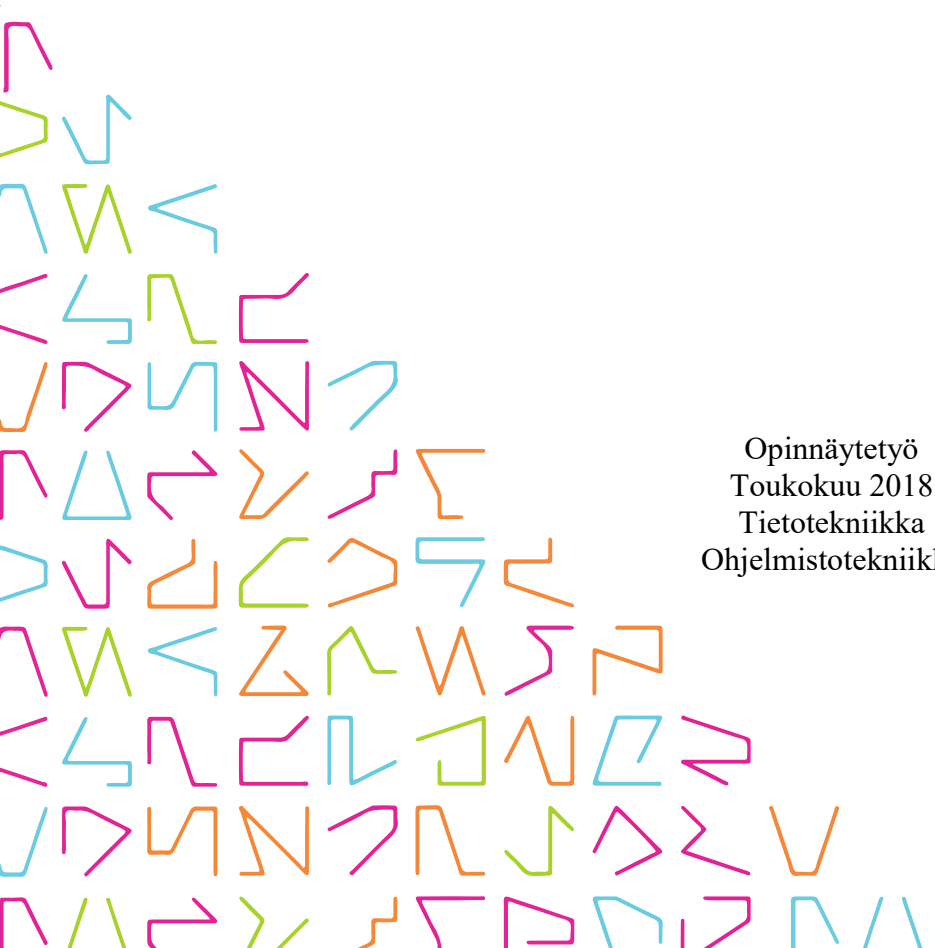
TAMPEREEN
AMMATTIKORKEAKOULU

Talotekniikan IoT

Sensorihallinta ja tiedon välittäminen

Heikki Alho



Opinnäytetyö
Toukokuu 2018
Tietotekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

Alho Heikki
Talotekniikan IoT
Sensorihallinta ja tiedonvälittäminen

Opinnäytetyö 24 sivua, joista liitteitä 0 sivua
Toukokuu 2018

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa tilakoneena ja välityspalvelimena toimiva ohjelma Tampereen ammattikorkeakoulun talotekniikan koulutukselle. Ohjelma kehitettiin ohjaamaan tulevan olosuhdelaboratorion mittauskokonaisuutta ja välittämään arvoja pilvipalveluun.

Ohjelma tehtiin Raspberry Pi -tietokoneelle C++ -ohjelmointikielellä käyttäen MQTT-viestintäprotokollaa sensoritiedon noutamiseen ja Wapice Oy:n IoT-ticketin tekemää viestirajapintaa tiedon lähettämiseen pilvipalveluun. Käyttäjä voi konfiguroinnilla liittää ohjelmaan useita sensoreita käytettäväksi.

Lopputuloksena oli määrittelyn mukainen käyttövalmis ohjelma ja käyttöohjeistus, jolla olosuhdelaboratorion sensorien asentaminen ja käyttöönotto onnistuu henkilöltä, jolla ei ole osaamista ohjelmoinnista.

Asiasanat: ohjelmistosuunnittelu, iot

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information and Communications Technology
Software Engineering

Alho Heikki
Building services engineering IoT
Sensor control and transfer of data

Bachelor's thesis 24 pages, appendices 0 pages
May 2018

Purpose of this thesis was to design and develop a program to act as a state-machine and a proxy server for Tampere University of Applied Sciences building services engineering programme. Program was developed to control an air quality laboratorys measuring system and send data to a cloud service.

Program was made for Raspberry Pi -computer with C++ -programming language using MQTT-messaging protocol to read sensor data and a program library made by Wapice Ltd. to deliver data to a IoT-Ticket -cloud service. User can configure multiple sensor to be used with the program.

Result of this thesis was a program that fulfilled the specifications and instructions for its use to allow a user without programming knowledge to install and use it.

Key words: software engineering, iot

SISÄLLYS

LYHENTEET JA TERMIT

Arduino	Ohjelmoitava mikrokontrolleri
IoT	Internet of Things, esineiden internet
MQTT	Message Query Telemetry Transport – TCP/IP -protokollalle suunniteltu kevyt viestintäprotokolla
C++	Ohjelmointikieli
Raspberry Pi	Pieni, yhden piirilevyn tietokone
JSON	JavaScript Object Notation – Tekstipohjainen tiedontalletus-muoto

1 JOHDANTO

Tämä opinnäytetyö on osa Tampereen ammattikorkeakoulun talotekniikan IoT-projektia, jonka tavoitteena on rakentaa luokkatilaan oppimisympäristö muun muassa olosuhteiden mittausta, LVI-järjestelmien mittausta ja säätöä, sekä olemassa olevien tilamallinnuksien todentamista varten. Projektin tekninen toteutus on jaettu kahteen osaan, talotekniikassa yleisesti käytettyjen sensorien liittämisestä sovittuun viestiprotokollaan ja sensorikokonaisuutta hallitsevaan tilakoneohjelmaan, jota tässä opinnäytetyössä käsitellään.

Välityspalvelin toimii teknisen toteutuksen runkona. Ohjelma ajoittaa systeemiin liitettyjen sensorien tiedonkeräämisen ja lähettämisen pilvipalveluun konfigurointitiedoston perusteella, jota käyttäjä muokkaa aina lisätessään tai poistaessaan sensoreita systeemistä. Ohjelma on suunniteltu laajennettavaksi tukemaan useampia viestiprotokollia ja pilvipalveluita, jos tulevaisuudessa se nähdään tarpeelliseksi.

2 TALOTEKNIIKAN IOT-PROJEKTI

Projektin tavoitteena on luoda oppimisympäristö Tampereen ammattikorkeakoulun talotekniikan koulutukseen. Oppimisympäristö koostuu yhdestä luokkatilasta, joka muutettaisiin laboratoriotilaksi eristämällä se täysin rakennuksen normaalista ilmastointijärjestelmästä ja käyttämällä tilan omaa ilmastointikonetta ilmanlaadun hallitsemiseen. Oppimisympäristöstä mitataan sen ja ympäröivien tilojen olosuhteita ja käytetään saatuja mittaustuloksia LVI-järjestelmien mittaukseen sekä säätämiseen, sisätilojen LVI-mallintamiseen, että energiatehokkuuden ja elinkaaritalouden määrittämiseen. Mallia voidaan käyttää varmentamaan jo olemassa olevia kaupallisten ohjelmien, esimerkiksi Equan IDA Indoor Climate and Energy:n (EQUA Simulation AB, 2017), tarjoamia malleja ja hyödyntämään kaikkea edellä mainittua opetuksessa, tutkimuksessa ja kehittämisessä.

Olosuhdemittauksiin käytettävät sensorit rajattiin talotekniikan ammattilaiskäyttöön suunniteltuihin sensoreihin, joita esimerkiksi Vaisala Oyj (Vaisala Oyj, 2018) valmistaa. Mitattaviin arvoihin kuuluu muun muassa lämpötila, ilmanlaadut, ilmanpainesuhteet ja energiankulutus. Tämän vuoksi teknisen toteutuksen suunnittelussa lähtökohtana oli helposti muunneltava järjestelmä, johon uusien ja erilaisten sensorien lisääminen olisi helppoa ja mahdollisimman vähän rajoitettua. Käyttäjän pitää pystyä systeemin muokkaukseen ilman vaativamman tason tietoteknisiä taitoja, kuten ohjelmointia tai sulautettujen järjestelmien osaamista.

3 TEKNIIKKA

Projekti päätettiin toteuttaa mahdollisimman avoimilla, helposti muokattavilla ja joustavilla alustoilla. Runkona toimiva välityspalvelin Kaspos suunniteltiin Linux -ytimeen perustuvaan Raspbian -käyttöjärjestelmään konsolipohjaiseksi C++ -sovellukseksi. Sensoreita hallitseva ohjelma on toteutettu Arduino -mikroprosessorille. Laitteiden välinen viestintä perustuu MQTT-protokollaan internetyhteyden kautta sisäverkossa ja tiedon lähettäminen pilvipalveluun CURL-tekniikan päälle rakennetulla rajapinnalla.

3.1 Laitteisto

3.1.1 Raspberry Pi

Raspberry Pi on Raspberry Pi Foundationin kehittämänä, noin luottokortin kokoinen yhden piirilevyn tietokone. Se kehitettiin alun perin edistämään tietotekniikan opetusta kouluissa ja kehitysmaissa, mutta sai valtavan suosion ja myi yli 10 miljoonaa kappaletta syyskuuhun 2016 mennessä. Malliksi valittiin Raspberry Pi 3 Model B, jonka järjestelmäpiirinä on Broadcomin (Broadcom Inc.) BCM2837, jossa on neliytiminen ARM:n (ARM Limited) Cortex A53 prosessori 1,2 GHz kellotaajuudella, näytönohjaimena VideoCore IV 300/400MHZ ja 1GB LPDDR2 -käyttömuistia. 3 Model B:ssä on USB-porttia, 10/100 Ethernet RJ45 -liitin, langaton 2.4GHz 802.11n WLAN -siru, HDMI-audio/videoliitin ja 3.5mm audioliitin. (Raspberry Pi Foundation, ei pvm) Rasberry Pi valittiin hardware-alustaksi sen monipuolisten liitäntämahdollisuuksien, pienen koon ja tehokkuuden takia.

3.1.2 Arduino

Arduino on avoimeen lähdekoodiin perustuva tuoteperhe, jonka kehitys alkoi vuonna 2003 Italiassa Ivrea Interaction Design Insitutessa helppokäyttöiseksi kehitysalustaksi opiskelijoille ilman osaamistaustaa ohjelmoinnista tai elektroniikasta. Avoin lisensointi mahdollistaa laitteiden ja ohjelmien tekemisen ja myymisen vapaasti ja vähentää kustannuksia huomattavasti. Ohjelmointikielenä käytetään C:n ja C++:n risteytystä, johon on

poimittu metodeja molemmista. (Arduino, 2017). Projektissa käytettiin Arduino Mega 2560 -kehitysalustaa, mutta alustat toimivat samoilla ohjelmakoodeilla, kun käytettävät pinnit vaihdetaan oikeiksi. Megassa on suuri valikoima digitaalisia sisään- ja ulosmenoja, analogisia sisääntuloja, 4 sarjaporttia, USB-liitin ohjelmointia ja jännitelähdettä varten sekä erillinen jänniteliitin. (Arduino, 2017) Arduinon valittiin helpon ohjelmoitavuuden, monipuolisten liitäntämahdollisuuksien ja halpuuden tuoman monistettavuuden takia projektin sensoritoteutukseen.

3.1.3 Sensorit

Järjestelmään kytkettäviä sensoreita ei ole rajoitettu muuten kuin käytettävän viestiväylän suhteen. Projektin Arduino-ohjelma tukee 0-10V jänniteviestiä käyttäviä sensoreita jänniiteenjakokytkennän ja AD-muuntimen avulla. Ohjelmaa voidaan kuitenkin tarvittaessa laajentaa tukemaan muitakin viestintätapoja, kuten 4-20mA virtaviestiä tai väyläratkaisuja. Tässä projektissa käytimme Vaisalan GMW83RP-sensoria (Vaisala Oyj, 2018), joka mahdollistaa ilman lämpötilan, kosteuden ja hiilidioksidipituuden mittaamisen ja tiedon lukemisen jänniteviestillä. Sensorista löytyy myös näyttö, josta edellä mainitut arvot ovat luettavissa.

3.2 Teknologiat

3.2.1 Raspbian -käyttöjärjestelmä

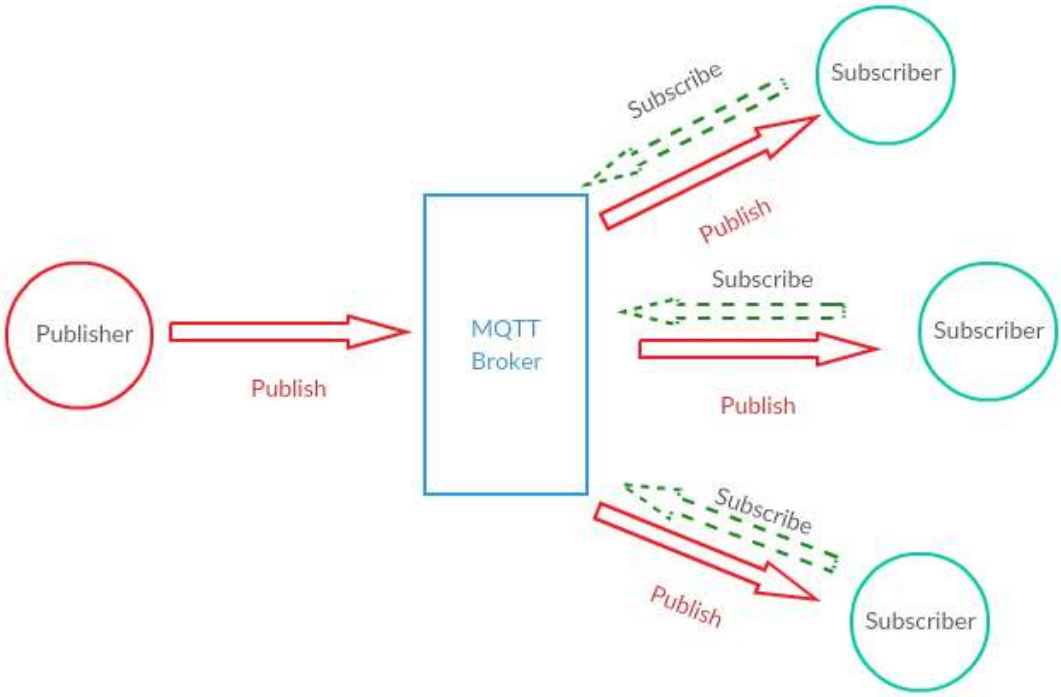
Raspbian on Debian Wheezy-pohjainen käyttöjärjestelmä, joka on suunniteltu erityisesti Raspberry Pi:lle. Debian -järjestelmät ovat vakaita ja suosittuja Linux-yhteisön keskuudessa, joten dokumentaatiota on saatavissa runsaissa määrin. Debianille on tarjolla yli 50000 ohjelmistopakettia (Treinen, 2016), josta suurin osa on ilmaisia. Raspbian on muokattu mahdollisimman kevyeksi kokonaisuudeksi mm. vaihtamalla application binary interface (ABI)n liukulukurekistereitä käyttäväksi. Käyttöjärjestelmällä on oma pakettivarasto, joka erikoistuu Raspberry Pi:lle optimoituihin ohjelmistoihin. (Raspbian, ei pvm) Käyttöjärjestelmän keveys ja muokattavuus tuovat synergiaetuja Rasperry Pi:n kanssa luoden erinomaisen rungon IoT-ympäristöille.

3.2.2 C++ -ohjelmointikieli

C++ on ohjelmointikieli, jonka kehitys alkoi vuonna 1973 C-kielen pohjalta Bjarne Stroustrupin toimesta. Ensimmäinen versio kirjasta The C++ Programming Language julkaistiin lokakuussa 1985 ja uusin, neljäs versio julkaistiin vuonna 2013. Kieleen on lisätty ominaisuuksia mahdollistamaan olio-, funktionaalisen- ja proseduaalisen ohjelmoinnin. Sen kehittämisen näkökulmina pidetään joustavuutta, suorituskykyä ja tehokkuutta. Kieli on ISO-standartisoitu (ISO/IEC 14882:2014)

3.2.3 Message Queueing Telemetry Transport – MQTT

MQTT on TCP/IP protokollan päällä toimiva ISO-standardisoitu (ISO/IEC PRF 20922) viestintäprotokolla, joka kehitettiin mahdollisimman kevyeksi laitteisto- ja yhteysvaatimuksien minimoimiseksi. Protokolla perustuu julkaise/tilaa periaatteeseen, jossa yksi laite toimii viestien välittäjänä ja muut laitteet lähettäjinä tai vastaanottajina.



Kuva 1: MQTT -viestintämalli, julkaisijan (publisher) viesti jaetaan kaikille tilaajille (subscriber) serverinä toimivan jakajan (broker) toimesta.

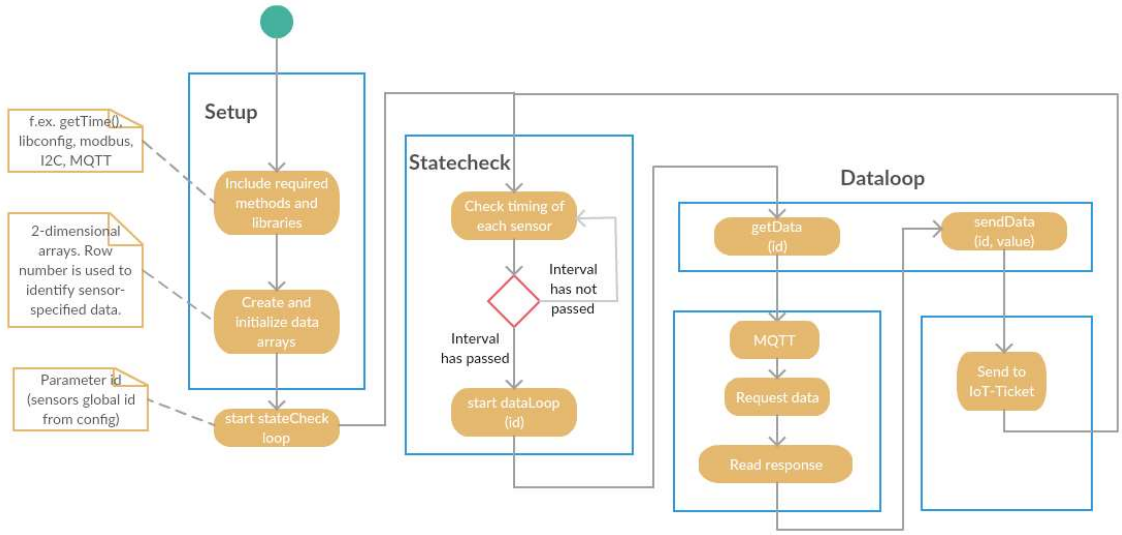
Jokainen tilaajana toimiva laite pystyy tilaamaan erilaisia aiheita ja välittäjä tuo kaikki siihen julkaistut viestit tilaajille. Samalla jokainen tilaaja pystyy myös toimimaan julkaisijana. Protokollalle on standardisoitu viestimalli, jossa jokainen laite yleensä toimii omalla kanavallaan toimittaen aktiivisesti tietoa, jota välityspalvelin kuuntelee. Tässä projektissa kehitimme oman viestimallimme MQTT:n päälle, joka avataan kohdassa 4.1.3.

4 OHJELMA

Välityspalvelin suunniteltiin toimimaan koko teknisen toteutuksen runkona. Sen tehtäviksi jäisi toimiminen konfiguroitavana tilakoneena, tiedon pyytäjänä, sekä välittäjänä. Suunnittelun pohjana oli ohjelman tekeminen mahdollisimman helposti muokattavaksi, jotta esimerkiksi erilaisia tapoja noutaa sensoridataa tai lähettää sitä tietovarastoihin voidaan myöhemmin lisätä konfiguroitaviksi osiksi ohjelmaan.

4.1 Toiminta

Ohjelman toiminta alkaa setup -osiosta (main.cpp), jossa luetaan konfiguraatiotiedoston sisältö dynaamisesti luotaviin tietotauluihin ajastuksia, sensoreita, viestintäprotokollia ja tietovarastoja varten, sekä käynnistetään omilla ohjelmasäikeillä pyörivät palvelut ennen tilakoneen käynnistystä. Tilakone seuraa sensorien lukufrekvenssiä ja käynnistää lukuoperaation omaan ohjelmasäikeeseen, kun aikaväli on kulunut. Seuraavaksi suoritetaan tiedon nouto sensorilta määriteltyä viestintäprotokollaa käyttäen ja saatu tieto lähetetään määriteltyyn tietovarastoon. Kun nämä operaatiot ovat valmistuneet, sen ohjelmasäie päättyy ja se siivotaan. Jokainen lukuoperaatio on omassa ohjelmasäikeessään, jotka tapahtuvat samanaikaisesti muiden kanssa. Tilakoneen toiminta jatkuu koko ajan, vaikka lukuoperaatioita on käynnissä.



Kuva 2. Ohjelman prosessikaavio

4.1.1 Konfiguraatio

Konfiguraatiotiedosto rakentuu JavaScript Object Notation (JSON) -muotoisista objekteista, jotka luetaan tiedostosta libconfig -kirjaston (Hyperrealm) tarjoamilla työkaluilla. Kirjasto mahdollistaa konfiguraatioiden lukemisen esimerkiksi otsikon perusteella ja luetusta objektista voidaan edelleen poimia yksittäisiä palasia, esimerkiksi sensorin konfiguraation tiettyjä arvoja. Konfiguraatiot luetaan ohjelmaan main.cpp -tiedoston mukaisesti ohjelman alussa.
Konfiguraatiotiedostoon voidaan lisätä otsikoilla useampia kohtia tarpeen mukaan. Toteutuksessa käytettiin kolmea osiota: RPI, Wapice ja sensors. RPI, eli Raspberry Pi, käytetään laitteen tunnistamiseen esimerkiksi käyttämämme pilvipalvelun IoT-Ticketin sisältä.

Muuttujan nimi	Suomennos	Selite
deviceName	Laitteen nimi	Käytetään pilvipalvelimella tunnistamaan tietoja lähettävä laite
manufacturer	Laitteen valmistaja	Käytetään pilvipalvelimella tunnistamaan tietoja lähettävän laitteen valmistaja tai asentaja

Taulukko 1. Konfiguraatiotiedoston laiteosio (RPI)

Wapice- otsikon alla on IoT-Ticketiin yhdistämiseen tarvittavat arvot. IoT-ticket (Wapice Ltd) on Wapice Oy:n (Wapice Ltd) IoT-alusta, jolla voidaan seurata ja hallita etänä laitteita ja laiteverkkoja. Palvelussa on mahdollista tehdä esimerkiksi reaaliaikaisen seurannan näyttöjä, joista selviää tämänhetkinen ja määrätyn ajan tilastolliset tiedot valituilta laitteilta, sallien helpon ja selkeän mittariston seurantaa varten.

Muuttujan nimi	Suomennos	Selite
user	Käyttäjätunnus	IoT-Ticketin tunnistautumisessa käytetty tunnus
password	Salasana	IoT-Ticketin tunnistautumisessa käytetty salasana
deviceID	Laitteen tunnistenumero	IoT-Ticketin automaattisesti luotu laitteen tunnistenumero
url	Verkko-osoite	IoT-Ticketin verkko-osoite tiedonlähetykseen

Taulukko 2. Konfiguraatiotiedoston IoT-Ticketin pilvipalvelun asetusosio (wapice)

Sensors- otsikon alla ohjelma käyttää sensorien yksilöintiä, joka muodostuu listasta sensoriobjekteja: Sensorin tunnistetietoja, kuten nimi, fyysinen sijainti ja mitattavan mitattavan suureen yksikkö. Muuntokerrointa käytetään mitatun arvon muuttamiseksi todelliseksi arvoksi, esimerkiksi AD-muuntimelta luetut arvot ovat tässä projektissa välillä 0 – 26667. (Turunen, 2017) Aikavälillä tarkoitetaan sitä, kuinka usein sensorilta luetaan mitattu arvo. Palvelimen nimellä ohjelma erottelee käytettävän lähetyslogiikan. Verkkoa, viestikehystä laitenumeroa ja tiedon sijaintia käytetään tiedonsiirtoprotokollasta riippuen eri tavoilla, näistä kohdassa 4.1.3 lisää. Viimeiseksi vapaa kommenttikenttä, jossa voi kertoa tarkemmin sensorin ominaisuuksista, jotka eivät selkene aiemmista asetuskohdista.

Muuttujan nimi	Suomennos	Selite
name	Nimi	Suureen nimi tai muu sille haluttu tunniste
path	Reitti	Sensorin fyysinen sijainti
type	Tyyppi	Mitattavan suureen yksikkö
server	Palvelin	Palvelin, jolle sensorin tiedot lähetetään
modifier	Muuntokerroin	Kerroin, jolla luetusta lukuarvosta saadaan mitattavan suureen oikea arvo
interval	Aikaväli	Kuinka usein sensorin arvo luetaan, millisekunneissa
network	Verkko	Käytettävä tiedonsiirtoprotokolla
deviceName	Laitteen nimi	Sensorin nimi
messageFormat	Viestikehys	Käytettävä viestikehys, jos tiedonsiirtoprotokolla sen vaatii (MQTT tässä työssä)
deviceNumber	Laitenumero	Looginen laitenumero tiedonlukua varten (tässä työssä MQTTn viestikehystä varten)
dataLocation	Tiedon sijainti	Looginen tiedonsijainti tiedonlukua varten (tässä työssä MQTTn viestikehyksessä ja Modbus-väylässä)
freeComment	Vapaa kommentti	Vapaa viestikenttä lisätietoa varten

Taulukko 3. Konfiguraatiotiedoston sensorien asetusosio (sensors)

```
rpi :
{
  deviceName = "kaspos-rpi";
  manufacturer = "Tamk-IOT";
};
wapice :
{
  user = "käyttäjä";
  password = "salasana";
  deviceID = "";
  url = "https://iot-ticket.fi/api/v1";
};
sensors = (
  {
    name = "Temperature";
    path = "TAMK/A2-19/N-corner/Temperature";
    type = "C";
    server = "Wapice";
    modifier = 0.001875;
    interval = 5000;
    network = "mqtt";
    devicename = "Lämpö1";
    messageFormat = 1;
    deviceNumber = 1;
    dataLocation = 1;
    freeComment = "Vapaa sana sensorista";
  },
  {
    name = "CO2";
    path = "TAMK/A2-19/E-corner/Temperature";
    type = "ppm";
    server = "Wapice";
    modifier = 0.07500;
    interval = 5000;
    network = "mqtt";
    devicename = "Hiilidioksidi1";
    messageFormat = 1;
    deviceNumber = 1;
    dataLocation = 2;
    freeComment = "Vapaa sana sensorista";
  }
 );
```

Tekstileike 1. Kahden sensorin konfiguraatioesimerkki

Konfiguraatiot luetaan ohjelman käynnistyksen jälkeen niille varattuihin vektoritaulukoihin, joissa taulukon alkiota käytetään jäsentämään tietoa. Jokaisen taulukon samansuuruinen alkio vastaa aina saman sensorin tietoja, joten eri taulukoiden käyttö ristiin on helppoa ja ne on jaoteltu niiden käyttötarkoitusten mukaan osiin. Tätä alkiota käsitellään myöhempänä tilakonetunnuksena.

Taulukossa neljä on ajastintaulukon (timerArray) mallirivi. Ensimmäisessä solussa on konfiguraatiotiedostosta luettu mittausaikaväli ja toisessa solussa viimeisin ajanhetki, jolloin sensori on luettu. Alustuksessa viimeisimpään mittausaikaan asetetaan

arvoksi senhetkinen kellonaika vähennettynä aikavälillä, jotta ensimmäinen lukukerta tapahtuu heti ohjelman käynnistyttyä. Jokaista arvoa muutetaan sattumanvaraisella luvulla väliltä 0…500, jotta ensimmäinen lukukierros ei tapahdu samalla millisekunnilla.

Aikaväli	Viimeisin mittausaika

Taulukko 4. Ajastintaulukon yksi rivi

Taulukossa viisi on väylätaulukon eli sensorin lukemiseen käytettävien arvojen mallirivi. Laitenumeroa, tiedon sijaintia ja viestikehystä käytetään eri tavoilla riippuen sitä, mikä verkko eli viestiprotokolla on käytössä. Verkko muutetaan tekstimallisesta arvosta kokonaisluvuksi vastaamaan ohjelmoitua kokonaisuutta, esimerkiksi 0 vastaa Modbus RTU -väylää ja 3 vastaa ohjelmoitua MQTT -verkkoa.

Laitenumero	Tiedon sijainti	Verkko	Viestikehys

Taulukko 5. Väylätaulukon yksi rivi

Taulukossa kuusi on ominaisuustaulukon eli sensorin lisätietojen mallinnus. Pääsääntöisesti kaikkia arvoja käytetään, kun sensorilta luettu arvo sidotaan järkevään kokonaisuuteen käyttöä varten, mutta toiminnallisesti ainoastaan network-arvolla on looginen tarkoitus määrätä vastaanottava palvelin. Muita käytetään helpottamaan tiedon käyttöä ja ymmärtämistä.

Nimi	Sijainti	Tyyppi	Verkko	Laitenimi	Palvelin	Kommentti

Taulukko 6. Ominaisuustaulukon yksi rivi.

Taulukossa seitsemän on muuntotaulukon tiedot, johon tallennetaan ainoastaan muuntokerroin.

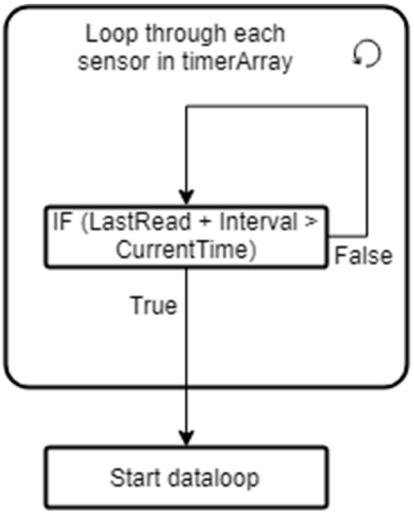
Muuntokerroin

Taulukko 7. Muuntotaulukon yksi rivi

Viimeisenä ohjelman asetusvaiheessa otetaan yhteys IoT-Ticketin palvelimelle ja rekisteröidään laite, jos sitä ei ole vielä tehty. Samalla käynnistetään MQTT-säikeet tiedon pyytämistä ja vastaanottamista varten.

4.1.2 Tilakone

Asetusvaiheen jälkeen käynnistetään ohjelman runko eli tilakone, joka löytyy statecheck.cpp -tiedostosta. Kun sensorin konfiguroitu aikaväli tulee täyteen, tilakone käynnistää dataloop -metodin, joka erottelee käytettävän viestiprotokollan mukaan käynnistettävän lukuprosessin. Tiedon lukukäsky aloitetaan erillisessä prosessisäikeessä, joten tilakoneen toiminta jatkuu normaalisti, vaikka lukuoperaatio onkin käynnissä.



Kuva 3. Tilakoneen prosessikaavio

```
void
stateCheck()
{
                        for (int id = 0; id < AoS; id++){
                        if ((timerArray[id][1] + timerArray[id][0])  <= getTime()){
                                                timerArray[id][1] = getTime();
                                                            globalid = id;
                                                            dataLoop(id);
                                                            usleep(50);
                                                }
                                    }
                        return;
            }
```

Tekstileike 2. Tilakone

```
void
dataLoop(int
id){

                if (busArray[id][2] == 3){ //Used for MQTT, senddata is called at
on_message callback - See mqtt.cpp
                        readData(id);
                }
                else{
                        float _value = readData(id);
                        std::thread t2 (senddatathread, id, _value);
//Practically anything else.
                        t2.detach();
                }
                return;
}
```

Tekstileike 3. Dataloop eli viestiprotokollien mukainen erottelu ja tiedon lähetys

4.1.3 Tiedon lukeminen

Ohjelmaan on kaksi erilaista viestiprotokollaa tiedon noutamiselle: MQTT:n päälle ohjelmoitu viestikehys ja Modbus RTU. Tässä projektissa käytettiin vain MQTT-viestikehystä, mutta ohjelmarakenteeseen pystytään lisäämään uusia viestiprotokollia tarpeen mukaan. Tiedon noutavalle metodille annetaan parametrina ainoastaan tilakoneen tunnistenumero, jonka avulla metodi saa tarvittavat tiedot noutoa varten.

MQTT on kevyt viestiprotokolla, joka tässä projektissa on sovellettu toimimaan kahdella eri aiheella eli topicilla. Tilakoneen käynnistäessä sensorin lukuoperaation asetusvaiheessa luotu lähetyssäie lähettää sensoria hallitsevalle Arduinolle pyynnön ja kuuntelusäie ottaa vastauksen vastaan ja luo prosessisäikeen tiedon pilvipalveluun lähettämistä varten. Pyynnön viestikehyksenä käytetään viestikehys 01:stä taulukon 8 mukaisesti, tilakoneen tunnisteella tarkoitetaan sensorille automaattisesti annettua järjestysnumeroa, eli konfiguraatiotaulukoiden alkiota. Toisella rivillä on konfiguraatiota vastaava muuttuja ja kolmannella rivillä on esimerkki yhdestä pyynnöstä. Pyyntö tapahtuu aiheessa *kysely,* laitteen tunnisteella erotellaan miltä Arduino-laitteelta tietoa pyydetään ja

tiedon sijainnilla 1…4 erotellaan Arduinoon liitetyt sensorit. Format, deviceNumber ja dataLocation ovat jokaiselle sensorille konfiguroitavia arvoja konfiguraatiotiedostossa.

Viestikehys	**Tilakoneen tunniste**	**Laitteen tunniste**	**Tiedon sijainti**
format		deviceNumber	dataLocation
01	001	01	01

Taulukko 8. Lukupyynnön viestikehys

Projektissa käytetyt Arduinot, jotka lähettävät sensoritiedon takaisin aiheella *vastaus* pyynnöstä ohjelmoitiin vastaamaan taulukon 9 viestikehystä 01.

Viestikehys	**Tilakoneen tunniste**	**Luettu arvo**
01	001	12345

Taulukko 9. Vastauksen viestikehys

Vastauksena saatu luettu arvo kerrotaan muuntokertoimella, josta saadaan oikea mitattu arvo suureelle. Arvo ja tilakoneen tunniste välitetään senddatathread -metodille, joka omassa säikeessään lähettää tiedon palvelimelle. Kaikki MQTT:ssä käytetyt viestikehyksen ovat vakiomittaisia, jolloin esimerkiksi tilakoneen tunniste yksi (1) on kehyksessä 001. Ainoa poikkeus tästä on luettu arvo, jolle on varattu tarkempia sensoreita varten tilaa 15-merkkiselle arvolle, jonka ei tarvitse olla vakiopituinen.

4.1.4 Tiedon lähetys

Kun sensorilta on saatu mitattu arvo, ohjelma kutsuu senddatathread -metodia sensorin tilakonetunnisteella ja mitatulla arvolla. Tässä projektissa käytetään vain TAMK:lle hankittua omaa palvelinta, jolle asennettiin oma IoT-Ticket -palvelu tiedon tallennusta ja visualisointia varten. Metodi asettaa mitatun arvon lisäksi sensorin tunnistetiedot lähetettävään objektiin: Nimen, tyypin ja fyysisen sijainnin, sekä lähetysajan. Lähetystä yritetään maksimissaan kaksi kertaa. Lähetyksen jälkeen käytetty tieto-objekti nollataan ja lähetyssäie päätetään. Lähetykseen käytettiin IoT-Ticketin tarjoamaa ohjelmakirjastoa (Wapice Ltd., ei pvm). Ohjelmaan voidaan lisätä muita pilvipalveluita lisäämällä lähetysmetodi

senddatathread-metodin if-funktioihin halutulla nimellä. Tässä kohdassa voidaan määritellä haluttavien tietojen välittäminen konfiguraatiosta kutsuttavalle lähetysfunktiolle.

```
void
wapiceSend(std::string
name, std::string
path, std::string
type, float value){

                        //Vector for wapice IOT_API, store data and send.
                        std::vector<IOT_WriteData> dataVector;
                        IOT_WriteData data;
                        data.SetName(name);
                        data.SetUnit(type);
                        data.SetValue(value);
                        data.SetPath(path);
                        data.SetTimeToNow();
                        dataVector.push_back(data);


                        for (int i = 0; i < 2; i++){
                                                //Attempt to send data for 2 times
                        IOT_API sendApi (wapiceUrl, wapiceUser, wapicePassword);
                        if (sendApi.SendData(wapiceDevID, dataVector) != IOTAPI::IOT_ERR_OK)
                        {
                            std::cerr << "Failed to send data for "<< i + 1 << " times." <<
                        std::endl;
                            std::cout << "Failed to send data for " << i + 1 << " times." <<
                        std::endl;

                            wapicecounter = wapicecounter +1;
                            i++;
                        }
                        else {
                            i = 2;
                        }
                        }
                        wapicecounter = wapicecounter - 1;
                        dataVector.clear();

                        return;
                        }
```

Tekstileike 4. Tiedon lähettäminen IoT-Ticketiin

5 POHDINTA

Tämän projektin teknisessä toteutuksessa onnistuttiin hyvin. Tilakoneohjelma ja sensorilaitteet toimivat suunnitellusti ja niitä voidaan käyttää olosuhdelaboratoriossa tai missä tahansa ympäristössä, jossa on mahdollista rakentaa LAN-verkko käyttöä varten. Ohjelman asentaminen ja käyttäminen ohjeiden avulla on mahdollista myöskin sellaisille käyttäjille, joilla ei ole osaamista ohjelmoinnista ja ohjelmaan pystytään lisäämään uusia ominaisuuksia käyttötarkoitusten niin vaatiessa. Projektiin jäi kuitenkin huomattavan paljon parannettavaa. Vaikka ohjelman rakenne mahdollistaakin uusien viestiprotokollien ja pilvipalvelimien lisäämisen, niille ei rakennettu selkeitä, yhteneväisiä rajapintaratkaisuja, joiden kanssa osien lisääminen olisi yksinkertaisempaa. Toiseksi, ohjelmassa käytettiin olio-ohjelmoinnin mahdollistavaa ohjelmointikieltä, sen luokkarakenteiden hyödyntäminen jäi puuttumaan. Käytännöllisin tapa toteuttaa sensorien konfigurointi ohjelman sisällä olisi ollut sensoriluokan rakentaminen tarvittavilla ominaisuuksilla. Näin konfiguraatiotaulukkoja ei olisi tarvittu. Samoilla ratkaisuilla viestiprotokollien ja pilvipalveluiden lisääminen olisi yhdenmukaistunut.

Projektin aikana opin paljon IoT-järjestelmistä ja talotekniikan ilmanlaatuun liittyvistä mittausteknologioista. Sensoriverkon suunnitteleminen ja kokoaminen alusta asti vaati huomattavan määrän tutkimusta ja opettelua erilaisista teknologioista, sekä tutustumista talotekniikan sensorien liittimien teknisiin ominaisuuksiin. Huomasin, että suunnitellessa tietoteknisiä ratkaisuja muille aloille substanssiosaamisen hankkiminen on tärkeä osa kokonaisuutta, vaikka työnkuvaan kuuluisikin vain ohjelmistokehitystä. Kokonaisuudessa olen tyytyväinen projektin tuloksiin ja oppimiini asioihin. Tämän kaltaisissa projekteissa on useita mahdollisia toteutustapoja, joiden valintaan vaikuttaa esimerkiksi käyttökohde, suoritusvaatimukset ja tilaavan tahon kokemukset teknologioista. Yleensä mikään valinta ei ole yksin täysi oikea, vaan vaatimukset täyttäviä vaihtoehtoja on useita. Olen tyytyväinen tässä projektissa tekemiini valintoihin ja niiden mahdollistamaan oppimiskokemukseen.

LÄHTEET

Alho, H. (2017). Noudettu osoitteesta https://github.com/Kromii/kaspos/

Arduino. (2017). Haettu 21. 4 2017 osoitteesta https://www.arduino.cc/en/Guide/Introduction

Arduino. (2017). Haettu 21. Huhtikuu 2017 osoitteesta https://www.arduino.cc/en/Main/ArduinoBoardMega2560

ARM Limited. (ei pvm). Noudettu osoitteesta https://www.arm.com/

Broadcom Inc. (ei pvm). Noudettu osoitteesta https://www.broadcom.com/

EQUA Simulation AB. (2017). Noudettu osoitteesta https://www.equa.se/fi/ida-ice

Hyperrealm. (ei pvm). Haettu Huhtikuu 2017 osoitteesta https://hyperrealm.github.io/libconfig/

JSON. (ei pvm). Haettu Huhtikuu 2018 osoitteesta http://www.json.org/

Raspberry Pi Foundation. (ei pvm). Raspberrypi.org. Haettu 20. Huhtikuu 2017 osoitteesta https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md

Raspbian. (ei pvm). Raspbian.org. Haettu 21. 4 2017 osoitteesta https://www.raspbian.org/RaspbianFAQ

Treinen, R. (8. Helmikuu 2016). Debian mailing lists. Haettu 21. Huhtikuu 2017 osoitteesta https://lists.debian.org/debian-devel/2016/02/msg00122.html

Turunen, V. (2017). Noudettu osoitteesta http://urn.fi/URN:NBN:fi:amk-201705158273

Upton, E. (8. Syyskuu 2016). Raspberry Pi Foundation. Haettu 11. Huhtikuu 2017 osoitteesta https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/

Vaisala Oyj. (2018). Noudettu osoitteesta https://www.vaisala.com

Vaisala Oyj. (2018). GMW80-Series-Datasheet. Noudettu osoitteesta https://www.vaisala.com/sites/default/files/documents/GMW80-Series-Datasheet-B211435EN.pdf

Wapice Ltd. (ei pvm). Haettu Huhtikuu 2018 osoitteesta http://www.wapice.com

Wapice Ltd. (ei pvm). IoT-Ticket. Haettu Huhtikuu 2017 osoitteesta IoT-Ticket: https://iot-ticket.com

Wapice Ltd. (ei pvm). Github/IoT_Ticket. Haettu Huhtikuu 2018 osoitteesta https://github.com/IoT-Ticket/IoT-LinuxCppClient