IOT-YHDYSKÄYTÄVÄLAITTEEN TOTEUTTAMINEN

Case: ITKO-hanke

LAB-AMMATTIKORKEAKOULU
Insinööri (AMK)
Tieto- ja viestintätekniikka
Ohjelmistotekniikka
Syksy 2020
Mikael Björke

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Björke, Mikael	Opinnäytetyö, AMK	Syksy 2020
	Sivumäärä	
	34	

Työn nimi

IoT-yhdyskäytävälaitteen toteuttaminen
Case: ITKO-hanke

Tutkinto

Insinööri (AMK)

Tiivistelmä

Opinnäytetyössä suunniteltiin ja toteutettiin IoT-yhdyskäytävälaitteille tarkoitettu ohjelmisto, joka kerää ja lähettää sensoreilta mitattua dataa palvelimelle aikasarjatietokantaan tallennettavaksi. Opinnäytetyö tehtiin toimeksiantona LAB-ammattikorkeakoulun ITKO-hankkeelle, jonka tarkoituksena on parantaa Päijät-Hämeen alueen yritysten IoT ja big data valmiuksia.

Opinnäytetyön teoriaosuudessa perehdytään erilaisiin viestintätapoihin ja datan kuljetusmenetelmiin. Lisäksi käydään läpi toteutuksessa käytettyjen sensoriteknologioiden toimintaperiaatteita ja ominaisuuksia.

Yhdyskäytävälaitteen ohjelmistokehityksessä päädyttiin käyttämään moniprosessista arkkitehtuuria, jonka myötä ohjelmiston toiminnallisuus jaettiin erillisiin prosesseihin. Ohjelmiston eri osat viestivät toisilleen käyttäen prosessien väliseen kommunikointiin tarkoitettua menetelmää hyödyntäen ZeroMQ-viestintäkirjastoa. Sensoreilta kerätyn datan lähettämiseen käytettiin MQTT-protokollaa, joka mahdollisti tietoturvallisen ja luotettavan kommunikointiväylän yhdyskäytävälaitteen ja palvelimen välille. Ohjelmointikielenä käytettiin pääasiassa Python-ohjelmointikieltä.

Lopputuloksena saatiin toteutettua yleiskäyttöinen IoT-yhdyskäytävälaite, joka mahdollistaa datan keräämisen ja lähettämisen erilaisilta sensoriteknologioilta käyttäen niiden rajapintoja. Ohjelmistokehityksessä käytetty arkkitehtuuriratkaisu tukee ohjelmiston skaalautuvuutta helpottamalla uusien sensoritukien ja ominaisuuksien kehittämistä.

Asiasanat

IoT, tiedonkeruujärjestelmä, ohjelmistosuunnittelu, ohjelmistokehitys, Python

Abstract

Author(s)	Type of publication	Published
Björke, Mikael	Bachelor's thesis	Autumn 2020
	Number of pages	
	34	

Title of publication

Implementation of IoT Gateway Device
Case: ITKO project

Name of Degree

Bachelor of Engineering

Abstract

The goal of this thesis was to design and implement a software for IoT gateway devices that collects data from sensors and sends it to a server for storage in a time series database. The thesis was commissioned by the ITKO project of LAB University of Applied Sciences. The purpose of the ITKO project is to improve the IoT and big data capabilities of companies in the Päijät-Häme region.

The theoretical part of the thesis introduces various communication and data transport methods. In addition, it reviews operating principles and features of the sensors used in the implementation.

The software of the gateway device ended up implementing a multi-process architecture, which divided the software functionality into separate processes. The different parts of the software communicate with each other using an inter-process communication method utilizing the ZeroMQ communication library. The collected data was transmitted from gateway devices using the MQTT protocol, which offered a secure and reliable communication connection between gateway devices and the server. The software was mainly developed with the Python programming language.

The result of the thesis was general-purpose IoT gateway device that allows data to be collected and transmitted from various sensor technologies using different interfaces. The software architecture solution used in the implementation also supports software scalability by facilitating the development of new sensor supports and features.

Keywords

IoT, data gathering system, software design, software development, Python

SISÄLLYS

Sanasto

IoT	Internet of Things. Verkkoon liitetty esine tai asia
ITKO	Yrityslähtöiset IoT-ratkaisut ja koneoppiminen -hanke
IPC	Inter-process Communication. Prosessien välinen kommunikointi
MPA	Multi-process architecture. Ohjelmistoarkkitehtuuri, joka koostuu useasta eri prosessista
MQTT	Message Queuing Telemetry Transport. Sovelluskerroksen viestintäprotokolla
Node	Z-Wave verkossa oleva Z-Wave laite
P2P	Point-to-Point. Kahden laitteen välinen yhteys
QoS	Quality of Service. Mahdollistaa tietoliikenteen luokittelun eri laatutasoille
Socket	Verkon kommunikaatioyhteyden alku- tai päätepiste
TCP	Transmission Control Protocol. Vikasietoinen kurjetuskerroksen tiedonsiirtoprotokolla
TLS	Transport Layer Security. Salausprotokolla tietoliikenteen salaamiseen
WSS	WebSocket Secure. Salattu WebSocket yhteys
X.509	Salauskirjoitukseen käytetty standardi, jota käytetään erilaisissa internet protokollissa

1 JOHDANTO

IoT, eli esineiden internet muodostuu sensoreiden, ohjelmiston, sekä internetyhteyden kokonaisuudesta. Sensorilaitteet eivät aina ole suorassa yhteydessä internettiin, vaan ne vaativat läheisyyteensä laitteen, joka toimii yhdyskäytävänä niiden ja palvelimen välillä.

Eri toimialojen yritykset tarvitsevat toisistaan poikkeavia IoT-ratkaisuja omiin käyttökohteisiinsa. Sensoreita voidaan hyödyntää muun muassa kiinteistöautomaatiossa, ympäristön tarkkailussa sekä teollisuuden ohjausjärjestelmissä. Sensorilaitteita ja niiden käyttämiä teknologiarajapintoja löytyy näin ollen huomattava määrä, joka tuo haasteita yleiskäyttöisen yhdyskäytävälaitteen kehittämiseen.

Opinnäytetyö suoritetaan osana ITKO-hanketta (Yrityslähtöiset IoT-ratkaisut ja koneoppiminen), jonka tavoitteena on kehittää Päijät-Hämeen alueen pienten ja keskisuurien yritysten IoT, data pipeline ja koneoppimisen hyödyntämismahdollisuuksia. Hankkeen aikana muodostettavalla yleiskäyttöisellä sovellusalustalla toteutetaan data pipeline sensoreilta kerätylle datalle. Pipeline koostuu datan tallentamisesta, analysoinnista, koneoppimisen mallien hyödyntämisestä, sekä tekoälyn soveltamisesta. (LAB ammattikorkeakoulu, 2019.)

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa yleiskäyttöinen ohjelmisto IoT-yhdyskäytäväkäytävälaitteelle. Laitteen tulee suorittaa datan kerääminen siihen joko langattomasti tai langallisesti liitetyiltä sensoreilta ja lähettää hankittu data tallennettavaksi palvelimelle käyttäen tietoturvallisuutta noudattavia menetelmiä. Ohjelmiston tulee tukea useita erilaisia projektissa käytettäviä sensoriteknologioita ja mahdollistaa tulevaisuudessa uusien sensoritukien ja ominaisuuksien lisääminen.

Työn teoriaosuudessa käsitellään yhdyskäytäväratkaisussa käytettyjä teknologioita ja niiden toimintaperiaatteita. Näihin kuuluvat muun muassa erilaiset sensorit ja niiden käyttämät protokollat, sekä yhdyskäytävälaitteen suorittamat sisäiset ja ulkoiset kommunikointimenetelmät. Yhdyskäytävälaitteen ohjelmistokehitys toteutetaan pääosin Python ohjelmointikielellä.

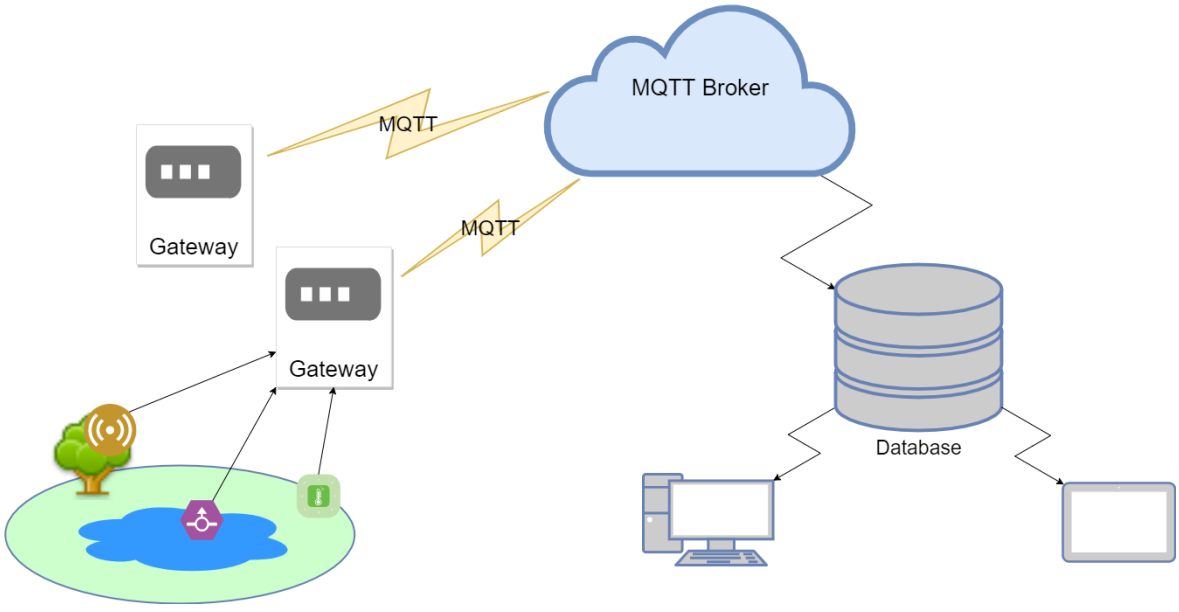
2 ASIAKASVAATIMUKSET

Opinnäytetyön aikana käytettäviin menetelmiin ja ratkaisuihin päädytään ITKO hankkeelle asetettujen tavoitteiden johdattelemina. Tämän luvun tavoitteena on esitellä suurempaa kokonaiskuvaa hankkeen tavoitteista.

ITKO hankkeen aikana tullaan kehittämään infrastruktuuri, joka koostuu kolmesta seuraavasta työpaketista:

- IoT ja datansiirto
- sovellusalusta (universal platform)
- koneoppiminen ja tekoäly. (EURA 2014, 2019.)

IoT ja datansiirto on osuus, jota tullaan toteuttamaan osana opinnäytetyötä. Paketti pitää sisällään datan mittauksen sensoreilta, sen keräämisen yhdyskäytävälaitteelle, valmistelun lähetettävään muotoon sekä tietoturvallisen siirron palvelimen aikasarjatietokantaan tallennettavaksi (Kuva 1). Tavoitteena on myös tutkia IoT viestintätapoja, johon kuuluu erityisesti MQTT-protokollan toiminnallisuus ja ominaisuudet. Protokollaa tullaan käyttämän IoT-datan siirtämisessä yhdyskäytävälaitteelta palvelimelle.



Kuva 1. Kokonaiskuva IoT datansiirron toiminnallisuudesta

Sovellusalustapaketin tavoitteena on kehittämää yleiskäyttöinen alusta, joka pystyy toimimaan joko yritysten omissa järjestelmissä tai ulkoistettuna kolmannen osapuolen pilvipalveluissa. Sovellusalusta ei saa kuitenkaan olla ulkoistettuna riippuvainen pilvipalveluiden palveluntarjoajista tai niiden ominaisuuksista, vaan sen tulee olla siirrettävissä vapaasti

palvelimelta toiselle. Alusta pitää sisällään välittäjän, joka vastaanottaa yhdyskäytävälaitteilta saatavaa dataa ja ohjaa sen sovellusalustalla olevaan tietokantaan.

Koneoppiminen ja tekoäly ovat osana sovellusalustalle rakennettavaa data pipelinea, joka koostuu kerätyn datan analysoinnista, visualisoinnista sekä koneoppi- ja tekoälymallien hyödyntämisestä. Näitä käyttämällä kerättyä dataa voidaan käyttää hyödyksi esimerkiksi havaitsemalla ihmisille tai ympäristölle haitallisia vahinkotekijöitä tai vaikka ennakoimalla kiinteistöjen energiatehokkuutta.

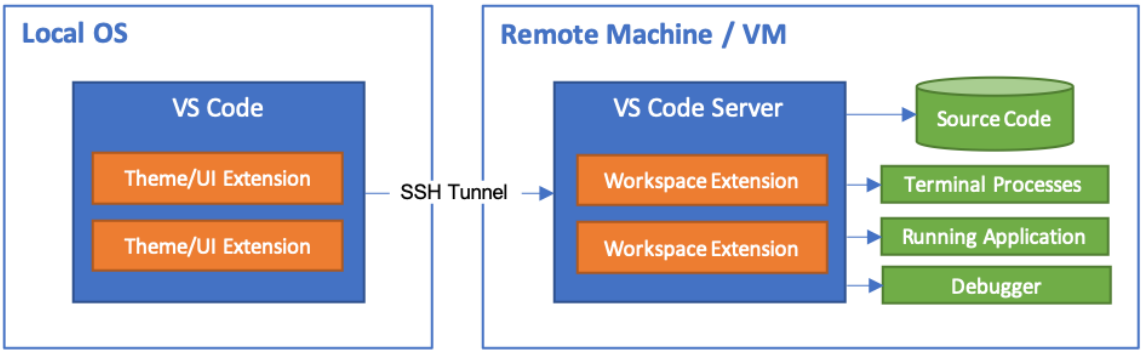
Infrastruktuurin eri osien käyttöönottoprosessien tulisi olla mahdollisimman automatisoitua, jotta sovellusalustojen lisääminen uusille palvelimille olisi nopeata ja yksinkertaista. Tarve automatisoinnille koskee myös IoT-yhdyskäytävälaitteiden asennus- ja ylläpitotoimintoja. Lisäksi yhdyskäytävälaitteen kehityksessä tulee ottaa huomioon IoT alustan mahdollinen laajentuminen, jotta uusien sensoritukien ja -ominaisuuksien kehittäminen olisi helposti toteutettavissa.

Infrastruktuuria tullaan testaamaan hankkeen aikana suoritettavissa piloteissa, joihin osallistuvat mukana olevat yhteistyöyritykset. Tämä mahdollistaa tarvittavien ominaisuuksien määrittelyn ja testaamisen jo toteutuksen alkuvaiheessa.

3 TEKNOLOGIAT

3.1 Kehitysympäristö

Yhdyskäytävälaitteen ohjelmistokehitys suoritettiin käyttäen Visual Studio Code koodieditoria ja siihen kehitettyä Remote SSH lisäosaa. Remote SSH mahdollistaa ohjelmistokehityksen suoraan yhdyskäytävälaitteelta käsin muodostamalla siihen etäyhteyden SSH tunnelin kautta. Kuva 2 havainnollistaa kyseisen lisäosan toiminnallisuutta.



Kuva 2. Remote SSH lisäosan toiminnallisuus (Visual Studio Code 2020)

SSH tunnelin muodostamisen yhteydessä etälaitteelle asentuu VS Code palvelin, joka mahdollistaa VS Coden ominaisuuksien käytön myös etäkehityksessä. Ominaisuuksiin kuuluu muun muassa IntelliSense eli tekstin viimeistely, lisäosien hyödyntäminen etälaitteella, sekä etäterminaalin käyttö. Tiedostoja ja kansiota voidaan selata editorilla samaan tapaan kuin paikallisessa kehityksessä. (Visual Studio Code, 2020.)

Ohjelmiston kehityksessä käytettiin pääasiassa Python ohjelmointikieltä, jota ajettiin omasta virtuaaliympäristöstä. Virtuaaliympäristöllä pyritään helpottamaan ohjelmiston käyttöönottoa uusilla alustoilla huolehtimalla Python version ja käytettävien pakettien yhteensopivuudesta. Virtuaaliympäristö sisältää oman sisäisen Python version sekä listan ohjelmiston käyttämistä kirjastopaketeista ja niiden versionumeroista. Paketit ladataan ja asennetaan ohjelmiston käyttöönoton yhteydessä pip työkalulla, joka on tarkoitettu Pythonin pakettienhallintaa varten.

3.2 Datan kuljetustavat

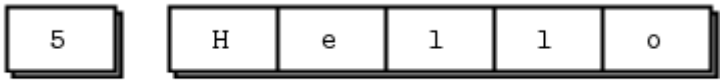
Yhdyskäytävälaitteen ohjelmistolle asetettu vaatimus pystyä skaalautuvuuteen johti päätökseen jakaa ohjelma pienempiin osiin. Tämän myötä ilmeni tarve käyttää menetelmää, joka siirtää sensoreilta kerättävää dataa ohjelmiston osilta toisille.

Tässä osiossa esitellään kahta eri sovellustason kommunikointitekniikka, joita yhdyskäytävälaitteen ohjelmistokehityksessä päädyttiin käyttämään. Tekniikoita hyödynnettiin ohjelmiston sisäisessä kommunikoinnissa sekä palvelimen ja yhdyskäytävälaitteen välisessä viestinnässä.

3.2.1 ZeroMQ

ZeroMQ on asynkroniseen kommunikointiin tarkoitettu avoimen lähdekoodin viestintäkirjasto, joka on kehitetty alustariippumattomaksi usealle eri ohjelmointikielelle. Viestintä tapahtuu käyttäen socketteja, joiden avulla viestejä voidaan kuljettaa erilaisten kuljetusmenetelmien avulla. Kommunikointi tapahtuu yksisuuntaisesti oletuksena ilman välittäjää. (ZeroMQ.)

Viestit sisältävät ZeroMQ:ssa yhden tai useamman viestikehyksen, joka sisältää tiedon viestin koosta tavuissa sekä viestin sisällön (Kuva 3). ZeroMQ ei tiedä viestien sisällöstä mitään muuta kuin viestin pituuden, joten ohjelmistokehittäjän tulee itse huolehtia datan oikeanlaisesta formatoinnista.



Kuva 3. ZeroMQ viestikehys (ZeroMQ)

Kuljetusmenetelmät

Socketteja yhdistäessä niille annetaan osoite, jota käytetään yhteyden päätepisteenä (endpoint). Osoitteen etuliitteenä ilmoitetaan viestin toimittamiseen käytetty kuljetustapa, joka määrittää myös käytettävän protokollan. Taulukko 1 esitellään mahdollisia kuljetustapoja ja niiden käyttökohteita.

Kuljetusmenetelmä	**Kuvaus**
IPC (inter-process communication)	Prosessien välinen kommunikointiyhteys. Socketille määritellään järjestelmästä tiedostopolku, jonka välityksellä viestit lähetetään. Esimerkki osoitteesta: `ipc:///tmp/socket-name`

Inproc (in-process)	Prosessien sisäinen kommunikointiyhteys. Voidaan käyttää esimerkiksi monisäikeisissä ohjelmissa, jossa kommunikointia tarvitaan säikeiden välillä. Esimerkki osoitteesta: `inproc://socket-name`
TCP	TCP yhteydellä toimiva viestintäyhteys. Socketteja luodessa määritetään osoite ja portti. Mahdollistaa myös laitteiden välisen kommunikoinnin verkon yli. Esimerkki osoitteesta: `tcp://192.168.1.150:60088`
PGM (pragmatic general multicast)	Monilähetysprotokolla, jota käytettäessä viesti lähetetään usealle vastaanottajalle verkossa. Osoitteeseen määritellään verkkosovitin, monilähetysosoite sekä portti. Esimerkki osoitteesta: `pgm://192.168.1.1;239.1.1.1:5555`
VMCI	Virtuaalikoneiden ja isäntäkoneen väliseen kommunikointiin tarkoitettu rajapinta. Osoitteessa määritellään kommunikaation tunnus ja rajapinnan portti. Esimerkki osoitteesta: `vmci://communication-id:5555`

Taulukko 1. ZeroMQ kuljetusmenetelmät (mukailtu ZeroMQ API)

Viestintämallit

Socketteja luodessa niille annetaan tyypit, jotka määrittävät niiden tehtävän viestinnässä. Erilaiset socket tyypit muodostavat keskenään erilaisia viestintämalleja. Jokaisella viestintämallilla on omat ominaisuutensa ja käyttökohteensa. Taulukko 2 vertaillaan erilaisia viestintämalleja ja luetellaan niihin käytettäviä socket tyyppejä.

Viestintämalli	**Kuvaus**
Julkaisija–tilaaja	Tietojenjakelumalli, jossa kaikki tilaajat vastaanottavat julkaisijan lähettämän viestin. Tilaajasocketit voivat suodattaa tilaamiaan viestejä määrittämällä vastaanotettavaksi vain halutuilla tavujonoilla alkavia viestejä. Socket tyypit: `publish, subscribe`

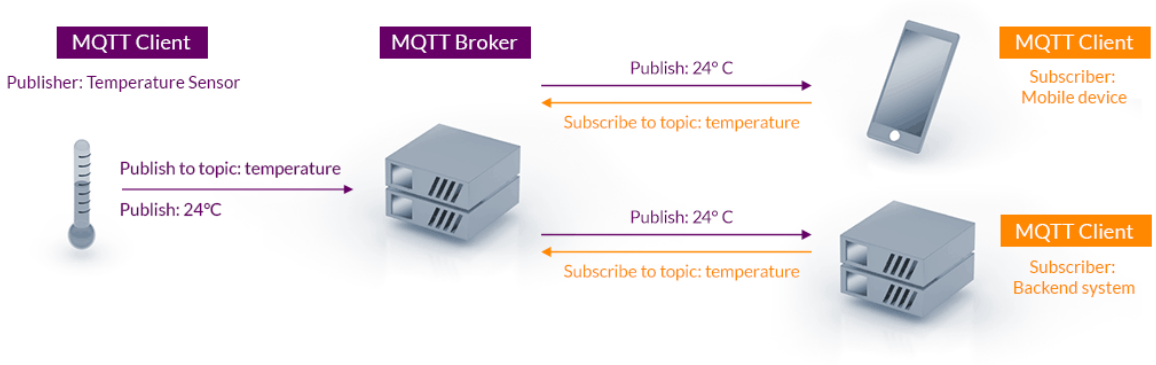
Pyyntö–vastaus	Tehtävienjakelumalli, jossa pyyntö lähetetään socketilta vastaajasocketeille. Vastaajasockettien joukosta vain yksi lähettää vastausviestin pyytäjäsocketille. Pyytäjien ja vastaajien välissä voi olla pyyntöjen reitittäjiä sekä jakajia. Socket tyypit: `request, reply, router, dealer`
Pipeline	Nimensä mukaisesti pipeline-malli, jossa viestit lähetetään yksisuuntaisten sockettien kautta niiden välissä oleville nodeille. Toisin kuin muissa viestintämalleissa, pipelinea käytettäessä viestien vastaanottoa ei tarvitse erikseen käsitellä. Socket tyypit: `push, pull`
Eksklusiivinen pari	Eroten muista viestintämalleista, jotka voivat sisältää halutun määrän socketteja, nämä socketit voivat muodostaa yhteyden vain yhteen vastakappaleeseen. Tarkoitettu käytettäväksi prosessien sisäiseen kommunikointiin. Socket tyypit: `pair`

Taulukko 2. ZeroMQ viestintämallit (mukailtu ZeroMQ API)

3.2.2 MQTT

MQTT (Message Queuing Telemetry Transport) on OASIS standardoitu sovelluskerrokseen kuuluva viestintäprotokolla, joka käyttää viestien kuljettamiseen TCP-yhteyttä. MQTT:tä käytetään pääasiallisesti IoT kommunikointiin ja se on laajalti käytetty monilla eri teollisuuden aloilla. (MQTT.)

Viestintäarkkitehtuuri (Kuva 4) pohjautuu julkaisija–tilaaja malliin, jossa asiakas (client) vastaanottaa viestejä tilaamistaan aiheista (topic). Asiakkaat voivat myös julkaista omia viestejä. Asiakkaiden välisenä viestinviejänä toimii välittäjä (broker), johon asiakkaat muodostavat yhteyden. Välittäjä toimittaa aiheeseen julkaistun viestin jokaiselle aiheeseen tilanneelle asiakkaalle. Uusimmissa MQTT-versioissa välittäjästä käytetään myös nimitystä palvelin (server).



Kuva 4. MQTT viestintäarkkitehtuuri (MQTT)

Laatutasot

Yksi MQTT:n avainominaisuuksista on sen tarjoamat viestityksen laatutasot (Quality of Service). Kolmella eri laatutasolla voidaan määrittää sopimus viestien lähettäjien ja vastaanottajien välille. Sopimukset määrittävät varmuustason, jolla viestien toimituksesta huolehditaan. Erilaisia laatutasoja käytetään tapauskohtaisesti riippuen viestien tärkeydestä ja verkkoyhteyden vakaudesta.

Viestien laatutasot:

- Viesti lähetetään saamatta vahvistusta sovelluskerroksessa tapahtuvasta vastaanotosta (taso 0).
- Viesti lähetetään vähintään kerran varmistaen, että vastaanotosta saadaan vahvistus (taso 1).
- Viesti lähetetään varmistaen, että viesti vastaanotetaan vain kerran (taso 2).

Laatutasoja käytettäessä tulee ottaa huomioon viestien toimituksen kaksi eri vaihetta, eli viestin julkaisu asiakkaalta välittäjälle ja viestin toimittaminen välittäjältä tilaaville asiakkaille. Julkaisijan määrittämä laatutaso pätee vain julkaisijan ja välittäjän välillä, kun taas viestien toimittaminen välittäjältä tilaajille määräytyy vastaanottajien asettaman laatutason mukaan. (HiveMQ, 2015a.)

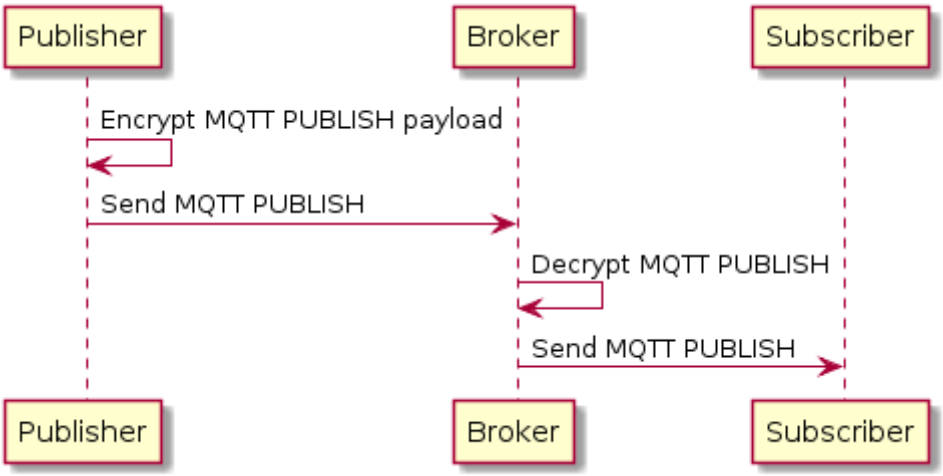
Tietoturva

Tietoturva on kriittinen osa IoT-laitteiden käyttöä, mutta se jätetään silti liian usein huomioimatta järjestelmiä kehittäessä (IoT Security Foundation). Dataa on tärkeä suojata ulkopuolisilta tahoilta ja huolehtia sen turvallisesta toimituksesta palvelimelle. MQTT mahdollistaa erilaisia kuljetus- ja sovelluskerroksen tietoturvaratkaisuja, joilla voidaan todentaa asiakkaan pääsyoikeuksia, suojata lähetettävää dataa tai huolehtia muista tietoturvariskeistä.

Asiakkaille voidaan asettaa todennusvaatimuksia, jotka toimivat kriteereinä välittäjään yhdistämiseksi ja estävät samalla ulkopuolisia kuuntelemasta aiheisiin lähetettyjä viestejä. Todennusmenetelmiksi voidaan määritellä käyttäjänimen ja salasanan perusteella tunnistautuminen, asiakastunnuksen (client ID) käyttö, sekä X.509 sertifikaatin esittäminen välittäjälle. Todennuksia käytettäessä tulee ottaa huomioon, että kirjautumistiedot ja viestit lähetetään välittäjälle salaamattomana, joten kommunikointiin tulee käyttää myös kuljetuskerroksen salausmenetelmiä.

Todennustapojen käyttäminen mahdollistaa myös asiakkaille määriteltävien käyttöoikeuksien hyödyntämisen. Käyttöoikeuksilla pystytään rajaamaan asiakkaiden pääsy vain tiettyihin MQTT ominaisuuksiin. Asiakkaan käyttämille tunnuksille voidaan esimerkiksi antaa lupa julkaista tai tilata vain rajattuihin aiheisiin. Todennuksien ja oikeuksien hallitsemiseen voidaan käyttää erilaisia pääsynhallintaratkaisuja, joista yleisimpiä ovat roolipohjaiset käyttöoikeudet (RBAC) tai pääsynvalvontaluettelo (ACL). (HiveMQ, 2015b.)

MQTT toimii TCP yhteyden päällä, joka on oletuksena salaamaton. TLS:n (Transport Layer Security) avulla tietoliikenne pystytään salaamaan niin, että asiakkaan ja välittäjän välinen yhteys on suojattu (Kuva 5).

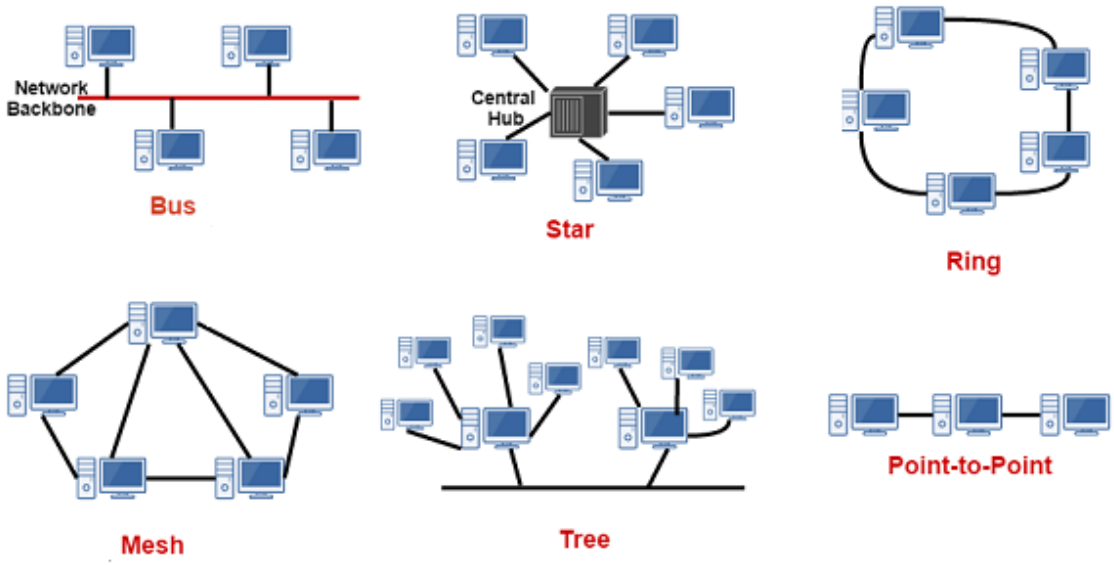


Kuva 5. Asiakkaan ja välittäjän välisen yhteyden suojaaminen (Hive MQ 2015c)

MQTT mahdollista myös tietoliikenteen WebSockettien kautta, jolloin salattu yhteys voidaan muodostaa WebSocket Secure protokollan avulla. Tämä mahdollistaa MQTT protokollan käytön oletuksena olevien HTTP ja HTTPS porttien läpi. (HiveMQ, 2015d.)

3.3 Sensorit

IoT laitteille ja sensoreille löytyy lukuisia eri käyttötarkoituksia. Niitä voidaan hyödyntää muun muassa kiinteistöautomaatiossa, ympäristön tarkkailussa sekä teollisuuden ohjausjärjestelmissä. Sensorit ovat joko langallisia tai langattomia, ja ne pystyvät toteuttamaan erialisia verkkotopologioita toistensa ja yhdyskäytävälaitteen kanssa (Kuva 6).



Kuva 6. Erilaisia verkkotopologioita (Sayeed 2017)

Tässä osiossa käydään läpi erilaisten sensoriteknologioiden teoriaa ja toimintaperiaatteita. Käsitellyille laitteille kehitettiin myös tiedonkeruutuki projektin aikana.

3.3.1 Z-Wave laitteet

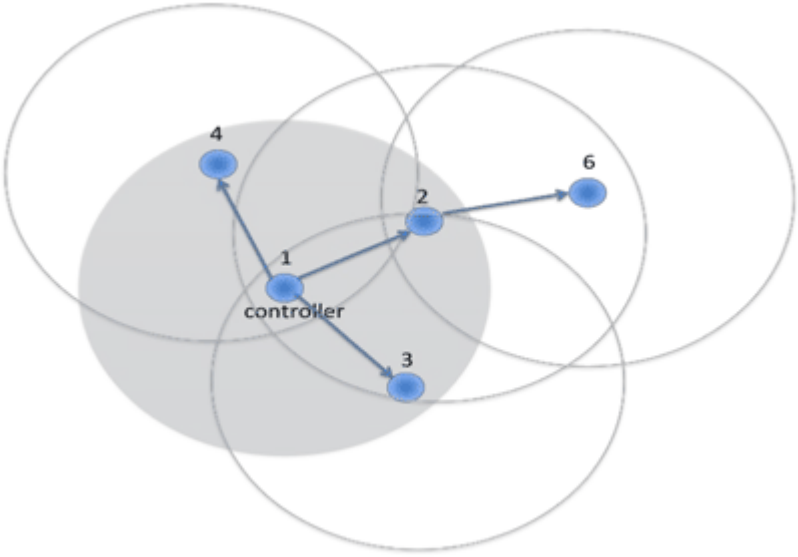
Z-Wave on lyhyen kantaman langaton protokolla, joka on kehitetty käytettäväksi asuintilojen automatisointiin tarkoitetuissa laitteissa. Kiinteistöautomaation laitteisiin sisältyy muun muassa älylukkoja, -valoja, -termostaatteja, sekä sensoreita. Vuonna 2020 markkinoilla oli yli 3000 erilaista sertifioitua Z-Wave laitetta. (Z-Wave.)

Z-Wave signaalin kantavuus vaihtelee riippuen laitteista ja ympäristötekijöistä, mutta tyypillinen kantama on noin 15–30 metriä ilman fyysisiä esteitä. Signaalin matkan pidentämiseksi on myös olemassa signaaleiden toistimia ja vahvistimia, jotka edelleenlähettävät signaaleja ja samalla vahvistavat niitä mahdollistamalla kantaman pidentymisen jopa 150 metriin asti (Aeotec Group 2020).

Jokaisen Z-Wave laitevalmistajan tuottamat laitteet käyttävät samaa Z-Wave tuoteperheeseen kuuluvaa järjestelmäpiiriä, joka mahdollistaa yhteensopivuuden eri laitevalmistajien laitteiden välillä. Laitteiden kesken pystytään muodostamaan mesh-verkko, jossa laitteet kykenevät viestimään toisilleen. Verkossa olevista Z-Wave laitteista käytetään nimitystä node.

Verkossa olevat kontrollerit ovat laitteita, jotka ohjaavat nodejen toimintaa. Yhdessä Z-Wave verkossa voi olla useampi kontrolleri, mutta yhden niistä on oltava pääkontrolleri, joka ohjaa verkon tapahtumia ja hallitsee nodeja muun muassa käsittelemällä niiltä saatavat viestit. Pääkontrolleri on myös vastuussa verkkoon liitettävien nodejen lisäämisestä antamalla niille oman verkkotunnuksensa. Verkossa olevat nodet yksilöidään myös pääkontrollerin toimesta jakamalla nodeille omat uniikit node-tunnukset. (Vesternet 2012.)

Z-Wave teknologia mahdollistaa myös viestien reitittämisen, eli laitteet pystyvät kommunikoimaan myös kantamansa ulkopuolella oleville nodeille välittämällä viestejä niiden välissä olevien nodejen välityksellä (Kuva 7). Reitittäminen tapahtuu hyödyntämällä nodejen ylläpitämää listaa, joka sisältää tiedon niiden kantaman sisäpuolella olevista naapureista.



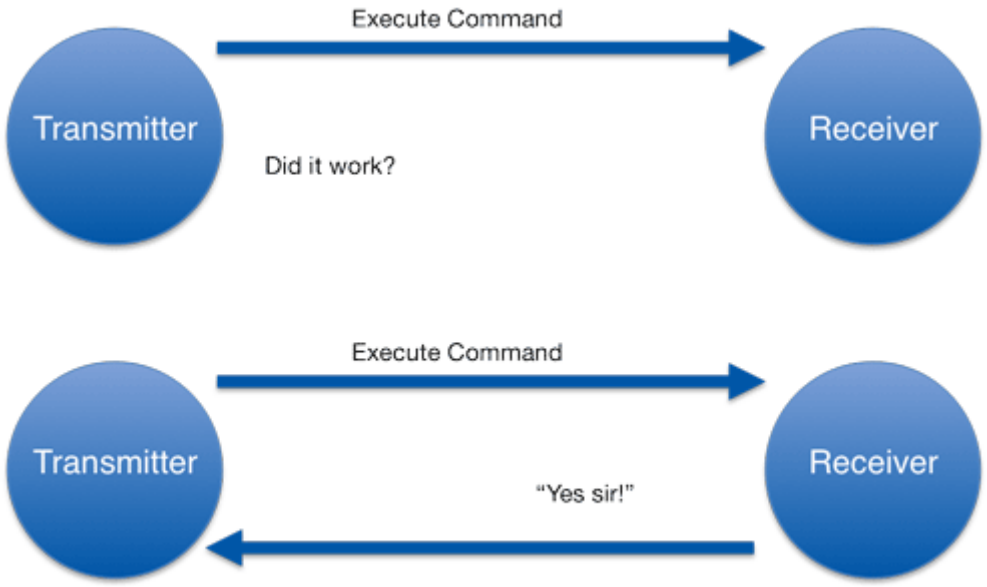
Kuva 7. Z-Wave mesh-verkko (Vesternet 2012)

Kuten kuvasta huomataan, node 6 on kontrollerin (node 1) kantaman ulkopuolella. Reititysominaisuuden ansiosta kontrolleri pystyy kuitenkin viestimään node 6:n kanssa node 2:n välityksellä, joka on molempien nodejen kantaman alueella. Kyseinen ominaisuus toimii kuitenkin vain verkkovirrassa olevilla laitteilla johtuen siitä, että akkukäyttöiset laitteet ovat virtaa säästävässä horrostilassa, jolloin ne heräävät vain tietyin väliajoin päivittääkseen tilaansa ja lähettääkseen tietoja itsestään.

Z-Wave teknologia koostuu kolmesta eri kerroksesta, jotka muodostavat yhdessä protokollan toiminnallisuuden:

- Radiokerros sisältää Z-Wave signaalin taajuuden, koodauksen sekä siirron verkon ja fyysisen radiolaitteen välillä.
- Verkkokerros pitää sisällään tiedon siitä, miten dataa vaihdetaan kahden noden välillä. Näihin kuuluu esimerkiksi viestien osoitukset ja reititykset.
- Sovelluskerros sisältää tiedot laitteiden tekemien toimintojen suoritusohjeista.

Verkkokerros sisältää myös kolme omaa alikerrosta, jotka ovat MAC-kerros, kuljetuskerros sekä reitityskerros. Kerrokset toimivat yhdessä käyttäjältä näkymättömissä ja varmistavat nodejen välisen virheettömän kommunikoinnin käyttämällä viestinnässä hyödyksi vastaanottojen kuittauksia ja viestien ohjaamista lyhyimpien reittien kautta. Kuva 8 esitetään kahden Z-Wave laitteen välistä viestinlähetystä. Laite lähettää komentoviestin langattomasti, jonka jälkeen se jää odottamaan kuittausta viestin vastaanotosta. Mikäli kuittausta ei saada, viesti yritetään lähettää uudestaan kolmanteen yritykseen asti. Onnistuneen vastaanoton tapahtuessa vastaanottaja lähettää kuittaussignaalin takaisin viestin lähettäjälle.



Kuva 8. Lähetin-vastaanotin kommunikaatio (Vesternet 2012)

3.3.2 RuuviTag

RuuviTag on suomalaisen Ruuvi Innovations Ltd. Oy:n suunnittelema avoimen lähdekoodin ympäristösensori, joka mittaa lämpötilaa, ilmankosteutta, ilmanpainetta sekä liikettä.

Laite on sijoitettu vesitiiviin kotelon sisään, joka hengittää kannessa olevan vedenkestävän tarran läpi (Kuva 9). RuuviTag toimii täysin langattomasti ja saa virtansa vaihdettavissa olevalta CR2477 paristolta. RuuviTagin pienen virrankulutuksen ansiosta paristojen käyttöiän luvataan olevan useita vuosia. (Ruuvi.)

Kuva 9. RuuviTag sensori (Ruuvi)

RuuviTag toimii majakkana (beacon), eli se jakaa mittaustuloksia ympäristöönsä sen lähistöllä olevien laitteiden kuunneltavaksi. Tämän ominaisuuden ansiosta datan kerääminen ei vaadi laitteiden parittamista tai yhteyden muodostamista. RuuviTag käyttää oletuksena datan lähettämiseen Bluetooth protokollaa, mutta tukee myös Wirepas, Mira OS ja Quuppa yhteyksiä sekä muita 2.4 GHz taajuudella toimivia protokollia. (Ruuvi, 2019.)

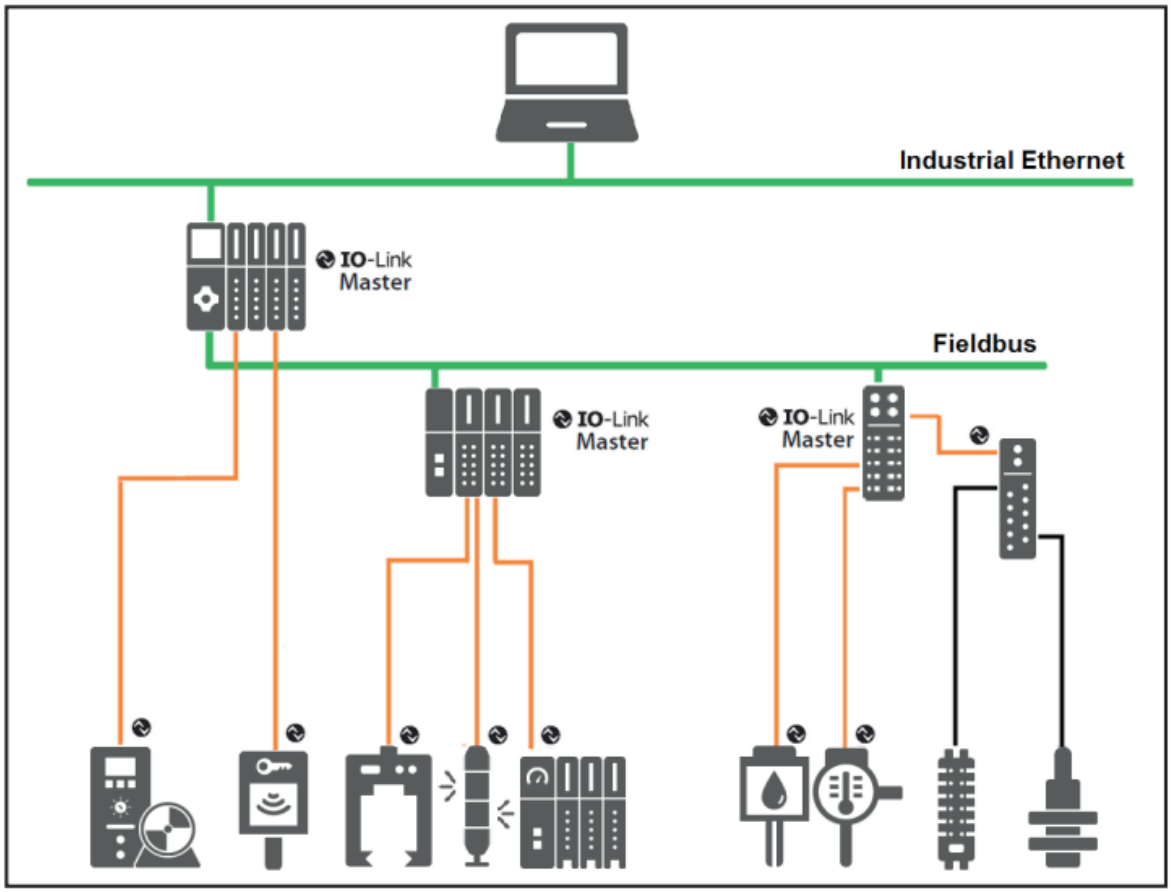
Lähetettävän viestin formaatti määräytyy laitteeseen asennetun laiteohjelmiston (firmware) pohjalta. Virallista formaattia noudattavat viestit sisältävät tavujonon, joka muodostuu seuraavista osista:

- viestin tyyppi
- laitevalmistajakohtaiset tiedot
- dataformaatin versio
- datan tietosisältö.

Datan tietosisällön rakenne määrittyy käytetyn formaattiversion mukaan. Viimeisin formaattiversio 5 sisältää tiedon lämpötilasta, relatiivisesta ilmankosteudesta, paineesta, liikkeestä, paristojen varauksesta, signaalinvoimakkuudesta, liiketunnistuslaskurista, mittaustunnisteesta ja RuuviTagin MAC-osoittesta. Mittaustunniste muuttuu joka kerta, kun mittaus suoritetaan sensoreilta ja sitä voidaan käyttää samojen mittaustulosten tallentamatta jättämiseksi, mikäli mittaustulosten kuunteluväli on tiheä. (Jousimaa, 2020.)

3.3.3 IO-Link laitteet

IO-Link on teollisuuden viestinnän verkkostantardi (IEC 61131-9), jota käytetään sensoreiden ja toimilaitteiden (actuators) yhdistämiseksi joko kenttäväylään tai Ethernetiin. IO-Link järjestelmä koostuu IO-Link masterista (isäntä), sekä siihen liitetyistä IO-Link laitteista (Kuva 10). Laitteet voivat olla esimerkiksi erilaisia sensoreita, venttiilinavaajia tai RFID-lukijoita. IO-Link laitteet liitetään isäntään kaapeleilla, jotka mahdollistavat kaksisuuntaisen kommunikointitavan point-to-point (P2P) sarjaliitännän kautta. Kaksisuuntaisuus mahdollistaa masteriin liitettyjen laitteiden ohjaamisen ja konfiguroinnin (IO-Link, 2018)



Kuva 10. Esimerkki IO-Link arkkitehtuurista (IO-Link 2018)

P2P ominaisuuden ansiosta mastereita voidaan myös liittää toisiinsa Ethernet porttien välityksellä, joka vähentää tarvittavien johtojen määrää, sillä jokaista laitetta ei tarvitse yhdistää erikseen järjestelmän ohjaajaan. Mastereita varten on myös olemassa niihin liitettäviä lisävarusteita, jotka mahdollistavat langattoman yhteyden laitteiden ja mastereiden välillä (IO-Link wireless, 2018).

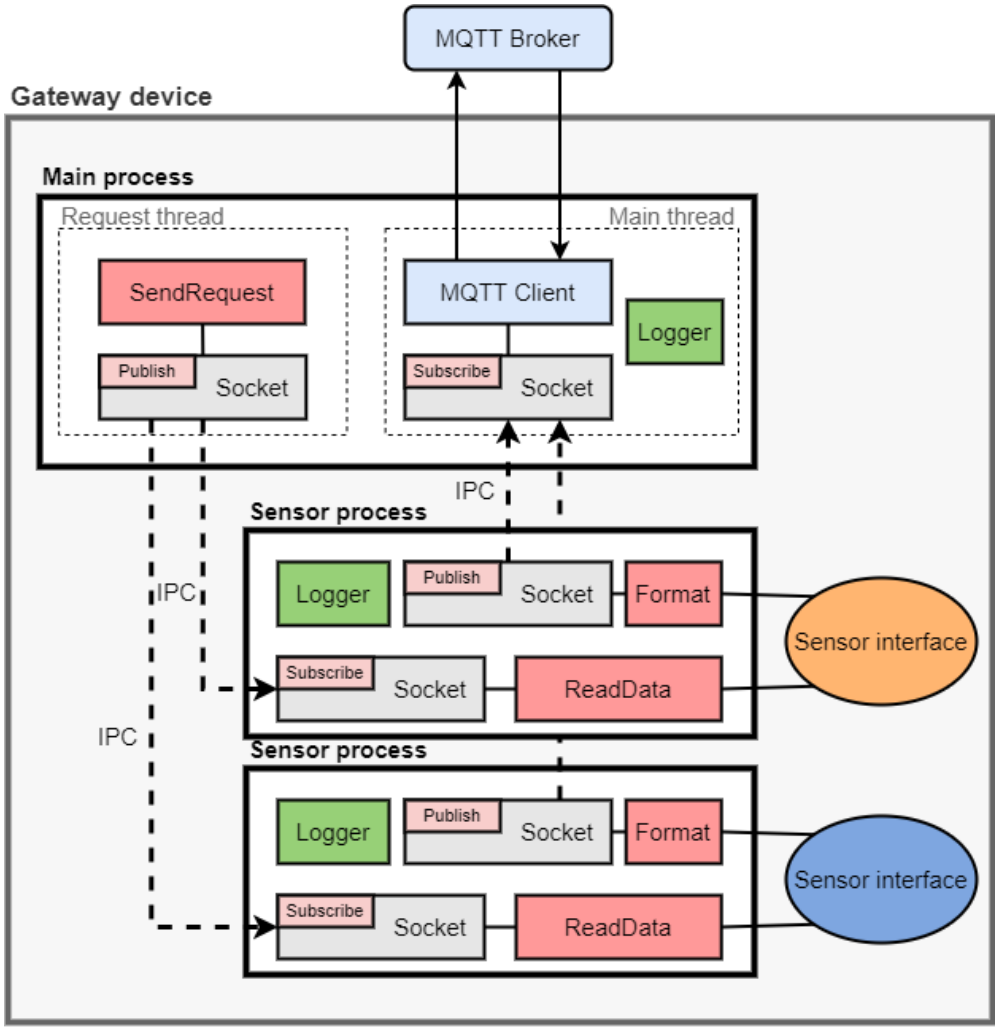
IO-Link laitteiden valmistajat tarjoavat erilaisia rajapintoja tuotteilleen. Esimerkiksi projektissa käytettävä ifm:n valmistama IO-Link master mahdollistaa laitteille kommunikoinnin JSON formaatilla joko MQTT-protokollaa tai REST rajapintaa hyödyntäen.

4 YHDYSKÄYTÄVÄLAITTEEN TOIMINTAYMPÄRISTÖ

4.1 Arkkitehtuuri

Yhdyskäytävälaitteen ohjelmistoa kehitettiin ja pilotoitiin Raspberry Pi tietokoneilla. Raspberryt osoittautuivat hyviksi testauslaitteiksi niiden tarjoamien erilaisten laitteistorajapintojen variaatioiden takia. Erilaisissa sensoritoteutuksissa tultiin hyödyntämään muun muassa Bluetoothia, USB portteja sekä GPIO pinnejä hyödyntämällä.

Ohjelmistokehityksessä päädyttiin toteuttamaan moniprosessista arkkitehtuuria (multi-process architecture). Kyseisessä arkkitehtuuriratkaisussa ohjelmistokokonaisuus on jaettu pieniin osiin, jotka muodostuvat yksittäisistä ohjelmista (Kuva 11). Ohjelmiston ajon aikana prosessit toimivat toisistaan riippumatta ja toteuttavat niille yksilöityjä käyttötarkoituksia.



Kuva 11. Arkkitehtuurin kokonaiskuva

Arkkitehtuurivalinta mahdollistaa vakaamman toiminnan ohjelmistolle, kun prosessit eivät ole riippuvaisia toistensa tilasta. Tämän takia yhdessä prosessissa mahdollisesti tapahtuva ohjelmointivirhe tai prosessin kaatuminen ei vaikuta muiden prosessien toimintaan. Arkkitehtuuri tekee ohjelmiston rakenteesta myös selkeämmän, kun erilaiset toiminnalliset osat ovat eroteltu toisistaan.

Moniprosessinen arkkitehtuurin helpottaa ohjelmiston laajennettavuutta, kun uusia sensoritukia ja ominaisuuksia voidaan kehittää muuttamatta olemassa olevaa lähdekoodia. Uudet ohjelmat saadaan liitettyä ohjelmistokokonaisuuteen luontevasti ilman tarvetta käynnistää prosesseja uudestaan. Ohjelmia kykenee myös kehittämään useilla eri ohjelmointikielillä, sillä prosessien välinen kommunikointi suoritetaan tavalla, joka tukee universaalisti kaikkia ohjelmointikieliä.

4.1.1 Prosessit

Ohjelmiston eri osien suorittamiseksi päädyttiin käyttämään Linuxin systemd työkalua, joka on tarkoitettu järjestelmän ja prosessien hallintaan. Systemd mahdollistaa prosessien suorittamisen palveluina ja tarjoaa erilaisia ratkaisuja niiden ylläpitoon. Palveluita varten luotavissa konfiguraatiotiedostoissa pystytään määrittämään erilaisia parametrejä, jotka vaikuttavat palvelun toimintaan. Parametrien avulla pystytään muun muassa osoittamaan Kuva 12 mukaisesti ohjelman ajamiseen käytettävä prosessi sekä sen käyttämät ympäristömuuttujat tai ohjeistamaan palvelun uudelleenkäynnistäminen tietyllä aikavälillä virhetilan tapahtuessa.

```
[Unit]
Description=service name
After=network.target

[Service]
EnvironmentFile=env/file/path
Type=simple
ExecStart=/python/path /program/file/path
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target
```

Kuva 12. Palvelun konfiguraatiotiedosto

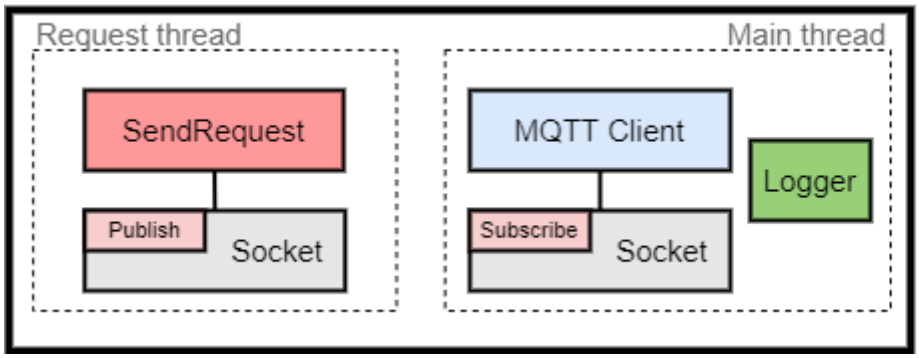
Palvelut saadaan aloitettua yhdyskäytävälaitteen käynnistymisen yhteydessä systemctl komennon enable -toiminnon avulla. Tämä luo symbolisen linkin palvelun konfiguraatiosta, jota käytetään automaattisesti käynnistettävien palveluiden aloittamiseen boottauksen yhteydessä.

Jokainen ohjelmiston prosessi pitää sisällään omaa tapahtumien kirjausjärjestelmää, joka kerää lokitietoja prosessin toiminnasta. Lokitietoja kirjataan erillisiin lokitiedostoihin, mutta niitä voi myös seurata reaaliajassa systemd:n journalctl komennon avulla. Kirjausjärjestelmän toteuttamiseen käytettiin Pythonissa vakiona olevaa lokimoduulia.

Ohjelmien ajamiseen käytetään Pythonin virtuaaliympäristöä, jonka sijainti osoitetaan palveluiden konfiguraatiotiedostoissa ExecStart parametrin arvon ensimmäisessä osassa. Virtuaaliympäristö suorittaa ohjelmistokokonaisuutta, joka muodostuu pääprosessista ja sensoriprosesseista.

Pääprosessi

Pääprosessi on vastuussa sensoreilta mitattavan datan pyytämisestä, mittaustulosten vastaanottamisesta sekä niiden lähettämisestä palvelimelle. Prosessi koostuu Kuva 13 mukaisesta kahdesta säikeestä (thread). Pyyntösäikeen tehtävänä on lähettää sensoriprosesseille mittauspyyntöjä, jotta ne suorittaisivat datan keräämisen sensoreilta. Kerätty data vastaanotetaan pääsäikeen toimesta, joka sisältää MQTT asiakkaan ja lokijärjestelmän. Datan pyyntö ja vastaanotto suoritetaan molemmissa säikeissä olevien sockettien kautta.

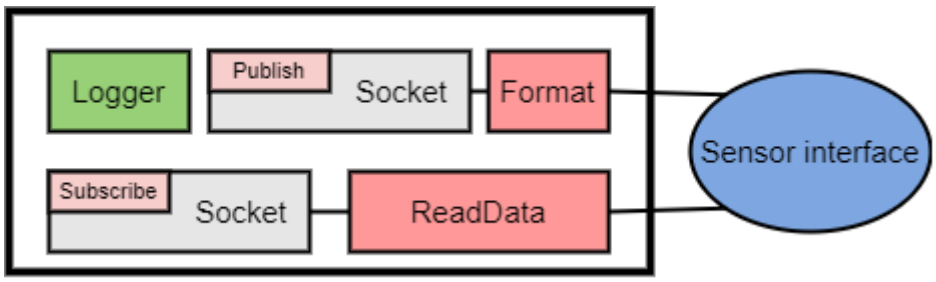


Kuva 13. Pääprosessin rakenne

Sensoriprosessit

Sensoriprosessien tehtävänä on kerätä kullekin prosessille yksilöityjen sensorityyppien mittaustuloksia ja tarvittaessa ohjata suurempia laitekokonaisuuksia, kuten esimerkiksi Z-

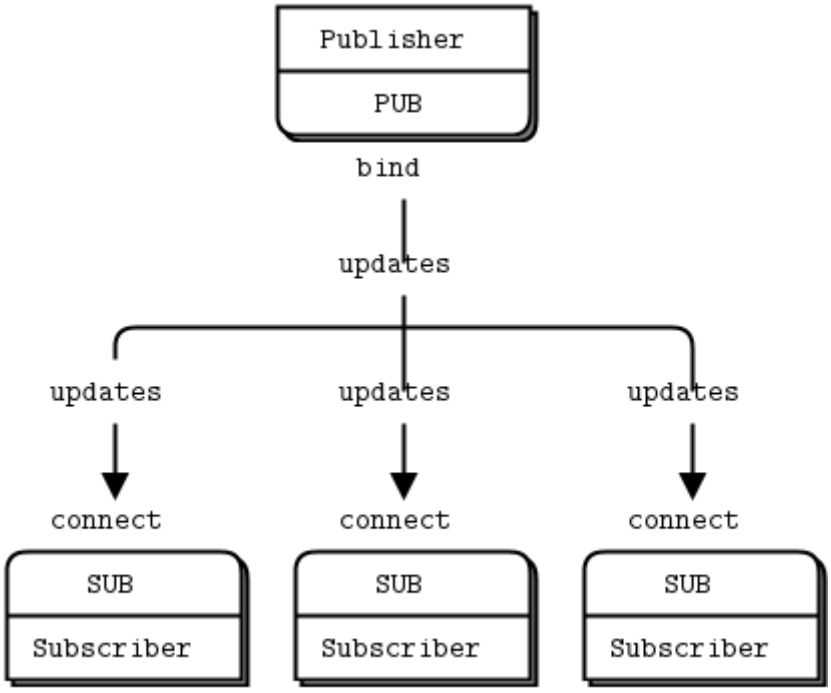
Wave mesh-verkkoa. Prosessit sisältävät socketit pääprosessin kanssa viestimiseen, sekä sensorikohtaiset rajapintaratkaisut mittausten lukemista varten (Kuva 14). Eri sensorikohtaisista ratkaisuista löytyy tarkempaa tietoa luvusta 4.2.2.



Kuva 14. Sensoriprosessin rakenne

4.1.2 Prosessien välinen kommunikointi

Prosessit viestivät keskenään ZeroMQ -kirjaston sockettien välityksellä. Viestintämallina päädyttiin käyttämään julkaisija–tilaaja -mallia (Kuva 15), joka mahdollistaa datapyyntöjen lähettämisen kaikille sensoriprosesseille kerrallaan. Kyseinen viestintämalli ei kuitenkaan mahdollista kaksisuuntaista kommunikaatiota, joten ongelma ratkaistiin käyttämällä jokaisessa prosessissa viestien kuunteluun ja lähettämiseen omia socketteja.



Kuva 15. Toteutettu julkaisija–tilaaja viestintämalli (ZeroMQ)

Viestien kuljetukseen käytettiin prosessien väliseen kommunikointiin tarkoitettua IPC kuljetustapaa. Pääprosessi luo päätepisteet kahdelle socketilleen väliaikaiseen tiedostopolkuun, joihin sensoriprosessit yhdistävät itsensä prosessien käynnistyttyä.

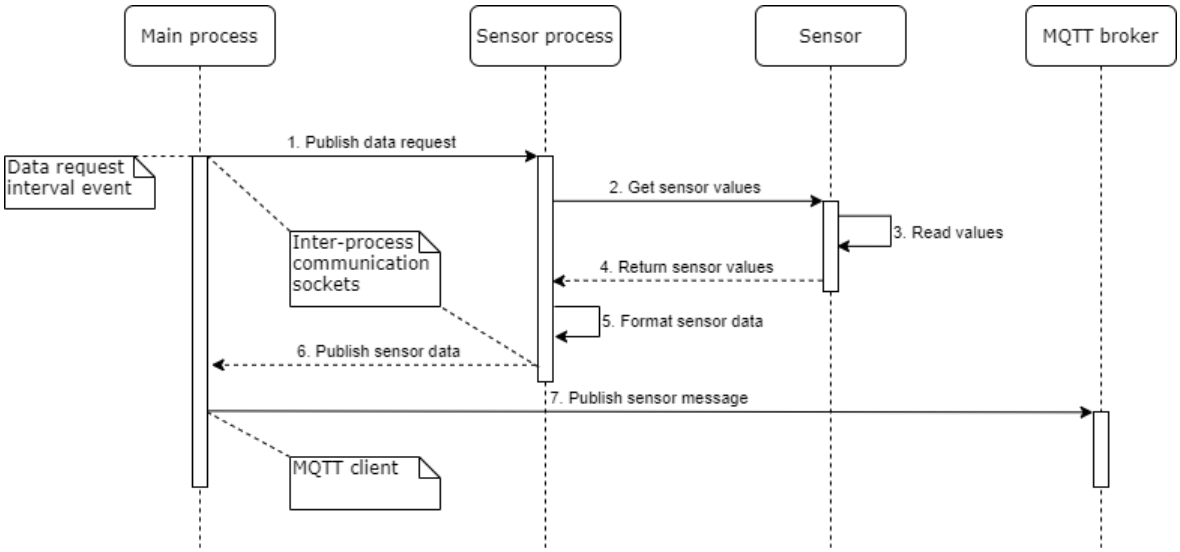
4.1.3 MQTT asiakas

Pääprosessissa olevaa MQTT-asiakasta käytetään viestinnässä yhdyskäytävälaitteen ja palvelimen välillä. Mittaustuloksia saadessa pääprosessi ohjaa datan MQTT:n välityksellä palvelimella olevalle MQTT välittäjälle. MQTT mahdollistaa myös kaksisuuntaisen yhteyden, joten viestejä voidaan myös vastaanottaa yhdyskäytävälaitteen ulkopuolelta. Välittäjältä saatavat viestit voivat olla muun muassa sensorikohtaisia komentoja tai laitteen konfigurointia koskevia ohjeita.

MQTT asiakas käyttää tietoturvalliseen viestintään protokollan tarjoamia erilaisia ominaisuuksia. Välittäjään yhdistämiseksi vaaditaan todennus käyttäjänimellä, salasanalla sekä TLS sertifikaatilla. Palvelimen ja yhdyskäytävälaitteen välinen yhteys salataan WebSocket Secure protokollaa käyttäen. MQTT:tä päädyttiin käyttämään WSS:n välityksellä, koska kommunikointi tapahtuu silloin oletusportin 443 kautta, jolloin palvelimille ei tarvitse erikseen avata omia MQTT portteja.

4.2 Sensoridatan kerääminen

Kuva 16 oleva sekvenssikaavio antaa kokonaiskuvaa sensoreilta mitattavan datan sijainnista yhdyskäytävälaitteella keräyksen eri vaiheiden aikana alkaen sen lukupyynnöstä ja päättyen palvelimelle lähettämiseen. Seuraavissa alaluvuissa käydään sekvenssikaavion vaiheet yksityiskohtaisemmin läpi ja esitellään erilaisille sensorirajapinnoille kehitettyjä toteutuksia.



Kuva 16. Sensoridatan sekvenssikaavio

Datan hakeminen alkaa pääprosessin julkaistaessa lukupyynnön sensoriprosesseille pyyntösocketin kautta. Sensoriprosessien vastaanottaessa pyynnön, ne lukevat sensoreilta mittaustulokset ja formatoivat datan, jonka jälkeen se lähetetään pääprosessille datasocketin kautta. Pääprosessilta datan edelleenlähetetään MQTT asiakkaan toimesta palvelimella olevalle MQTT välittäjälle.

4.2.1 Mittauspyynnöt ja toistovälit

Sensoreilta haettavien mittaustietojen lukuvälit määräytyvät pääprosessin lähettämien mittauspyyntöjen aikavälin mukaan. Pääohjelma sisältää mittauspyyntöjen lähettämiseen tarkoitetun säikeen (thread), joka julkaisee tietyin aikavälein sensoreille mittauspyyntöjä. Mittauspyynnöt lähetetään niille tarkoitetun socketin kautta, joka toimittaa viestin IPC kuljetustapaa hyödyntäen kaikille sensoriprosesseille.

4.2.2 Sensoriohjelmat ja mittaukset

Sensoriprosessit odottavat mittauspyyntöjen saapumista tilaamiinsa datapyyntösocketteihin. Pyyntöjen saapuessa mittaustietojen lukemiseen käytetään jokaiselle eri sensoriteknologialle toteutettua omaa datan lukumenetelmää, joka käyttää hyödyksi sensoreille tarkoitettuja kirjastoja ja rajapintoja.

Yhdyskäytävälaitteen metriikat

Yhdyskäytävälaitteen metriikoita keräämällä voidaan seurata laitteen tilaa, käytettyjä resursseja ja havaita epänormaaleja tapahtumia, kuten esimerkiksi suorituskykyongelmia. Laitteelta kerätään muun muassa tilastoa suoritinkäytöstä, muistista, levytilasta sekä mahdollisesti käytössä olevan akun varauksesta.

Z-Wave

Projektin aikana suoritetuissa piloteissa oli käytössä useita erilaisia Z-Wave laitteita. Z-Wave verkon pääkontrollerina käytettiin RaZberry lisäkorttia (Kuva 17), joka on yhdistettävissä Raspberry Pin GPIO-liittimiin. Kontrollerin toimintaa ohjattiin yhdyskäytävälaitteelle kehitetyn Z-Wave prosessin välityksellä.

Kuva 17. RaZberry Z-Wave lisäkortti (Z-Wave.me)

Prosessi ylläpitää mesh-verkkoa ja seuraa sen tilaa. Verkon tilasta pidetään yllä XML-konfiguraatiotiedostoa, joka sisältää tiedon verkossa olevista nodeista, nodejen laitetiedoista sekä käyttäjän määrittelemistä node yksilöinneistä. Yksilöinteihin voidaan määrittää noden nimi, sijainti ja tuotenimi. Laitetiedot saadaan toteutuksessa käytettävän Z-Wave-kirjaston

konfiguraatiotiedostoista, jotka sisältävät listan olemassa olevista Z-Wave laitteista ja niiden ominaisuuksista. Verkkoon liitetyn laitteen tunnuksen pohjalta voidaan muun muassa selvittää laitteen nimi, valmistaja sekä sensorien suorittamien mittausten nimikkeet ja yksiköt. Laitteista olemassa olevien listausten ansiosta ohjelmasta saatiin aikaan geneerinen toteutus, joka tukee monia erilaisia Z-Wave laitteita, jolloin jokaiselle sensorille ei tarvitse erikseen kehittää omaa tukea.

Sensoridatan lukupyynnön saadessaan ohjelma palauttaa kaikilta Z-Wave verkossa olevilta nodeilta saadut sensorilukemat. Lukemat eivät tule suoraan nodeilta, vaan ne haetaan ylläpidettävän verkon välimuistista, johon pääkontrollerille lähetettyjen nodejen mittaustiedot päätyvät. Sensorilaitteiden suorittamien mittausten aikavälit ovat laitekohtaisia, mutta aikaväleihin vaikuttavat myös laitteiden käyttämät virtalähteet. Ohjelmaa kehittäessä huomattiin, että kun Multisensor 6 laitteita pidettiin toiminnassa paristoilla, niin laitteet siirtyivät horrostilaan vähentääkseen virrankulutusta. Multisensorit heräsivät tunnin välein horroksesta suorittaakseen mittauksia ja palasivat takaisin nukkumaan lähetettyään tulokset kontrollerille.

IO-Link

IO-Link laitteille suoritetuissa piloteissa käytettiin ifm:n valmistamaa IO-Link masteria (Kuva 18) sekä lämpötila- ja etäisyyssensoria. Master- ja yhdyskäytävälaitteen välinen kommunikointi toteutettiin EtherNet/IP ja IoT Core rajapintoja käyttäen. IoT Core on IO-Link masterissa oleva REST-rajapinta, jota käytetään laitteiden ohjaamista ja sensoridatan keräämistä varten.

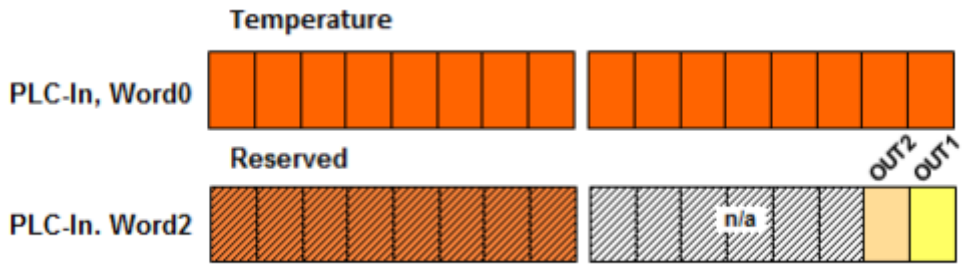
Kuva 18. IO-Link master (ifm)

Mittaustulokset haettiin rajapintaan lähetettävien HTTP-pyyntöjen avulla. Pyynnöt lähetetään masterin IP-osoitteeseen POST-metodeilla, jotka sisältävät palveluluokan tunnuksen, korrelaatio ID:n, datapisteen sekä käytettävän palvelun nimen (Kuva 19). Datapisteessä määritellään myös masterin portti, johon laite on kytketty.

```
body = '''{
    "code":10,
    "cid":1,
    "adr":"/iolinkmaster/port[%s]/iolinkdevice/pdin/getdata"
}''' % (port)
```

Kuva 19. POST-pyynnön rakenne

IO-Link masterilta palautettava vastaus saadaan JSON formaatissa, joka sisältää pyydetyn sensoridatan. Data on kuitenkin koodattuna heksadesimaaliksi, joten se tulee muuntaa luettavaan muotoon. Data saattaa kuitenkin sisältää myös muita sensorin antamia arvoja, joten heksadesimaalista muunnettavan bittijonon biittejä tulee siirtää (shift) päästääkseen halutun mittausarvon alueelle. Bittijonojen pituudet ovat laitekohtaisia ja oikealle alueelle päästäkseen tulee viitata kyseiselle laitteelle löytyvään bittitaulukkoon (Kuva 20), joka löytyy laitteen valmistajan dokumenteista.



Kuva 20. Lämpötilasensorin bittitaulukko (ifm 2017)

Lämpötilasensorin mittausarvon saamiseksi bittejä tulee siirtää tässä tapauksessa oikealle 16 askelta käyttämällä loogista sivuttaissiirtoa. Kyseinen määrä saadaan selvitettyä taulukosta, josta huomataan bittijonon kokonaispituuden olevan 32 bittiä (0–31). Jono alkaa taulukon viimeisestä solusta, joten mittaustuloksen kannalta tärkeät bitit ovat 16–31.

RuuviTag

RuuviTag sensoreiden lähettämien viestien vastaanottamiseen käytettiin Raspberry Pin sisäänrakennettua Bluetooth sovitinta sekä RuuviTag sensoreita varten kehitettyä Python kirjastoa. Kirjaston avulla data saadaan dekoodattua suoraan sensoreiden lähettämistä Bluetooth paketeista, jolla vältetään raa'an binääridatan manuaalinen käsittely.

RuuviTagien lähettämien datojen keräämiseen oli seuraavia vaihtoehtoja:

- Jokaisen sensorin lähettämä data käsitellään erikseen takaisinkutsua hyödyntäen.
- Dataa kerätään vain tietyiltä RuuviTageilta MAC-osoitteen perusteella.
- Data kerätään yhdeltä RuuviTagilta MAC-osoitteen perusteella.
- Data kerätään RuuviTageilta mittaustuloksen muuttuessa.
- Data kerätään kaikilta RuuviTageilta, jotka lähettävät viestejä vastaanottajan kantaman sisäpuolella.

Parhaimmaksi vaihtoehdoksi osoittautui viimeisimmäksi mainittu menetelmä, joka toteutettiin kuuntelemalla dataa aikakatkaisun kanssa kaikilta kantaman sisällä olevilta RuuviTageilta. Tämä mahdollistaa uusien mittalaitteiden lisäämisen järjestelmään ilman erillisiä toimenpiteitä tai MAC-osoite määrittelyjä.

4.2.3 Valmistelu

Sensoridatojen valmistelun tarkoituksena on luoda mittaustuloksista tietorakenne, jota voidaan kuljettaa yhtenäisenä pakettina palvelimelle asti. Dataan voidaan tässä vaiheessa lisätä sen tietokantaan tallentamisen kannalta tärkeää olevaa tietoa, kuten esimerkiksi sensorilaitteiden tunnisteita. Z-Wave laitteiden tapauksessa yksilöinteihin käytettiin nodejen nimityksiä sekä laitteiden tuotenimiä, kun taas RuuviTagit eroteltiin MAC-osoitteiden perusteella.

Lisäksi datan objektirakenteeseen lisätään tunnus, joka kertoo datan alkuperästä. Tunnusta käytetään myöhemmin MQTT-viestiä julkaistaessa. Datan valmistelun tapahduttua se lähetetään sensoriprosessilta pääprosessille Python objektina.

4.2.4 Lähettäminen

Pääohjelman vastaanotettaessa dataobjektin socketin kautta, se julkaistaan MQTT asiakkaan toimesta palvelimelle. Ennen lähettämistä objektista poimitaan tunnus, joka asetettiin siihen sensoriohjelmassa. Tunnuksen tarkoituksena on ohjata viesti sille kuuluvaan sensoriaiheeseen. Aiheen etuliitteenä käytetään yhdyskäytävälaitteen tunnusta, jota seuraa sensorikohtainen tunnus. Dataobjekti muunnetaan viestin julkaisun yhteydessä Python objektista JSON formaattiin.

Datan lähettämisessä tulee ottaa huomioon mahdolliset MQTT asiakkaan ja välittäjän välisen yhteyden häiriötilat. Mahdolliset häiriötilat voidaan ratkaista datan puskuroinnilla (buffering), jolla tarkoitetaan datan väliaikaista tallentamista muistiin, kun sitä ollaan siirtämässä toiseen paikkaan. Puskuroinnin avulla voidaan varmistaa datan säilyvyys mahdollisissa vikatilanteissa, kuten esimerkiksi verkkoyhteyden hetkittäisten katkosten tapahtuessa.

Yhdyskäytävälaitteen lähettämien mittaustuloksien puskuroimiseen käytetään MQTT-protokollan tarjoamaa mahdollisuutta valita eri viestinnän laatutasojen välillä (QoS). Asetettaessa QoS vähintään tasolle 1, voidaan määrittä viestien uudelleenlähetysyritykset välittäjälle, kunnes viesti on vastaanotettu.

Puskuroinnissa täytyy ottaa myös huomioon mittaustulosten hetkittäisyys. Verkkoyhteyden katketessa teoreettisesti kahdeksi minuutiksi välittäjä saa yhteyden uudelleenmuodostuessa takautuvia mittausarvoja, jolloin niitä normaalisti kirjoittaessa tietokantaan aikaleimat eivät pidä enää paikkaansa. Tämä voidaan ratkaista sisällyttämällä mittausarvoja julkaistaessa viestiin aikaleima ISO 8601 formaatissa, jota käytetään tietoja tallettaessa tietokantaan. Aikaleima lisätään aina UTC muodossa mahdollisten aikavyöhyke-erojen varalta.

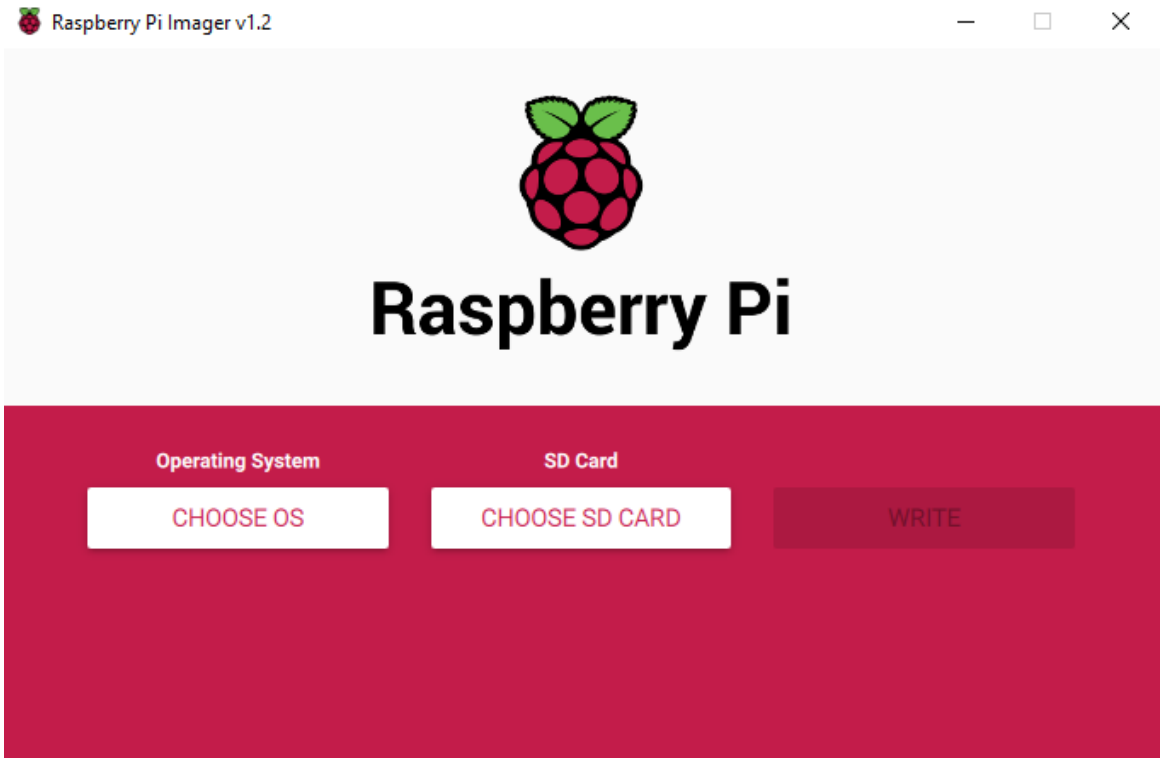
5 TUOTTEISTUS

5.1 Käyttöönotto

Yhdyskäytävälaitteen tuotteistamisella pyritään saavuttamaan ominaisuuksia ja ratkaisuja, joilla helpotetaan loppukäyttäjän laitteen käyttöä ja mahdollistetaan uusien laitteiden vaivaton käyttöönotto ja hallitseminen. Laitteen ohjelmisto on kehitetty tukemaan Linux pohjaisia käyttöjärjestelmiä. Ohjelmiston kehitysvaiheessa ja projektin aikana suoritetuissa piloteissa yhdyskäytävälaitteina käytettiin erilaisia ARM-mikroprosessoriarkkitehtuurilla toimivia Raspberry Pi 3, Raspberry Pi 4 ja Revolution Pi tietokoneita.

Raspberry Pit käyttävät massamuistinaan SD-muistikorttia, jolle käyttöjärjestelmä tulee myös asentaa manuaalisesti ennen laitteen käyttöönottoa. Käyttöjärjestelmän kirjoittaminen muistikortille vaatii koneesta kortinlukijan sekä asennustyökalun.

Asentamista varten löytyy useita erilaisia työkaluja, joista projektin aikana hyödyllisimmäksi osoittautui Raspberry Pi Foundationin kehittämä asennusohjelma Rapberry Pi Imager (Kuva 21). Kyseinen ohjelma mahdollistaa asennusvalinnan viimeisimpien käyttöjärjestelmäversioiden välillä, mutta mahdollistaa myös erillisen levykuvan asentamisen.



Kuva 21. Raspberry Pi Imager (Raspberry Pi Foundation)

5.2 Automatisointi

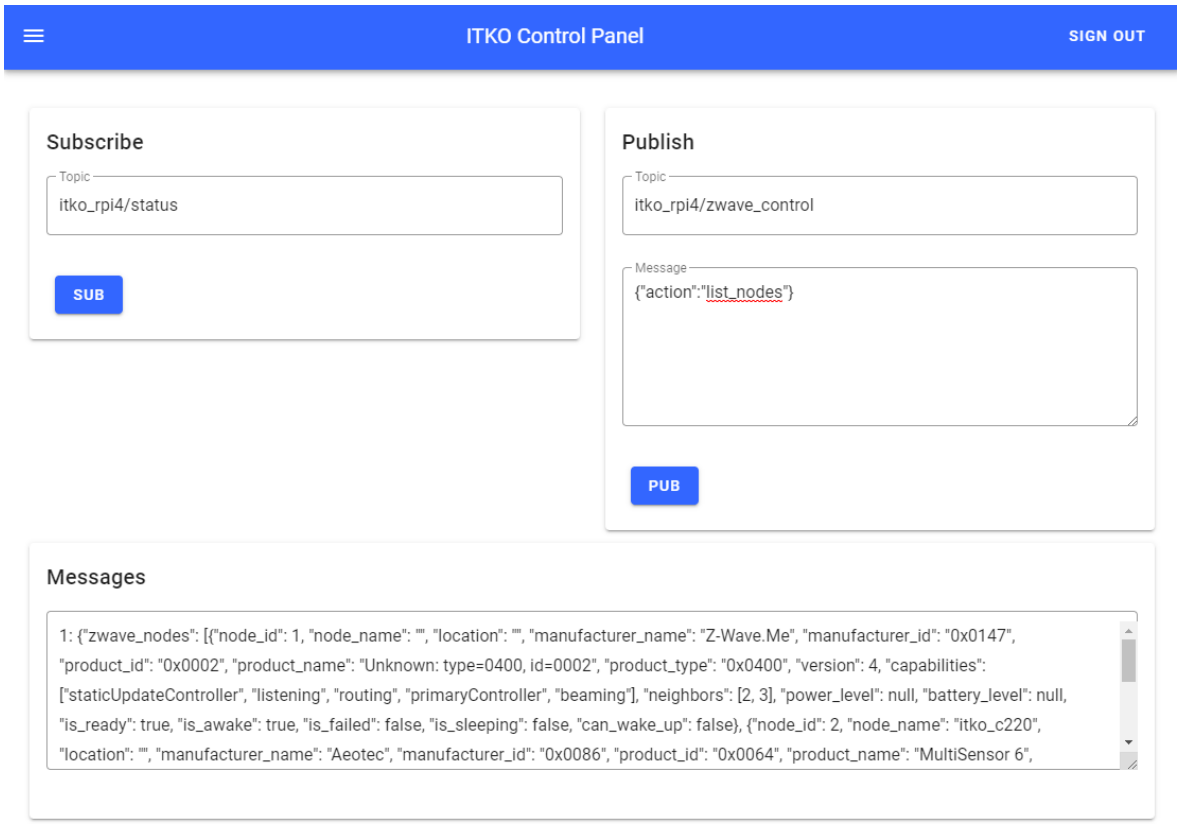
Eri asentamisvaiheiden automatisoinnilla pyritään vähentämään mahdollisten inhimillisten virheiden tapahtumista ja asennusprosessin nopeuttamista. Käyttöjärjestelmän asennuksen valmistuttua muistikortille on mahdollista määrittää laitteen käynnistyksen yhteydessä ajettavia skriptejä. Skriptien avulla laitteelle voidaan esimerkiksi konfiguroida verkkoasetuksia, antaa laitteelle isäntänimi (host name) tai käynnistää SSH palvelin, jolloin etäyhteyden muodostaminen on mahdollista heti ensimmäisen käynnistyksen yhteydessä ilman laitteelta vaadittavia erillisiä toimenpiteitä.

Laitteen käynnistyttyä sille tulee asentaa yhdyskäytäväohjelmisto. Ohjelmiston eri osien asentamista varten kehitettiin komentoriviskriptejä, jotka suorittavat ohjelmien eri asentamisvaiheet. Skripteillä luodaan muun muassa tarvittavat konfiguraatiotiedostot palveluiden ajamista varten sekä pystytetään Pythonin virtuaaliympäristö prosesseille.

5.3 Konfigurointi

Ohjelmiston parametrejä on mahdollisuus muokata sen käyttämien ympäristömuuttujien avulla. Ympäristömuuttujat sisältävät muun muassa MQTT julkaisuissa käytettävän yhdyskäytävälaitteen tunnuksen, joka assosioi palvelimella käsiteltävän datan lähteen oikealle laitteelle. Konfigurointia voidaan myös toteuttaa suoritettavien mittausten aikaväliä muuttamalla sekä vaihtamalla käytettävää palvelinta muokkaamalla välittäjään yhdistämisessä käytettäviä parametrejä.

Z-Wave laitetukea kehitettäessä huomattiin, että verkon hallitseminen on haastavaa yhdyskäytävälaitteelta käsin. Verkkoon tehtävien muutosten, kuten esimerkiksi inkluusiotilan aktivointi laitteiden lisäämistä varten, vaatisi käyttäjältä etäyhteyden muodostamista laitteeseen. Ratkaisuna päädyttiin kehittämään ITKO hallintapaneeli, joka sisältää MQTT-asiakkaan viestien tilaus- ja julkaisuominaisuuksineen (Kuva 22).



Kuva 22. ITKO hallintapaneeli

Hallintapaneelin kautta pystytään viestimään yhdyskäytävälaitteen kanssa ja ohjaamaan sen Z-Wave verkkoa lähettämällä laitteelle suunnattuun aiheeseen hallintakomentoja. Hallintakomennoilla mahdollistetaan laitteiden lisääminen ja poistaminen verkosta, sekä nodejen nimeäminen mittausdatojen yksilöintiä varten.

5.4 Tarkkailu

Yhdyskäytävälaitteen resursseja voidaan tarkastella sen lähettämien metriikoiden pohjalta. Metriikoista ilmenee muun muassa yhdyskäytävälaitteen suorittimen, muistin sekä levytilan käyttö (Kuva 23). Lisäksi prosessit kirjoittavat toiminnastaan tapahtumarekisteriä laitteen muistiin. Tapahtumarekistereistä voidaan tarpeen tullen jäljittää mahdollisia vikatilanteita.

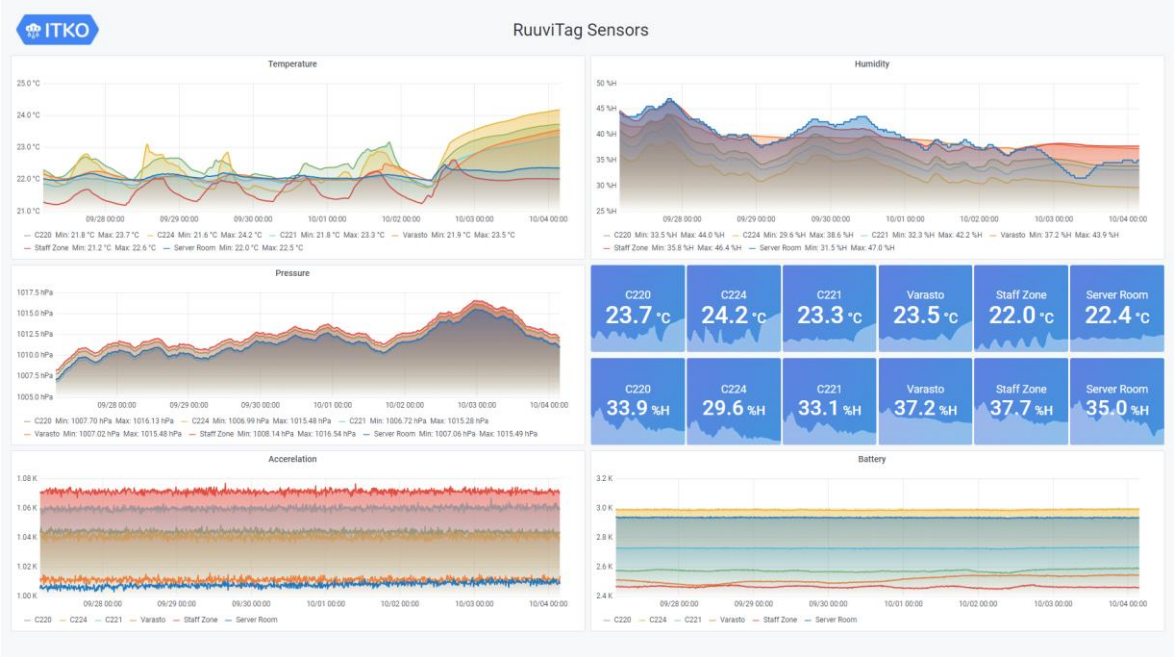
Kuva 23. Yhdyskäytävälaitteen metriikat dashboardilla

Z-Wave verkon laitteiden tilanseurantaa varten kehitettiin MQTT-viestillä toimitettava nodejen listauspyyntö, joka osoitetaan yhdyskäytävälaitteen tilaamaan aiheeseen. Pyynnön käsiteltyään yhdyskäytävälaite julkaisee vastauksen omaan statusaiheeseensa. Vastaus sisältää listan Z-Wave verkossa olevista nodeista sekä niiden yksilöllisistä tiedoista, kuten esimerkiksi nodejen ID:stä, naapureista ja statuksista.

5.5 Datan visualisointi

Kerätyn datan visualisointiin käytettiin Grafana nimistä verkkosovellusta, joka mahdollistaa erilaisten dashboardien luomisen datan tarkkailua ja analysointia varten (Kuva 24). Datan esittämistä voidaan toteuttaa erilaisilla mittareilla, kuvaajilla sekä muilla käyttöön otettavilla lisäosakomponenteilla. Grafana tukee datalähteenä monia tietokantoja, mukaan lukien projektissa käytettävää InfluxDB aikasarjatietokantaa.

Grafana dashboardeja on myös mahdollista esittää Grafana Kiosk työkalun avulla. Kiosk on tarkoitettu käytettäväksi dashboardien näyttämiseen suurilla näytöillä ilman tarvetta interaktiivisuuteen. Projektissa päädyttiin hyödyntämään Grafana Kioskeja datan esittämistä varten ajamalla Kiosk sovelluksia Raspberry Pi tietokoneilla ja liittämällä niitä televisioihin HDMI-kaapeleiden välityksellä.



Kuva 24. Dataa visualisoituna Grafana dashboardilla

6 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa usean eri laitevalmistajien ja teknologioiden sensoreita tukeva IoT-yhdyskäytävälaiteohjelmisto, joka kerää ja lähettää datan tallennettavaksi tietokantaan tietoturvallisesti. Tavoitteissa onnistuttiin ja lopputuloksena saavutettiin ratkaisu, joka tukee useaa eri sensorirajapintaa ja toteuttaa MQTT-protokollan tarjoamia tietoturvaratkaisuja.

Ohjelmistokehityksessä päädyttiin moniprosessiseen arkkitehtuuritoteutukseen, jossa ohjelmistokokonaisuuden toiminnallisuus on jaettu erillisiin prosesseihin. Arkkitehtuurivalinta mahdollisti vikasietoisemman toiminnan ohjelmistolle ja paransi sen skaalautuvuutta helpottamalla eri jatkokehityksen aspekteja. Arkkitehtuuri tukee kehityksessä myös eri ohjelmointikieliä johtuen IPC kommunikointitavan universaalista ominaisuudesta.

Yhdyskäytävälaitteita saatiin pilotoitua useamman kuukauden ajan eri olosuhteissa. Mittauksia suoritettiin muun muassa kerrostaloyksiössä, oppilaitoksella sekä yhteistyöyrityksen tuotantotiloissa.

Tärkeimpiä jatkokehityksen kohteita ovat yhdyskäytävälaitteen automatisointia koskevat ominaisuudet, joita opinnäytetyön aikana ei ehditty toteuttamaan. Huolimatta siitä, että asennusskriptit mahdollistavat laitteiden käyttöönoton vaivattomasti, niitä joudutaan ajamaan manuaalisesti etäyhteyden välityksellä. Kyseisen ominaisuuden voisi toteuttaa hakemalla palvelimelta boottaus skripteillä ensimmäisen käynnistyksen yhteydessä tarvittavat konfiguraatiot ja asennusskriptit. Lisäksi ohjelmiston päivittäminen tulee suorittaa manuaalisesti jokaiselle laitteelle erikseen. Tämän ratkaisuksi voitaisiin käyttää hyödyksi yhdyskäytävälaitteiden suuntaan toimivaa MQTT viestintää julkaisemalla päivityspyyntöjä niille tarkoitettuun aiheeseen.

ITKO-hanke jatkaa toimintaansa vuoden 2021 loppuun asti. Projektin seuraavina välietappeina on sovellusalustan data pipelinen muiden osien kehittäminen, joihin kuuluu kerätyn datan muodostaminen analytiikaksi sekä koneoppimisen ja tekoälyn mallien hyödyntäminen.

LÄHTEET

LAB ammattikorkeakoulu, 2019. ITKO - Yrityslähtöiset IoT-ratkaisut ja koneoppiminen. Viitattu 19.9.2020. Saatavissa https://lab.fi/fi/projekti/itko-yrityslahtoiset-iot-ratkaisut-ja-koneoppiminen

EURA 2014, 2019. Euroopan aluekehitysrahaston (EAKR) rahoittaman hankkeen kuvaus. Viitattu 29.9.2020. Saatavissa https://www.eura2014.fi/rrtiepa/projekti.php?projekti-koodi=A75244

Visual Studio Code, 2020. Remote Development using SSH. Viitattu 1.10.2020. Saatavissa https://code.visualstudio.com/docs/remote/ssh

ZeroMQ. An open-source universal messaging library. Viitattu 2.10.2020. Saatavissa https://zeromq.org/

ØMQ - The Guide.Chapter 1 – Basics. Viitattu 2.10.2020. Saatavissa https://zguide.zeromq.org/docs/chapter1/

ØMQ API. zmq_connect(3). Viitattu 3.10.2020. Saatavissa http://api.zeromq.org/4-2:zmq-connect

ØMQ API. zmq_socket(3). Viitattu 3.10.2020. Saatavissa http://api.zeromq.org/4-2:zmq-socket

MQTT. MQTT: The Standard for IoT Messaging. Viitattu 10.10.2020. Saatavissa https://mqtt.org/

HiveMQ, 2015a. Quality of Service 0,1 & 2 - MQTT Essentials: Part 6. Viitattu 10.10.2020. Saatavissa https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

IoT Security Foundation. Establishing Principles for Internet of Things Security. Viitattu 20.10.2020. Saatavissa https://www.iotsecurityfoundation.org/

HiveMQ, 2015b. Authorization - MQTT Security Fundamentals. Viitattu 22.10.2020. Saatavissa https://www.hivemq.com/blog/mqtt-security-fundamentals-authorization/

HiveMQ, 2015c. Securing MQTT Systems - MQTT Security Fundamentals Viitattu 22.10.2020. Saatavissa https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption/

HiveMQ, 2015d. TLS/SSL - MQTT Security Fundamentals. Viitattu 22.10.2020. Saatavissa https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/

Sayeed, A. 2017. Computer Network Topology Outline. Viitattu 22.10.2020. Saatavissa https://systemzone.net/computer-network-topology-outline/

Silicon Laboratories, 2020. Safer, smarter homes start with Z-Wave. Viitattu 19.9.2020. Saatavissa https://www.z-wave.com/learn

Aeotec Group, 2020. Range Extender 7 technical specification. Viitattu 25.9.2020. Saatavissa https://aeotec.freshdesk.com/support/solutions/articles/6000226828-range-extender-7-technical-specification-

Vesternet, 2012. Understanding Z-Wave Networks, Nodes & Devices. Viitattu 19.9.2020. Saatavissa https://www.vesternet.com/pages/understanding-z-wave-networks-nodes-devices

Ruuvi. What is RuuviTag? Viitattu 21.9.2020. Saatavissa https://ruuvi.com/ruuvitag-specs/

Ruuvi, 2019. RuuviTag Technical Specifications. Viitattu 21.9.2020. Saatavissa https://ruuvi.com/files/ruuvitag-tech-spec-2019-7.pdf

Jousimaa, O. 2020. Data Format 5 Protocol Specification (RAWv2). Viitattu 22.10.2020. Saatavissa https://github.com/ruuvi/ruuvi-sensor-protocols/blob/master/dataformat_05.md

ØMQ - The Guide. Chapter 2 - Sockets and Patterns. Viitattu 24.10.2020 Saatavissa https://zguide.zeromq.org/docs/chapter2/

Z-Wave.me. RaZberry. Viitattu 22.10.2020. Saatavissa https://z-wave.me/products/razberry/

IO-Link, 2018. IO-Link System Description. Viitattu 20.10.2020. Saatavissa https://io-link.com/share/Downloads/At-a-glance/IO-Link_System_Description_eng_2018.pdf

IO-Link, 2018. IO-Link Wireless Exposé. Viitattu 23.10.2020. Saatavissa https://io-link.com/share/Downloads/At-a-glance/IO-Link_Wireless_Expos%c3%a9_eng_2018.pdf

ifm. AL1350 IO-Link master with IoT interface. Viitattu 23.10.2020. Saatavissa https://www.ifm.com/us/en/product/AL1350

ifm, 2017. Temperature sensor TV7105 Interface Description. Viitattu 23.10.2020. Saatavissa https://www.ifm.com/download/files/ifm-TV7105-20170308-IODD11-en/$file/ifm-TV7105-20170308-IODD11-en.pdf

Raspberry Pi Foundation. Raspberry Pi Imager for Windows. Viitattu 21.9.2020. Saatavissa https://www.raspberrypi.org/downloads/