetyypit 13

2.3.2 Digitan LoRa-verkko 14

2.4 Datan käsittely ja visualisointi 15

2.4.1 Actility ThingPark 15

2.4.2 Node-RED 16

2.4.3 ThingsBoard 16

3 Mittalaitteen suunnittelu ja toteutus 17

3.1 Vaatimusmäärittely 17

3.2 Komponentit 18

3.3 Fyysinen rakenne 19

3.4 Mittalaitteen toteutus 20

3.4.1 Lämpötilan ja kosteuden mittaus 20

3.4.2 Kotelon ja putken rakennus 21

4 Mitattujen arvojen lähetys pilveen 24

4.1 Laitteen lisäys LoRa-verkkoon 24

4.2 Viestin lähetys laitteelta LoRa-verkkoon 26

4.3 LoRa-viestien käsittely ja ohjaus pilveen 27

4.4 Arvojen visualisointi käyttäjälle 30

4.5 Laitteen viimeistely 31

4.5.1 Virrankulutuksen optimointi ja akunkesto 31

5 Testaus ja johtopäätökset 32

6 Pohdinta 34

Lähteet 36

Liitteet 39

Liite 1. LoPy4 ohjelmointi 39

Kuviot

Kuvio 1 IoT-arkkitehtuuri 7

Kuvio 2 Viserrysten muodostuminen (Ruano 2016) 10

Kuvio 3 Kaistanleveyden ja hajautuskertoimen vaikutus viserrykseen (Ruano 2016) 11

Kuvio 4 LoRaWAN-verkon rakenne 12

Kuvio 5 OTAA-aktivoinnin toiminta 13

Kuvio 6 Päätelaitetyyppien eroavaisuudet. 14

Kuvio 7 Digitan IoT-verkon kuuluvuus (IoT:n kartta 2020) 15

Kuvio 8 Thingsboardin arkkitehtuuri (Colasante n.d) 17

Kuvio 9 Fyysisen rakenteen suunnitelma 19

Kuvio 10 Kotelon suunnitelma 19

Kuvio 11 Anturin ja näytön toiminta 20

Kuvio 12 Kannen hahmotelma 21

Kuvio 13 Kannen lopputulos 21

Kuvio 14 Putken pään valmistelu anturille 22

Kuvio 15 Kotelon ja kahvan kiinnitys 22

Kuvio 16 Rakennettu mittalaite 23

Kuvio 17 Device EUI selvitys 24

Kuvio 18 Laitteen lisääminen 25

Kuvio 19 LoRa-verkkoon liittyminen 25

Kuvio 20 LoRa-verkkoon liittymisen tarkastus 26

Kuvio 21 Viestin lähetys LoRa-verkkoon....26
Kuvio 22 Viesti LoRa-verkossa....27
Kuvio 23 Päätepalvelimen määritys....28
Kuvio 24 Reititysasetukset....28
Kuvio 25 Hexadesimaalin purku....29
Kuvio 26 Käsittelyketju Node-Redissä....29
Kuvio 27 Lämpötilan ja kosteuden arvot Thingsboardissa....30
Kuvio 28 Arvojen taulukko....30
Kuvio 29 Boot.py....31
Kuvio 30 Akunkesto....32
Kuvio 31 Mittaukset tunnin välein....33

Lyhenteet

ABP	Activation By Personalization
AS	Autonomous System
CoAP	Constrained Application Protocol
CSS	Chirp Spread Spectrum
HTTP(S)	Hypertext Transfer Protocol (Secure)
IBM	International Business Machines Corporation
IoT	Internet of Things
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
MQTT	Message Queuing Telemetry Transport
OTAA	Over-The-Air Activation
SF	Spreading Factor
USB	Universal Serial Bus

1 Johdanto

1.1 Tavoitteet

Opinnäytetyön tavoitteena on suunnitella ja rakentaa mittalaite, joka mittaa sekajätekasasta lämpötilan ja kosteuden. Sekajätekasoissa tapahtuu biojätteen hajoamista ja se voi aiheuttaa kasaan liian korkean lämpötilan. Korkea lämpötila altistaa sekajätekasan itsesyttymiselle ja syttymisen tapahduttua se on vaikea sammuttaa kasojen koon vuoksi. Mittalaitteella ennaltaehkäistään syttymistä ajoissa seuraamalla lämpötilaa ja kosteutta kasasta.

Mitatut arvot voi käyttäjä halutessaan lähettää pilvipalveluun, josta arvoja voidaan tarkastella etänä. Laitteeseen täytyi valita komponentit, suunnitella fyysinen rakenne, ohjelmoida ja rakentaa komponenteista mittalaite. Lopputuloksena syntyy laite, joka tulee toimeksiantajan käyttöön.

1.2 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimii Jyväskylän Yliopiston ja Jyväskylän ammattikorkeakoulun IT-instituutin yhteinen eÄlytelli-hanke. Hankkeen tarkoituksena on kehittää ja tuoda tietoisuutta teollisen internetin ja data-analyysin mahdollisuuksista (eÄlytelli-hanke n.d). Osana hanketta on tutkia sekajätekasan itsesyttymistä Sammakkokangas Oy:n kanssa.

1.3 Tutkimusmenetelmät

Opinnäytetyön tutkimusmenetelmänä on kehittämistutkimus. Kehittämistutkimuksessa pohjaudutaan teoriaan ja esitetään ongelma, jota tutkimuksessa aletaan kehittämään ja ratkaisemaan. Ongelmaa tarkastellaan ja kehitetään todellisissa olosuhteissa. Kehittämisen aikana syntyy uutta teoriaa, jota voidaan hyödyntää muissa tutkimuksissa. (Pernaa 2013.)

Opinnäytetyössä esitettiin ongelma, jota varten tutustuttiin eri aiheiden teoriaan ja niiden avulla kehitettiin ja toteutettiin fyysinen mittalaite. Työssä syntyi myös IoT-ympäristö, jota voitiin halutessa muokata ja kehittää opinnäytetyön tietoja hyödyntämällä.

1.4 Aiemmat toteutukset

Opinnäytetyössä hyödynnettiin muita IoT-toteutuksia, joita löytyi internetistä. Toteutuksia oli LoRa-verkosta, työssä käytettävistä komponenteista ja pilvipalveluista. Näitä toteutuksia hyödyntämällä ja yhdistämällä saatiin paljon hyödyllistä tietoa opinnäytetyön toteuttamiseen. Suurin hyöty saatiin IoT-toteutuksista, joissa mitattiin maaperän kosteutta anturin ja LoPy4-piirin avulla. Täysin samanlaista toteutusta opinnäytetyön kanssa ei ollut olemassa, joten työssä vaadittiin paljon kehitystyötä.

Aikaisempia opinnäytetöitä löytyi ja niitä hyödynnettiin mittalaitteen rakennuksessa. Esineiden internettiin, LoRaan ja LoPy4-alustaan liittyviä opinnäytetöitä löytyi ja niistä sai hyvän käsityksen IoT:n ja LoRan toiminnasta. Harri Jäntin kirjoittama opinnäytetyö IoT-projektin dokumentoinnista auttoi mittalaitteen rakennuksessa, koska Jäntin työssä tutkittiin esimerkiksi LoPy4-alustan ja LoRan toimintaa IoT-ympäristössä (Jäntti 2019).

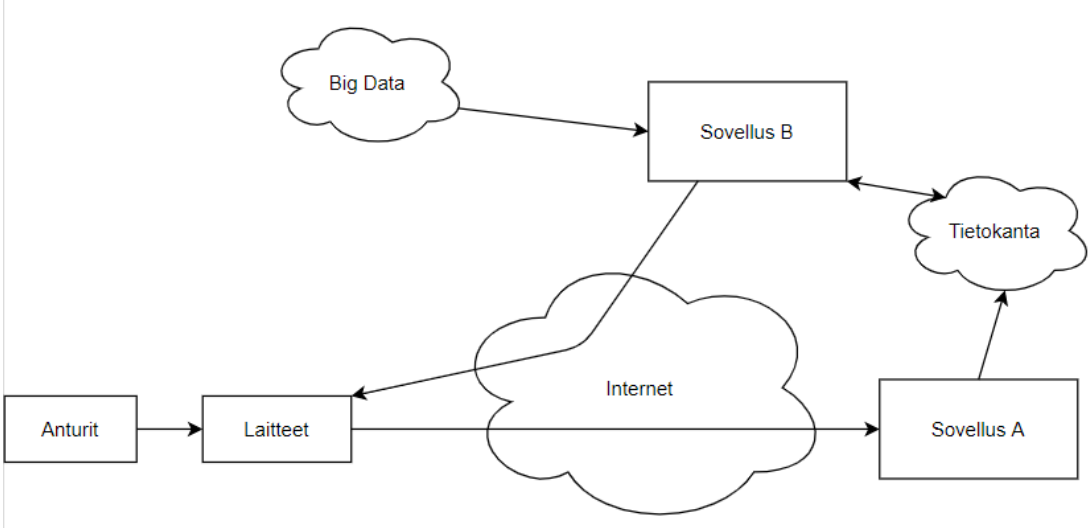
Kaupallisia toteutuksia löytyi, mutta yksikään ei vastannut täysin opinnäytetyön mittalaitteen vaatimuksia. Kaupallisissa toteutuksissa usein puuttui näyttö, tai anturi oli vääränlainen. Actility valmistaa LoRa-verkossa toimivia mittauslaitteita ja ne ovat ostettavissa verkkokaupasta, mutta ei löytynyt sopivaa laitetta opinnäytetyön tarkoitusta varten. Opinnäytetyön mittalaitteen fyysiseen rakenteeseen saatiin mallia TEMPPI-nimisestä kaupallisesta mittalaitteesta. TEMPPI on Kasvutaito Oy:n valmistama lämpömittari turpeen ja kompostin lämpötilan mittaamiseen (TEMPPI Mallit 2014).

2 Työn teoria

2.1 IoT

IoT (Internet of Things), eli esineiden internet on termi, jolla tarkoitetaan järjestelmää, jossa laitteita on kytketty internet-verkkoon ja ne voivat kommunikoida keskenään älykkäästi. Laitteet voivat esimerkiksi kerätä, analysoida ja visualisoida dataa. Esineiden internetin laitteita ei ihmisen tarvitse aina ohjata, vaan laitteet pystyvät toimimaan itsenäisesti. (Chaudhuri 2019.) IoT:n päätarkoitus ei ole teknologian kehittäminen, vaan kustannusten ja turhan työn vähentäminen sekä tehokkuuden parantaminen (Mitä IoT tarkoittaa? n.d).

IoT-järjestelmän toimintaa voidaan esittää arkkitehtuurikuvalla, josta ilmenee laitteiden tehtävät ja niiden välinen kommunikointi järjestelmässä. Kuviossa 1 on esitetty esimerkki IoT-arkkitehtuurista, jossa anturit lähettävät dataa laitteille ja laitteet lähettävät sen eteenpäin verkkoon sovellus A:lle. Sovellus käsittelee dataa ja tallentaa sen tietokantaan, josta sovellus B alkaa käsittelemään sitä. Sovellus B voi myös halutessaan lähettää laitteille käskyjä verkon ylitse tai poimia dataa internetistä.



Kuvio 1 IoT-arkkitehtuuri

IoT-laitteiden määrä on ollut kovassa kasvussa viime vuosina. Vuonna 2018 laitteita oli arviolta 7 miljardia ja vuonna 2019 26,6 miljardia. Ennusteiden mukaan 2020 laitteiden määrä ylittää 31 miljardia ja vuoteen 2025 mennessä laitteita olisi 75 miljardia. (Maayan 2020.)

IoT-järjestelmät voidaan jakaa karkeasti kahteen ryhmään. Kaupallinen IoT ja teollinen IoT. Kaupalliset IoT-tuotteet ovat pääosin tarkoitettu yksittäisille ihmisille tai perheille. Esimerkiksi älykodin tuotteet, puettavat älylaitteet ja puheentunnistusjärjestelmät ovat kaupallisia IoT-tuotteita. Kaupalliset tuotteet helpottavat ihmisten elämää esimerkiksi kotona, kaupoissa tai missä tahansa. (IoT: Consumer & Commercial vs. Industrial - Main overview 2019.)

Teolliset tuotteet pyrkivät parantamaan tuotettavuutta ja tehokkuutta teollisuudessa. Esimerkiksi tehtaat, kaupungit ja maatilat käyttävät teollisia IoT-järjestelmiä. Teollisuudessa on ollut valmiina jo automaattisia järjestelmiä ja niitä pyritään parantamaan IoT:n avulla. (IoT: Consumer & Commercial vs. Industrial - Main overview 2019.)

2.2 LoRa

LoRa (long range) on modulaatioratkaisu, jonka on kehittänyt yhdysvaltalainen Semtech-yritys. LoRan ominaisuuksiin kuuluu pieni tehonkulutus ja erinomainen radiokuuluvuus pitkillä etäisyyksillä. Lisäksi se on tosi häiriösietoinen ja se mahdollistaa pienien datapakettien siirtämisen. LoRa on myös avoimen lähdekoodin tekniikka, eli jokainen voi käyttää sitä vapaasti. (Seneviratne 2019.)

LoRan etu on sen mahdollistama pitkä kantama, pieni tehonkulutus ja edullisuus. Pienellä hinnalla voidaan rakentaa IoT-laite, jonka akku kestää jopa vuosia. LoRa-signaali toimii hyvin myös rakennusten läpi, koska sen taajuus on niin pieni.

LoRa käyttää Euroopassa 868 MHz taajuutta. Tiedonsiirto onnistuu 0,3-22 kbit/s nopeudella ja kantama on jopa 15-45 kilometriä tasaisessa maastossa. (Ruano 2016.)

Suomessa toimii kaupallinen LoRa-verkko, joka on Digita Oy:n ylläpitämä. Digitan verkko toimii koko Suomen alueella 868 MHz taajuudella. Suomessa ja myös muualla maailmassa toimii myös ilmainen The Things Network, johon kuka tahansa voi lisätä tukiaseman. The Things Networkin verkko ei kuitenkaan kata koko Suomea, joten sitä voidaan käyttää vain tietyissä paikoissa.

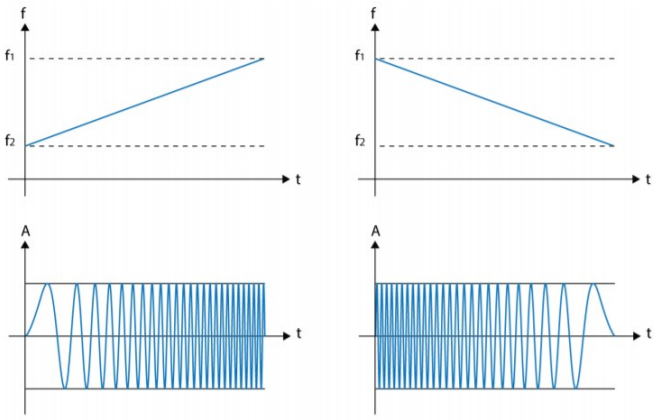
2.2.1 Modulaatio

Modulaatio tarkoittaa prosessia, jossa radiosignaalia muokataan niin, että se mahdollistaa datan siirron. Data voidaan lisätä signaalin korkeataajuiseen kantoaaltoon. Vastaavasti vastaanotin voi demodulaation avulla poimia kyseisen datan signaalista. Modulaatiota tarvitaan, koska tiedonsiirto matalilla taajuuksilla on erittäin haastavaa. Modulaatioita on analogisia ja digitaalisia. (What is modulation? 2018.)

LoRa perustuu CSS (Chirp Spread Spectrum) modulaatioon, eli chirp-hajaspektrimodulaatioon. Kyseinen tekniikka on kehitetty jo 1940-luvulla ja sitä on käytetty esimerkiksi sotilasympäristössä tutkissa. (Seneviratne 2019.)

Chirp-hajaspektrimodulaatiossa ajan muuttuessa taajuutta muutetaan, mutta amplitudi pysyy samana. Modulaatiossa muodostuu viserryksiä (engl. chirp). Taajuuden muuttuessa ylöspäin muodostuu up-chirp ja alaspäin muuttuessa down-chirp. Näiden avulla saadaan data välitettyä signaalissa. (Ruano 2016.)

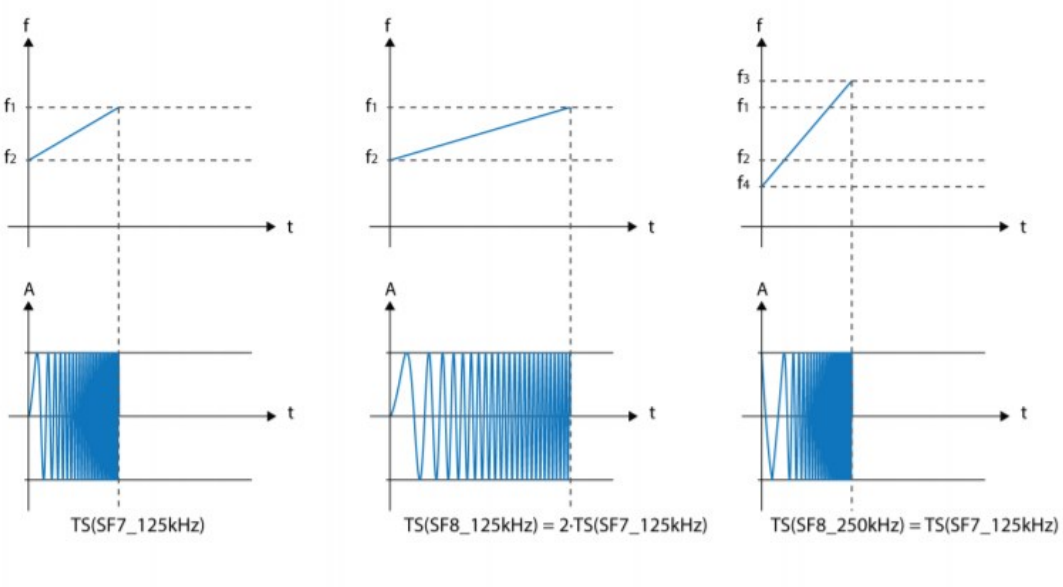
Kuviossa 2 näkyy viserrysten muodostuminen taajuutta (pystyakseli) muuttamalla ajan (vaaka-akseli) suhteen.



Kuvio 2 Viserrysten muodostuminen (Ruano 2016)

LoRa tukee kaistanleveyksiä 125kHz, 250kHz ja 500 kHz. Alhaiset kaistanleveydet mahdollistavat pitkän kantaman signaalille ja ne vaikuttavat viserryksen kestoon. Kaistanleveyden tuplaantuessa viserryksen kesto puolittuu. Viserryksen kestoon vaikuttaa kaistanleveyden lisäksi hajautuskerroin (Spreading Factor, SF). Hajautuskertoimia on kuusi, SF7-SF12. Yksi isompi hajautuskerroin nostaa aina viserryksen kestoa kaksinkertaiseksi. Suuremmalla hajautuskertoimella voidaan myös lähettää enemmän dataa. (Ruano 2016.)

Kuviossa 3 havainnollistetaan kaistanleveyden ja hajautuskertoimen vaikutusta viserrykseen. Kuviosta voidaan todeta, että SF8 viserryksen kesto on kaksinkertainen SF7 verrattuna, mutta jos käytetään tuplasti isompaa taajuutta, niin viserrysten kestot ovat samat.



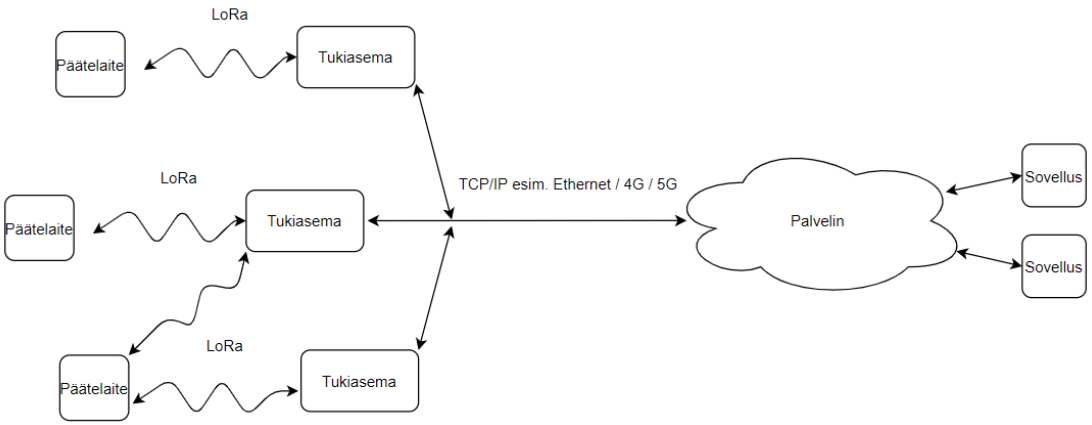
Kuvio 3 Kaistanleveyden ja hajautuskertoimen vaikutus viserrykseen (Ruano 2016)

2.3 LoRaWAN

LoRaWAN (Long Range Wide Area Network) on verkko, joka koostuu päätelaitteista, tukiasemista, palvelimesta ja sovelluksista. Nämä kommunikoivat keskenään ja näin muodostavat oman verkkonsa.

Päätelaite kommunikoi LoRa-signaalin avulla tukiasemiin, jotka kuuntelevat liikennettä. Viestin saatuaan tukiasema välittää sen palvelimelle internet-yhteydellä. Palvelin määrittää, mitä tukiasemaa päätelaite voi käyttää liikenteeseen ja saako kyseinen päätelaite liittyä verkkoon. Palvelin voi sisältää sovelluksia, jotka käsittelevä päätelaitteilta saatuja viestejä. Sovellukset voivat olla myös erillisissä palvelimissa, joihin voidaan lähettää viestejä LoRa-verkon palvelimelta. (What is LoRaWAN? 2015.)

Kuviossa 4 havainnollistetaan LoRaWAN-verkon rakenne, jossa on päätelaitteita, tukiasemia, palvelin ja sovelluksia. Tukiasemat ovat yhteydessä palvelimeen esimerkiksi Ethernet-, 4G- tai 5G-yhteydellä.

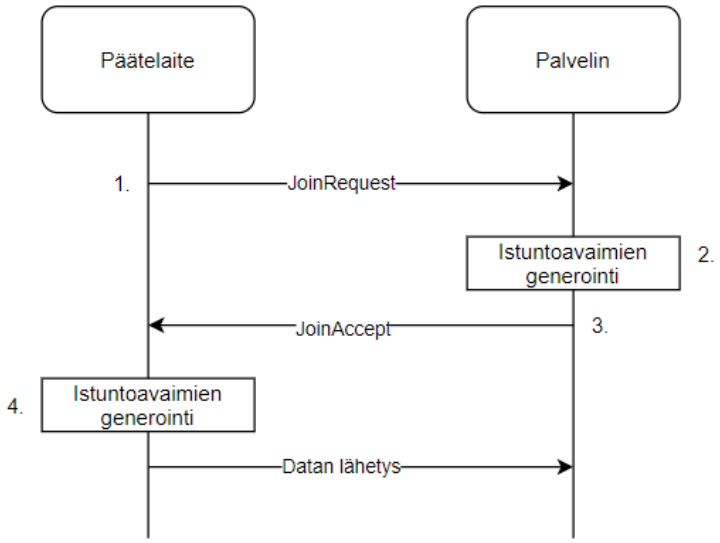


Kuvio 4 LoRaWAN-verkon rakenne

Päätelaitteen yhdistyminen verkkoon vaatii tunnistautumisen. Laitteen tunnistautuminen toteutetaan kahdella eri aktivointimenetelmällä.

OTAA (Over-The-Air Activation) on yleisempi aktivointitapa, jossa päätelaite lähettää viestin tukiasemalle, että se haluaa liittyä verkkoon. Tätä viestiä sanotaan JoinRequest-viestiksi. Viesti sisältää DevEUI- ja AppEUI-avaimet, sekä DevNonce luvun. DevEUI on jokaisen päätelaitteen uniikki avain, jolla voidaan tunnistaa, mikä päätelaite on kyseessä. AppEUI-avaimella määritetään, mihin sovellukseen päätelaite haluaa lähettää viestejä. DevNonce on luku, jolla voidaan tarkastaa, ettei samaa viestiä lähetetä useampaan kertaan. (LoRa – Device Activation Call Flow (Join Procedure) using OTAA and ABP 2018.)

JoinRequest-viestin saatuaan tukiasema välittää sen palvelimelle laitteen liittymisen sallimiseksi. Sallittuaan liittymisen palvelin generoi laitteen tunnistautumisavaimen ja lähettää laitteelle JoinAccept-viestin. JoinAccept-viesti sisältää tiedot, joilla päätelaite voi generoida istuntoavaimet. Viesti lähetetään sen tukiaseman kautta, jolla on vahvin signaali päätelaitteelle. Tämän jälkeen päätelaite generoi istuntoavaimen JoinAccept-viestin avulla ja sen jälkeen datan lähettäminen palvelimelle on mahdollista. (LoRa – Device Activation Call Flow (Join Procedure) using OTAA and ABP 2018.) Kuviossa 5 esitetään OTAA-aktivoinnin toiminta.



Kuvio 5 OTAA-aktivoinnin toiminta

ABP (Activation By Personalization) on aktivointitapa, joka ei ole niin yleinen kuin OTAA. ABP-aktivoinnissa ei käytetä DevEUI- tai AppEUI-avaimia, eikä päätelaite suorita samanlaista liittymisprosessia kuin OTAA-aktivoinnissa. ABP-aktivoinnissa laitteelle ja palvelimelle on konfiguroitu samat istuntoavaimet, joten niitä ei generoida automaattisesti. Istuntoavaimien ollessa samat voi päätelaite lähettää dataa palvelimelle ilman liittymisprosessia. (LoRa – Device Activation Call Flow (Join Procedure) using OTAA and ABP 2018.)

ABP-aktivointi ei ole yhtä tietoturvallinen kuin OTAA istuntoavaimien vuoksi. Päätelaitteeseen voi joku päästä käsiksi ja se mahdollistaa istuntoavaimien kopioimisen. Istuntoavaimilla voi lähettää mistä laitteesta tahansa viestejä kyseiseen istuntoon.

2.3.1 Päätelaitetyypit

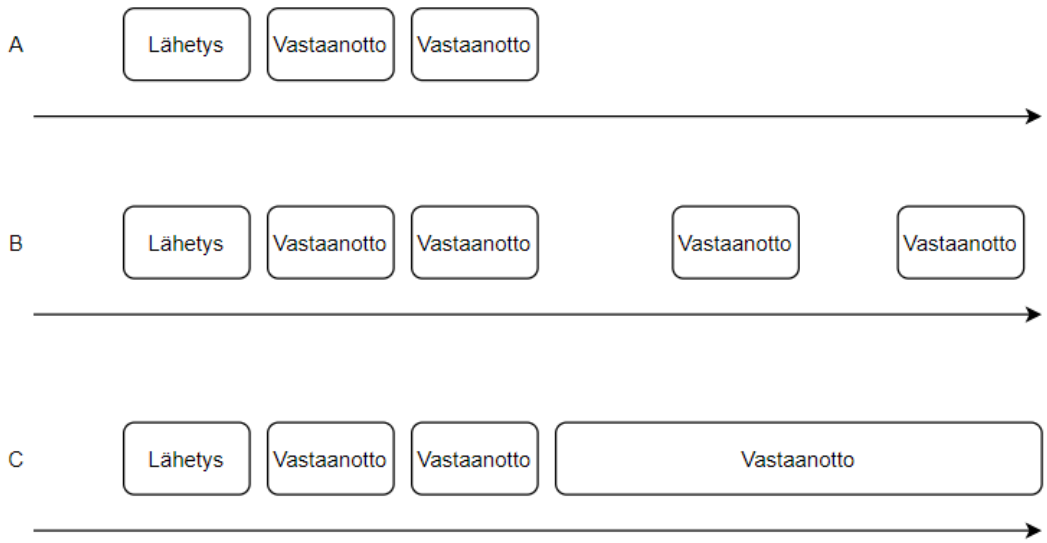
LoRa-päätelaitteet voidaan jakaa kolmeen eri luokkaan A, B ja C.

A-luokan päätelaitteet voivat kommunikoida molempiin suuntiin, eli ne voivat lähettää ja vastaanottaa dataa. Laite aloittaa viestin lähetyksen jälkeen kuuntelemaan tulevaa liikennettä kahden lyhyen aikaikkunan ajan. Tuleva viesti ei tule perille, jos se lähetetään aikaikkunan ulkopuolella. (What is LoRaWAN? 2015.)

B-luokan päätelaitteet toimivat kuten A-luokan laitteet, mutta B-luokan laitteissa on ylimääräinen aikataulutettu ikkuna, jolloin laitteisiin voidaan lähettää viestejä. Päätelaitteet osaavat avata ikkunan oikeaan aikaan Beacon-viestien avulla. Beacon-viesti tulee tukiasemalta ja viestin saatuaan päätelaite avaa aikaikkunan viestin vastaanottamiseen. (What is LoRaWAN? 2015.)

C-luokan päätelaitteet ovat myös kaksisuuntaisia ja ne kuuntelevat koko ajan tulevaa liikennettä, paitsi silloin, kun ne itse lähettävät viestejä. (What is LoRaWAN? 2015.)

Päätelaitetyypin valinta vaikuttaa akunkestoon ja viiveeseen. A-luokan laitteet ovat parhaita akunkestoa ajatellen, mutta niissä on isompi viive C-luokan laitteisiin verrattuna. Kuviossa 6 havainnollistetaan päätelaitetyyppien eroavaisuudet. Kuviossa esitetään lähetyksen ja vastaanoton aikaikkunat ajan kuluessa.

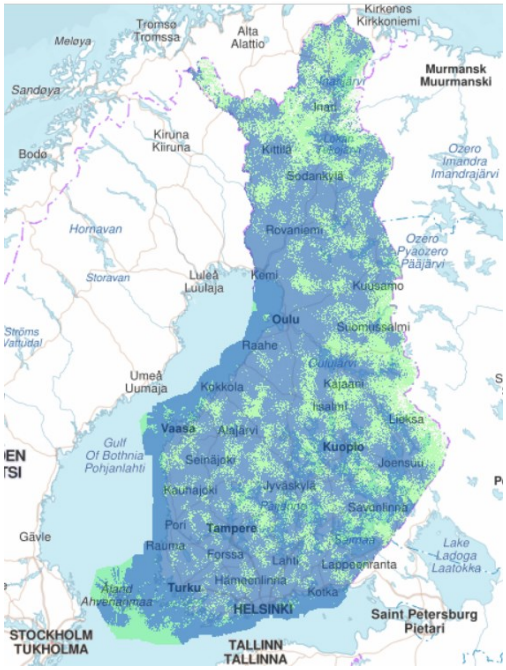


Kuvio 6 Päätelaitetyyppien eroavaisuudet.

2.3.2 Digitan LoRa-verkko

Digita tarjoaa IoT-verkkoa koko Suomen alueella ja tämä verkko tukee LoRa-teknologiaa. Verkko on Digitan valvonnassa ympäri vuoden. Verkon kuuluvuus on todella hyvä ja kehittyy jatkuvasti. Kuviossa 7 näkyy Digitan IoT-verkon kuuluvuus Suomessa.

Sininen väri tarkoittaa sisäkuuluvuutta ja vihreä väri ulkokuuluvuutta. Kuviosta voidaan todeta, että verkko on koko Suomen kattava.



Kuvio 7 Digitan IoT-verkon kuuluvuus (IoT:n kartta 2020)

Digitan verkkoa hyödyntää Suomessa monet vesi- sähkö ja lämmönjakeluyhtiöt. Verkko mahdollistaa esimerkiksi etäluettavat vesi-, lämpö- ja sisäilmamittarit. Uutta käyttäjäkuntaa syntyy koko ajan lisää. (Esineiden internet valtaa jokaista toimialaa 2020.)

2.4 Datan käsittely ja visualisointi

2.4.1 Actility ThingPark

Actility ThingPark on IoT-alusta, jota Digita Oy:n LoRa-verkko käyttää. Se tarjoaa palveluntarjoajille erilaisia työkaluja LoRa-verkon käyttöön, valvontaan ja hallintaan. Actilityn tavoitteena on tuottaa IoT-palveluita joka puolelle maapalloa. (The Actility product portfolio n.d.)

Alustaa käyttää tällä hetkellä yli 50 eri maiden isoa LoRaWAN-palveluntarjoajaa ja näiden verkoissa on yhteensä yli 35000 LoRa-antennia (Actility is the world leader in IoT networks management n.d.) Lähes jokainen LoRa-verkko käyttää Actility ThingPark-alustaa, joten kyseessä on todella suosittu ja maailmanlaajuinen palvelu.

2.4.2 Node-RED

Actility ThingPark-alustasta voidaan lähettää data eteenpäin johonkin, missä se voidaan muuttaa selkokieliseen muotoon. Tähän tarkoitukseen Node-RED sopii hyvin. Se on kevyt palvelu, jota voidaan ajaa palvelimella. Node-RED voidaan asettaa kuuntelemaan viestejä, ja viestin saatuaan palvelu poimii sen ja käsittelee viestin dataa ja lähettää sen eteenpäin.

Node-RED-palvelun on kehittänyt IBM (International Business Machines Corporation) vuonna 2013. Se on rakennettu avoimeen lähdekoodiin perustuvan Node.js-ympäristön päälle, joka mahdollistaa JavaScript-koodikielen ajamisen palvelimella. Palvelussa on käyttöliittymä, johon pääsee esimerkiksi selaimella. Käyttöliittymässä voidaan määrittää funktioita datan ohjaamiseksi eteenpäin. (About n.d.)

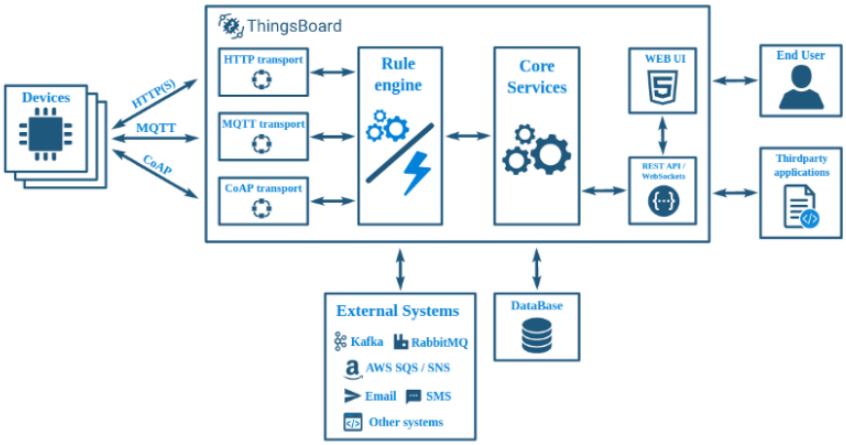
Node-Redin voi ladata ilmaiseksi palvelun nettisivuilta. Sivuilla on ohjeet palvelun asentamiseen esimerkiksi dockeriin, raspberry pi -alustaan tai virtuaalikoneelle. (Get Started n.d.)

2.4.3 ThingsBoard

Node-RED-palvelusta voidaan lähettää data eteenpäin visualisoitavaksi esimerkiksi ThingsBoardiin, joka on avoimen lähdekoodin IoT-alusta. Alustassa voidaan kerätä, analysoida ja visualisoida dataa. Päätelaitteille voidaan myös lähettää viestejä alustasta. ThingsBoard on skaalautuva, luotettava ja muokattavissa oleva alusta. Käyttäjä voi tehdä laitteiden keräämästä datasta eri näköisiä widgettejä, joilla visualisoidaan esimerkiksi lämpötilan ja kosteuden arvoa. (What is ThingsBoard? n.d.)

ThingsBoardiin voidaan lähettää dataa käyttäen eri protokollia, kuten HTTP(S), MQTT ja CoAP. Datan saavuttua alustaan viesti kulkeutuu Rule engine -toiminnon läpi, jossa

dataviestiä voidaan muokata. Tämän jälkeen data kulkeutuu laitteen telemetriasivulle, josta voidaan tarkastella laitteelta saatuja viestejä. Kuviossa 8 on ThingsBoardin arkkitehtuuri, josta nähdään sen toiminta.



Kuvio 8 Thingsboardin arkkitehtuuri (Colasante n.d)

ThingsBoardin voi ladata ilmaiseksi sen verkkosivuilta. Asennuksen voi suorittaa moneen eri käyttöjärjestelmään. Palvelua voi myös testata ilman asentamista käyttämällä live demo -versiota. (Installation n.d.)

3 Mittalaitteen suunnittelu ja toteutus

3.1 Vaatimusmäärittely

Mittalaitteen täytyi pystyä mittaamaan sekajätekasan lämpötilan ja kosteuden arvot. Arvot täytyi myös voida lähettää pilvipalveluun visualisoitavaksi. Laitteessa täytyi olla virtakytkin, USB-liitin, antenni ja mittaustilakytkin.

Ensimmäisessä mittaustilassa laitteen tuli mitata koko ajan kosteutta ja lämpötilaa, sekä esittää arvot näytöllä. Käyttäjän täytyi pystyä lähettämään painonappia painamalla viimeksi mitatut arvot pilvipalveluun langattomasti LoRa-verkon avulla.

Toisessa mittaustilassa laitteen täytyi mitata arvot tunnin välein ja mennä mittausten välillä nukkumistilaan virran säästämiseksi. Mitattuaan arvot laitteen täytyi automaattisesti lähettää ne pilvipalveluun. Painonappia painamalla näytön täytyi syttyä ja käyttäjä pystyi tarkastella mitattuja arvoja näytöltä.

3.2 Komponentit

Sulautetuksi kehitysalustaksi valittiin LoPy4, koska se on helppokäyttöinen ja siinä on samassa myös LoRa-lähetin ja antenni. LoPy4 käyttää ohjelmointikielenään micropythonia. Alustaan kytkettiin akku, anturi, kaksi kytkintä, painonappi, näyttö, antenni ja USB-liitin.

Anturiksi valikoitui SHT-10, koska sillä saadaan mitattua kosteus ja lämpötila luotettavasti. Anturi kestää vettä ja siinä on kestävä kotelo, joten se kestää paremmin mekaanista rasitusta.

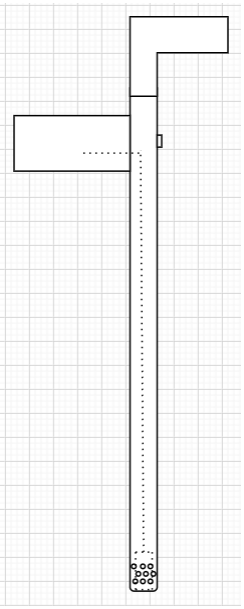
Näytöksi valittiin LCD1602, koska se on erittäin suosittu ja helppo asentaa. Näyttöön mahtuu 16x2 merkkiä ja siinä on taustavalo, joten näyttöä näkee lukea myös kirkkaalla säällä. Näyttö vaatii 5V jännitteen, joten näyttöä varten tarvittiin myös step-up-muunnin, joka muuntaa jännitteen 3,7 voltista 5 voltiksi.

Laitteen akuksi valittiin 3,7V ja 2000 mAh kokoinen litium polymeeriakku. Käyttäjä voi ladata akkua kytkemällä USB-johdon laitteeseen. Akun koko oli riittävä, koska laite käytti erittäin vähän virtaa.

Komponentit asennettiin Hammondin eloksoituun alumiinikoteloon. Kotelo valittiin sen kestävyyden ja ulkonäön vuoksi. Kotelo kiinnitettiin 1,5 metriä pitkän alumiiniputken päähän. Putken materiaaliksi valittiin alumiini keveyden ja lämmönjohtavuuden vuoksi.

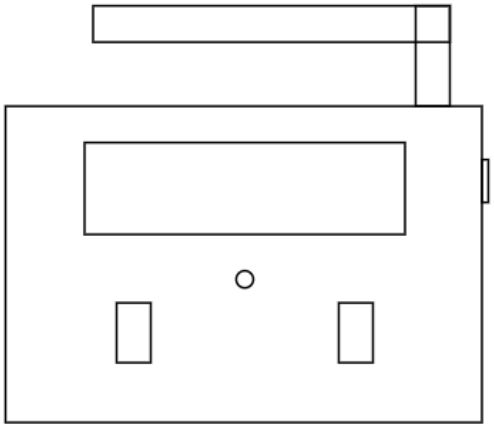
3.3 Fyysinen rakenne

Sekajätekasat voivat olla korkeita, joten alumiiniputken pituudeksi valittiin 1,5 metriä. Putken päähän tuli anturi, jota varten putkeen täytyi porata reikiä. Reiät auttoivat anturin ja sekajätekasan kontaktin parantamiseen. Putken toiseen päähän kiinnitettiin kotelo ja kahva, jotta laitetta on helpompi työntää kasaan. Laitteen rakenteen suunniteltu toteutus esitetään kuviossa 9.



Kuvio 9 Fyysisen rakenteen suunnitelma

Koteloon täytyi tehdä reiät näytölle, kytkimille, painonapille, antennille ja USB-liittimelle. Komponenttien paikkojen suunnitelma kotelon kannessa esitetään kuviossa 10.



Kuvio 10 Kotelon suunnitelma

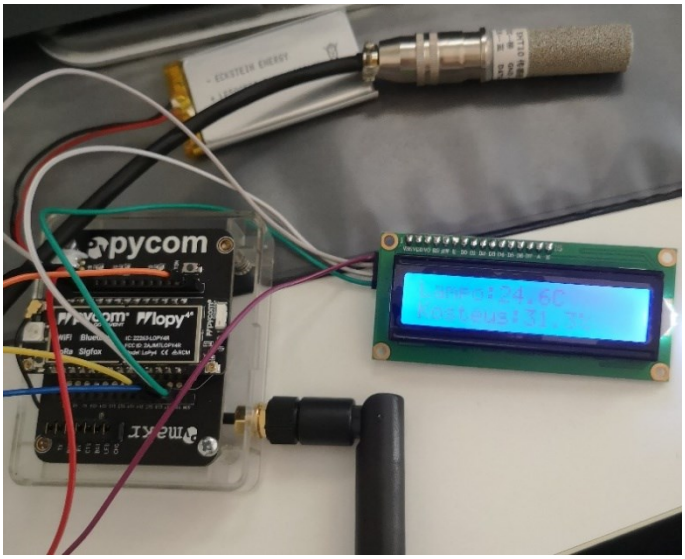
3.4 Mittalaitteen toteutus

3.4.1 Lämpötilan ja kosteuden mittaus

Laitteen toteutus aloitettiin päivittämällä Lopy4-alustan ohjelmistoversio uusimpaan. Tämän jälkeen testattiin anturin ja Lopy4-alustan toimintaa. Anturi kytkettiin alustaan ja ohjelmoitiin se mittaamaan lämpötilan ja kosteuden arvot.

Ohjelmointi tehtiin käyttämällä avoimeen lähdekoodiin perustuvaa sovellusta Atom, joka on ilmainen tekstieditori. Atom-sovellukseen asennettiin LoPy4-alustan tunnistautumista varten Pybytes-lisäosa. Lisäosan avulla tekstieditori osasi ajaa ohjelman LoPy4-alustalla.

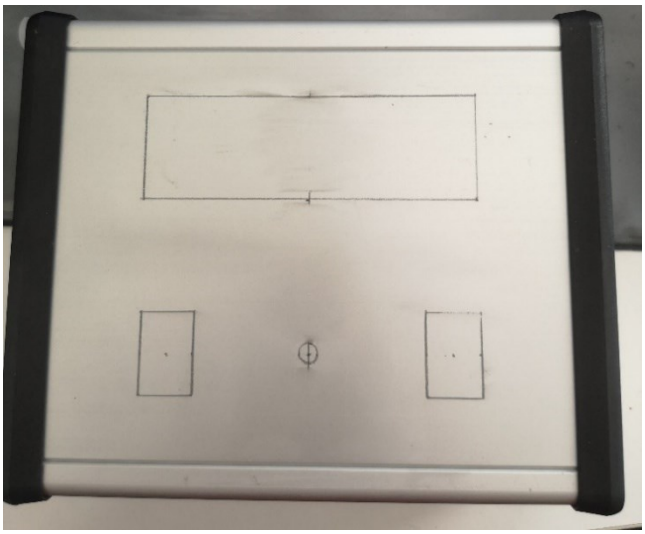
Seuraavaksi piiriin kytkettiin näyttö arvojen tarkastelua varten. Anturin ja näytön ohjelmointi onnistui hyödyntämällä muiden tekemiä ohjelmia, joita löytyi keskustelupalstoilta. Kuviossa 11 näkyy laitteen toiminta tässä vaiheessa.



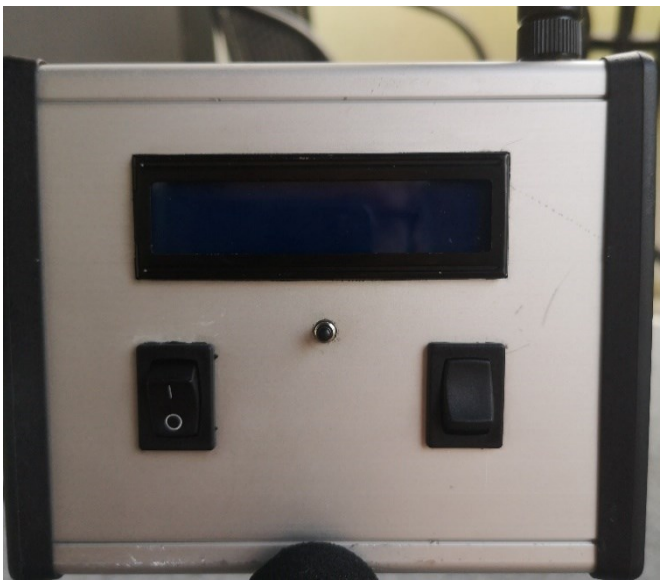
Kuvio 11 Anturin ja näytön toiminta

3.4.2 Kotelon ja putken rakennus

Komponenttien asentaminen vaati reikien tekemistä koteloon. Reikien teko onnistui poraamalla ja viilaamalla. Komponentit asennettiin ja liimattiin tehtyjen reikien kohdalle. Toteutus esitetään kuviossa 12 ja 13.



Kuvio 12 Kannen hahmotelma



Kuvio 13 Kannen lopputulos

Alumiiniputken päähän asennettiin anturi, jota varten täytyi putkeen porata reikiä. Reiät paransivat anturin tarkkuutta. Putken päähän asennettiin muovikärki, kun an-

turi oli saatu paikalleen. Kuviossa 14 näkyy putken pään toteutus, jossa anturi on paikallaan.

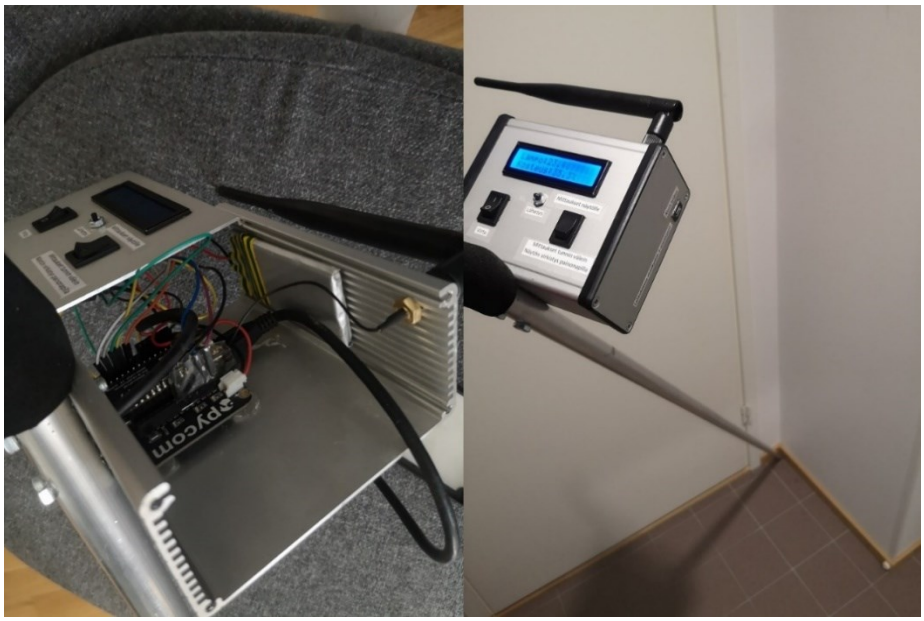
Kuvio 14 Putken pään valmistelu anturille

Kotelo kiinnitettiin putkeen poraamalla reiät ja kiinnittämällä osat toisiinsa pulteilla ja muttereilla. Lopuksi putkeen asennettiin kahva, jotta laitetta olisi helppo käsitellä. Kahva tehtiin taitetusta alumiiniputkesta, joka kiinnitettiin ruuvilla. Kuviossa 15 näkyy kiinnityksen ja kahvan toteutus.

Kuvio 15 Kotelon ja kahvan kiinnitys

Tämän jälkeen koteloon merkattiin ohjetekstit jokaisen kytkimen kohdalle käyttämisen helpottamiseksi. Lopuksi Lopy4 sekä akku kiinnitettiin koteloon, jolloin laitteen

rakennusvaihe oli valmis. Kuviossa 16 näkyy mittalaitteen lopputulos.



Kuvio 16 Rakennettu mittalaite

4 Mitattujen arvojen lähetys pilveen

4.1 Laitteen lisäys LoRa-verkkoon

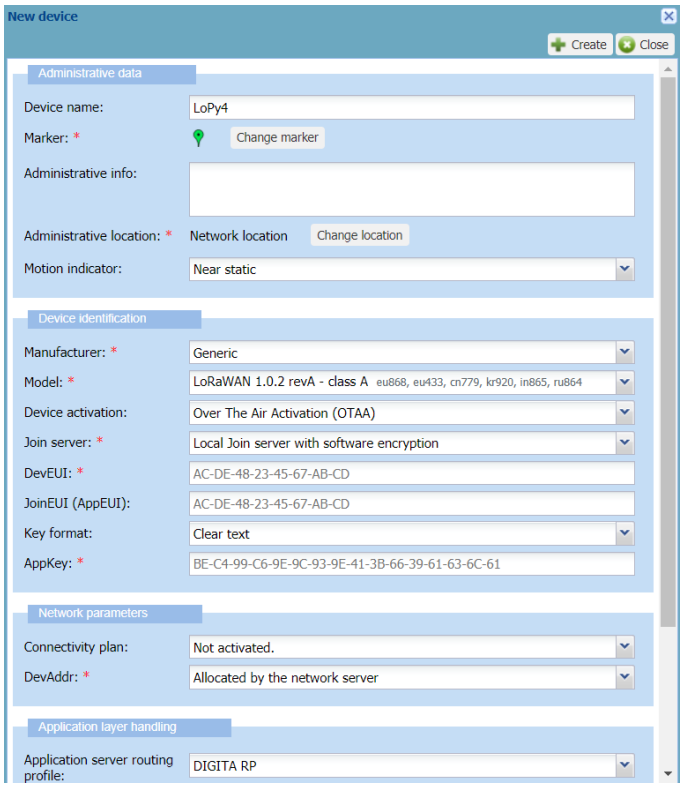
Jokaisella LoRa-laitteella on uniikki 8 tavun pituinen Device EUI arvo, jolla voidaan tunnistaa laite LoRa-verkossa. Laitteen Device EUI saatiin ajamalla laitteeseen kuvion 17 mukainen koodi.

```
from network import LoRa
import binascii
lora = LoRa(mode=LoRa.LORAWAN)
print(binascii.hexlify(lora.mac()).upper().decode('utf-8'))
```

Kuvio 17 Device EUI selvitys

Device EUI arvon selvittyä laite voitiin lisätä Digitan ThingPark-palvelun Device Manager -sivulle. Lisäystä varten piti asettaa ThingParkissa laitteelle nimi, valmistaja, malli, sekä laitteesta selvitetty Device EUI. Laitteen aktivointiin käytettiin OTAA-aktivointia. Lopuksi täytyi asettaa 8 tavun pituinen AppEUI-avain ja 16 tavuinen AppKey-avain. AppEUI- ja AppKey-avaimet keksittiin itse, koska ThingPark ei luonut niitä itsestään. Lopuksi laitteelle valittiin reititysasetukset, mutta niitä ei ollut vielä tehtynä. Kuviossa

18 esitetään laitteen lisäykseen tarvittavat tiedot.



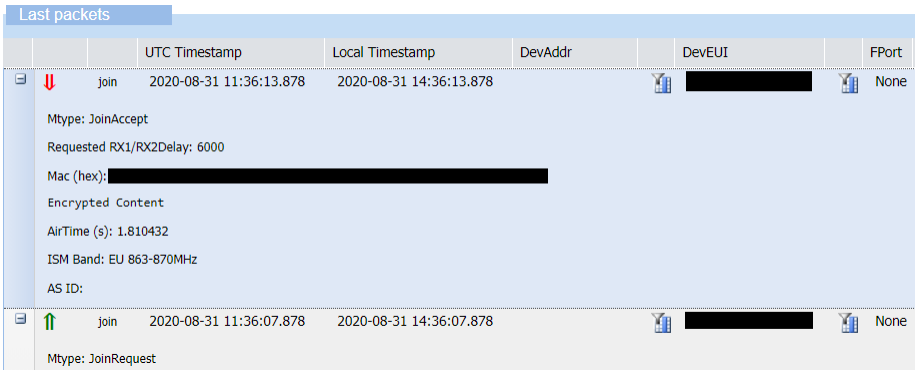
Kuvio 18 Laitteen lisääminen

Tämän jälkeen laitteen ohjelmaan voitiin lisätä äsken asetetut AppEUI ja AppKey. Lisäyksen jälkeen voitiin testata laitteen liittymistä verkkoon. Kuviossa 19 näkyy laitteen ohjelmasta funktio, jolla laite liittyi LoRa-verkkoon. AppEui ja AppKey ovat peitettynä tietoturvasyistä.

```
def JoinLora(): #LoRaan liittyminen
    lora = LoRa(mode=LoRa.LORAWAN)
    app_eui = binascii.unhexlify('[redacted]')
    app_key = binascii.unhexlify('[redacted]')
    lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout=0)
    while not lora.has_joined():
        for i in range(30):
            time.sleep(0.5)
            if lora.has_joined():
                break
```

Kuvio 19 LoRa-verkkoon liittyminen

Laitteen liittyessä verkkoon se lähetti ensin liittymispyynnön, jonka tukiasema vastaanotti. Tukiasema välitti pyynnön palvelimelle ja palvelin päätti, saiko laite liittyä verkkoon. Liittymisen sallittua tukiasema lähetti laitteelle hyväksymisviestin ja laite oli tämän jälkeen liittynyt verkkoon. Kuviossa 20 näkyy ThingParkin Wireless Logge-rista verkon viestit laitteen liittyessä verkkoon.



Kuvio 20 LoRa-verkkoon liittymisen tarkastus

4.2 Viestin lähetys laitteelta LoRa-verkkoon

Lämpötilan ja kosteuden arvojen lähettämiseksi LoRa-verkkoon täytyi ne muuttaa tavuiksi. Muuttaminen onnistui käyttämällä ustruct-nimistä ohjelmakirjastoa, jolla lämpötilan ja kosteuden arvot pakattiin yhdeksi muuttujaksi. Kuviossa 21 esitetään testausohjelma, jolla lähetettiin arvoja LoRa-verkkoon. Paketin lähetys oli mahdollista vain silloin, kun verkkoon oli liitytty.

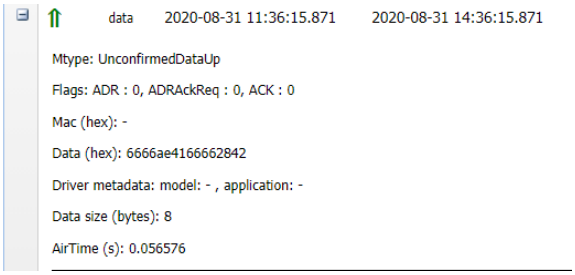
```
def sendpacket(packet): #LoRa-paketin lähetys
    s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
    s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
    s.setblocking(False)
    s.send(packet)

packet = ustruct.pack('2f', 21.8, 42.1)
sendpacket(packet)
```

Kuvio 21 Viestin lähetys LoRa-verkkoon

ThingParkin Wireless Loggerista voitiin tarkastaa, saako verkko laitteelta viestejä ohjelman ajettuaan. Viestin saavuttua LoRa-verkkoon alkuperäinen lämpötilan ja kosteuden muuttuja muuttui heksadesimaaliksi ja viestiin lisättiin metatietoja, kuten

laitteen Device EUI. Kuviossa 22 näkyy, että viesti saapui LoRa-verkkoon onnistuneesti.



Kuvio 22 Viesti LoRa-verkossa

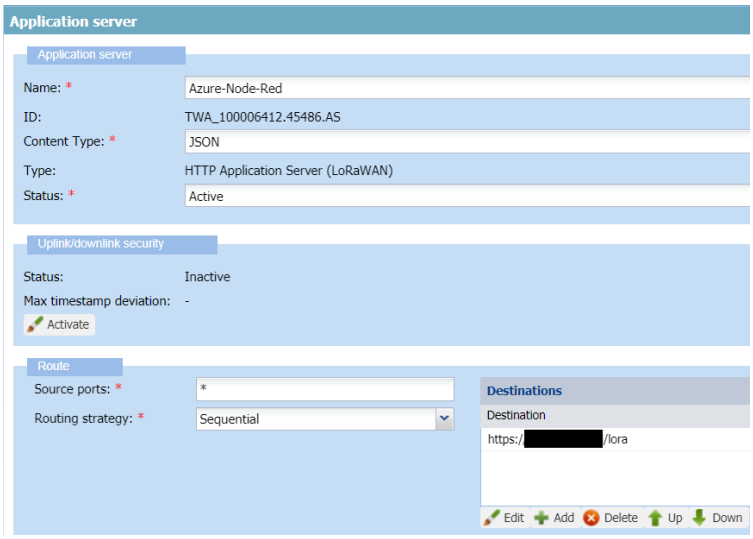
4.3 LoRa-viestien käsittely ja ohjaus pilveen

Laitteelta saadut viestit täytyi lähettää eteenpäin pilvipalveluun. Lähettämistä varten ThingParkissa tehtiin Application Server, eli päätepalvelin, johon viestit lähetettiin eteenpäin. LoRa-viestit olivat heksadesimaalimuodossa, joten viestit täytyi ensin muuttaa takaisin numeroiksi. Muuntamista varten asennettiin Microsoft Azuressa olevaan Jyväskylän ammattikorkeakoulun virtuaalikoneelle Node-Red-palvelu. LoRa-palvelimelta voitiin lähettää viesti Node-Rediin, jossa viestiä muokattiin ja lähetettiin eteenpäin.

Node-Redin asennuksen jälkeen muokattiin tietoturva-asetuksia esimerkiksi lisäämällä käyttäjätunnistautuminen, jotta kuka tahansa ei voinut kirjautua palveluun. LoRa-verkosta voitiin lähettää viesti eteenpäin päätepalvelimelle, jos palvelimen osoite oli HTTPS-muodossa. HTTPS-osoitetta varten varten tarvittiin sertifikaatti, joka opinnäytetyössä oli jo valmiiksi asennettuna virtuaalikoneeseen käyttämällä Lets Encrypt -palvelua.

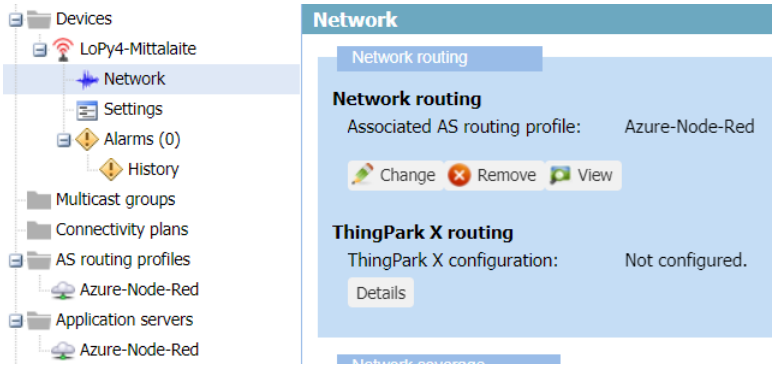
Node-Redin asennuksen jälkeen sille määritettiin oma domain-osoite, eli Node-Rediin pääsi kirjoittamalla sen osoite internet selaimeen. Osoite lisättiin myös ThingPark-palveluun päätepalvelimen osoitteeksi. Kuviossa 23 näkyy päätepalvelimen

määritys ThingParkissa.



Kuvio 23 Päätepalvelimen määritys

Tämän jälkeen tehtiin AS routing profile, eli reititysprofiili. Reititysprofiililla määritettiin laitteen reititysasetukset. Tämän jälkeen laiteelta tulleet viestit reititettiin eteenpäin Node-Red-palvelimelle. Kuviossa 24 näkyy laitteen lopulliset reititysasetukset.



Kuvio 24 Reititysasetukset

Node-Red asetettiin kuuntelemaan http-viestejä. Viestin saavuttua se siirtyi eteenpäin funktiolle, joka tarkasti viestin alkuperäisen lähteen. Lähteen tarkistaminen onnistui poimimalla viestin metatiedoista Device EUI ja vertaamalla sitä mittalaitteen Device EUI-arvoon. Viesti ohjattiin eteenpäin seuraavalle funktiolle, jos Device EUI-arvo täsmäsi. Seuraavalla funktiolla poimittiin viestistä heksadesimaali ja se muutet-

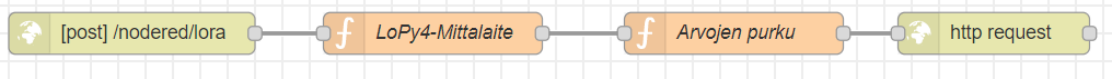
tiin takaisin lämpötilan ja kosteuden arvoksi. Kuviossa 25 esitetään funktio, jolla heksadesimaalista poimittiin lämpötilan ja kosteuden arvot.

```
function hextobytes(hex) {
    bytepacket = [];
    for (var i = 0, len = hex.length; i < len; i+=2) {
        bytepacket.push(parseInt(hex.substr(i,2),16));
    }
    return bytepacket;
}
function bytestofloat(bytes) {
    bits = bytes[3]<<24 | bytes[2]<<16 | bytes[1]<<8 | bytes[0];
    sign = (bits>>>31 === 0) ? 1.0 : -1.0;
    e = bits>>>23 & 0xff;
    m = (e === 0) ? (bits & 0x7fffff)<<1 : (bits & 0x7fffff) | 0x800000;
    f = sign * m * Math.pow(2, e - 150);
    return +f.toFixed(1);
}
bytes = hextobytes(msg.payload.DevEUI_uplink.payload_hex);

temp = bytestofloat(bytes.slice(0, 4));
moist = bytestofloat(bytes.slice(4, 8));
msg.payload =
{
    Lampotila: temp,
    Kosteus: moist
}
return msg;
```

Kuvio 25 Hexadesimaalin purku

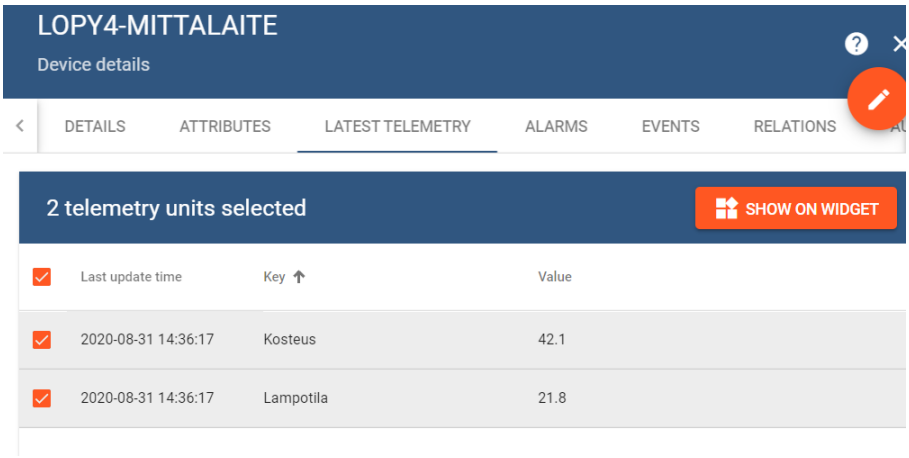
Tämän jälkeen viesti oli muutettu oikeanlaiseksi ja se voitiin lähettää ThingsBoardiin. ThingsBoardissa arvoja voitiin tarkastella internetistä. Lähetystä varten täytyi laite lisätä ThingsBoardiin ja poimia sieltä Device ID, jonka avulla arvot voitiin lähettää oikeaan osoitteeseen Node-Red-palvelusta. Viestit voitiin lähettää osoitteeseen http://localhost:8080/api/v1/*Device ID*/telemetry, koska ThingsBoard ja Node-Red olivat samalla palvelimella. Kuviossa 26 on Node-Redin valmis viestinkäsittelyketju, jossa alkuperäinen viesti muutettiin ja ohjattiin eteenpäin. Kuviossa on 4 eri toimintoa, jotka suoritettiin Node-Red-palvelussa. Ensimmäinen kuunteli http-viestejä ja välitti ne seuraavalle, jossa tarkistettiin viestin alkuperäinen lähde. Kolmas toiminto purkaa viestistä heksadesimaalin ja muuttaa sen lämpötilan ja kosteuden arvoiksi. Lopuksi neljäs toiminto lähettää arvot eteenpäin ThingsBoardiin.



Kuvio 26 Käsittelyketju Node-Redissä

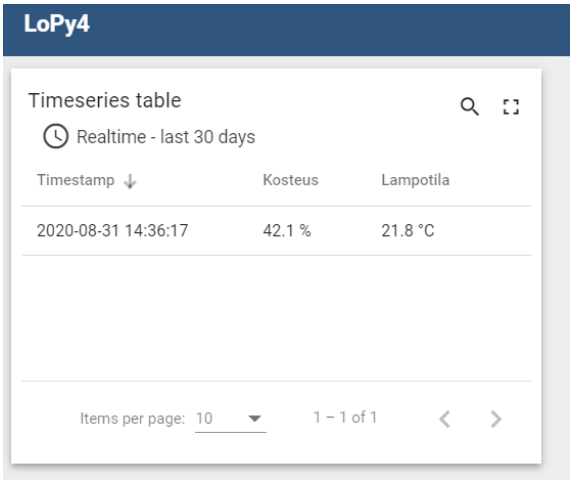
4.4 Arvojen visualisointi käyttäjälle

ThingsBoardista voitiin tarkastella, minkälaisia viestejä laite lähetti sinne. Kuviossa 27 näkyy laitteelta lähetetty viesti, joka saapui ThingsBoardiin. Node-Redin ja ThingsBoardin osuus toimi oikein.



Kuvio 27 Lämpötilan ja kosteuden arvot Thingsboardissa

Seuraavaksi arvoista tehtiin taulukko, joka toteutettiin valitsemalla arvot telemetri-asivulta ja painamalla show on widget -painiketta. Viestin lähetyksen jälkeen voitiin arvoja tarkastella ThingsBoardin taulukosta. Kuviossa 28 on kuvattu näkymä, johon mitatut arvot tulivat esille.



Kuvio 28 Arvojen taulukko

4.5 Laitteen viimeistely

Laitteen toiminta oli todettu toimivaksi ja se voitiin viimeistellä ohjelmoimalla painonappi, kytkimet ja näyttö toimimaan oikein. Liitteessä 1 on laitteen lopullinen ohjelma. Laitteelle ohjelmoitiin kaksi eri mittaustilaa. Ensimmäisessä mittaustilassa laite mittasi arvot ja meni sen jälkeen tunniksi nukkumistilaan. Toisessa mittaustilassa arvoja mitattiin jatkuvasti ja arvot näkyivät näytöllä.

Lopuksi lisättiin ohjelmaan ominaisuus, jolla käyttäjää tiedotettiin akun tilasta. Akun kapasiteetin pudotessa liian matalalle, käyttäjää kehotettiin lataamaan akku. Akkua ladattaessa näytölle päivittyi akun latauksen tilanne.

4.5.1 Virrankulutuksen optimointi ja akunkesto

Olennainen osa LoRa-laitteita on saada ne kuluttamaan mahdollisimman vähän virtaa. Tässä vaiheessa laitteella kului ohjelman suorittamiseen käynnistyksen jälkeen yli 15 sekuntia, kun mittaustilana oli mittaukset tunnin välein. Virrankulutus on suuri silloin, kun laite on päällä, joten käynnistysaikaa täytyi parantaa.

LoPy4-alustassa on boot.py ohjelma, joka ajetaan laitteen käynnistyksen yhteydessä. Ohjelmaan kirjoitettiin eri komentoja, joilla poistettiin LoPy4-alustasta esimerkiksi WiFi ja muita ominaisuuksia. Kaikki ylimääräiset ominaisuudet hidastivat laitteen käynnistymistä, joten ne poistettiin. Kuviossa 29 näkyy boot.py ohjelma, jolla käynnistysaikaa parannettiin.

```
import pycom

if pycom.pybytes_on_boot() == True:
    pycom.pybytes_on_boot(False)

if pycom.heartbeat() == True:
    pycom.heartbeat(False)

if pycom.wifi_on_boot() == True:
  pycom.wifi_on_boot(False)
```

Kuvio 29 Boot.py

Käynnistysmääritysten muokkaamisen jälkeen laite käynnistyi ja suoritti ohjelman kahdeksassa sekunnissa, eli aika lyheni lähes puoleen.

Akunkesto laskettiin mittaamalla laitteen virrankulutus ohjelmaa suorittaessa ja nukkumistilassa. Laskemisessa käytettiin mittaukset tunnin välein -mittaustilaa, koska sitä tullaan käyttämään laitteella eniten. Tällöin laite on nukkumistilassa tunnin ja sen jälkeen kahdeksan sekuntia päällä.

Kuviossa 30 on mittaustulokset, joista nähdään, että nukkumistilassa laite kuluttaa virtaa 0,160mA ja käynnissä ollessa 50mA. Näillä arvoilla voitiin laskea arvioitu akunkesto, jonka tulokseksi tuli 2000mAh akulla 246 päivää. Tuloksessa huomioitiin, että akkua ei haluta täysin tyhjäksi, joten tulos on laskettu 1600mAh kapasiteetilla. Tulokset ovat laskettu pelkästään mittaukset tunnin välein -mittaustilassa. Näytön ollessa päällä laitteen virrankulutus oli 80-90 mA.

Virrankulutus laite päällä	**Kesto**	**Kulutus tunnisssa (mAh)**	**Akunkesto**
50mA	8s	0,111	
Virrankulutus laite Deepsleep-tilassa			
160µA = 0,160mA	1h	0,16	
		Yhteensä 0,271	n.5904h = 246 päivää

Kuvio 30 Akunkesto

5 Testaus ja johtopäätökset

Laitteen mittaustilat testattiin ja todettiin toimiviksi. Mittauksia otettiin tunnin välein ja kaikki mittaukset tulivat ThingsBoardiin näkyville. Kuviossa 31 näkyy ThingsBoardin taulukko, josta näkee kosteuden ja lämpötilan. Mitatut arvot olivat realistisia ja lähes

identtisiä toiseen lämpömittariin verrattuna.

	Kosteus	Lampotila
02:06:54	52.3 %	18.2 °C
01:07:05	58.1 %	18.9 °C
00:07:14	59.2 %	19.8 °C
23:07:25	50.8 %	21 °C
22:07:36	45.4 %	22.2 °C
21:07:47	41.9 %	23.1 °C
20:07:59	38 %	25.1 °C
19:08:05	39.8 %	24.9 °C
18:08:09	42.7 %	24.6 °C

Kuvio 31 Mittaukset tunnin välein

Toinen mittaustila testattiin myös toimivaksi, eli mitatut arvot tulivat näytölle näkyviin ja painonapilla voitiin lähettää viimeksi mitatut arvot ThingsBoardiin. Arvojen tullessa näytölle huomattiin anturin toimivan hitaasti. Laitteella kesti kauan oikean lämpötilan näyttämiseen. Hitaus ei ollut iso ongelma, koska laitteella mitattiin suurimmaksi osaksi tunnin välein.

Laite vietiin testikäyttöön, mutta se altistui vesisateelle ja laite lakkasi toimimasta. Kotelon sisään oli päässyt vettä ja akku oli hajonnut todennäköisesti oikosulun takia. Tämän vuoksi laitteeseen täytyi uusia akku ja miettiä laitteen vedenkestävyyttä. Akku vaihdettiin ja kotelon komponenttien reiät tiivistettiin. Kotelon sisällä oleva LoPy4-alusta asennettiin muovikoteloon, jotta se ei varmasti altistu kosteudelle. Tämän jälkeen laite oli valmis ja se toimi paremmin.

Laite vietiin uudestaan testikäyttöön ja sen toiminta varmistettiin. Mittaukset tunnin välein toimi kuten piti ja laite oli valmis. Lopputuloksena syntyi toimiva laite, jota voidaan vielä jatkokehittää paremmaksi. Mahdollisia kehityskohteita laitteessa on anturi, kotelo ja paikkatiedon välitys. Paikkatiedolla voisi kertoa, mistä kasasta mittaukset ovat peräisin.

6 Pohdinta

Työn tarkoituksena oli suunnitella ja toteuttaa mittalaite sekajätekasan itsesyttymisen ennaltaehkäisyyn. Tämä onnistui ja laite saatiin rakennettua. Työ vaati paljon suunnittelua ja tutkimista, koska siinä käytettiin useita eri järjestelmiä ja komponentteja. Järjestelmiin tutustuminen vei aikaa, mutta se oli pakollista laitteen rakennuksen kannalta.

Komponenttien valinta onnistui melko hyvin. LoPy4-piiri oli yksinkertainen käyttää ja siihen löytyi paljon apua valmistajan nettisivuilta. Anturin valinta olisi voinut olla parempi, koska valitulla anturilla kesti kauan näyttää oikea lämpötila. Tämä ei kuitenkaan ollut ongelma, jos laitetta käytettiin mittaukset tunnin välein -mittaustilassa.

Laitetteella oli aluksi tarkoitus mitata lämpötilaa ja kostetutta katoksessa olevista sekajätekasoista, mutta myöhemmin selvisi, että kasoja on myös katoksen ulkopuolella. Tämä vaikeutti työtä paljon, koska komponentit oli valittu niin, että vedenkestävyyttä ei juuri mietitty, joten laitteesta hajosi akku kosteuden takia. Kotelon vedenkestävyyttä parannettiin myöhemmin ja tämän jälkeen laite oli kestävämpi.

Mittalaitteen tietoliikenneosuus oli haastava toteuttaa. Täytyi tutustua LoRan toimintaan ja moneen eri järjestelmään, kuten ThingParkiin, Node-Rediin ja ThingsBoardiin. Digitan käyttämästä ThingParkista löytyi todella vähän apua netistä, joten siihen täytyi tutustua itse. Laitteen yhdistäminen Digitan verkkoon onnistui ongelmitta, mutta datan lähetys eteenpäin ThingParkista vaati paljon kokeiluja ja uusien järjestelmien opettelua. Kun data saatiin järjestelmästä ohjattua Jyväskylän Ammattikorkeakoulun palvelimelle, dataa voitiin visualisoida asiakkaalle luettavaksi. Tuloksena syntyi selkeä taulukko, johon mitatut tulokset päivittyivät ThingsBoardissa.

Päätepalvelimen määrityksestä on varmasti hyötyä jatkossa, koska siitä ei löydy juurikaan tietoa netistä. Työn tuloksena syntyi toimiva LoRaWAN-ympäristö, josta on

hyötyä tulevaisuudessa muissa IoT-projekteissa, joissa käytetään LoRaa ja Digitan LoRa-verkkoa.

Lähteet

Actility is the world leader in IoT networks management. N.d. Actilityn verkkosivut. Viitattu 26.10.2020. https://www.actility.com/.

About. N.d. Node-RED verkkosivut. Viitattu 26.10.2020. https://nodered.org/about/.

Chaudhuri, A. 2019. Internet of things, for things, and by things. E-kirja. Florida: CRC Press. Viitattu 5.10.2020. https://library-books24x7-com.ezproxy.jamk.fi:2443/toc.aspx?bookid=138743.

Colasante, F. N.d. How to setup an IoT system using ThingsBoard. Artikkeli Things-Boardin asennuksesta. Viitattu 26.10.2020. https://medium.com/@colasante.fran-cesco/how-to-setup-an-iot-system-using-thingsboard-b705c9189e37.

Esineiden internet valtaa jokaista toimialaa. 2020. Artikkeli Digitan verkkosivuilla. Vii-tattu 16.11.2020. https://www.digita.fi/asiakastarinat/valtakunnallinen-iot-verkko-tuo-yrityksille-rajattomasti-uusia-mahdollisuuksia-liiketoiminta-on-kasvanut-vauhdik-kaasti/.

eÄlytelli-hanke. N.d. eÄlytellin verkkosivut. Viitattu 14.9.2020. https://ealytelli.fi/.

Get Started, N.d. Node-Redin verkkosivut. Viitattu 16.11.2020. https://node-red.org/#get-started.

Installation. N.d. ThingsBoardin verkkosivut. Viitattu 16.11.2020. https://things-board.io/docs/user-guide/install/installation-options/.

IoT: Consumer & Commercial vs. Industrial - Main overview. 2019. Artikkeli ubidotsin verkkosivuilla. Viitattu 16.11.2020. https://ubidots.com/blog/iot-consumer-vs-com-mercial-vs-industrial-main-overview/.

IoT:n kartta. 2020. Digitan verkkosivujen materiaali. Viitattu 5.10.2020. https://www.digita.fi/iotn-kartta/.

Jäntti, H. 2019. IoT-projektin dokumentointi. Opinnäytetyö. Viitattu 24.11.2020. http://urn.fi/URN:NBN:fi:amk-2019053113671.

LoRa – Device Activation Call Flow (Join Procedure) using OTAA and ABP. 2018. Verkkoartikkeli. Viitattu 28.10.2020. http://www.techplayon.com/lora-device-activation-call-flow-join-procedure-using-otaa-and-abp/.

LoRaWAN-teknologia. N.d. Digitan verkkosivut. Viitattu 5.10.2020. https://www.digita.fi/etusivu/palvelut-yrityksille/iot/lorawan-teknologia/.

Maayan, G. 2020. The IoT Rundown For 2020: Stats, Risks, and Solutions. Artikkeli securitytoday-verkkosivulla. Viitattu 5.10.2020. https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx?Page=1.

Mitä IoT tarkoittaa?. N.d. Telian verkkosivut. Viitattu 16.11.2020. https://www.telia.fi/yrityksille/iot/esineiden-internet.

Pernaa, J. 2013. Kehittämistutkimus tutkimusmenetelmänä. Artikkeli kehittämistutkimuksesta. Viitattu 18.11.2020. https://tuhat.helsinki.fi/ws/files/127650174/2013_Pernaa_KT_tutkimusmenetelmana_KT_kirja.pdf.

Ruano, E. 2016. LoRaTM protocol Evaluations, limitations and practical test. Opinnäytetyö. Viitattu 27.10.2020. https://upcommons.upc.edu/handle/2117/98853.

Seneviratne, P. 2019. Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power Wireless IoT Networks. E-kirja. California: Apress. Viitattu 24.9.2020. https://www.pdfdrive.com/beginning-lora-radio-networks-with-arduino-build-long-range-low-power-wireless-iot-networks-e183964838.html.

TEMPPI Mallit. 2014. Kasvutaito.fi nettisivut. Viitattu 12.11.2020. http://kasvutaito.fi/?page_id=2842.

The Actility product portfolio. N.d. Actilityn verkkosivut. Viitattu 26.10.2020. https://www.actility.com/products/.

What is LoRaWAN?. 2015. Verkkoartikkeli. Viitattu 26.10.2020. https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf.

What is modulation?. 2018. Blogiartikkeli. Viitattu 26.10.2020. https://www.carritech.com/news/what-is-modulation-in-telecommunications/.

What is ThingsBoard?. N.d. ThingsBoardin verkkosivut. Viitattu 24.9.2020. https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/

Liitteet

Liite 1. LoPy4 ohjelmointi

```
import binascii, pycom, socket, time, struct, ustruct, machine
from network import LoRa
from sht1x import SHT1X
from machine import I2C, Pin, ADC
from esp8266_i2c_lcd import I2cLcd

def JoinLora(lora): #LoRaan liittyminen
    app_eui = binascii.unhexlify('[redacted])
    app_key = binascii.unhexlify('[redacted])
    lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout=0)
    while not lora.has_joined():
        for i in range(30):
            time.sleep(0.5)
            if lora.has_joined():
                break

def measures(): #Lämpötilan ja kosteuden mittaus
    sht1x = SHT1X('P9', 'P10')
    result = sht1x.measure()
    return result

def batteryvoltage():   #Akun jännite
    adc = ADC()
    bat = adc.channel(pin='P16', attn=ADC.ATTN_11DB)
    bat_vol = int(bat.voltage() * 3.055)
    return bat_vol

def main():
    #Kytkimien ja näytön pinnien sekä muuttujien määritys
    naytto=Pin('P22',mode=Pin.IN)
    naytto2=Pin('P23',mode=Pin.IN)
    DEFAULT_I2C_ADDR = 0x27
    charged = False
    reset_reason = machine.wake_reason()
    mode_switch = Pin('P8', mode=Pin.IN, pull=Pin.PULL_DOWN)
    sendswitch = Pin('P20', mode=Pin.IN, pull=Pin.PULL_DOWN)
    machine.pin_sleep_wakeup(['P20'], mode=machine.WAKEUP_ANY_HIGH,
enable_pull=False)
    lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)
    lora.nvram_restore()
    s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
    s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
    s.setblocking(True)
    while True:
        while mode_switch() == 1:   #Kun mittaustila on "mittaukset
näytölle"
            naytto=Pin('P22',mode=Pin.OUT)
            naytto2=Pin('P23',mode=Pin.OUT)
            naytto.value(1)
            naytto2.value(1)
            i2c=I2C(0, pins=('P11','P12'))
            lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
            while True:
                bat_vol = batteryvoltage()
                usb = Pin('P0', mode=Pin.IN, pull=Pin.PULL_DOWN)
                if usb() == 1 and charged == False:
                    lcd.clear()
```

```
                    lcd.putstr("Akku
latautuu\n"+str(bat_vol)+"mV/4100mV")
                    time.sleep(5)
                if bat_vol > 4100:
                    charged = True
                if charged == True and usb() == 1:
                    lcd.clear()
                    lcd.putstr("Akku latautunut")
                    time.sleep(5)
                if bat_vol < 3350 and usb() == 0:
                    lcd.clear()
                    lcd.putstr("Lataa akku!")
                    time.sleep(7)
                temp_hum = measures()
                temp=round(temp_hum[0],1)
                hum=round(temp_hum[1],1)
                lcd.clear()
                lcd.putstr("Lampo:"+str(temp)+"C\n"+"Kosteus:"+
str(hum)+"%")
                for i in range(200):
                    time.sleep(0.1)
                    if sendswitch() == 1:   #Jos painetaan painonappia,
niin lähetetään LoRa-viesti
                        packet = ustruct.pack('2f', temp, hum)
                        if not lora.has_joined():
                            lcd.clear()
                            lcd.putstr("Yhdistaa\nverkkoon")
                            JoinLora(lora)
                        lcd.clear()
                        lcd.putstr("Lahettaa arvot\npilveen")
                        s.send(packet)
                        time.sleep(1)
                        lora.nvram_save()
                        lcd.clear()
                        lcd.putstr("Lahetetty")
                        time.sleep(1)
                        break
                    elif mode_switch() == 0:
                        break
                if mode_switch() == 0:  #Jos vaihetaan mittaustilaa
                    lcd.clear()
                    lcd.putstr("Mittaukset\ntunnin valein")
                    time.sleep(2)
                    lcd.clear()
                    lcd.putstr("Arvot pilveen\nautomaattisesti")
                    time.sleep(2)
                    lora.nvram_save()
                    lcd.clear()
                    lcd.putstr("Naytto sammuu")
                    time.sleep(2)
                    naytto.value(0)
                    naytto2.value(0)
                    naytto=Pin('P19',mode=Pin.IN)
                    naytto2=Pin('P23',mode=Pin.IN)
                    machine.deepsleep(3600000)
        while mode_switch() == 0: #Kun mittaustila on "Mittaukset tunnin
välein"
            while True:
```

```
                if reset_reason[0] == 1:   #Jos näyttö virkistetään
painonapilla
                    naytto=Pin('P22',mode=Pin.OUT)
                    naytto2=Pin('P23',mode=Pin.OUT)
                    naytto.value(1)
                    naytto2.value(1)
                    i2c=I2C(0, pins=('P11','P12'))
                    lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
                    bat_vol = batteryvoltage()
                    changemode = False
                    if bat_vol < 3350:
                        lcd.clear()
                        lcd.putstr("Lataa akku!")
                        time.sleep(7)
                    for i in range(2):
                        time.sleep(3)
                        temp_hum = measures()
                        temp=round(temp_hum[0],1)
                        hum=round(temp_hum[1],1)
                        lcd.clear()
                        lcd.putstr("Lampo:"+str(temp)+"C\n"+"Kosteus:"+
str(hum)+"%")
                        if mode_switch() == 1:
                            lcd.clear()
                            lcd.putstr("Mittaukset\nnaytolle")
                            time.sleep(3)
                            changemode = True
                            break
                    if changemode == True:
                        break
                    lcd.clear()
                    lora.nvram_save()
                    lcd.putstr("Naytto sammuu")
                    time.sleep(1)
                    naytto.value(0)
                    naytto2.value(0)
                    naytto=Pin('P22',mode=Pin.IN)
                    naytto2=Pin('P23',mode=Pin.IN)
                    machine.deepsleep(3600000)
                if reset_reason[0] != 1:    #Jos laitteeseen ei ole
koskettu ja tunti on mennyt
                    temp_hum = measures()
                    temp=round(temp_hum[0],1)
                    hum=round(temp_hum[1],1)
                    packet = ustruct.pack('2f', temp, hum)
                    if not lora.has_joined():
                        JoinLora(lora)
                    s.send(packet)
                    lora.nvram_save()
                    naytto.value(0)
                    naytto2.value(0)
                    naytto=Pin('P22',mode=Pin.IN)
                    naytto2=Pin('P23',mode=Pin.IN)
                    machine.deepsleep(3600000)

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