Niilo Niskanen TTV17SA

IoT-anturiverkon toteutus tietoverkkojen opetusympäristöön

Insinööri (AMK)

Tieto- ja viestintätekniikka

Kevät 2021

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Niilo Niskanen

Työn nimi: IoT-anturiverkon toteutus tietoverkkojen opetusympäristöön

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: esineiden internet, anturisolmu, Docker, Docker swarm, klusteri, kontti, tietoverkot

Työn tavoitteena oli kehittää esimerkkiratkaisu IoT-anturiverkosta opinnäytetyön tilaajan, Kajaanin Ammattikorkeakoulun opetuskäyttöön. Opinnäytetyö on osa suurempaa Kajaanin Ammattikorkeakoulun tietoverkkojen opetusympäristöhanketta. Esimerkkiratkaisu toteutettiin tilaajan määrittelemien vaatimuksien pohjalta. Työn käytännön toteutus jakautui kahteen osa-alueeseen, anturisolmuun ja taustajärjestelmään.

Anturisolmun tehtävä on kerätä ja lähettää anturidataa säännöllisin väliajoin langattomasti hyödyntäen MQTT-protokollaa. Anturisolmulla päätettiin mitata lämpötilaa, kosteutta ja ilmanpainetta. Suureita varten valittiin sopivat anturit. Koodi kehitettiin käyttäen Arduino IDE-ohjelmistoa. Anturisolmu suunniteltiin mahdollisimman yksinkertaiseksi, koska pääpaino on anturisolmun tiedonsiirron demonstroimisessa, ei itse mittaamisessa.

Taustajärjestelmä, josta käytetään opinnäytetyössä termiä klusteri, toteutettiin anturisolmun jälkeen. Klusteri muodostuu neljästä Raspberry Pi 3B -mikrotietokoneesta, jotka on liitetty samaan lähiverkkoon. Raspberry Pi -mikrotietokoneiden käyttöjärjestelmät käynnistettiin USB-muistista SD-kortin sijaan elinkaaren pidentämistä varten. Yhdelle Raspberry Pi -mikrotietokoneelle asennettiin Ubuntu Server 18.04.5 LTS-käyttöjärjestelmä, muille asennettiin Raspberry Pi OS. Klusteriominaisuus luotiin Docker-ohjelmistoon sisäänrakennetulla Docker swarm-työkalulla. Raspberry Pi -mikrotietokoneet liitettiin Docker swarm-klusteriin, jonka jälkeen klusterille kehitettiin tilaajan vaatimusmäärittelyn mukaiset palvelut. Palveluihin kuuluu MQTT-protokollan mukainen välittäjäpalvelu, tilaajapalvelu ja tietokantapalvelu. Klusteriin lisättiin työn kehityksen aikana myös palvelut, jotka esittävät ajettavien palveluiden tilaa visuaalisesti ja anturidataa visuaalisesti.

Opinnäytetyön tuloksena Kajaanin Ammattikorkeakoulu saa käyttöönsä toimivan esimerkkiratkaisun siitä, kuinka Docker swarm-pohjainen klusteri voidaan toteuttaa. Opinnäytetyö toimii myös perusteellisena dokumentaationa siitä, kuinka kyseinen klusteri kehitettiin. Dokumentaatio voi olla hyödyllinen projektin jatkokehitystä ja ylläpitoa varten.

Abstract

Author(s): Niilo Niskanen

Title of the Publication: IoT Sensor Network For Network Teaching Environment

Degree Title: Bachelor of Engineering, Information and communication technologies

Keywords: internet of things, sensor-node, Docker, Docker swarm, cluster, container, networks

The goal of this Bachelor's thesis was to develop an example solution of an IoT-sensor-network for use in education by the client of the thesis, Kajaani University of Applied Sciences. The thesis is part of a larger network teaching environment project, conducted by the client of the thesis. The example solution was implemented according to the requirements defined by the client. The practical part of the thesis was divided into two parts, the sensor-node and the background system.

The function of the sensor node is to collect and transmit sensor data at regular intervals wirelessly using the MQTT protocol. The chosen values to collect were temperature, humidity and air pressure. Suitable sensors were selected for use. The code for the sensor node was developed using the Arduino IDE software. The sensor node was not the main area of development, so it was decided that it would be developed as simply as possible.

The background system, which is referred to as a cluster, was implemented after the sensor node. The cluster consists of four Raspberry pi 3B microcomputers connected to the same network. The raspberry Pi's operating systems were booted from USB sticks instead of SD cards to extend their life. One raspberry Pi microcomputer was installed with Ubuntu Server 18.04.5 LTS operating system, the others were installed with the Raspberry Pi OS operating system. The cluster was initialized with the Docker swarm tool built into the Docker software. The Raspberry Pi microcomputers were connected to the Docker swarm cluster, after which services were developed for the cluster according to the client's requirements. The services include a broker service according to the MQTT protocol, a subscriber service and a database service. During the development of the cluster, other services were also added to the cluster that show the status of the services to be run and the sensor data visually.

As a result of the thesis, the client will have a working example of how a Docker swarm-based cluster can be implemented. The thesis also serves as a thorough documentation of how the cluster was developed. The documentation can be used to further develop and maintain the cluster.

Sisällys

Symboliluettelo

Broadcast-liikenne	Lähetettävä datavirta ennalta määräämättömälle vastaanottajamäärälle.
Debian	Suosittu Linux-käyttöjärjestelmä jakelupaketti.
Docker	Työkalu, joka helpottaa sovelluksien ajamista konteissa.
I2C	Inter-Integrated Circuit Bus. Tietoliikenneprotokolla, jota käytetään piirilevyn sisäisessä kommunikaatiossa.
IoT	Internet of Things. Esineiden internet. Kuvaa älykkäiden laitteiden muodostamaa verkkoa.
Kaistanleveys	Yhtenäisen taajuusalueen ylemmän ja alemman rajataajuuden erotus. Tietoverkkojen tapauksessa tiedonsiirtonopeus bit/s.
Kontti	Tapa virtualisoida käyttöjärjestelmä.
LAN	Local Area Network. Rajoitetulla alueella toimiva tietoliikenneverkko.
Latenssi	Paketin edestakaiseen matkaan kuluva aika.
MAC	Media Access Control. Verkkolaitteen fyysinen osoite.
MQTT	Message Queuing Telemetry Transport. Tietoliikenneprotokolla, jota käytetään IoT-tyylisessä viestinnässä.
OSI	Open Systems Interconnection. Tarjoaa ohjeistuksia siitä, kuinka tietoteknisten laitteiden tulisi kommunikoida tietoverkoissa.
OTP-muisti	One Time Programmable Memory. Muistin tyyppi, jonka sisällön voi ylikirjoittaa.
Paketti	Pieni määrä dataa, joka siirretään verkossa, sisältää alkuperän, määränpään ja sisällön. Termiä käytetään OSI-mallin verkkokerroksessa.
Portti	16-bittinen numero, jolla tunnistetaan sovelluksia ja palveluita.
Protokolla	Ryhmä sääntöjä, jotka mahdollistavat sähköisten laitteiden välisen kommunikoinnin.

Segmentti	Pieni määrä dataa, termiä käytetään OSI-mallin kuljetuskerroksessa.
Standardi	Määrittelee tietoliikenteelle säännöt, joita tarvitaan teknologioiden yhteensopivuuteen.
TCP	Tietoliikenneprotokolla, joka vaatii yhteyden muodostamisen toimilaitteiden välillä.
UDP	Tietoliikenneprotokolla, joka ei vaadi yhteyden muodostamista toimilaitteiden välillä.
WAN	Wide Area Network. Suurella alueella toimiva ulkoverkko.
WLAN	Wireless Local Area Network. Langaton, paikallinen tietoliikenneverkko.
Ydin	Kernel. Käyttöjärjestelmän ydin, linkki laitteiston ja ohjelmiston välillä.

1 Johdanto

Kevään 2020 aikana Suomeen rantautuneen covid-19-pandemian johdosta Kajaanin ammattikorkeakoulun tuli siirtää kaikki opiskelu etämuotoon. Tämä äkkinäinen muutos toi mukanaan ongelman, kuinka parhaiten toteuttaa etäopetusta insinööriopiskelijoille. Esimerkiksi Kajaanin ammattikorkeakoulun Älykkäät järjestelmät -suuntautumisvaihtoehdon opinnot painottuvat IoT-järjestelmien ohella langattoman tietoliikenteen ja tietoverkkojen ymmärtämiseen.

Opinnäytetyön tilaajana toimii Kajaanin Ammattikorkeakoulu Oy (KAMK). Tilaajan idea on toteuttaa tietoverkkojen opetusympäristö tieto- ja viestintätekniikan insinööriopetukseen. Tietoverkkojen opetusympäristö tarjoaa insinööriopiskelijoille mahdollisuuden opiskella tietoverkkoihin liittyviä käsitteitä ja konsepteja täysin etäopiskelun muodossa. Tieto- ja viestintätekniikan opettajat tarvitsevat tietoverkkojen opetusympäristöä myös siksi, että heidän ei aina tarvitse rakentaa uusia työkaluja uusille kursseille. Opetusympäristöä tullaan siis käyttämään jatkuvassa opetuksessa, koululla paikan päällä ja etänä. Opetusympäristön avulla voi kenelle tahansa opettaa tietoverkkojen perusteita, mutta opetusympäristö on pääsijaisesti suunniteltu insinööriopiskelijoiden käytettäväksi. Se, kuinka opetusympäristöä voidaan käyttää opetukseen, jää itse opettajien päätettäväksi. Tämä opinnäytetyö keskittyy opetusympäristön IoT-kokonaisuuden suunnitteluun ja toteutukseen.

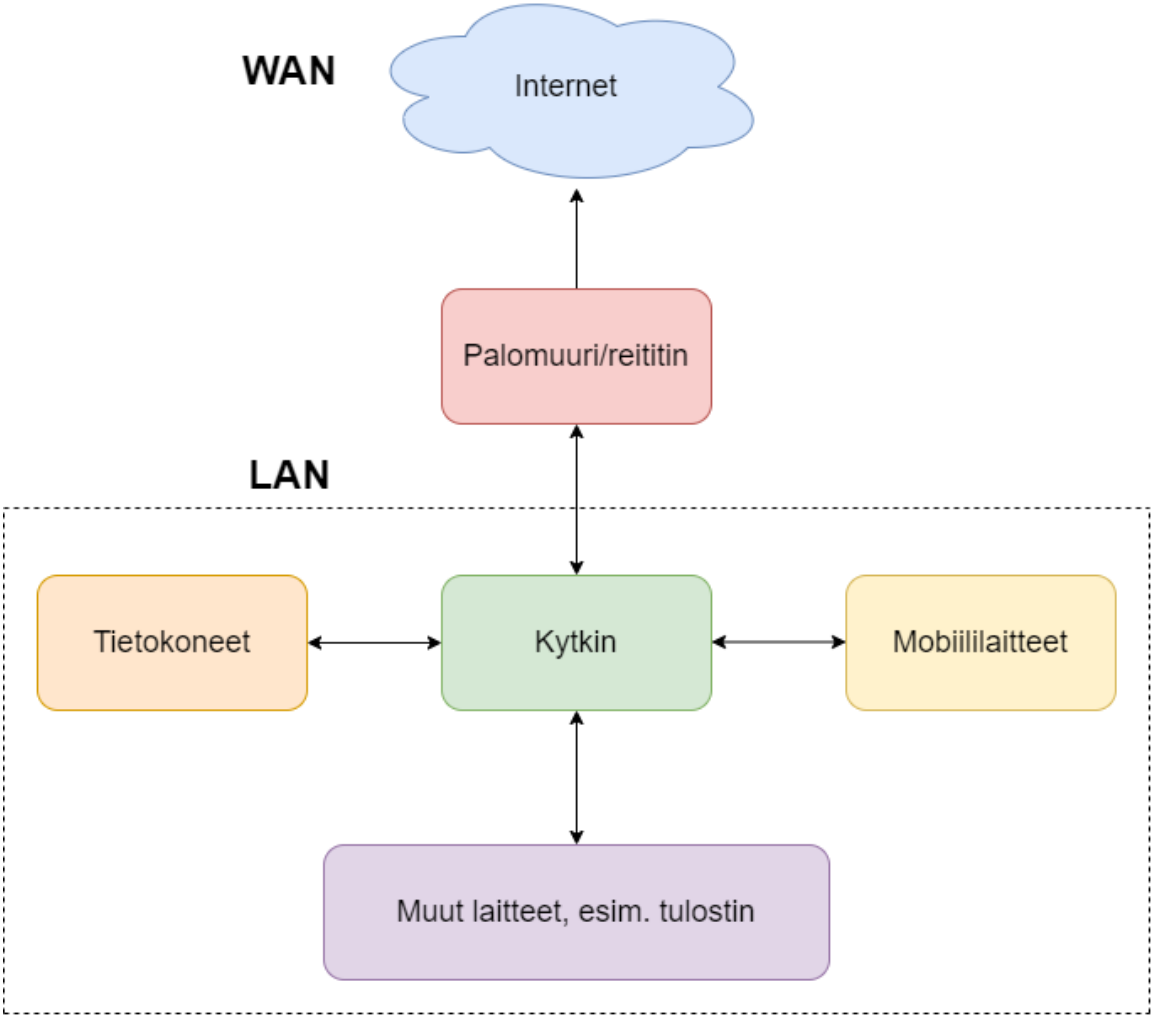
Opinnäytetyö on kirjoitettu niin, että alan ammattilaiset ja opiskelijat saavat dokumenttia lukemalla perusteellisen ymmärryksen siitä, kuinka tietoverkkojen opetusympäristö on suunniteltu ja toteutettu. Opinnäytetyötä voi tällöin käyttää mahdollisen jatkokehityksen ja ylläpidon perusteena.

Opinnäytetyölle asetettiin seuraavia vaatimuksia. Tietoverkkojen opetusympäristöä varten tulee toteuttaa kaksi identtistä klusteria, jossa voi ajaa Docker-ohjelmiston avulla kontteja. Opetusympäristöä varten tulee myös toteuttaa antureilta dataa keräävä laite, joka lähettää keräämänsä datan langattomasti säännöllisin väliajoin talteen tietokantaan. Jokaiselle klusterille toteutetaan oma anturisolmu. Opetusympäristössä tulee olla tietokanta, johon tallennetaan anturilaitteelta kerätty data. Kaikille opetusympäristön osa-alueille tulee luoda kattava dokumentaatio, josta selviää kaikki tarvittava tieto ylläpitoa ja mahdollista jatkokehitystä varten.

Tietoverkkojen opetusympäristö toteutetaan opinnäytetyön käytännön osuutena kehitystyönä. Tilaaja on määrittänyt määräajan niin, että kehitystyön tulee olla valmis kesäksi 2021.

2 Tietoverkot

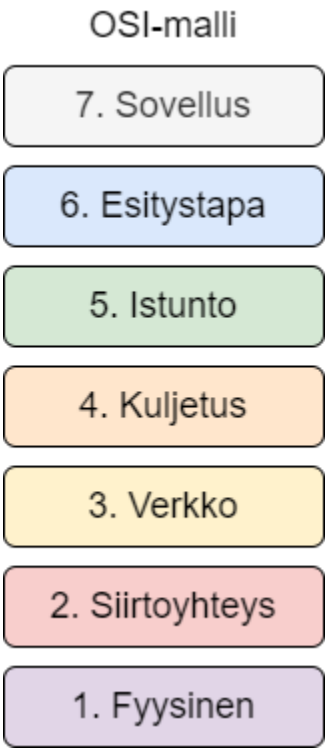
Tietoverkkoja on kaikkialla ympäröivässä maailmassa. Eräs tunnetuimmista tietoverkoista on internet, jossa toisiinsa kytketyt tietoverkot kommunikoivat hyödyntäen protokollaa eli yhteyskäytäntöä. Internetin toiminta perustuu TCP/IP-viitemalliin, jonka kahdesta tärkeimmästä protokollasta TCP ja IP nimi on johdettu. Kuvassa 1 esitetään tietoverkkoja periaatetasolla.



Kuva 1. Tietoverkkojen periaate.

2.1 OSI-malli

OSI (engl. Open Systems Interconnection) malli tarjoaa ohjeistuksia siitä, kuinka tietoteknisten laitteiden tulisi kommunikoida tietoverkon yli. OSI-malli ei tarjoa yksityiskohtaisia menettelytapoja tai protokollia kommunikoinnin toteutukseen. Malli on luonteeltaan teoreettinen. [1.] Kuvassa 2 on esitetty OSI-mallin eri kerrokset.



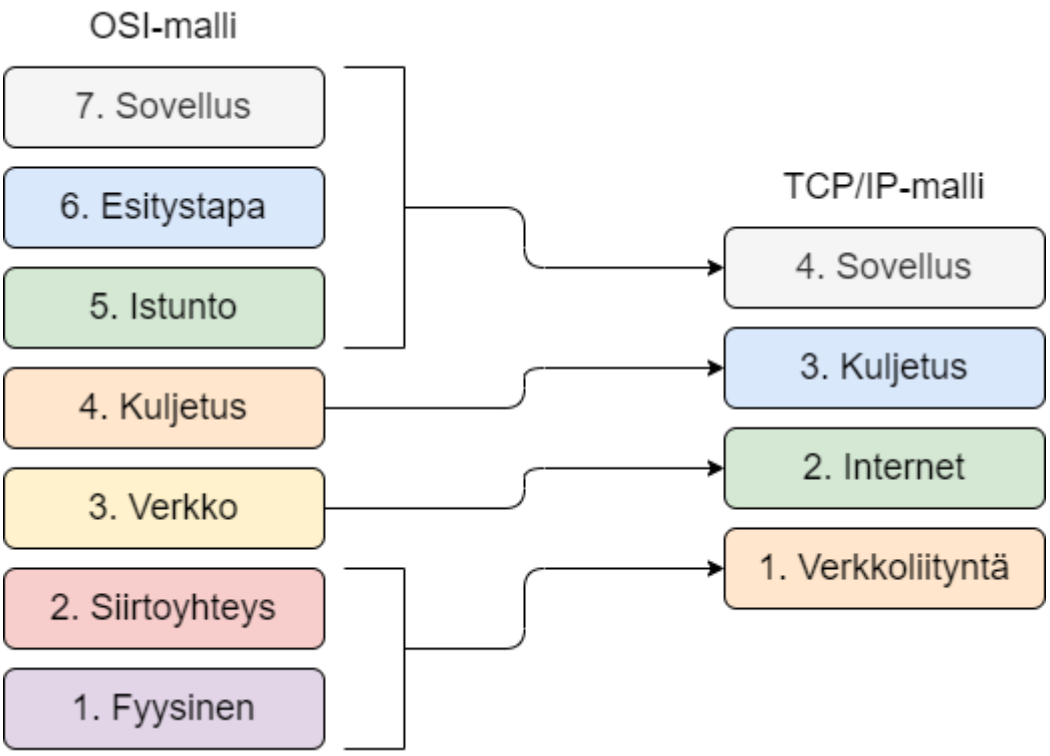
Kuva 2. OSI-mallin kerrokset.

Fyysisessä kerroksessa (engl. physical layer) määritetään, kuinka raaka bittivirta siirretään vastaanottajalle. Bittivirran siirto tapahtuu sähköisenä signaalina, sähkömagneettisena säteilynä tai valopulssina. Fyysisessä kerroksessa määritetään siis fyysisiä komponentteja, kuten kaapelointia tai radiotaajuuksia. Siirtoyhteyskerros vastaa samassa verkossa olevien toimilaitteiden välisestä tietoliikenteestä. Toimilaitteet tunnistetaan MAC-osoitteiden (engl. Media Access Control) avulla. MAC-osoitteet ovat yksilöllisiä verkon toimilaitteelle, kuten verkkosovittimille ja reitittimen porteille. Siirtoa varten tietopaketit paketoidaan kehyksiin (engl. frame). Siirtoyhteyskerros vastaa myös mahdollisten virheiden hallinnasta, jotka voivat tapahtua verkkojen välisessä kommunikoinnissa. Verkkokerros (engl. network layer) on vastuullinen tietopakettien lähetyksestä eri verkkojen välillä. Verkkokerros paketoi kuljetuskerrokselta saamansa segmentit tietopaketeiksi lähettäjällä, kerros purkaa tietopaketit segmenteiksi vastaanottajalla. Verkkokerros on myös vastuussa

parhaan reitin löytämisestä vastaanottajalle, eli reitityksestä. Kuljetuskerros (engl. transport layer) ottaa vastaan dataa istuntokerrokselta ja jakaa sen segmentteihin. Kuljetuskerros vastaanottajan päässä on vastuussa segmenttien muutoksesta takaisin dataksi. Kerros on myös vastuussa tiedonvirran nopeuden säädöstä ja virheiden hallinnasta. Kuljetuskerroksen yhteydellinen TCP-protokolla huolehtii, että tietopaketit saapuvat perille oikeassa järjestyksessä. Jos protokollana käytetään UDP-protokollaa, yhteyttä ei muodosteta ollenkaan. Istuntokerroksessa (engl. session layer) avataan ja suljetaan istunto kahden laitteen välillä. Aikaa avaamisen ja sulkemisen välillä kutsutaan istunnoksi. Kerros varmistaa, että istunto pysyy käynnissä niin kauan, että kaikki haluttu tieto saadaan siirrettyä. Siirron jälkeen istunto suljetaan. Istuntokerros on myös vastuussa tiedonsiirron synkronoinnista käyttämällä välipisteitä (engl. checkpoint). Esitystapakerroksessa (engl. presentation layer) saatu tietopaketti muutetaan esitettäväksi sovelluksille. Esitystapakerros on vastuussa tietopaketin käännöksestä, salauksesta ja tietopaketin pakkaamisesta. Sovelluskerros (engl. application layer) on ainut kerros, joka näkyy käyttäjälle. Sovelluskerroksessa sijaitsee protokollat ja tiedon manipulaatio, joita sovellukset käyttävät viestintään. [1.]

2.2 TCP/IP-protokollapino

TCP/IP on kokoelma protokollia, jotka muodostavat internetin ja muiden verkkojen ”selkärangan”. Pinoon kuuluu noin 500 eri tietoliikenneprotokollaa. Protokollapino on nimetty kahden tärkeimmän internetkommunikaatioprotokollan mukaan, TCP ja IP. [2.] TCP/IP termiä voi kuvailla myös mallina, kuten OSI-mallia. TCP/IP-mallissa on neljä kerrosta, toisin kuin OSI-mallissa, jossa on seitsemän kerrosta. TCP/IP-malli on ollut kauemmin käytössä kuin OSI-malli, kumpaakin mallia käytetään nykypäivänä. [2.] Kuvassa 3 verrataan OSI-mallin kerroksia TCP/IP-mallin kerroksiin.



Kuva 3. OSI-mallin kerrokset verrattuna TCP/IP-mallin kerroksiin.

TCP/IP-mallin verkkoliityntäkerros (engl. network access layer) sisältää OSI-mallin siirtoyhteyskerroksen ja fyysisen kerroksen. Verkko-liityntäkerroksen käytetyimmät protokollat ovat Ethernet (langallinen) ja IEEE 802.11 (langaton). Kerroksen tavoite on määrittää, kuinka eri verkot voidaan liittää toisiinsa, kuten kuinka kodin verkon voi liittää internetiin modeemin kautta. [2.] Internetkerroksen merkittävimpiä protokollia on IP-protokolla, jonka avulla varmistetaan, että lähetetyt tietopaketit saapuvat oikeaan määränpäähän. Tietopaketit reititetään IP-osoitteiden avulla. [2.] Kuljetuskerros huolehtii pakettien lähetyksestä ja vastaanotosta kahden verkossa olevan toimilaitteen välillä. Kuljetuskerroksen kaksi merkittävintä protokollaa ovat yhteydellinen TCP ja yhteydetön UDP. Käytettäessä TCP-protokollaa huolehtii se kuittauksin pakettien perillemenosta, uudelleen lähettämisestä sekä järjestämisestä oikeaan järjestykseen. [2.] Sovelluskerros (engl. application layer) hallitsee sovelluksien välistä kommunikointia verkossa. Protokollat, kuten HTTP, HTTPS ja SMTP, toimivat tässä kerroksessa [2].

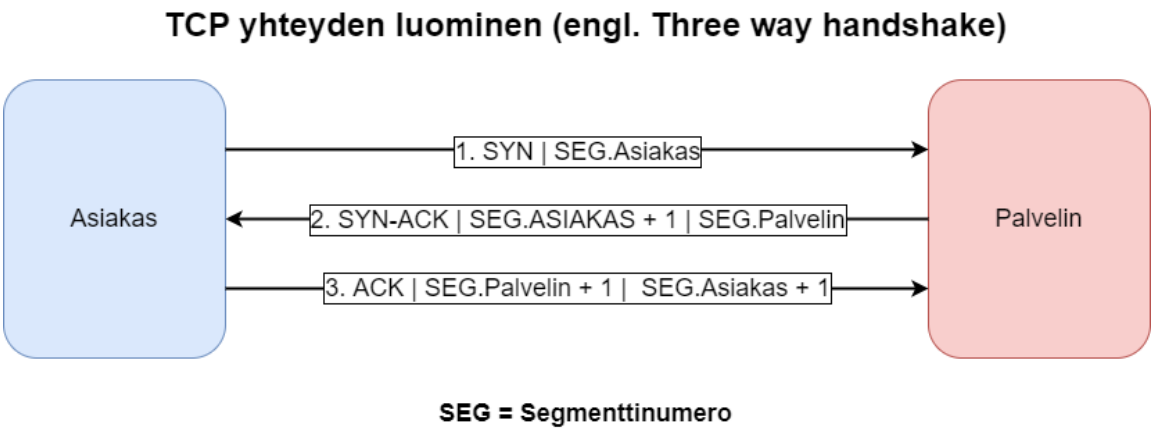
TCP/IP-mallin protokollapinon merkittävä etu on, että ne toimivat erillään laitteistosta ja sovelluksista. Protokollat on standardisoitu niin, että millä tahansa käyttöjärjestelmällä tai toimilaitteella voi päästä verkkoon. [2.]

2.2.1 TCP-protokolla

Kuljetuskerroksen TCP (engl. Transmission Control Protocol) protokolla on yksi TCP/IP-protokollapinon tärkeimpiä protokollia, Protokolla tarjoaa standardisoidun tavan laitteiden välillä tapahtuvaan tietoliikennekommunikaatioon [3]. TCP-protokolla sallii samanaikaisen kaksisuuntaisen tietoliikenteen, eli toimilaite voi vastaanottaa tietopaketteja samaan aikaan, kun se lähettää toiselle laitteelle omia tietopakettejaan [3].

Ennen kuin yhteys voidaan luoda toimilaitteiden välillä, tulee kummallakin olla IP-osoite ja haluttu portti avattuna. IP-osoite toimii toimilaitteiden tunnisteena, kun taas portti toimii käytettävän sovelluksen tunnisteena. Palomuurin säännöissä tulee sallia haluttu sovellusportti, jotta kommunikointi kahden laitteen sovelluksen välillä on mahdollista. Yhteydessä laitteet tunnetaan asiakkaana ja palvelimena, mutta sillä ei ole väliä, kumpi laite aloittaa yhteyden luomisen. [3.] Luvussa oletetaan, että asiakas aloittaa yhteyden luomisen ja sulun.

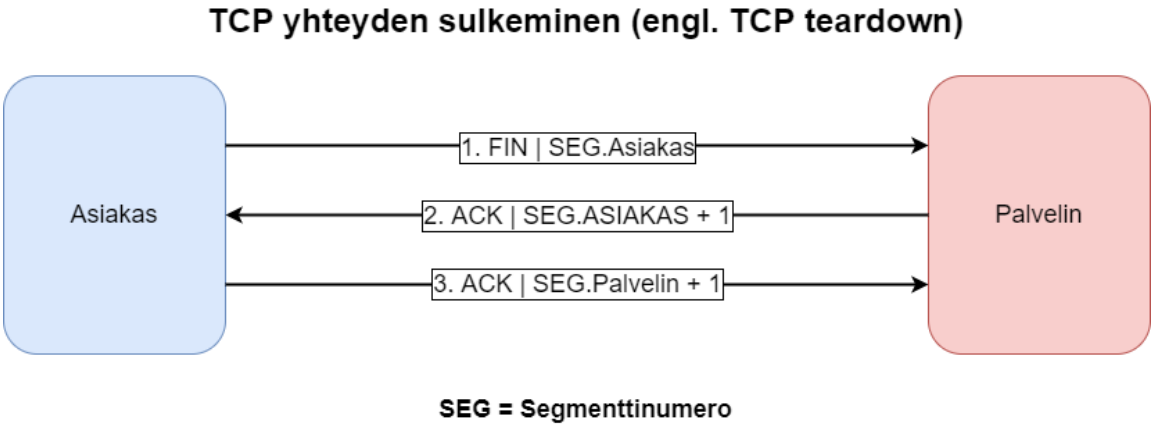
Kun yhteyden luominen alkaa, ensimmäisenä asiakas lähettää palvelimelle SYN (engl. Synchronize)-segmentin. Paketti sisältää segmenttinumeron, tämä numero varmistaa, että kaikki tietopaketit lähetetään oikeassa järjestyksessä. Jos palvelin vastaanottaa SYN-paketin, se hyväksyy yhteyden lähettämällä SYN-ACK (engl. Synchronize-Acknowledge) -segmentin takaisin asiakkaalle ja plussaa asiakkaan segmenttinumeron arvoa yhdellä. Palvelin lähettää myös oman numeronsa asiakkaalle. Asiakaslaite hyväksyy SYN-ACK -paketin lähettämällä palvelimelle ACK-paketin, joka sisältää palvelimen segmenttinumeron plussattuna yhdellä. Asiakas voi samaan aikaan aloittaa tietopakettien siirron palvelimelle. Jos palvelin ei ole saatavilla, asiakas vastaanottaa TCP RTS (engl. TCP Reset) -paketin. [3.] Kuva 4 esittää asiakkaan aloittaman yhteyden luomisen.



Kuva 4. TCP-protokollan mukainen yhteyden luominen.

Kumpikin osapuoli voi aloittaa yhteyden sulkemisen, yksipuolinen sulku on myös mahdollista. Yksipuolinen sulku tunnetaan puoliaukeana (engl. half-open) yhteytenä, jossa toinen osapuoli voi lähettää tietopaketteja, vaikka yhteys on jo suljettu. [3.]

Yhteyden sulkeminen alkaa, kun asiakas lähettää FIN-paketin, joka kertoo palvelimelle, että se voi lopettaa tietopakettien siirron. Asiakas lähettää oman segmenttinumeronsa. Palvelin lähettää ACK segmentin takaisin, joka sisältää asiakas-segmenttinumeron plussattuna yhdellä. Kun palvelimen tietopakettien siirto on valmis, se lähettää FIN-paketin ja oman segmenttinumeronsa. Asiakas vastaanottaa segmentit ja lähettää ACK-segmentin palvelimelle ja palvelimen segmenttinumeron plussattuna yhdellä. Paketin saapuessa yhteys suljetaan. [3.] Kuva 5 esittää asiakkaan aloittaman yhteyden sulkemisen.



Kuva 5. TCP-yhteyden sulkeminen.

2.2.2 IP-osoitteet

Jokainen laite, joka on liitetty verkkoon, tulee pystyä tunnistamaan toisista laitteista luotettavasti. TCP/IP-verkkojen maailmassa käytetään tunnisteena sekä MAC- että loogisia IP-osoitteita (engl. Internet Protocol). IP-osoitteet voivat esiintyä kahdessa eri formaatissa, IPV4 tai IPV6. [4.]

IPV4-standardin mukainen IP-osoite on neljä numeroa erotettuna desimaalipisteellä. Yksittäinen numero voi olla väliltä 0–255. Numero voi olla maksimissaan 255, koska toimilaitteet eivät käytä numeron desimaaliarvoa, toimilaitteet käyttävät numeron kahdeksanbittistä binaariarvoa. Numero esitetään desimaalissa luettavuuden helpottamiseksi. [4.] IPV4-standardin mukainen IP-osoite on 32-bittinen, eli verkossa voi esiintyä maksimissaan 2^{32} eli noin 4,3 miljardia uniikkia IP-osoitetta. Tämä määrä osoittautui ongelmaksi, kun internetiin liittyvien laitteiden määrä alkoi kasvaa eksponentiaalisesti. Ongelma ratkaistiin kehittämällä uusi IP-osoitteiden standardi, IPV6. [4, 5.]

IPV6-standardin IP-osoitteet koostuvat kahdeksasta heksadesimaalina esitetystä numerosta, jotka ovat eritettynä toisistaan kaksoispisteellä. IPV6-standardin IP-osoitteet ovat 128-bittisiä, verkossa voi siis esiintyä 2^{128} eli noin 3,4028237e+38 uniikkia IP-osoitetta. Vaikka uusi standardi luotiin, nykypäivänä IPV4-osoitteet ovat vielä laajassa käytössä, koska yksityiset verkot voivat käyttää omia rajattuja IP-osoitteitaan. IPV6-osoitteiden suosio kasvaa kuitenkin koko ajan. [4, 5.]

IP-osoitteet voivat olla staattisia tai dynaamisia. Staattinen IP-osoite voidaan antaa toimilaitteelle, jos halutaan, että sen IP-osoite ei muutu aina, kun se liittyy uudelleen verkkoon. IP-osoitteet jaetaan automaattisesti dynaamisesti, jos IP-osoitetta ei manuaalisesti aseteta staattiseksi. IPV4-IP-osoitteiden dynaamisesta jaosta vastaa DHCP (engl. Dynamic Host Control Protocol) -palvelin. IPV6-standardi tukee IP-osoitteiden itsenäistä jakoa. [4, 5.]

IPV4-standardin IP-osoitteet voidaan myös jakaa luokkiin, Luokan A verkossa ensimmäinen numero ilmaisee verkon, loput numerot ilmaisevat laitteen. Esim. 203.0.113.112 osoitteessa 203 esittää verkon, 0.113.112 ilmaisee laitteen osoitteen. Luokan B verkossa kaksi ensimmäistä numeroa ilmaisevat verkon ja kaksi muuta numeroa ilmaisevat laitteen. Samalla esimerkillä verkko olisi 203.0 ja laite olisi 113.112. Luokan C verkossa kolme ensimmäistä numeroa ilmaisee verkon, viimeinen numero ilmaisee laitteen. Esim. verkko on 203.0.113 ja laite olisi 112. On myös olemassa D ja E luokkien verkot, mutta ne eivät ole laajassa käytössä. [5.]

2.3 Aliverkot & aliverkkojen peitteet

Aliverkko on verkko suuremman verkon sisässä. Aliverkot tekevät verkoista tehokkaampia, koska tietopakettien ei tarvitse kulkea koko suuremman verkon läpi. Tietopaketit voivat siis kulkea mahdollisimman suoraa reittiä määränpäähänsä. Aliverkot ovat tarpeellisia, koska jos verkossa on teoreettisesti miljoonia laitteita, tietoliikenteestä tulisi hidasta Broadcast-liikenteen takia. Aliverkot rajaavat IP-osoitteiden määrää tiettyihin rajoihin, jotta laitteiden määrästä ei tulisi liian suuri. Aliverkot auttavat myös helpottamaan verkkojen organisointia. Koska IP-osoite voi kertoa vain verkon ja laitteen osoitteen, IP-osoitteella ei voi ilmaista, mihin aliverkkoon liitetty laite kuuluu. Verkon reitittimet käyttävät aliverkon peitteitä aliverkkojen erittelyyn. [6.]

Aliverkon peite on maski, joka määrittelee, mikä osa IP-osoitteesta on verkko-osaa ja mikä laitekohtaista numeroa, jotka ovat saatavilla verkossa. Aliverkossa laitteet voivat kommunikoida keskenään MAC-osoitteiden avulla, mutta eri aliverkkojen väliseen kommunikointiin pitää tietoliikenteen kulkea reitittimen läpi. [6.]

2.4 Palomuurit

Palomuuri on tietotekniikan turvallisuusjärjestelmä, jonka tehtävänä on estää asiattomien tietopakettien liikenteen verkosta toiseen, kuten internetistä lähiverkkoon. Palomuurin tehtävä on siis suodattaa tietoliikenteestä paketit, jotka eivät ole haluttuja. Palomuuri luo yksityisen verkon ja internetin väliin ”suodattimen”, jonka läpi kaikkien pakettien on pakko kulkea. Palomuurin voi asentaa isäntälaitteelle tai itse verkon reitittimeen. Palomuuri voi olla ohjelmisto tai laite. [7.]

Palomuurit voidaan luokitella viiteen eri tyyppiin. Paketteja suodattava palomuuri (engl. packet filtering firewall) vertaa vastaanotettuja paketteja ennalta luotuihin kriteereihin, kuten sallittuihin IP-osoitteisiin, tietopakettien tyyppiin tai porttiin. Suodatetut paketit poistetaan välittömästi. Paketteja suodattavat palomuurit ovat käytössä verkon reitittimessä, eli ne voivat vahtia koko verkon tietoliikennettä. [8.]

Piiritason yhdyskäytävä (engl. circuit-level gateway) on nopea tapa suodattaa paketteja. Piiritason yhdyskäytävät vahtivat verkon istuntojen aloittamisviestejä, kuten TCP-protokollan yhteyksien luomista. Kyseiset palomuurit eivät tarkasta pakettien sisältöä, joten niitä käytetään yleensä muiden turvajärjestelmien yhteydessä. [8.]

Sovellustason yhdyskäytävä (engl. application-level gateway), joka tunnetaan myös välityspalvelin palomuurina (engl. proxy firewall), toimii yksityisen lähiverkon ainoana saapumis- ja poistumispisteenä. Sovellustason yhdyskäytävät suodattavat paketteja määritellyn palvelun portin mukaan ja muiden ominaisuuksien, kuten HTTP-kyselymerkkijonon mukaan. Kyseiset palomuurit tarjoavat hyvää turvallisuutta, mutta hidastavat tietoliikennettä ja ovat yleisesti hankalia ylläpitää. [8.]

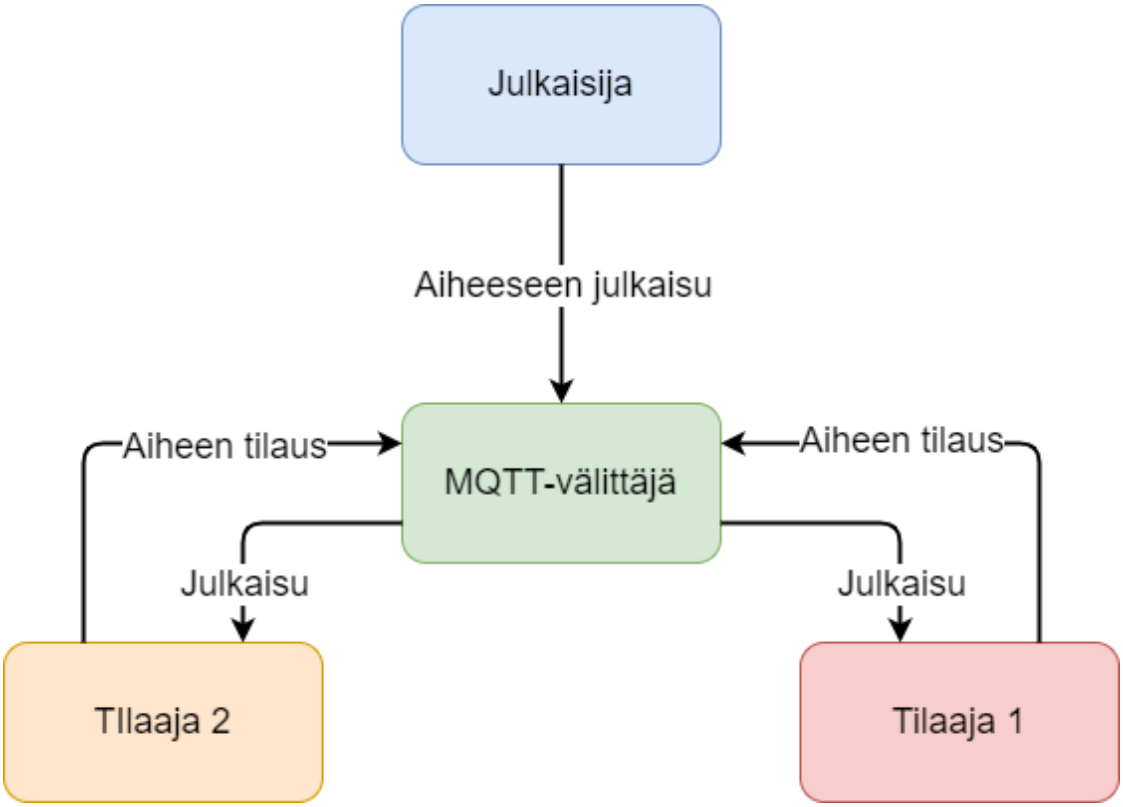
Tilallinen tarkastuspalomuuri (engl. stateful inspection firewall) tarkastaa tietopakettien sisällön lisäksi myös sen, että kuuluvatko tietopaketit hyväksyttyyn TCP-yhteyteen. Tämä tarjoaa parempaa turvallisuutta kuin pakettien suodatus itsestään, mutta hidastaa verkon tietoliikennettä huomattavasti. [8.]

Seuraavan sukupolven palomuuri (engl. next-generation firewall) yhdistää pakettien tarkastuksen ja tilallisen tarkistuksen ja muita verkon turvajärjestelmiä. Seuraavan sukupolven palomuurit voivat tarkistaa tietopakettien sisäisen dataa haittaohjelmien varalta. Ne ovat kalliita asentaa ja ylläpitää. Seuraavan sukupolven palomuurit pystyvät suodattamaan pois edistyneitä haittaohjelmia. [8.]

2.5 MQTT-protokolla

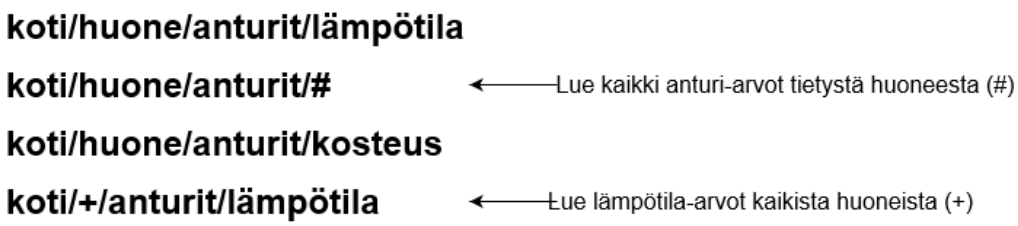
MQTT-protokolla (engl. Message Querying Telemetry Transport) on verkkotietoliikenneprotokolla, jota käytetään laajasti IoT-tyylisessä viestinnässä. MQTT-protokolla on suunniteltu toimimaan epäluotettavissa verkoissa, joissa alhainen kaistanleveys ja suuri latenssi estävät laajan tietoliikenteen toteutuksen. [9.] Protokollaan kuuluu monta spesifikaatiota, MQTT 3.1, MQTT 3.1.1, MQTT 5 ja MQTT-SN. MQTT-SN on optimoitu spesifikaatio vähävirtaisiin ympäristöihin, kuten anturiverkkoihin [10].

MQTT-protokollan mukaisessa viestinnässä toimilaitteet eivät viesti suoraan keskenään. MQTT-protokollan viestinnässä viestit lähetetään välittäjälle (engl. broker). Välittäjä on ohjelmisto, joka vastaanottaa yhden toimilaitteen lähettämät viestit ja lähettää ne toiselle toimilaitteelle, joka niitä on pyytänyt. Välittäjän kanssa viestinnässä käytetään julkaisu/tilaus (engl. publish/subscribe) menetelmää. Välittäjä ei lue viestien sisältöä. [9, 10.] Lähettäjä julkaisee viestinsä tiettyyn aiheeseen (engl. topic), jonka vastaanottaja tilaa. Kun lähettäjä julkaisee viestin aiheeseen, välittäjä vastaanottaa viestin ja lähettää sen vastaanottajalle, joka on tilannut kyseisen aiheen. [9, 10.] Kuva 6 esittää MQTT-viestinnän periaatetasolla.



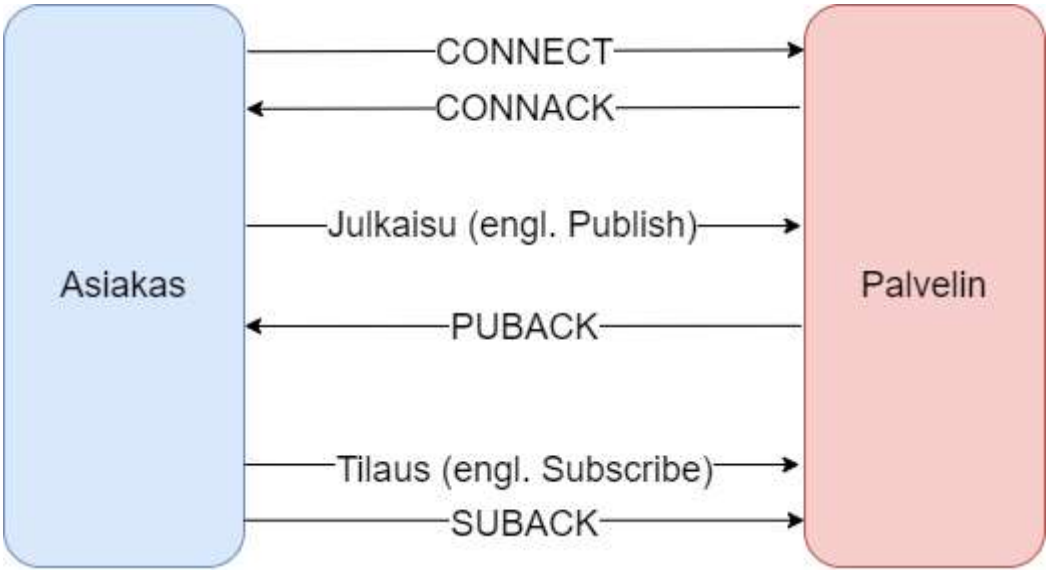
Kuva 6. MQTT viestintä periaatetasolla.

MQTT-protokollan viestinnässä käytetään aiheita viestien lajitteluun. Aihe on merkkijono, jossa aiheen kerrokset erotetaan vinoviivalla. Aiheessa pitää olla vähintään yksi kerros. Aiheita ei tarvitse erikseen luoda, välittäjä välittää viestit annettuun aiheeseen tarkistamatta, onko aihetta käytetty ennen. MQTT-laite voi tilata yhden tai useamman aiheen yhtä aikaa käyttämällä ”jokerimerkkejä” (engl. wildcard). Yhden kerroksen aiheesta voi korvata ”+” -merkillä, monta kerrosta voi korvata ”#” -merkillä. Kuvassa 7 esitetään esimerkkejä MQTT-aiheista. [9, 10.]



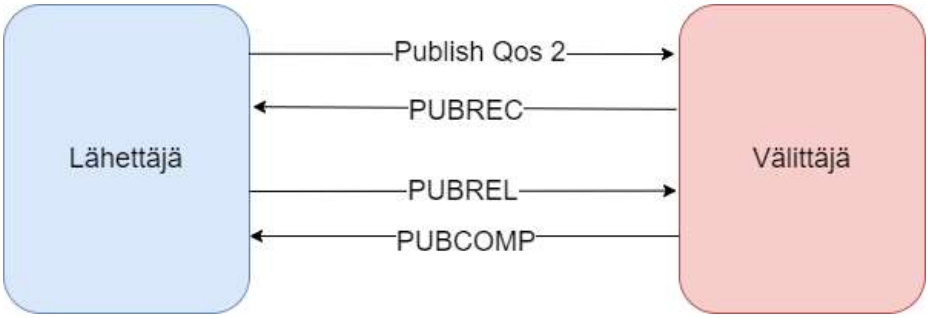
Kuva 7. Esimerkkejä MQTT-aiheista.

Asiakaslaitteen yhteys välittäjään luodaan lähettämällä ”CONNECT”-viesti välittäjälle, joka vastaa viestiin ”CONNACK” (engl. Connection Acknowledged) viestillä. Yhteys tarkistetaan ennen jokaista tiedonsiirtoa viestien lähetyksen varmistamiseksi. [9, 10.] Kuvassa 8 esitetään MQTT-yhteyden tarkistaminen ennen julkaisua tai tilausta.



Kuva 8. MQTT-yhteyden tarkistaminen.

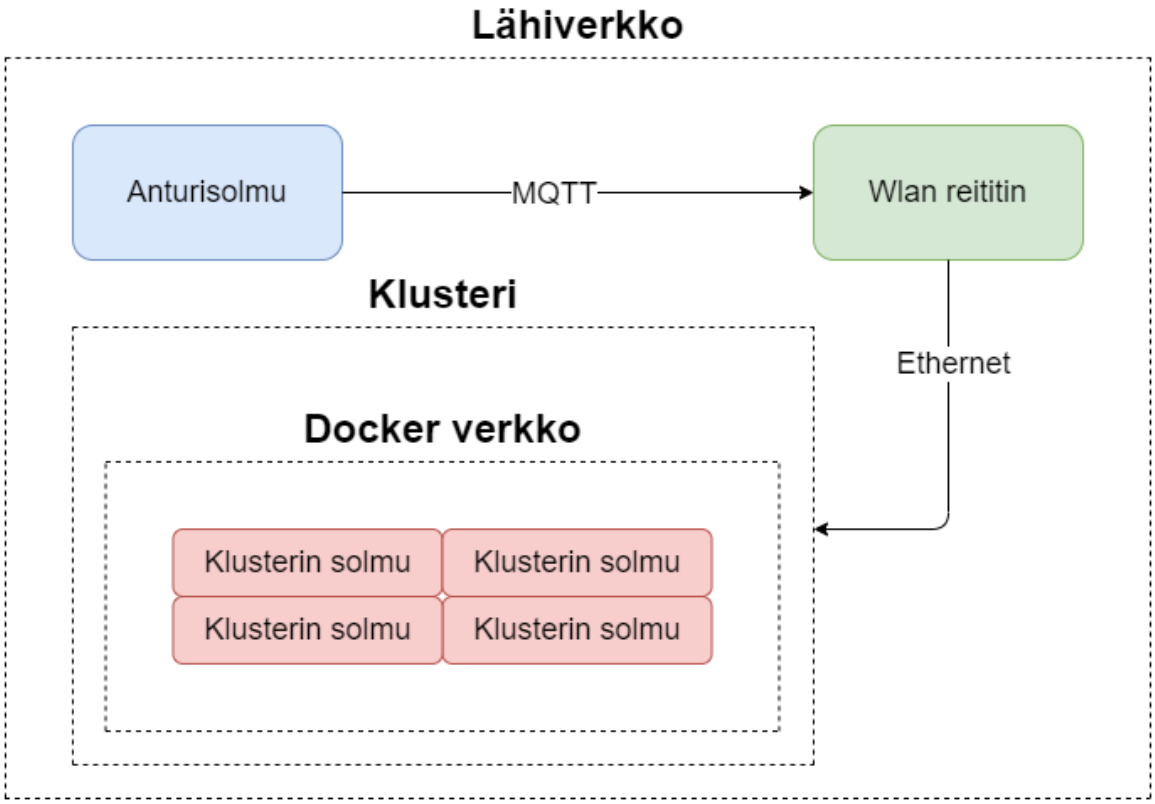
MQTT-protokolla varmistaa, että viesti saapuu perille käyttämällä palvelun laatu (engl. Qos, quality of service) -asetusta. Viestinnän laatu riippuu asetuksen tasosta. Taso ilmaistaan kokonaislukuna. Taso -1 on ideaalinen vähävirtaisille laitteille, jotka eivät välitä, saapuuko viesti perille. Viestin saapumista välittäjälle ei siis varmisteta. Edellä mainittu ominaisuus on satavilla vain MQTT-SN -spesifikaatiossa. [10.] Tason 0 viestin saapuminen varmistetaan, että se saapuu perille vain kerran. Viestin saapumisesta ei kerrota sen lähettäjälle, eikä viestiä yritetä lähettää uudelleen. Tason 0 viestejä ei laiteta jonoon odottamaan tilausta välittäjälle. Tason 1 viesti varmistetaan saapuneen perille ainakin kerran. Lähettäjä saa varmistuksen viestin vastaanotosta, viestit laitetaan jonoon odottamaan tilaajalle lähetystä. Viesti voidaan lähettää monta kertaa. Tason 2 viestin pitää saapua vain kerran ja ainoastaan kerran. Viestin lähetys varmistetaan neljäosaisella kättelyllä (engl. four-way-handshake). [10.] Kuvassa 9 esitetään tason 2 viestin lähetys.



Kuva 9. Tason 2 viestin lähettäminen.

3 IoT-anturiverkko

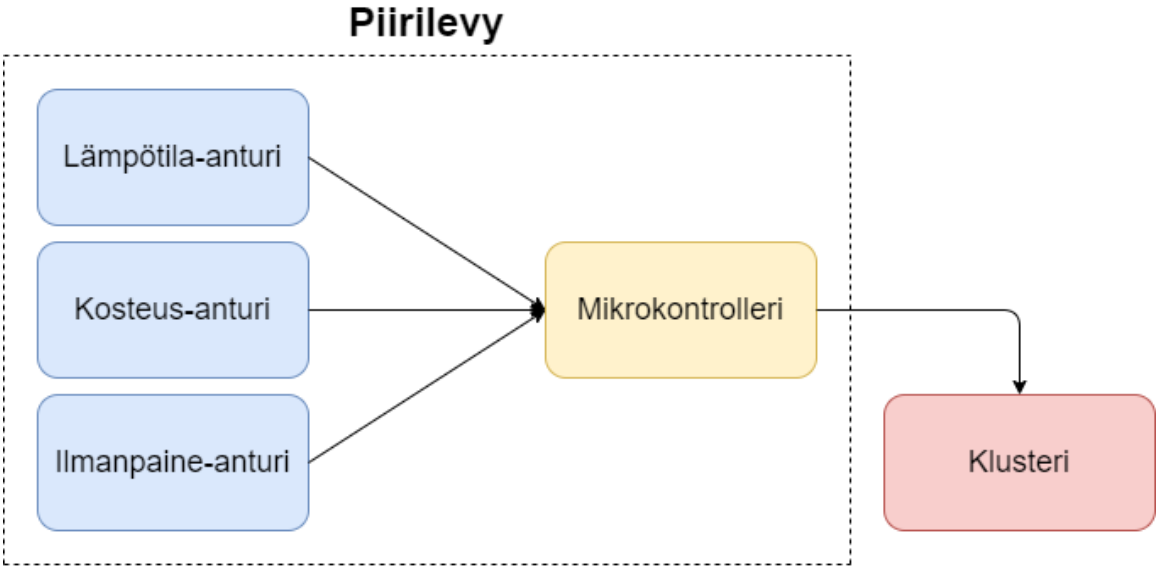
IoT eli esineiden internet käsite kuvaa fyysisten esineiden verkkoa, jotka on upotettu antureihin, ohjelmistoihin ja muihin teknologioihin tietojen yhdistämiseksi muiden laitteiden ja järjestelmien internetin kautta. Nämä laitteet vaihtelevat tavallisista kotitaloustuotteista teollisiin työkaluihin. [11.] Kuvassa 10 on esitetty havainnekuva opinnäytetyössä toteutetusta anturiverkosta. Anturisolmun data välitetään MQTT-protokollaa hyödyntäen tietokantaan. Anturisolmun data välitetään WLAN-reitittimellä MQTT-protokollan mukaisesti vastaanottopään klusterissa sijaitsevaan tietokantaan.



Kuva 10. Anturisolmun tietoliikenne kuvattuna.

3.1 Anturisolmu

Tässä luvussa käsitellään anturisolmun eri komponentteihin liittyvä teoria, joka mahdollistaa syvemmän ymmärryksen anturisolmun käytännön toiminnasta. Kuvassa 11 on esitetty tyypillisen anturisolmun periaatekuva. Kuvassa ei oteta kantaa tiedonsiirtoprotokolliin.



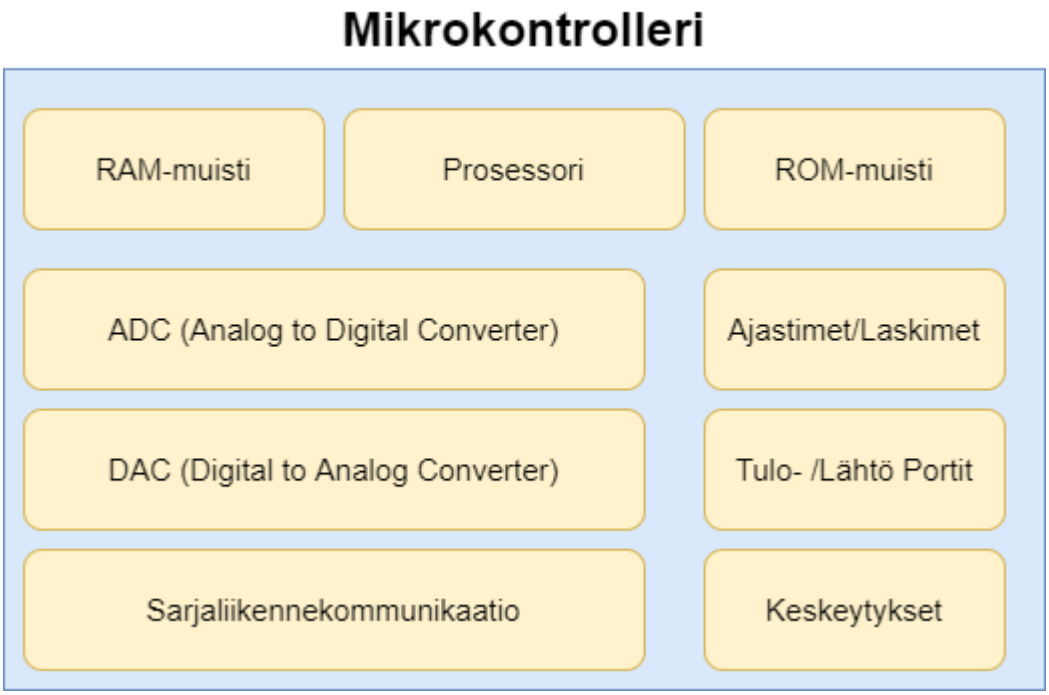
Kuva 11. Anturisolmun periaatekuva.

3.1.1 Mikrokontrollerit

Mikrokontrollereita löytyy nykypäivänä monenlaisista sulautetuista laitteista, kuten kaukosäätimestä, autoista, jääkaapeista ja pyykkikoneista. Mikrokontrollereista puhutaan ”Computer on chip” -tyylisenä laitteena, koska ne sisältävät monia normaalin tietokoneen ominaisuuksia, mutta paljon pienemmässä mittakaavassa. Mikrokontrollereista käytetään yleisesti lyhennettä MCU (engl. Micro Controller Unit). [12, 13.]

Mikrokontrollerit ovat kompakteja, integroituja piirejä, jotka on suunniteltu hallitsemaan sulautetun järjestelmän tiettyä toimintoa. Mikrokontrolleriin sisältyy ainakin prosessori, muistia ja tulo-/lähtö-(I/O, Input/output) oheislaitteet yhdellä mikropiirillä. [11, 13.]

Mikrokontrollerin merkittävin ero mikroprosessoriin verrattuna löytyykin mikrokontrollerissa samaan mikropiiriin integroiduista oheislaitteista. Mikroprosessorin mikropiirille on integroitu pelkkä suoritin. Ohjelmakoodi sijaitsee mikroprosessorissa erillisellä EPROM-sirulla. [13.] Kuvassa 12 esitetään komponentteja, joita mikrokontrollerista voi löytyä.



Kuva 12. Tyypillisen mikrokontrollerin toiminnallinen lohkokaavio.

Mikrokontrollerin tärkein sisäinen elementti on sen prosessori (engl. CPU, Central Processing Unit), joka toimii mikrokontrollerin ”aivoina”. Se prosessoi ja vastaa erilaisiin ohjeistuksiin, jotka ohjaavat mikrokontrollerin toimintaa. Ohjeistuksiin kuuluu mm. aritmetiikan laskenta, ohjelmalogiikka ja I/O-operaatiot. [12, 13, 14.]

Mikrokontrollerin muistiin tallennetaan dataa, jota prosessori vastaanottaa ja käyttää ohjeistuksissaan, jotka se on koodattu toteuttamaan. Mikrokontrollereissa on kahta erilaista muistia. ROM (engl. Read Only Memory) -tyyppiseen muistiin tallennetaan data, jota mikrokontrolleri aina tarvitsee, kuten mikrokontrolleriin asennettu ohjelmakoodi. ROM-muistiin tallennettu data säilyy, vaikka mikrokontrollerista sammutetaan virta. RAM (engl. Random Access Memory) -tyyppiseen muistiin tallennetaan väliaikaista dataa, jota ei ole tarkoitus säilyttää kauaa, kuten esim. lämpötila-anturilta luettu data. RAM-tyyppinen muisti pyyhkiytyy virran sammuessa. [12, 13, 14.]

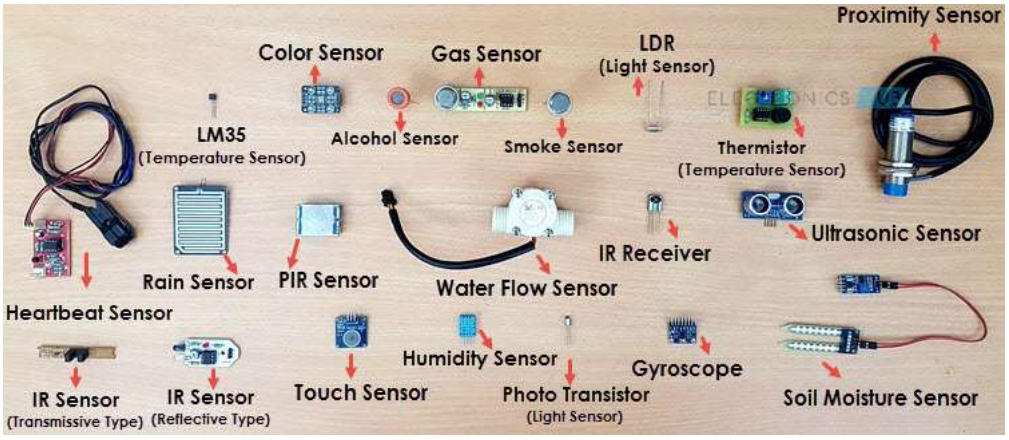
Mikrokontrolleri kommunikoi muiden laitteiden kanssa tulo- /lähtö (engl. I/O, Input/Output) -oheislaitteiden kautta. Tuloporttiin saapuu dataa, joka lähetetään prosessorille käsiteltäväksi. Tuloporttiin voi saapua esim. lämpötila-anturin dataa. Lähtöporttiin kirjoitetaan data, joka halutaan viedä mikrokontrollerin ulkopuolelle, esim. toiseen mikrokontrolleriin. [12, 13, 14.]

ADC (Analog To Digital Converter) eli AD-muunnin muuttaa saamansa analogisen signaalin digitaalisiksi tavuiksi. Analoginen signaali voi olla jännitteen tai virran muodossa. DAC (engl. Digital

To Analog Converter) -muunnin on AD-muuntimen laitteen vastakohta. Se muuttaa digitaalisen datan analogiseen jännitemuotoon. [12, 13, 14.]

3.1.2 Anturit

Yksinkertaisin tapa määritellä anturi on, että se on laite, joka tuottaa lähtösignaalia tiettyyn fyysiseen suureeseen nähden. Tuotettu lähtösignaali lähetetään muille laitteille kuten mikrokontrollerille käsiteltäväksi. Antureilla halutaan huomata muutoksia ympäristössä ja reagoida niihin. Esimerkiksi jos käyttäjä huomaa, että lämpötila-anturi kertoo, että lämpötila on liian suuri, hän voi reagoida kyseiseen lämpötilaan haluamallaan tavalla. Antureilla voidaan mitata monia eri suureita, kuten lämpötilaa, kosteutta, valonmäärää, ja ilman partikkeleja. [15.] Kuvassa 13 esitetään erilaisia antureita.



Kuva 13. Monenlaisia antureita esitettynä [15].

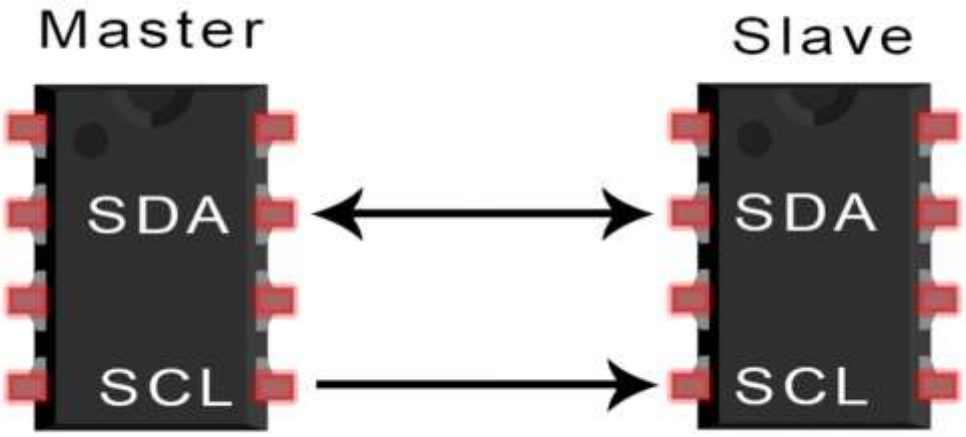
Antureita voidaan lajitella erilaisiin kategorioihin. Anturit voidaan esim. jakaa aktiivisiin ja passiivisiin antureihin. Aktiivinen anturi tarvitsee ulkoisen tehosignaalin, passiivinen anturi voi tuottaa lähtösignaalinsa ilman ulkoista virransyöttöä. [15.]

Anturit voidaan myös luokitella kategorioihin havaintotapansa avulla. Mahdollisia kategorioita on, esim. sähköinen, biologinen, kemikaalinen ja radioaktiivinen. Seuraava kategoriointi perustuu signaalin muuntamisen tapaan, kuten valosähköinen, lämpösähköinen, sähkökemiallinen, sähkömagneettinen ja lämpöoptinen. Kenties tärkein anturien luokittelu on antureiden jakaminen analogisiin ja digitaalisiin antureihin. Analogiset anturit tuottavat mitattavan suureen arvoon verran-

nollista, jatkuvaa, analogista signaalia. Digitaalisissa antureissa suureeseen verrannollinen sähköinen signaali muutetaan digitaaliseksi anturissa sijaitsevan AD-muuntimen avulla. Digitaalisen signaalin lukemiseen anturilta käytetään digitaalista tiedonsiirtoväylää, kuten I2C. [15.]

3.1.3 I2C-protokolla

I2C on sarjamuotoinen, kaksijohtiminen ja synkroninen tiedonsiirtoväylä. Väylä tunnetaan myös nimellä ”IIC-väylä (engl. Inter-Integrated Circuit Bus)” tai ” I^2C”. I2C-väylää käytetään yleisesti piirilevyn sisäisessä kommunikoinnissa. Synkronoitu väylä tarkoittaa, että toimintaan liittyy kellosignaali, jonka mukaan tietoa lähetetään. [17.] Kuva 14 esittää I2C-protokollan periaatetasolla.



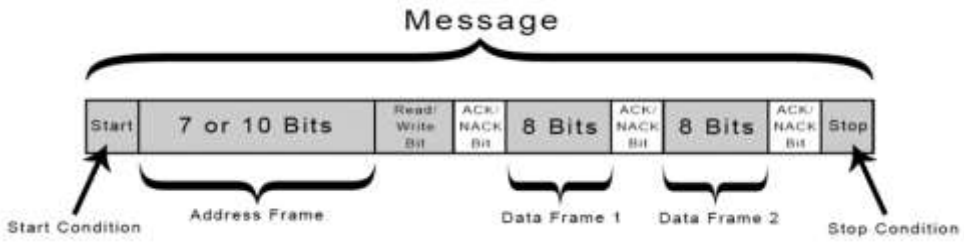
Kuva 14. I2C-protokollan pinnit kuvattuna [17].

I2C-väylässä tarvitaan maajohtimen lisäksi vain kaksi johdinta tietoliikennekommunikointiin. SDA (Serial Data Line) on linja, joka siirtää varsinaiset tietobitit. SCL (Serial Clock Line) vastaa kellopulssista, jonka tahdittamana tietobitit siirretään SDA-linjalla. I2C-laitteet voidaan jakaa kahteen rooliin: isäntälaite (engl. master) ja orjalaite (engl. slave). [17.]

Isäntälaite voi aloittaa tiedonsiirron, mutta orja ei. Kumpikin laite voi kuitenkin lähettää ja vastaanottaa tietoa. Jokaisella orjalaitteella on osoite, jota isäntä kutsuu aloittaessaan tiedonsiirron kyseisen laitteen kanssa. Väylä on ”multi-master” -tyyppinen, eli väylää voi komentaa useampi kuin yksi isäntälaite. Tämä toiminto tuo mukanaan mahdollisuuden tehdä virheitä, jos useampi isäntä yrittää lähettää tietoa samaan aikaan. Väylään on tämän takia kehitetty törmäyksenesto

(engl. Collision Detection), jolla väylä osaa toipua törmäystilanteesta hävittämättä lähetettyä tietoa. [17.]

I2C-väylän normaali tiedonsiirtonopeus on 100 Kb/s. Nopeassa toimintatilassa nopeus on 400 Kb/s. Sovellusten vaatiessa yhä nopeampia tiedonsiirtonopeuksia on kehitetty myös ”Fast-Mode Plus” (Fm+), joka siirtää tietoa 1 Mb/s nopeudella, ”High-speed Mode” siirtää tietoa 3,4 Mb/s nopeudella. [17.] Kuva 15 esittää I2C-viestin muodon.



Kuva 15. I2C-viestin muoto [17].

Viesti alkaa START-bitillä, joka kertoo, että viesti on alkanut. I2C käyttää osoitteita (engl. Addressing). Osoitekehys on aina ensimmäinen kehys START-bitin jälkeen viestissä. Isäntä lähettää osoitteen orjalle, jonka kanssa haluaa kommunikoida ja orja vertaa osoitetta omaansa. ACK-bitti lähetetään takaisin isännälle, jos osoitteet vastaavat toisiaan. Jos osoitteet eivät vastaa toisiaan, orja ei tee mitään ja SDA-linja pysyy HIGH-asennossa. Kirjoitus-/lukubitti kertoo orjalle, jos isäntä haluaa kirjoittaa dataa orjalle, tai lukea orjalta dataa. Kirjoitettaessa orjalle bitti on LOW, lukiessa HIGH. Sen jälkeen, kun isäntä vastaanottaa ACK-bitin orjalta, ensimmäinen datakehys voidaan lähettää. Datakehys on aina 8 bittiä eli 1 tavu, ja lähetetään merkitsevin (MSB) bitti ensin. Jokaisen datakehyksen jälkeen on ACK/NACK-bitti, joka kertoo, onko kehys vastaanotettu onnistuneesti. ACK-bitti on aina pakko vastaanottaa ennen kuin seuraava datakehys lähetetään. Sen jälkeen, kun kaikki datakehykset on lähetetty, lähettäjä STOP-bitin kertoakseen vastaanottajalle, että lähetys on loppunut. [17.]

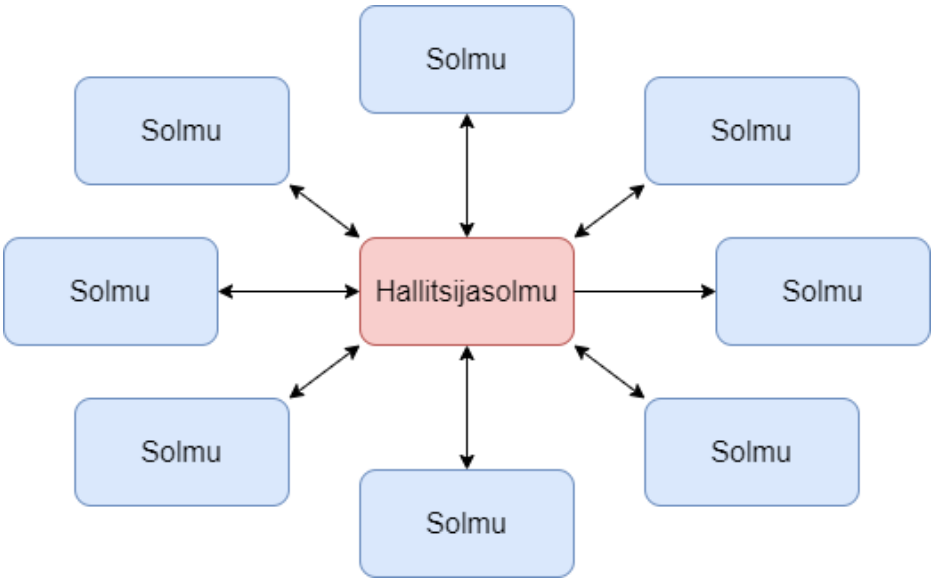
3.2 Anturiverkon taustajärjestelmä

Anturiverkon taustajärjestelmänä toimii klusteri. Taustajärjestelmää kutsutaan opinnäytetyössä termillä klusteri. Klusteri on usean tietokoneen muodostama ryhmittymä, jossa tietokoneet toimivat yhdessä suorittaakseen laskennallisesti intensiivisiä tehtäviä. Yksittäisestä klusterin laitteesta käytetään termiä solmu (engl. node). Klusterissa on aina vähintään yksi ”hallitsijasolmu”,

jonka tehtävä on jakaa klusterin solmujen resursseja, niin että tehtävät voidaan suorittaa mahdollisimman optimaalisesti. [18.]

Klusterin toimintaa varten solmujen tulee olla liitettynä samaan lähiverkkoon, jotta ne voivat kommunikoida keskenään. Klusteri tulee suunnitella niin, että solmujen välisessä kommunikaatiossa esiintyisi mahdollisimman vähän latenssia. Klusterit luodaan ohjelmiston avulla, muuten laitteet eivät voi jakaa resurssejaan keskenään. Klusteriin voidaan määrittää yhteinen massamuisti tai jokaisella solmulla voi olla oma muistinsa. Ohjelmistossa asetetaan hallitsijasolmu, jonka vastuu on jakaa klusterille annettu työ tasaisesti solmuille ja tarvittaessa kerätä työn tulokset yhteen ja esittää työn tulos käyttäjälle. [18.]

Ideaalisesti klusteri näkyisi käyttäjälle yksittäisenä järjestelmänä. Antaessaan työtä klusterille käyttäjän ei tarvitse tietää, onko järjestelmä klusteri vai yksittäinen tietokone. Tätä kutsutaan järjestelmän läpinäkyvyydeksi (engl. transparency). [18, 19.] Kuvassa 16 esitetään klusterien konsepti.



Kuva 16. Klusterin muoto konseptitasolla.

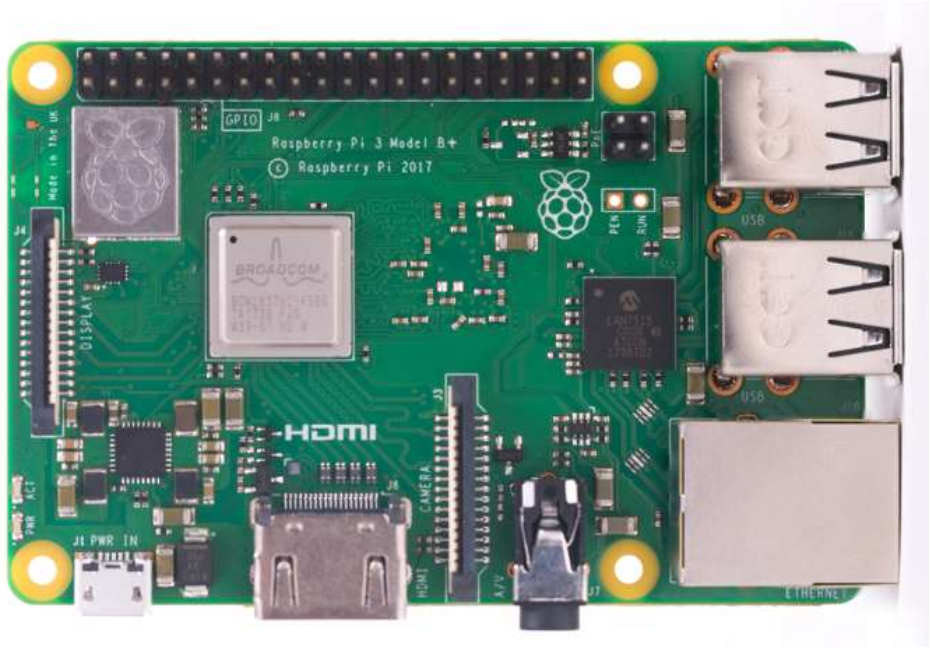
Klusterit voidaan jakaa kolmeen eri malliin, jotka ovat erittäin hyvän saatavuuden klusteri (engl. highly available cluster), kuormituksen tasapainottava klusteri (engl. load-balancing cluster) ja suorituskykyinen klusteri (engl. high performance cluster). Klusteri voi kuulua moneen kategoriaan samanaikaisesti. [24.] Erittäin saatavan klusterin tavoite on tarjota mahdollisimman keskeytymätön palvelu käyttäjälle hyödyntämällä klusterin ”tarpeettomuutta”. Yksittäisen solmun sammuessa muut klusterin solmut ottavat sammuneen solmun tehtävät itselleen, jolloin klusterin yl-

läpitäjillä on aikaa toteuttaa tarvittavat toimenpiteet klusterille. [19.] Kuormituksen tasapainottava klusteri jakaa verkkoliikenteensä tasaisesti klusterin solmuille. Tässä klusterin mallissa kaikki solmut ovat vastuussa tehtävien teosta. Jos yksittäinen solmu epäonnistuu, tulevat pyynnöt jaetaan muiden solmujen kesken. Erittäin saatava klusteri ja tasapainottava klusteri ovat myös oma mallinsa, joita käytetään yleensä verkkopalvelimina. [18, 19.] Suorituskykyisessä klusterissa pyritään maksimoimaan klusterin laskennallinen kapasiteetti. Suuri laskennallinen tehtävä jaetaan pienemmiksi tehtäviksi, jotka jaetaan solmujen kesken. Suorituskykyisiä klustereita käytetään tieteellisissä tehtävissä ja taloudellisessa analyysissa. [18, 19.]

Klustereilla on monia hyötyjä verrattuna yksittäisiin tietokoneisiin. Suurimpia etuja on klusterin helppo skaalaus, kustannustehokkuus, suuri tehokapasiteetti ja suuri saatavuus. Klusterit ovat kuitenkin hankalampia ylläpitää verrattuna yksittäiseen tietokoneeseen. Jokaiselle solmulle tulee asentaa täysi käyttöjärjestelmä, klusterin ohjelmisto ja sen riippuvuudet. Käyttöjärjestelmä tulee myös pitää päivitettynä. Resurssien käyttöä tulee seurata tarkasti jokaisella solmulla. Jaetun muistin hallinta on hankalaa hallita, pitää varmistaa, että solmut eivät ylikirjoita toistensa tiedostoja tai jaettuja tiedostoja. Näitä haasteita voi lieventää käyttämällä kontteja sovelluksen ajamiseen. [18.]

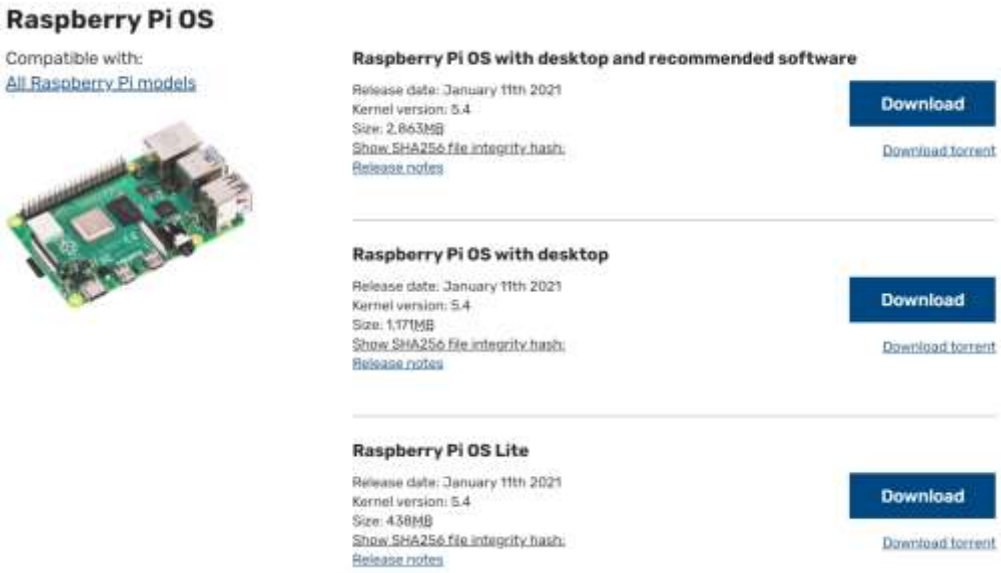
3.2.1 Raspberry Pi

Raspberry Pi -sarjan mikrotietokoneet ovat noin luottokortin kokoisia yhden piirilevyn tietokoneita, jotka ajavat Linux-pohjaisia käyttöjärjestelmiä. Raspberry Pi -mikrotietokoneita voidaan käyttää normaalina tietokoneena, mutta koska ne tarjoavat setin GPIO (engl. general purpose input/output) -pinnejä, niiden pääsijainen tarkoitus on olla osana elektroniikan projekteja. Raspberry Pi -tietokoneista on tehty monia versioita vuodesta 2012 lähtien. [20, 21.] Kuvassa 17 esitetään Raspberry Pi 3B+ -versio.



Kuva 17. Raspberry Pi 3B+ -tietokone [20].

Raspberry Pi voi ajaa monia eri Linux-pohjaisia käyttöjärjestelmiä ja omaa Raspberry Pi OS-käyttöjärjestelmäänsä. Raspberry Pi OS, joka tunnettiin ennen nimellä Raspbian, on Raspberry Pi -mikrotietokoneiden pääasiallinen käyttöjärjestelmä, koska se on suunniteltu juuri kyseisille laitteille toimivaksi. Raspberry Pi OS -käyttöjärjestelmästä tarjotaan eri versioita, sen voi asentaa graafisella käyttöliittymällä tai sen voi asentaa ilman käyttöliittymää, jolloin se käynnistyy komentoriville. [20, 21.] Käyttöjärjestelmä käynnistetään MicroSD-kortilta, johon se on asennettu. Kuvassa 18 esitetään Raspberry Pi OS -versiot.



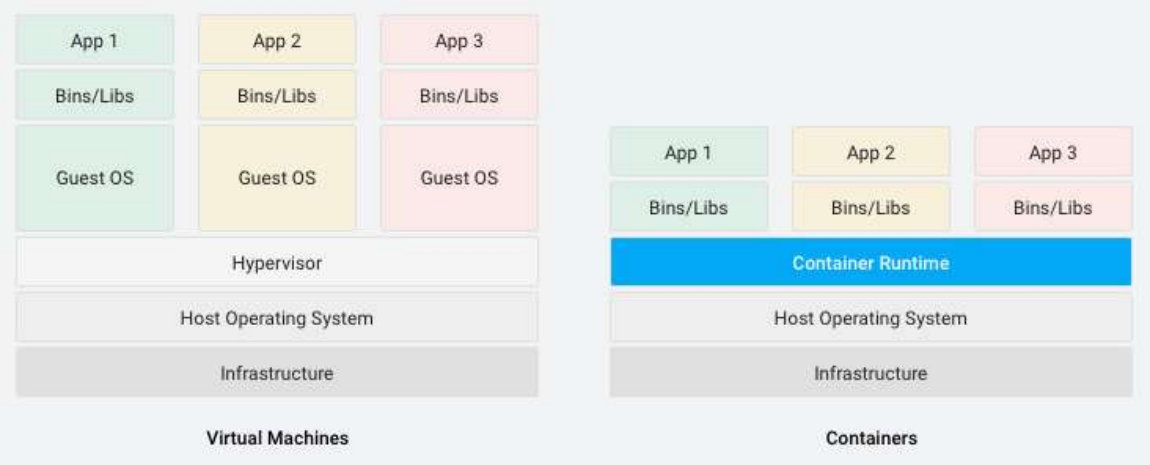
Kuva 18. Raspberry Pi OS -versiot [22].

Raspberry Pi -sarjan mikrotietokoneita kehittää Raspberry Foundation hyväntekeväisyysorganisaatio, jonka tavoite on edistää tietokoneiden opetusta. Raspberry Foundation toimii avoimen lähdekoodin ekosysteemissä, Raspberry Pi OS kuuluu esim. avoimeen lähdekoodiin. Käyttöjärjestelmä sisällyttää itseensä myös monia avoimen lähdekoodin ohjelmistoja. [20, 21.]

3.2.2 Kontit

Kontit (engl. containers) tarjoavat standardisoidun pakkausmekanismin, jossa sovellukset voidaan erottaa käyttöjärjestelmästä, missä konttia ajetaan. Kontit mahdollistavat sovellusten helpon ja johdonmukaisen käyttöönoton, riippumatta käyttöympäristöstä. [23.]

Kontteja voidaan verrata virtuaalisiin koneisiin. Virtuaalista konetta ajetaan isäntä-käyttöjärjestelmän päällä, sillä on virtualisoitu pääsy isännän taustalla olevaan laitteistoon. Kontit eivät tee virtualisaatiota laitteistotasolla, kuten virtuaalikoneet. Kontit virtualisoidaan suoraan isäntäkäyttöjärjestelmän ytimen (engl. kernel) päällä. Tämä tekee konteista kevyempiä ajaa kuin virtuaalikoneet. Ne ovat myös nopeampi käynnistymään ja vievät vähemmän muistia kuin virtuaalikoneet. [23, 24.] Kuvassa 19 esitetään konttien ja virtuaalisten koneiden eroavaisuuksia.



Kuva 19. Konttien ja virtuaalikoneiden eroavaisuus. [23].

Kontit antavat kehittäjille työkalun luoda eristettyjä ja ennalta-arvattavia järjestelmiä. Konttien sisäiset sovellukset ovat aina samassa versiossa, joten versiokonfliktit kehityksessä vähenevät huomattavasti. Kontteja voidaan ajaa Mac-, Windows- ja Linux-käyttöjärjestelmissä. Konttien ajamiseen tarvitaan ohjelmisto, jonka avulla kontteja voidaan luoda, hallita ja poistaa. Docker on yksi suosituimmista konttien hallintaohjelmista. [23.]

3.2.3 Docker

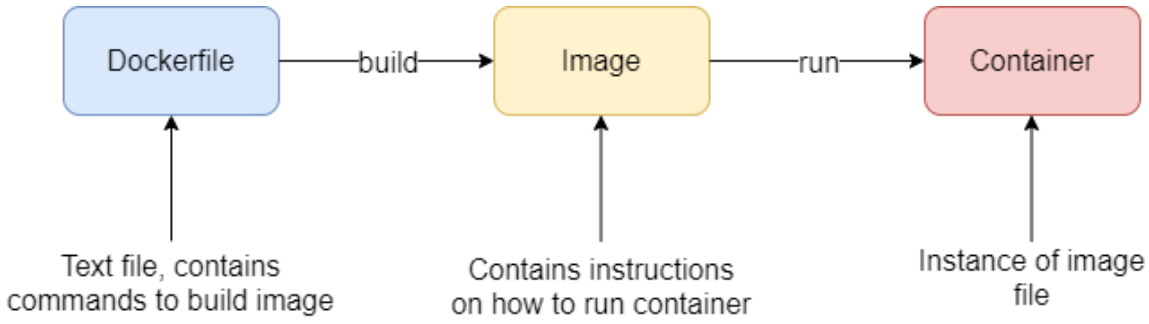
Docker on avoimessa lähdekoodissa oleva konttien hallintasovellusohjelmisto, joka on julkaistu vuonna 2013. Ohjelmiston avulla käyttäjä voi luoda, poistaa ja hallita kontteja. Docker on pääasiallisesti komentorivityökalu. Sen Windows- ja Mac -versiot tarjoavat myös Docker Desktop -nimisen graafisen sovelluksen, jonka avulla Docker-kontteja voi hallita. [25.]

Docker-ohjelmiston taustalla toimii Docker Engine -teknologia. Yleisesti ottaen, puhuttaessa Dockerista, tarkoitetaan Docker Engine-työkalua, eikä itse Docker-yhtiötä tai projektia. Docker Engine -ohjelmistosta on tarjolla kaksi versiota: ”Docker Engine Community” ja ”Docker Engine Enterprise”. Edellä mainittu on kaikille ilmainen versio, ”Enterprise”-versio on maksullinen yhtiöille tarkoitettu versio, johon on lisätty edistyneitä hallintaominaisuuksia ja työkaluja. [25.]

Kaikki Docker kontit alkavat Dockerfile-nimisestä tiedostosta. Tämä tekstitiedosto kirjoitetaan syntaksilla, joka toimii Docker Engine -ohjelmistolle ohjeena, kuinka Docker-kuva (engl. image) rakennetaan. Dockerfile spesifioi kontin sisäisen käyttöjärjestelmän, työkalut, koodikielet, verkon portit ja muut komponentit, joita voi tarvita. Dockerfile voi kertoa myös komennon, jonka kontti suorittaa käynnistäessään, esim. jonkin ohjelmiston käynnistäminen. [25, 26.]

Kun Dockerfile ajetaan ”docker build” -komennolla, työkalu luo kuvan, joka perustuu kyseiseen Dockerfile-tiedostoon. Kuva on siis siirrettävä tiedosto, joka sisältää spesifikaatiot siitä, kuinka ohjelmistoa ajetaan ja miten. Dockerfile siis kertoo ”build”-työkalulle, kuinka kuva rakennetaan, Kuva sisältää Dockerfile-tiedostossa kerrotut ohjelmistot ja komennot. [25, 26.]

Komento ”run” käynnistää kontit annetusta kuvasta. Jokainen kontti on siis instanssi kuvasta. Kontit on suunniteltu olemaan väliaikaisia, kontin voi pysäyttää ja käynnistää uudelleen. Kontti palaa tällöin samaan tilaan, josta se pysäytettiin. Samasta kuvasta voi ajaa monia eri kontteja, jos jokaisella kontilla on oma uniikki nimi. Jotkin asetukset, kuten verkon portit, voivat aiheuttaa konflikteja. [25, 26.] Kuvassa 20 esitetään, kuinka kontti luodaan Docker-ohjelmistolla.



Kuva 20. Docker-kontin luominen.

Luodut kuvat voidaan tallentaa Docker Hub-sivustolle rekisteriin (engl. registry). Docker Hub on Docker-yhtiön tarjoama palvelu kuvien löytämiseen ja jakamiseen muiden käyttäjien kanssa. Docker Hub-palvelun avulla muut käyttäjät voivat käyttää valmiiksi luotuja kuvia, jotta niitä ei aina tarvitse kirjoittaa itse alusta asti. Moni suuri yhtiö kehittää omista sovelluksistaan kuvia helppoa käyttöönottoa varten. [27.]

Docker-compose on Dockerin tarjoama työkalu, jolla voidaan luoda ”docker-compose.yaml”-tiedostoon palvelun kuvaus. Palvelu on yhden tai useamman kontin luoma ryhmä, joka tarjoaa jotakin toiminnallisuutta. Palvelua kuvatessa tiedostossa kerrotaan esim. liityttävä verkko, avattavat portit ja Docker-kuva. [27.] Jos Docker-kontteja halutaan ajaa klusterissa, yksi vaihtoehto on käyttää Docker Swarm-ohjelmistoa.

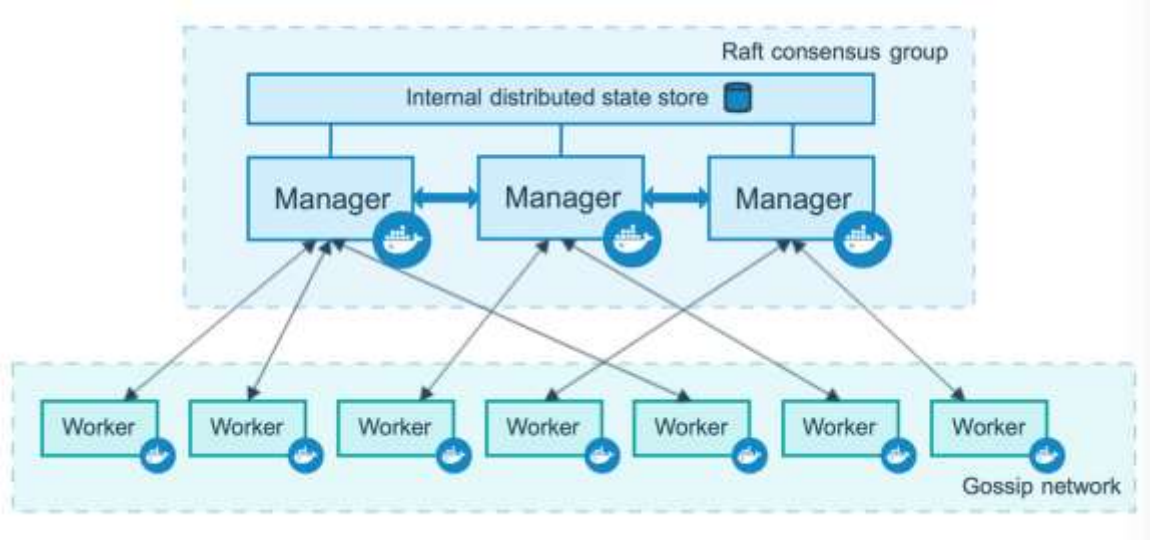
3.2.4 Docker Swarm

Docker swarm on hallinta- ja orkestraatiotyökalu, joka on sisäänrakennettu Docker Engine-ohjelmistoon. Työkalu on rakennettu Docker Engine-ohjelmiston sisään ”swarmkit”-työkalupaketin avulla. [28.]

Parvi (engl. swarm) on ryhmä fyysisiä tai virtuaalisia tietokoneita, jotka ajavat Docker-ohjelmistoa ja on konfiguroitu toimimaan yhdessä klusterina. Kun klusteri on muodostettu, käyttäjä voi ajaa Docker-komentonsa normaalisti, mutta nyt komennot toteutetaan klusterin solmuilla. Klusteria hallitsee parvihallitsija (engl. swarm manager), muista klusterin laitteista käytetään termiä solmu. Swarm-työkalun avulla ajettavista konteista käytetään termiä palvelu. [29.]

Palvelut (engl. services) ovat tehtäviä, joita solmut suorittavat. Palvelua luodessa sille annetaan Docker-kuva, jota käyttää, ja komennot, joita suorittaa kontin sisässä. Tehtävä kantaa sisällään

Docker-kontin ja kyseisen kontin sisäiset komennot. Parven hallitsijasolmu antaa tehtävän tietylle solmulle, Jonka jälkeen se ei voi siirtyä enää toiselle solmulle, Se voi vain suorittua onnistuneesti tai epäonnistua. Swarm-palveluiden avulla voidaan tarjota suuri sovelluksen saatavuus, koska jos yksittäinen solmu kaatuu, muut solmut voivat jakaa ajettavat kontit uudestaan keskenään. Näin voidaan välttää kriittisten palveluiden kaatuminen. [28, 29.] Kuvassa 21 esitetään Docker swarm-solmujen toimintaa.



Kuva 21. Docker swarm-solmujen toiminta [30].

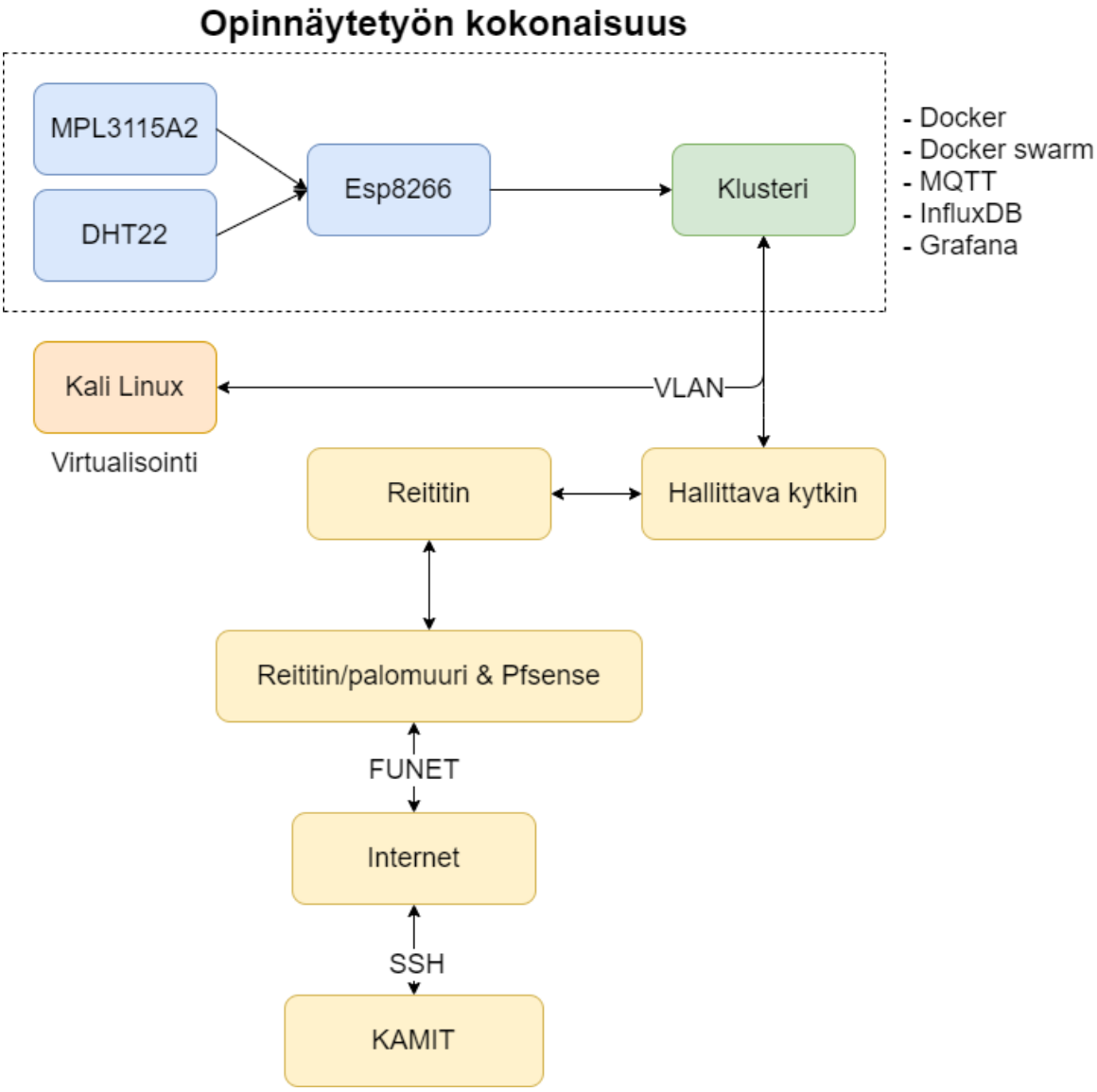
Parvessa sijaitsevalla solmulla voi olla kolme eri roolia: hallitsija (engl. manager), johtaja (engl. leader) tai työntekijä (engl. worker). Hallitsijasolmun tehtävä on jakaa parvelle annetut palvelut tasaisesti työntekijäsolmujen kesken, tätä kutsutaan automaattiseksi resurssien tasapainotukseksi (engl. automatic load-balancing). Hallitsijasolmun tehtäviin kuuluu myös hallita parven tilan ylläpitoa. Yhdessä parvessa voi olla 1–7 hallitsijasolmua, tällöin klusteri ei riipu yhdestä hallitsijasolmusta. [28, 29, 30.] Johtajasolmun vastuu on pitää lokia hallitsijasolmun päätöksistä ja varmistaa, että kaikilla solmuilla on sama versio lokista. Jos johtajasolmu kaatuu, toiselle solmulle annetaan johtajan rooli automaattisesti. Johtajasolmun vastuu on siis varmistaa, että kaikki hallitsijasolmut ylläpitävät samaa tilaa parven tilaa. [28, 29.] Työntekijäsolmun tehtävä on ajaa niitä palveluita, joita hallitsijasolmu antaa suoritettavaksi. Kaikki hallitsijasolmut ovat myös työntekijäsolmuja ja voivat ajaa palveluita, jos niillä on tarpeeksi vapaita resursseja tehdä niin. [28, 29, 30.]

Docker swarm työkalulla voidaan luoda kahdenlaisia palveluja: monistettuja (engl. replicated) ja globaaleja palveluita. Monistettu palvelu toimii spesifioimalla ajettavien palveluiden määrän,

jonka jälkeen parven hallitsija jakaa palvelut solmuille. Yhdellä solmulla voi siis ajaa monta identtistä palvelua [29]. Globaali palvelu toimii niin, että parven hallitsija ajaa yhden palvelun jokaisella saatavilla solmulla, joka voi täyttää palvelun vaatimukset. [29.]

4 Opinnäytetyön käytännön toteutus

Opinnäytetyö on osa suurempaa projektikokonaisuutta, jossa pyritään kehittämään tietoverkkojen opetuksen työkaluja Kajaanin ammattikorkeakoulun opetuskäyttöön. Opinnäytetyö on rajattu niin, että siihen kuuluu anturiverkon luominen ja klusterien luominen. Kuvassa 22 esitetään koko projektin kokonaisuus, jossa esiintyvät klusteri ja anturisolmu kuuluvat opinnäytetyön kokonaisuuteen.



Kuva 22. Projektin kokonaisuus kuvattuna.

4.1 Anturisolmun toteutus

Opinnäytetyö aloitettiin anturisolmun kehittämisellä. Tilaajan vaatimusmäärittelyyn kuului, että anturisolmu toteutetaan piirilevylle. Anturisolmu tuli toteuttaa yksinkertaisesti, koska se ei ole opinnäytetyön pääasiallinen kehitysalue. Anturisolmun tuli voida kommunikoida langattomasti lähiverkon kautta klusterin kanssa, jotta anturidata voidaan syöttää tietokantaan. Anturisolmussa tulee olla pieni näyttö, joka kertoo esim. laitteen IP-osoitteen tai muuta hyödyllistä tietoa. Ennen kuin anturisolmun kehitys pystyi alkamaan, tuli ensin valita käytettävä kehitysalusta, anturit ja koodaustyökalu.

4.1.1 Kehitysalustan valinta

Anturisolmun suunnittelu alkoi kehitysalustan valinnalla. Tilaajan pyynnöstä projektissa käytetään Heltec Wifi kit 8 -kehitysalustaa. Heltec WiFi kit 8 on esp8266-mikrokontrolleriin perustuva kehitysalusta. Siinä on sisäänrakennettu WiFi, joka toimii 2,4 GHz taajuudella ja käyttää 801.11n-standardia. Kehitysalusta saa virtansa Micro USB liittimestä, se käyttää viiden voltin käyttöjännitettä ja voi viedä virtaa maksimissaan 135 mA. Heltec WiFi kit 8 sisältää myös sisäänrakennetun pienen OLED-näytön, jolla voi esittää hyödyllistä tietoa. [31.] Kuvassa 23 esitetään kyseinen kehitysalusta.

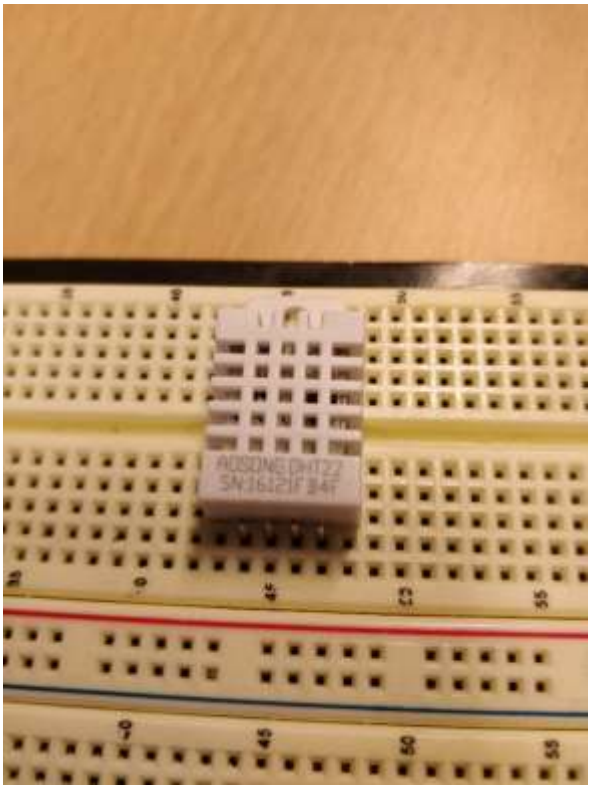
Kuva 23. Heltec WiFi kit 8 -kehitysalusta [31].

Kehitysalustoja oli saatavilla Kajaanin Ammattikorkeakoulun laboratoriotiloista ilman erillistä tilausta.

4.1.2 Mitattavat suureet & anturit

Mitattaviksi suureiksi valittiin lämpötila, kosteus ja ilmanpaine. Nämä suureet valittiin, koska niiden mittaaminen on yleistä, joten niille löytyy paljon anturivaihtoehtoja. Suureet ovat myös hyödyllisiä yleisessä elämässä, jos kerättyä dataa halutaan hyödyntää muihin tarkoituksiin.

Lämpötilaa ja kosteutta päätettiin mitata DHT22 anturilla (kuva 24). Anturi valittiin käytettäväksi, koska se on suosittu, helppo käyttää ja oli saatavilla Kajaanin Ammattikorkeakoulun laboratoriotiloista.



Kuva 24. DHT22-lämpötila- ja kosteusanturi.

Taulukossa 1 on esitetty DHT22-anturin ominaisuuksia.

Taulukko 1. DHT22-anturin yleistietoja [32].

Anturi	DHT22
Käyttöjännite	3–5 V
Toiminta-alue (Lämpötila)	-40–80 °C
Toiminta-alue (Kosteus)	0–100 %RH
Resoluutio (Lämpötila)	0,1 ℃
Resoluutio (Kosteus)	0,1 RH

Liitännät	VCC, GND & Data

Ilmanpainetta päätettiin mitata MPL3115A2-anturilla (kuva 25). Tämä anturi oli myös saatavilla Kajaanin Ammattikorkeakoulun laboratoriotiloista käytettäväksi.

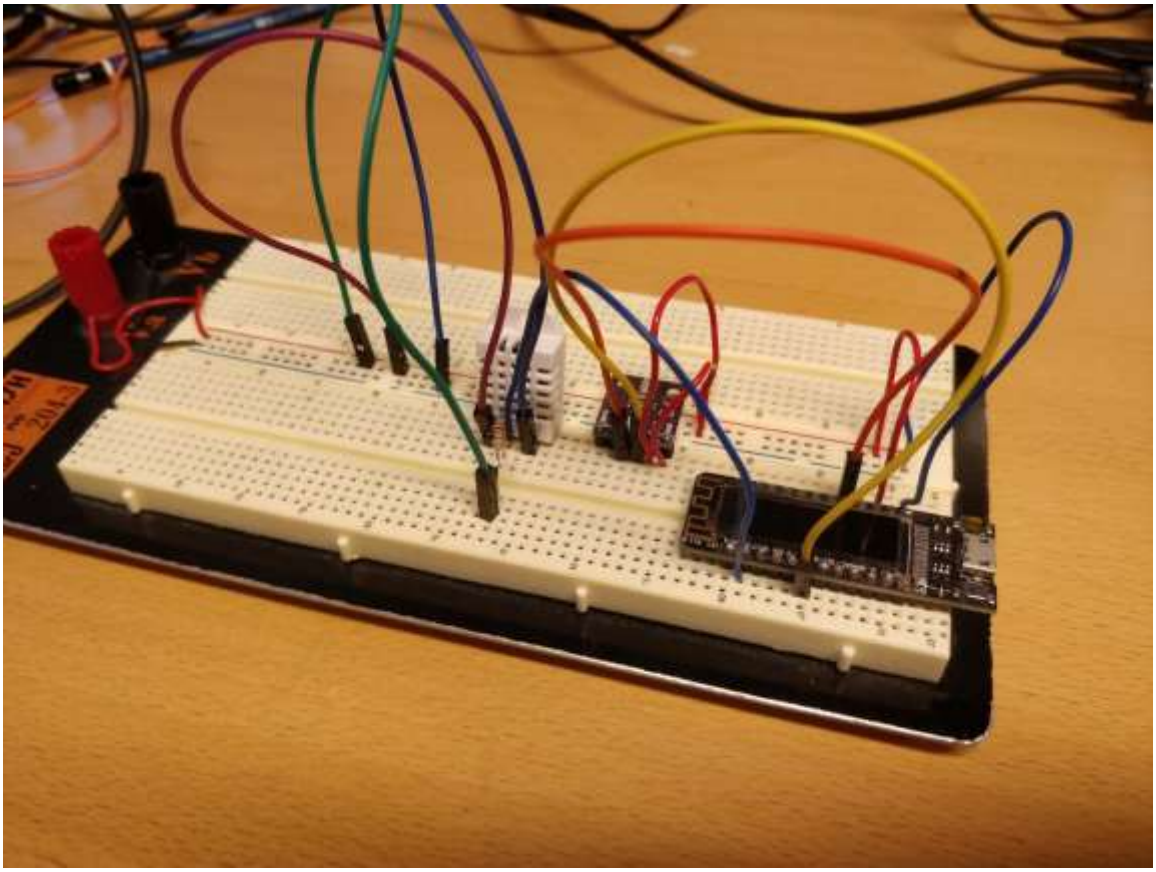
Kuva 25. CJMCU MPL3115A2 -anturi [33].

Taulukko 2 esittää anturin yleistietoja, kuten käyttöjännitteen ja liitännät [33].

Anturi	MPL3115A2
Käyttöjännite	1,95 – 3,6 V
Toiminta-alue (lämpö-tila)	-40-80 ℃
Resoluutio	1,5 Pa
Mittausväli (paine)	20 Pa - 110 kPa
Liitännät	INT2, INT1, SDA, SCL, VCC, GND

4.1.3 Kytkennät

Ennen kuin anturisolmulle voitiin suunnitella piirilevy, tuli komponenteista rakentaa koekytkentä, jolla voitiin varmistaa laitteiston toimivuus. Koekytkennän avulla voitiin myös aloittaa ohjelmisto-koodin kehitys (kuva 26).



Kuva 26. Anturisolmun koekytkentä.

Koekytkennän liitokset esitetään taulukossa 3, liitokset pysyvät samana piirilevyllä.

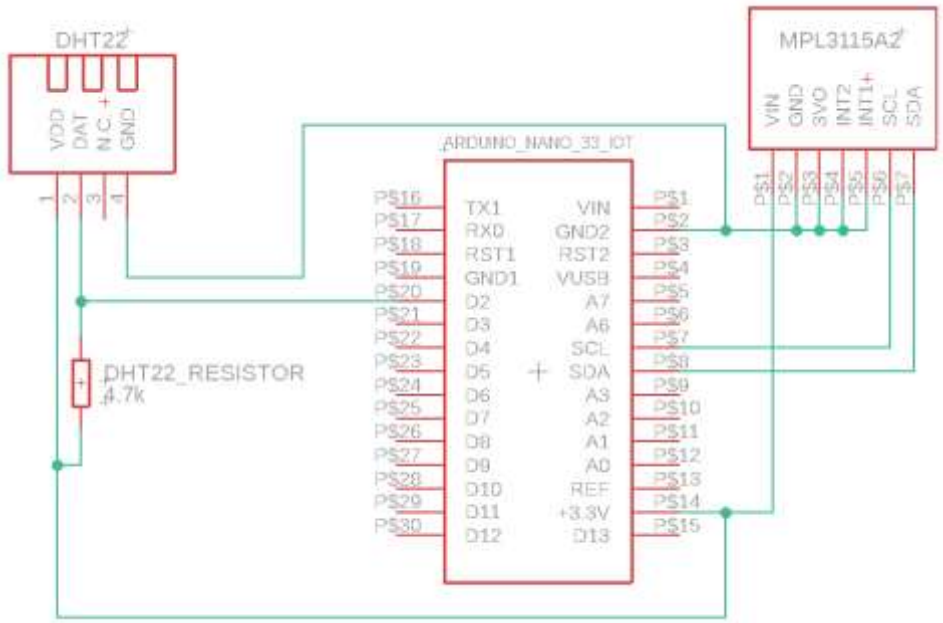
Taulukko 3. Komponenttien liitokset.

Anturi	Pinni	Kehitysalustan portti
DHT22		
	VCC	+3.3 V
	Data	GPIO 0
	GND	GND
MPL3115A2		
	VCC	+3.3 V
	GND	GND
	SDA	SDA, GPIO 4
	SCL	SCL, GPIO 5

4.1.4 Piirilevy

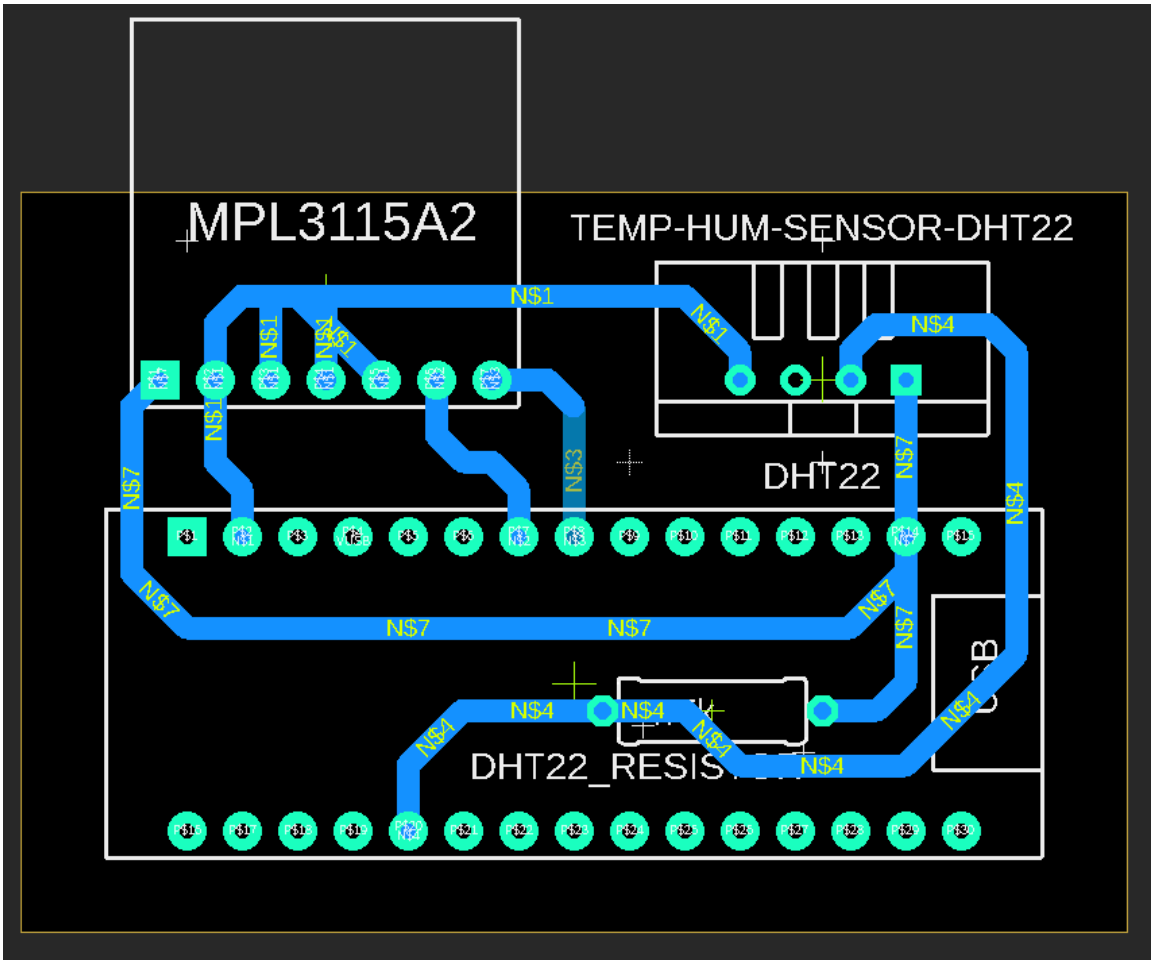
Anturisolmulle suunniteltiin tilaajan vaatimusmäärittelyn mukaan piirilevy. Anturisolmun komponenttien valinta pidettiin yksinkertaisena, jotta piirilevy olisi helppo ja nopea suunnitella. Anturisolmu haluttiin muutenkin pitää yksinkertaisena, koska se ei ole opinnäytetyön pääasiallinen kehitysalue.

Anturisolmun piirilevy suunniteltiin Autodesk Eagle -ohjelmistolla, jolle oli saatavilla opiskelijalisenssi Kajaanin Ammattikorkeakoulun kautta. DHT22-komponentille löytyi valmis kirjasto, jota voitiin hyödyntää. MPL3115A2-komponentille tuli luoda oma kirjasto. Heltec WiFi kit 8 -kehitysalustalle ei myöskään ollut saatavilla valmista kirjastoa, joten sillekin tuli luoda oma kirjasto. Kuva 27 esittää piirilevyn kaavakuvan (engl. schematic).



Kuva 27. Anturisolmun piirilevyn kaavakuva.

Kaavakuvassa vasemmalla puolella esiintyy DHT22-anturi ja sen PULL UP -vastus, jolle annettiin arvoksi 4,7 kilo-ohmia datalehden mukaan. Oikealla puolella esiintyy MPL3115A2-anturi, se ei tarvinnut muita komponentteja toimiakseen. Keskellä kaavakuvaa esiintyy Heltec WiFi kit 8 -kehitysalusta. Kuvassa 28 esitetään kaavakuvan pohjalta luotu piirilevykuvaus (engl. board).



Kuva 28. Kaavakuvan pohjalta luotu piirilevykuvaus.

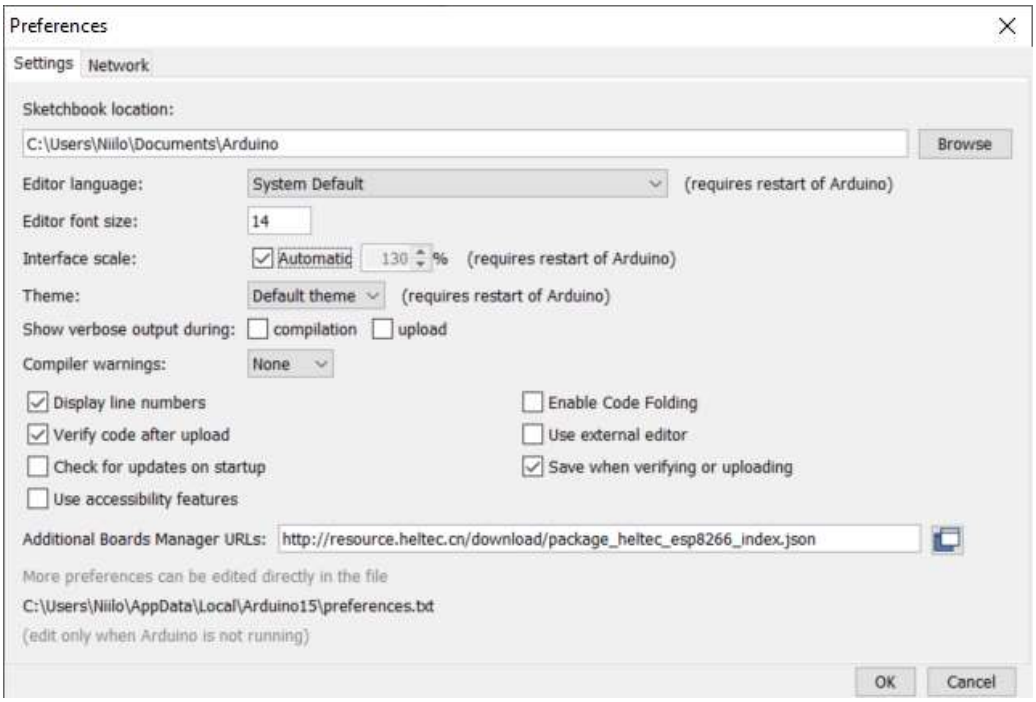
Kun tiedostot olivat valmiit, niistä luotiin ”Gerber”-tiedostot, joiden pohjalta fyysinen piirilevy voitiin jyrsiä. Piirilevy jyrsittiin Kajaanin Ammattikorkeakoulun tiloista löytyvällä ”Bungard” merkkisellä jyrsimellä. Valitettavasti piirilevyä ei ole saatu vielä kirjoituksen aikana toimimaan. Piirilevyn kehitysalusta ei saa luettua dataa antureilta luotettavasti. MPl3115A2-anturi palauttaa kehitysalustalle epämääräisiä merkkejä, DHT22-anturi ei palauta kehitysalustalle mitään arvoja. Piirilevyn kehitystä jatketaan vielä kirjoittamisen jälkeen.

4.1.5 Ohjelmistokoodi

Anturisolmun koodaukseen käytettiin Arduino IDE -koodaustyökalua. Työkalu valittiin käytettäväksi, koska se on hyvin yksinkertainen käyttää, ja sillä saa nopeasti koodia kirjoitettua ja ladattua kehitysalustalle. Tarjolla olisi ollut muitakin vaihtoehtoja kuin Arduino IDE. Toinen kannattava

vaihtoehto olisi ollut Visual Studio Code ja siihen manuaalisesti asennettava ”PlatformIO”-laajennus. Kyseistä työkalua kuitenkin vältettiin, koska se oli tarpeettoman monimutkainen asentaa ja käyttää. VS Code tukee Arduino-kirjastoja vain laajennuksen kautta, mutta laajennuksenkin kanssa kirjastojen lisääminen koodiin ei ole kovin helppoa, Koodauksessa haluttiin välttää turhaa asetuksien säätämistä ja päästä toteuttamaan koodia mahdollisimman nopeasti.

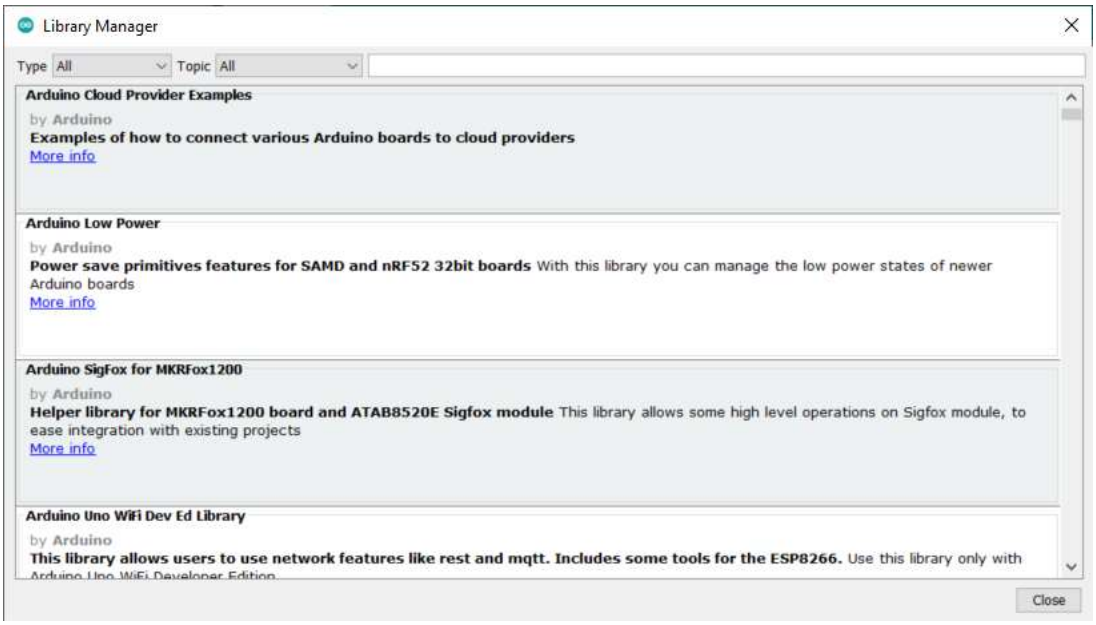
Ennen kuin ohjelmiston koodaus oli mahdollista aloittaa, Arduino IDE -ohjelmistoon piti lisätä valitun kehitysalustan tuki manuaalisesti. Kehitysalustan voi lisätä Arduino IDE-työkaluun valitsemalla yläreunasta ”File” -> ”Preferences”. Alareunassa on ”Additional Boards Manager URLs”-tekstikenttä, johon lisätään uusien kehitysalustojen linkit. Kehitysalustojen linkit haetaan alustan valmistajan sivuilta (kuva 29).



Kuva 29. Arduino IDE "File" -> "Preferences" näkymä.

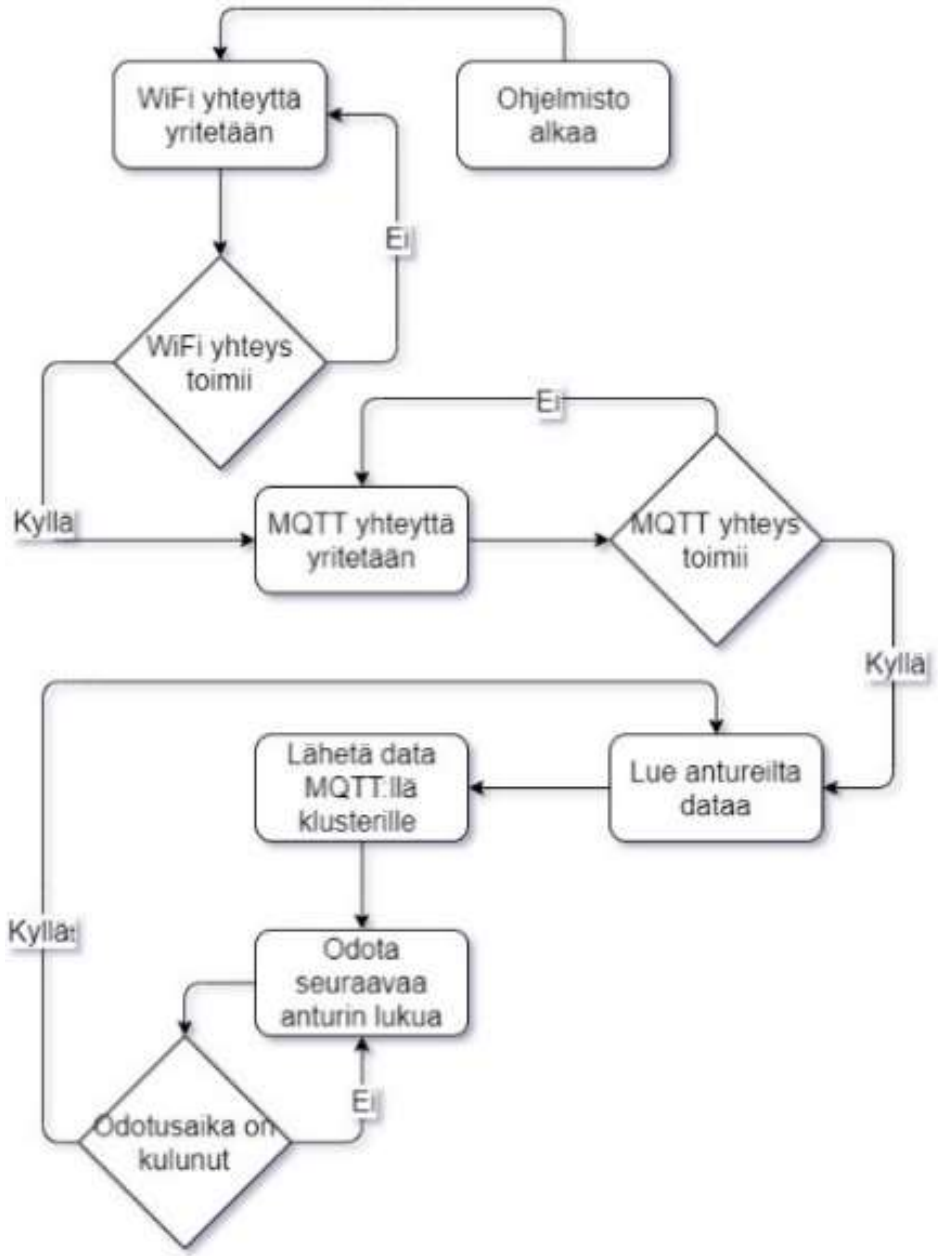
Kehitysalustan toiminnallisuuden lisäämisen jälkeen lisätään vielä koodissa tarvittavat kirjastot. Arduino IDE -ohjelmistossa on sisäänrakennettu työkalu, jonka avulla voi asentaa helposti tarvittavat kirjastot. Kirjastot voi asentaa valitsemalla yläreunasta ”Tools” -> ”Manage Libraries…”. Ohjelmistoon avautuu näkymä, jolla voi hakupalkin avulla etsiä tarvittavat kirjastot ja asentaa

kirjastosta haluttu versio (kuva 30). Ohjelmiston koodaukseen valittiin kaikista kirjastoista mahdollisimman uusi versio.



Kuva 30. "Tools" -> "Manage Libraries..." näkymä.

Ohjelmistokoodin suoritus on kuvattu kuvan 31 vuokaaviossa.



Kuva 31. Ohjelmistokoodin kulku esitetty vuokaaviona.

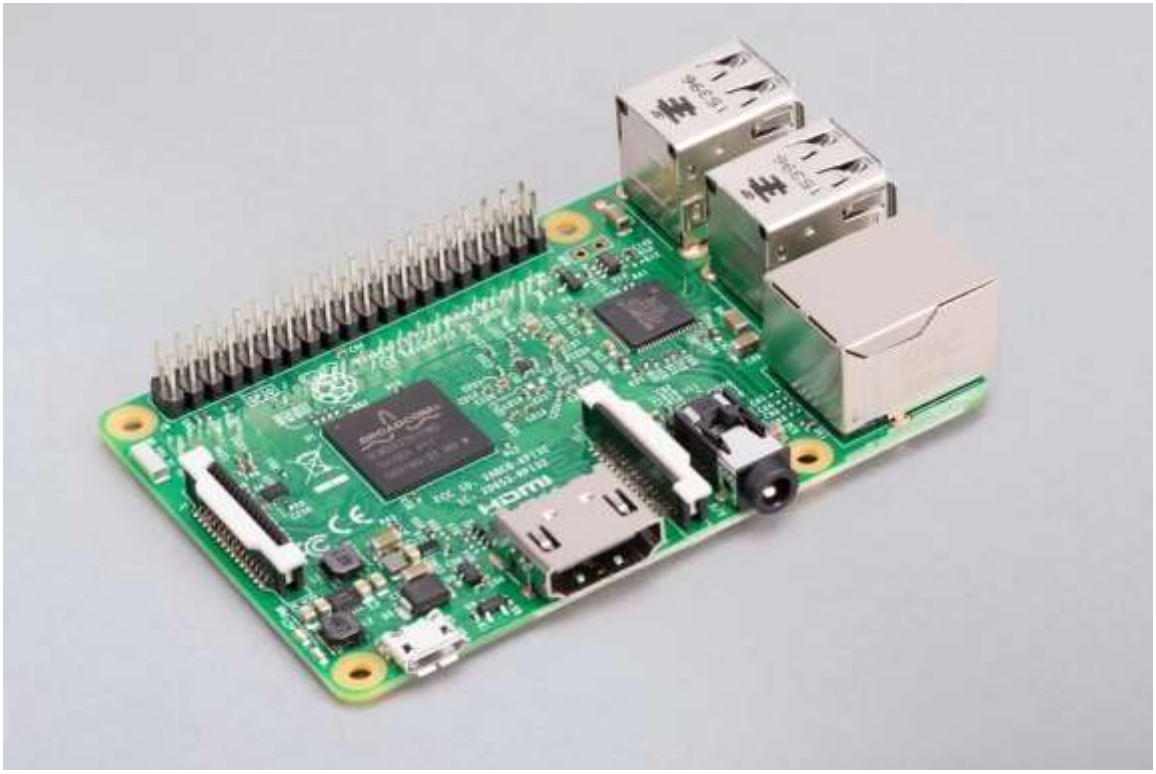
4.2 Klusterin toteutus

Tietoverkkojen opetusympäristön klusterin käytännön osuus aloitettiin rakentamalla yksittäinen esimerkkiklusteri, joka toimii esimerkkinä jatkokehitystä varten. Luku antaa lukijalle ohjeistuksen siitä, kuinka samanlainen klusteri voidaan toteuttaa luvussa toteutetun klusterin vierelle myöhempänä ajankohtana. Esimerkkiklusterin toteutus alkoi valitsemalla projektiin sopiva laitteisto ja ohjelmisto, joiden avulla klusteri voidaan toteuttaa.

4.2.1 Laitteiston ja ohjelmistojen valinta

Tietoverkkojen opetusympäristön laitteistoksi valittiin käytettäväksi Raspberry Pi -sarjan mikrotietokoneet. Sarja valittiin, koska ne ovat suhteellisesti halpoja mikrotietokoneita, jotka ovat helppoja alustaa nopeasti käyttöä varten. Projektia varten tilattiin kaksitoista Raspberry Pi 4B -mallin mikrotietokonetta klusteria varten. Projektissa päädyttiin kuitenkin käyttämään Raspberry Pi 3B -mallin mikrotietokoneita tilaajan pyynnöstä, koska niitä oli runsaasti saatavilla Kajaanin Ammattikorkeakoulun tiloissa. Tällöin 4B-mallin Raspberry Pi -tietokoneet voidaan säästää tulevaisuuden käyttöön, jossa suurempi prosessointiteho on tärkeämmässä roolissa kuin klusterissa.

Raspberry Pi 3B -mikrotietokoneessa on neljäytiminen ja 64-bittinen Broadcom BCM2837 -prosessori, joka toimii 1,2 GHz taajuudella [40]. Mikrotietokoneessa on 1 GB RAM-muistia [34]. Sen voi liittää verkkoon 10/100 Mbps Ethernet-liitännän kautta, 3B-mallissa on BCM43438 WLAN ja Bluetooth-sovitin [34]. 3B-mallissa on myös HDMI-videoliitäntä ja 3.5 mm ääniliitäntä, mutta niitä ei hyödynnetä tässä projektissa. Raspberry Pi 3B saa virtansa Micro USB-liitännästä, joka toimii viiden voltin käyttöjännitteellä, 3B voi maksimissaan ottaa 2.5 ampeerin edestä virtaa. [34.] Kuvassa 32 esitetään Raspberry Pi 3B -versio.



Kuva 32. Raspberry Pi 3B -versio [34].

Esimerkkiklusterissa voidaan käyttää 3B-mallin Raspberry Pi mikrotietokoneita, myös siksi, että ne ovat helposti korvattavissa, jos niiden alhainen prosessointiteho ja RAM-muistin määrä osoittautuu ongelmaksi. Klusterin laitteisto on muutenkin helposti vaihdettavissa, eikä kaikkien laitteiden tarvitse olla samaa mallia tai edes Raspberry Pi -tietokoneita. Klusteriin on mahdollista liittää myös virtuaalisia tietokoneita. Laitteiston ainoat pakolliset vaatimukset ovat, että se voi ajaa Docker-ohjelmistoa ja liittyä lähiverkkoon. Koska projektiin valittiin käytettäväksi Raspberry Pi -sarjan mikrotietokoneet, tulee palveluiden kehityksen aikana huomioida, että Docker-kuvien tulee tukea ARM32v7-prosessoriarkkitehtuuria.

Klusterin konttien hallintaan valittiin käytettäväksi Docker, koska se on hyvin suosittu ja helppokäyttöinen tapa hallita kontteja. Docker tarjoaa kotisivuillaan myös laajan dokumentaation. Klusterin orkestraatioon oli tarjolla kaksi eri vaihtoehtoa: Docker swarm ja Kubernetes. Projektissa päädyttiin käyttämään Docker swarm -ohjelmistoa, koska sen avulla saa nopeasti klusterin käyttöön, toisin kuin Kubernetes-ohjelmistolla, jossa on paljon opittavaa ennen kuin sen avulla voi luoda klusterin. Docker swarm on myös valmiiksi sisäänrakennettu Docker-ohjelmistoon. Klusteri on tilaajan pyynnöstä tarkoitus pitää yksinkertaisena, jotta sitä olisi helppo huoltaa ja jatkokehittää tulevaisuudessa.

4.2.2 Käyttöjärjestelmän ja ohjelmistojen asentaminen

Esimerkkiklusterissa toteutetaan neljän Raspberry Pi -mikrotietokoneen klusteri, jossa mikrotietokoneet on liitetty samaan lähiverkkoon yksinkertaisen Ethernet-kytkimen kautta. Mikrotietokoneet saavat virtansa virallisista Raspberry Pi -virtalähteistä. Mikrotietokoneiden käyttöjärjestelmät käynnistetään USB-tikuilta. Käyttöjärjestelmät päätettiin asentaa USB-tikuille SD-korttien sijaan, koska SD-korttien elinikä on lyhyt verrattuna USB-tikkuihin, kun niille kirjoitetaan paljon dataa muistiin. Mikrotietokoneiden käyttöjärjestelmiksi asennettiin Raspberry Pi OS -versio 11-01-21 ja Ubuntu 18.04.5 LTS -käyttöjärjestelmät. Kolmeen Raspberry Pi mikrotietokoneeseen asennettiin Raspberry Pi OS ja yhteen asennettiin Ubuntu. Ennen kuin Raspberry Pi mikrotietokoneet voitiin käynnistää USB-tikuilta, tuli se toiminnallisuus ottaa käyttöön jokaisella laitteella.

Toiminnallisuus otettiin käyttöön asentamalla Raspberry Pi OS SD-kortille, jolta Raspberry Pi käynnistettiin. Koodiesimerkki 1 kirjoittaa tekstirivin käynnistyksen konfiguraatio-tiedostoon.

```
echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt
```

Koodiesimerkki 1. Tiedostoon ”/boot/config.txt” kirjoittaminen.

Kun Raspberry Pi käynnistetään, komento kertoo laitteelle, että sen tulee kirjoittaa USB käynnistystä varten tarvittavat bitit OTP-muistiin. Komennon tuloksen voi tarkistaa koodiesimerkin 2 komennolla. Jos bittien kirjoittaminen on onnistunut, komento kirjoittaa tekstin ”17:3020000a” komentoriville.

```
vcgencmd otp_dump | grep 17:
```

Koodiesimerkki 2. Bittien kirjoituksen tarkistus.

Jokainen klusterin Raspberry Pi käynnistettiin SD-kortilta ja tarkistettiin koodiesimerkin 2 komennolla, että laite tukee USB-käynnistystä.

USB-tikuille asennettiin käyttöjärjestelmä käyttämällä Raspberry Pi Imager -ohjelmistoa. Ohjelmistossa valitaan asennettava käyttöjärjestelmä ja SD-kortti, jolle se asennetaan, mutta tässä tapauksessa valittiin USB-tikku SD-kortin sijaan. Kuva 33 esittää ohjelmiston käyttöä. ”Write” aloittaa käyttöjärjestelmän kirjoitusprosessin USB-tikulle.

Kuva 33. Raspberry Pi Imager -ohjelmisto.

Käyttöjärjestelmien SSH-palvelu otettiin käyttöön automaattisesti ensimmäisen käynnistyksen aikana luomalla tiedosto ”ssh” USB-tikun ”/boot” kansioon. Ubuntu-asennuksessa se luotiin kansioon ”system-boot”. Tällä tavalla säästettiin aikaa, koska laitetta ei tarvinnut liittää näyttöön ja

näppäimistöön ensimmäisellä käynnistyksellä. Laitteelle kirjauduttiin SSH-palvelun kautta oletusarvoilla (koodiesimerkki 3). Ubuntu vaatii, että oletussalasana vaihdetaan välittömästi kirjautumisen yhteydessä.

```
ssh pi@raspberrypi
ssh ubuntu@ubuntu
```

Koodiesimerkki 3. SSH kirjautuminen oletusarvoilla.

Kirjautumisen jälkeen komentoriviltä käynnistettiin ”raspi-config”-työkalu, jolla voitiin muuttaa tarvittavat asetukset (koodiesimerkki 4).

```
sudo raspi-config
```

Koodiesimerkki 4. ”raspi-config”-työkalun avaaminen.

Käyttäjän ”pi” salasana vaihdettiin ja isäntänimi muutettiin. Nimeämisessä käytettiin tiettyä kaavaa, isäntänimessä kerrotaan klusterin kirjain ja numero. Esimerkkiklusterin ensimmäisen Raspberry Pi mikrotietokoneen isäntänimi on ”rasp-node-a1”. isäntänimen etuliite ”rasp-node” on aina sama, sen jälkeinen kirjain kertoo missä klusterissa laite sijaitsee, numeron avulla voidaan erottaa klusterin laitteet toisistaan. Ubuntu käyttöjärjestelmään vaihdettiin isäntänimi koodiesimerkin 5 mukaisesti.

```
hostnamectl set-hostname rasp-node-a4
```

Koodiesimerkki 5. Isäntänimen vaihtaminen.

Tämän jälkeen asennettiin kaikki uudet päivitykset (koodiesimerkki 6).

```
sudo apt update && sudo apt upgrade
```

Koodiesimerkki 6. päivityksien hakeminen ja asentaminen.

Raspberry Pi käynnistettiin uudelleen tämän jälkeen. Seuraavaksi asennettiin Docker. Ennen asennusta asennustiedosto tuli ladata Dockerin sivuilta (koodiesimerkki 7).

```
curl -fsSL https://get.docker.com -o get-docker.sh
```

Koodiesimerkki 7. Docker-asennustiedoston hakeminen.

Kun tiedosto oli ladattu, se ajettiin koodiesimerkin 8 komennolla. Komennon ajamiseen tarvittiin ”root”-käyttäjän oikeudet.

```
sudo sh get-docker.sh
```

Koodiesimerkki 8. Asennustiedoston ajaminen.

Jotta Docker-komentoja ei tarvitse aina ajaa ”root” käyttäjänä, lisättiin käyttäjä ”pi” ”docker”-ryhmään koodiesimerkin 9 mukaisesti.

```
sudo usermod -aG docker pi
```

Koodiesimerkki 9. Käyttäjän ”pi” liittäminen ryhmään ”docker”.

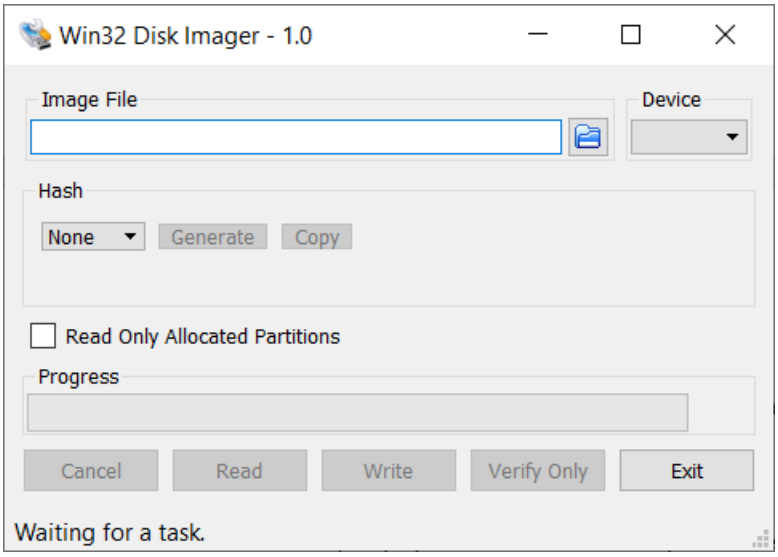
Viimeisenä muutoksena otettiin palvelu ”rsyslog” pois päältä käynnistyksessä. Kun palvelu ei käynnisty, käyttöjärjestelmälokeja ei kirjoiteta USB-muistiin, mikä pidentää sen elämää. Tämä toteutettiin koodiesimerkin 10 komennoilla. Ennen kuin palvelu voitiin ottaa pois päältä, palvelu piti pysäyttää.

```
sudo service rsyslog stop
sudo systemctl disable rsyslog
```

Koodiesimerkki 10. Palvelun pysäyttäminen ja käynnistyksen poistaminen.

Tässä vaiheessa Raspberry Pi käynnistettiin uudelleen, jotta kaikki muutokset tulisivat voimaan.

Kun Raspberry Pi OS -käyttöjärjestelmä oli alustettu, siitä otettiin ”.img”-tiedosto käyttämällä Win32 Disk Imager -ohjelmistoa. Kopioimalla käyttöjärjestelmä säästettiin aikaa, koska muihin käyttöjärjestelmiin ei tarvinnut asentaa kaikkea uudestaan. Käyttöjärjestelmä kopioitiin kahdesti, kummastakin käyttöjärjestelmästä vaihdettiin käyttäjän salasana ja isäntänimi. Kuvassa 34 esitetään Win32 Disk Imager -ohjelmiston käyttöliittymä.



Kuva 34. Win32 Disk imager -ohjelmisto.

4.2.3 Docker swarm -käyttöönotto

Kun laitteiston käyttöjärjestelmät oli asennettu, voitiin aloittaa Docker swarm -ohjelmiston käyttöönotto (koodiesimerkki 11). Komento liittää kyseisen solmun hallitsijasolmuna parveen.

```
Docker swarm init –advertise-addr 192.168.178.20
```

Koodiesimerkki 11. Docker swarm alustus.

Komento palauttaa komentoriville uuden komennon, jolla toiset solmut voidaan liittää työntekijäsolmuina parveen (koodiesimerkki 12). Komento ajettiin kaikilla muilla solmuilla.

```
docker swarm join --token SWMTKN-1-1o2o7pomjxo79504y235gaw3c4fq937bc40iv-mcdvrg9u1b63a-2snsmwu9x5j542acfi51skdj9 192.168.178.20:2377
```

Koodiesimerkki 12. Parveen liittyminen työntekijäsolmuna.

Jos parveen haluaa liittää uuden solmun myöhemmin, voidaan luoda uusi tunnus (engl. token) työntekijöille tai hallitsijoille (koodiesimerkki 13).

```
docker swarm join-token worker
docker swarm join-token manager
```

Koodiesimerkki 13. Uuden tunnuksen luominen.

Jos työntekijäsolmun haluaa ylentää hallitsijasolmuksi, sen voi tehdä koodiesimerkin 14 mukaisesti. Työntekijäsolmu ”ras-node-a2” ylennettiin hallitsijasolmuksi.

```
docker node promote rasp-node-a2
```

Koodiesimerkki 14. Solmun ylentäminen.

Lopuksi parvessa olevat solmut listattiin komentoriville (koodiesimerkki 15) (Kuva 28).

```
docker node ls
```

Koodiesimerkki 15. Solmujen listaaminen.

Kun kaikki solmut olivat liitettynä parveen, voitiin aloittaa itse palveluiden alustaminen, joita swarm ajaa.

4.2.4 Docker swarm-solmujen visualisointi

Jotta parvessa olevia solmuja voidaan visualisoida paremmin, esimerkkiklusterille käynnistetään palvelu, joka antaa visuaalisen esityksen kaikista parven solmuista ja niillä ajettavista palveluista. Palvelu perustuu GitHub käyttäjän ”ManoMarks" luomaan ”visualizer” Docker-kuvaan, josta on luotu Raspberry Pi mikrotietokoneille yhteensopiva versio [35] (kuva 35).

```
version: '3.8'

services:
  visualizer:
    image: alexellis2/visualizer-arm:latest
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
```

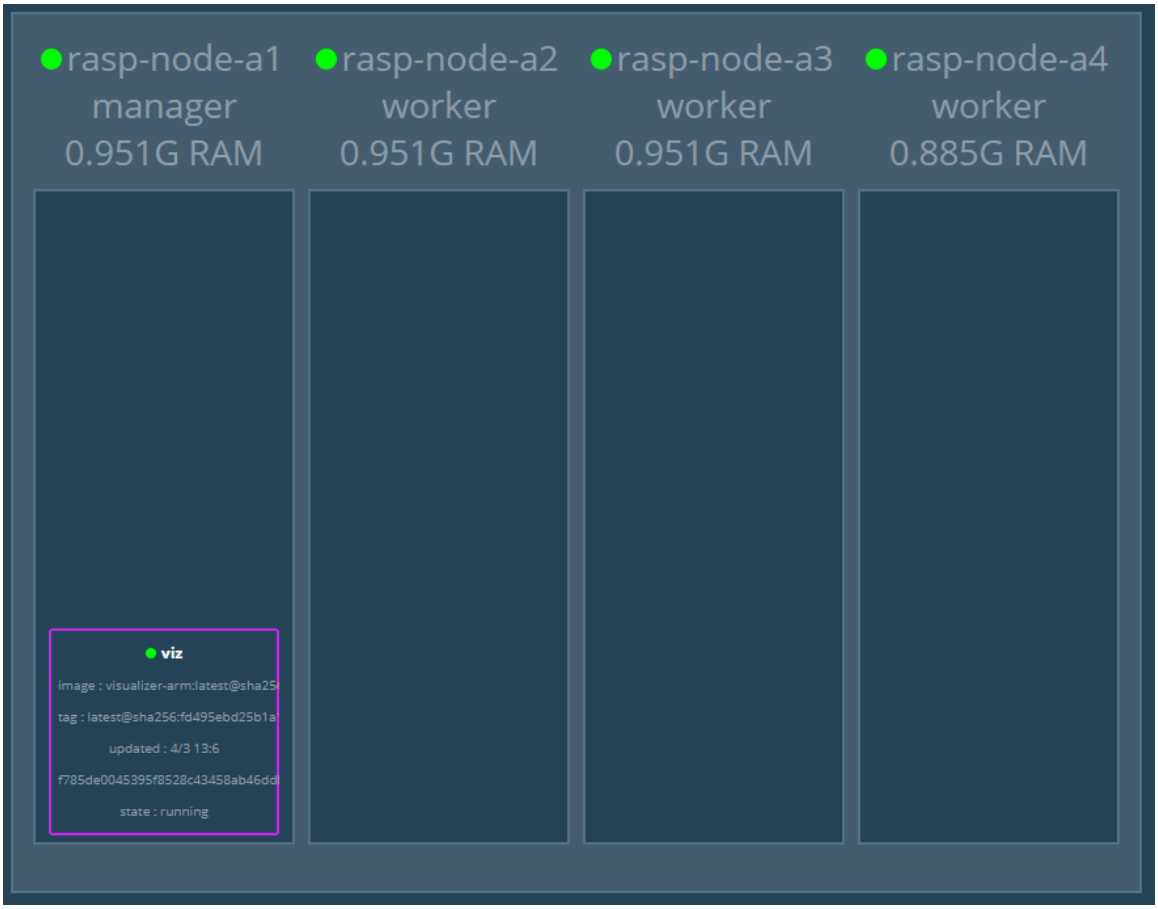
Kuva 35. Visualisointi palvelun ”docker-compose.yaml” -tiedosto.

Palvelu käynnistettiin koodiesimerkin 16 komennolla, samasta kansiosta, jossa tiedosto sijaitsi.

```
docker stack deploy -c docker-compose.yaml MONITOR
```

Koodiesimerkki 16. palvelun käynnistäminen.

Kun palvelu on käynnistetty, millä tahansa selaimella voidaan avata hallitsijasolmun IP-osoitteen portti 8080, tässä tapauksessa ”http://192.168.178.20:8080”. Selaimessa esitetään kuvan 36 mukainen parven palveluiden visualisointi. Parvessa näkyy tällä hetkellä vain visualisointipalvelu.



Kuva 36. Palveluiden visualisointi.

4.2.5 Tietokantapalvelu

Vaatimusmäärittelyn mukaisesti tuli klusterille pystyttää tietokanta, jonne anturidata voidaan tallentaa. Projektissa päädyttiin käyttämään InfluxDB-tietokantaa. InfluxDB on aikasarjatietokanta (engl. time series database), eli se on optimoitu tallentamaan ja noutamaan aikaleimaan liittyvää dataa, kuten projektissa kerättävää anturidataa [36]. Tietokannasta on tarjolla ARM32-prosessoriarkkitehtuurille yhteensopiva Docker-kuva. Kuvassa 37 esitetään InfluxDB-tietokannan Docker swarm -palvelun konfiguraatio.

```
influx:
  image: arm32v7/influxdb:1.8.4
  networks:
    - sensor-network
  ports:
    - "8086:8086"
  environment:
    - INFLUXDB_ADMIN_ENABLED=true
    - INFLUXDB_ADMIN_USER=${INFLUXDB_ADMIN_USER:-admin}
    - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_ADMIN_PASSWORD:-admin}
    - INFLUXDB_DB=sensor_data
    - INFLUXDB_USER=${INFLUXDB_USER:-mqtt_user}
    - INFLUXDB_USER_PASSWORD=${INFLUXDB_USER_PASSWORD:-mqtt_password}
  volumes:
    - ./influxdb.conf:/etc/influxdb/influxdb.conf
    - ./data:/var/lib/influxdb/data
  deploy:
    mode: replicated
    replicas: 1
    placement:
      constraints: [node.labels.influx == true]
```

Kuva 37. InfluxDB Docker-compose -konfiguraatio.

Konfiguraatiossa palvelu liitetään Docker verkkoon ”sensor-network” ja siltä avataan portti 8086 etäyhteyttä varten. Konfiguraatiossa asetetaan myös ympäristömuuttujia, jotka asettavat ”admin”-tunnukset, luovat tietokannan ja luovat uuden käyttäjän luotuun tietokantaan.

Palveluja käynnistettiin vain yksi, koska se tallentaa dataa solmun muistiin. Jos palvelua ajettaisiin eri solmuilla, tietokantaan tallennettu data ei aina olisi saatavilla. Tämä ratkaistiin ajamalla palvelua vain solmulla, jolle on asetettu etiketti (engl. label) (koodiesimerkki 17).

```
docker node update –label-add influx=true rasp-node-a1
```

Koodiesimerkki 17. Etiketin asettaminen solmulle.

Palvelu asetettiin ajamaan vain solmulla, jossa on etiketillä ”influx” on arvo ”true”. Palvelua käynnistäessä ympäristömuuttujille voidaan antaa arvot, tai voidaan käyttää ennalta määritettyjä arvoja jättämällä ne komennosta pois. Palvelu käynnistettiin koodiesimerkin 18 mukaisesti.

```
INFLUXDB_ADMIN_USER=root \
INFLUXDB_ADMIN_PASS=root_pass \
docker stack deploy -c docker-compose.yaml SENSOR
```

Koodiesimerkki 18. Palvelun käynnistys ympäristömuuttujilla.

4.2.6 MQTT-palvelu

MQTT-palvelu toimii MQTT-protokollan mukaisesti välittäjänä. Se ei siis syötä keräämäänsä dataa suoraan tietokantaan, Klusterille kehitetään toinen palvelu, joka toimii välittäjän tilaajana ja syöttää keräämänsä datan tietokantaan.

MQTT-palvelun haluttiin olevan mahdollinen ajaa millä tahansa solmulle, joten sille luotiin Dockerfile (Kuva 38). Tiedostossa kopioidaan ”mosquitto.conf” ja ”passwordfile” -tiedostot kuvan sisään, näin voidaan välttää tiedostojen jakaminen solmun muistista, joka sitoisi palvelun ajamaan aina samalla solmulla. Kontin käsketään ajaa kopioitua ”mosquitto.conf”-tiedostoa ”ENTRYPOINT”-avainsanalla.

```
#Eclipse Mosquitto is an open source message broker.
FROM eclipse-mosquitto:2.0.8

#Copy configuration & password files to docker container.
COPY mosquitto.conf /mosquitto/config/mosquitto.conf
COPY passwordfile /etc/mosquitto/passwd

# -v == Enable verbose logging
# -c == Use given custom .conf file
ENTRYPOINT [ "mosquitto", "-v", "-c", "/mosquitto/config/mosquitto.conf" ]
```

Kuva 38. MQTT-palvelun kuvan Dockerfile.

Koska Raspberry Pi -sarjan mikrotietokoneet ovat ARM-pohjaisia prosessoriarkkitehtuuriltaan, tuli kuva rakentaa kyseiselle arkkitehtuurille yhteensopivaksi. Tätä varten tuli luoda uusi rakentaja (engl. builder) ja ottaa se käyttöön (koodiesimerkki 19.).

```
docker buildx create --name arm32builder --use
```

Koodiesimerkki 19. Uuden rakentajan luonti ja käyttöönotto.

Komentorivillä navigoitiin kansioon, jossa Dockerfile sijaitsi, jonka jälkeen uusi kuva luotiin, sillä annetaan koodiesimerkin 20 mukaan tuki linux/amd64, linux/arm64, linux/arm/v7 prosessoriarkkitehtuureille. Komennolla työnnetään myös kuva Docker Hub-rekisteriin, josta Docker swarm voi hakea sen käytettäväksi.

```
docker buildx build \
        --platform linux/amd64,linux/arm64,linux/arm/v7 \
        -t niiloniskanen/mosquitto:1.0 \
        --push .
```

Koodiesimerkki 20. Kuvan rakentaminen ja työntö Docker Hub rekisteriin.

Seuraavaksi luotiin palvelulle ”docker-compose” -tiedosto. Tiedostossa käytetään uutta työnnettyä kuvaa, avataan portti 1883 ja liitytään ”sensor-network” -verkkoon. Tiedostossa kerrotaan myös palveluiden määrä ja kuinka palvelun tulee käynnistyä uudelleen virheen sattuessa (Kuva 39).

```yaml
services:
  broker:
    image: niiloniskanen/mosquitto:1.0
    networks:
      - sensor-network
    ports:
      - 1883:1883
    deploy:
      mode: replicated
      replicas: 1
      restart_policy:
        condition: on-failure
        delay: 5s
        max_attempts: 3
        window: 120s

networks:
  sensor-network:
    driver: overlay
    external: true
```

Kuva 39. MQTT-palvelun ”docker-compose”-tiedosto.

Docker-compose tiedoston luomisen jälkeen palvelupino käynnistettiin samasta kansiosta, jossa tiedosto sijaitsee (koodiesimerkki 21.)

```
Docker stack deploy -c docker-compose.yaml MQTT
```

Koodiesimerkki 21 Palvelupinon käynnistäminen.

Jotta MQTT-välittäjän keräämä anturidata voitiin tallentaa tietokantaan, tuli luoda toinen MQTT-palvelu, joka toimii välittäjän asiakkaana.

4.2.7 MQTT-tilaajapalvelu

MQTT-tilaajapalvelu kerää MQTT-välittäjältä sille lähetetyn anturidatan ja tallentaa sen tietokantaan. Anturidata kerättiin Python-kielellä toteutetulla koodilla. Palvelulle luotiin Dockerfile (kuva

37), joka kopioi kooditiedostot Docker-kuvaan ja asentaa tarvittavat python kirjastot ”requirements.txt”-tiedostosta (kuva 40).

```
FROM python:3.9-alpine3.13

WORKDIR /code

#Copy python file and requirements.txt to container.
COPY ./python/ ./

# Install python packages listed in .txt file
RUN pip install -r requirements.txt

ENTRYPOINT [ "python", "subscribe.py"]
```

Kuva 40. MQTT-tilaajapalvelun Dockerfile.

Dockerfile rakennettiin Docker-kuvaksi koodiesimerkin 22 mukaisesti.

```
docker buildx build \
        --platform linux/amd64,linux/arm64,linux/arm/v7 \
        -t niiloniskanen/mqtt_client:1.0 \
        --push .
```

Koodiesimerkki 22. MQTT-tilaajapalvelun rakennus.

Kun Docker kuva oli rakennettu ja työnnetty Docker Hub-rekisteriin, voitiin aloittaa docker-compose tiedoton luonti (kuva 41).

```yaml
version: '3.8'

services:
  client:
    image: niiloniskanen/mqtt-client:1.0
    environment:
      - PYTHONUNBUFFERED=1
      - BROKER_ADDRESS=${BROKER_ADDRESS:-192.168.178.20}
      - BROKER_PORT=${BROKER_PORT:-1883}
      - MQTT_USERNAME=${MQTT_USERNAME:-mosqCluster}
      - MQTT_PASSWORD=${MQTT_PASSWORD:-mosqPass}
      - CLIENT_NAME=${CLIENT_NAME:-cluster-client}
      - INFLUX_ADDRESS=${INFLUX_ADDRESS:-"http://192.168.178.24:8086"}
      - INFLUX_TOKEN=${INFLUX_TOKEN}
      - INFLUX_ORG=${INFLUX_ORG:-Cluster}
    networks:
      - sensor-network
    deploy:
      mode: replicated
      replicas: 1
      placement:
        constraints: [node.labels.mqtt-client == true]

networks:
  sensor-network:
    driver: overlay
    external: true
```

Kuva 41. MQTT-tilaajapalvelun ”docker-compose”-tiedosto.

Palvelu käynnistettiin (koodiesimerkki 23), parametrina annettiin InfluxDB-tietokannasta haettu ”Token”, joka toimii tietokannan tunnisteena. ”Token” antaa python-koodille luvan työntää dataa tietokantaan.

```
INFLUX_TOKEN=FDw4f1n9p9GgBxtDvh1tAhvFzTAinzz-
MMhRtPXXLgWLM2qbG_IuqFa3vNBtq30t8sKRy9j9EjXqkF9r4uWGm-w== \
Docker stack deploy -c docker.compose.yaml MQTT
```

Koodiesimerkki 23. Palvelun käynnistäminen ”Token” parametrilla.

Kaikki tilaajan vaatimusmäärittelyn mukaiset palvelut olivat nyt käynnistetty, seuraavaksi luotiin ylimääräinen palvelu, joka visualisoi tietokantaan tallennettua anturidataa.

4.2.8 Anturidatan visualisointipalvelu

Anturidataa visualisoidaan Grafana-ohjelmistoon perustuvan palvelun avulla. Grafana on avoimessa lähdekoodissa oleva datavisualisointi-työkalu [37]. Palvelulle luotiin ”docker-compose”-tiedosto (kuva 42).

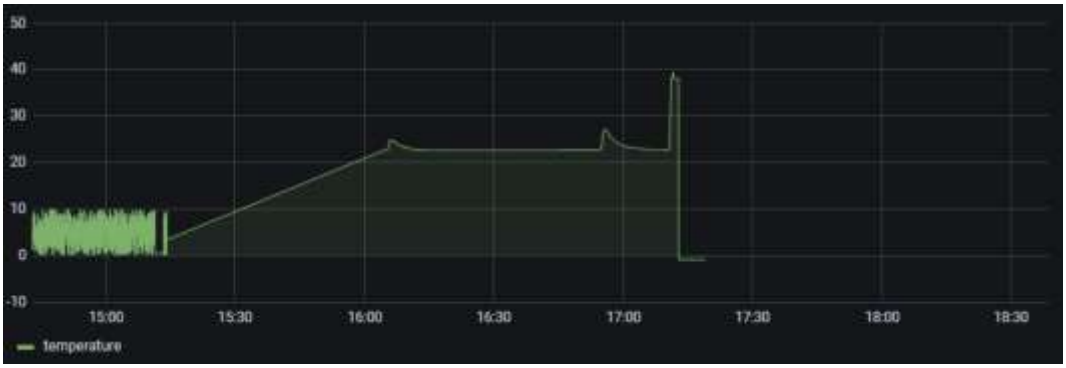
```
version: '3.8'

services:
  grafana:
    image:  grafana/grafana:7.4.3
    container_name: grafana
    ports:
      - 3000:3000
    networks:
      - sensor-network
    volumes:
      - ./data:/var/lib/grafana
    deploy:
      mode: replicated
      replicas: 1

networks:
  sensor-network:
    driver: overlay
    external: true
```

Kuva 42. Grafana-palvelun ”docker-compose” -tiedosto.

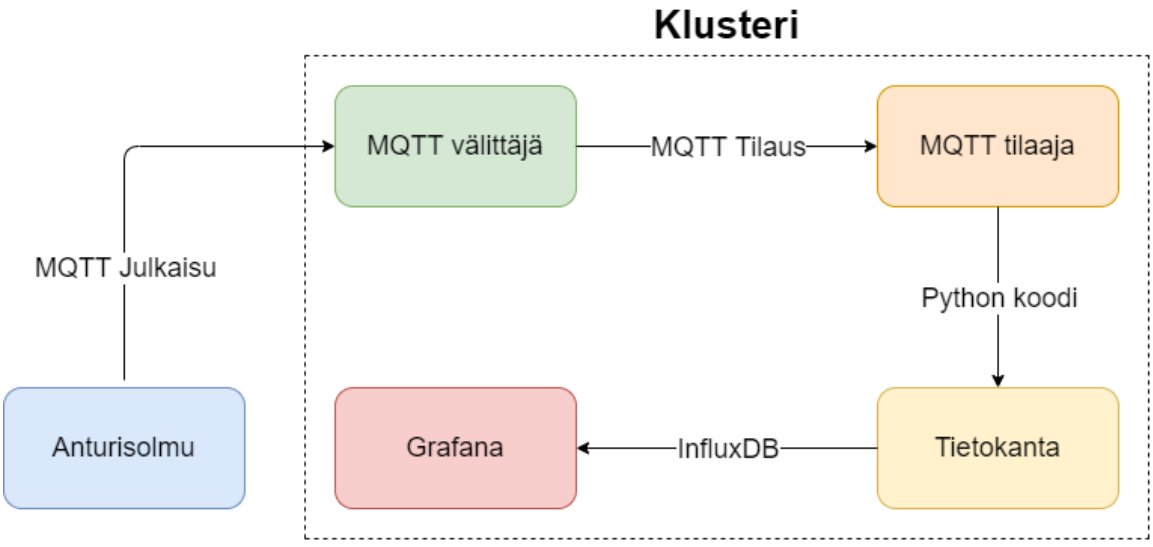
Kun palvelu oli käynnistynyt, sitä voitiin hallita graafisen käyttöliittymän kautta. Käyttöliittymään pääsee avaamalla selaimella parven solmun IP-osoitteen portin 8086. Käyttöliittymässä luotiin uusi InfluxDB ”Data source”, joka kertoo Grafana-ohjelmistolle mistä tietokannasta hakea anturidataa. Käyttöliittymässä luotiin myös uusi ”Dashboard”, joka on kokoelma paneeleita. Paneeli on yksittäisen arvon graafinen esitys. Jokaiselle anturisolmun lähettäneelle arvolle luotiin oma paneeli. Kuvassa 43 esitetään lämpötila-arvon paneeli.



Kuva 43. Lämpötila esitettynä Grafana-paneelissa.

4.3 Järjestelmän testaus

Järjestelmän toiminnallisuutta testattiin siirtämällä anturidataa anturisolmulta MQTT-välittäjäpalvelun kautta MQTT-tilaajapalvelulle. MQTT-tilaajapalvelu tallentaa anturidatan tietokantaan, josta Grafana-palvelu lukee kyseisen datan esitettäväksi graafisesti. Testauksessa huomioidaan siis kaikki klusterin palvelut, jotka tilaaja esitti vaatimusmäärittelyssään. Kuva 41 kuvaa testausprosessia.



Kuva 44. Järjestelmän testausprosessi.

Järjestelmän testausprosessi alkoi siirtämällä anturidataa anturisolmulta tietokantaan. Anturidata kirjoitettiin anturisolmulla Arduino IDE -ohjelmiston sarjaliikennemonitoriin (kuva 41).

```
18:09:21.209 -> .......IP Address:
18:09:25.028 -> 192.168.178.23
18:09:25.028 -> MQTT Client Started, Name == IOT-SENSOR-CLIENT
18:09:25.028 -> sensornodes/A/temperature | 23.40
18:09:25.028 -> sensornodes/A/humidity | 50.80
18:09:25.447 -> sensornodes/A/air_pressure | 98611.25
18:09:26.522 -> sensornodes/A/temperature | 23.40
18:09:26.522 -> sensornodes/A/humidity | 50.80
18:09:26.943 -> sensornodes/A/air_pressure | 98606.75
18:09:29.547 -> sensornodes/A/temperature | 23.40
18:09:29.547 -> sensornodes/A/humidity | 51.60
18:09:29.968 -> sensornodes/A/air_pressure | 98608.50
```

Kuva 45. Anturidata kirjoitettuna sarjaliikennemonitoriin.

Anturisolmu lähettää kaiken anturidatansa aiheeseen ”sensornodes/A/”. Anturidata välittyy anturisolmulta MQTT-välittäjäpalvelulle, joka Lähettää sen tilaajapalvelulle. Tilaajapalvelu vastaanotti anturidatan ja tallensi sen tietokantaan odotetusti. Kuvassa 42 esitetään InfluxDB tietokannan sisäistä dataa.

2021-03-22T16:12:00Z	23.46666666666667	temperature	sensor-node-A
2021-03-22T16:12:10Z	23.46666666666667	temperature	sensor-node-A
2021-03-22T16:12:20Z	23.5	temperature	sensor-node-A
2021-03-22T16:12:30Z	23.46666666666667	temperature	sensor-node-A
2021-03-22T16:12:40Z	23.43333333333334	temperature	sensor-node-A
2021-03-22T16:12:50Z	23.4	temperature	sensor-node-A
2021-03-22T16:13:00Z	23.366666666666664	temperature	sensor-node-A
2021-03-22T16:13:10Z	23.366666666666664	temperature	sensor-node-A
2021-03-22T16:13:20Z	23.325	temperature	sensor-node-A
2021-03-22T16:13:30Z	23.3	temperature	sensor-node-A
2021-03-22T16:13:40Z	23.3	temperature	sensor-node-A
2021-03-22T16:13:50Z	23.3	temperature	sensor-node-A
2021-03-22T16:14:00Z	23.3	temperature	sensor-node-A

Kuva 46. Anturidataa tallennettuna tietokantaan.

Kuvassa 42 esitetään vain anturidatan lämpötila-arvoja, koska tietokannan taulukko listaa tallennetun datan aakkosjärjestyksessä. Kaikkien kolmen anturien data saapui kuitenkin tietokantaan. Grafana-palvelussa luettiin dataa InfluxDB-tietokannasta ja esitettiin graafisesti. Grafana-palveluun luotiin jokaiselle anturiarvolle oma taulunsa, mutta kuvassa 43 esitetään vain lämpötila-arvojen graafinen esitys.

Kuva 47. Lämpötila-arvojen esitys graafisesti Grafana-palvelussa.

Järjestelmän testaus tuotti odotetut tulokset, jokainen osa järjestelmästä oli aiemmin testattu erillään muista järjestelmän osista. Opinnäytetyön käytännön kokonaisuus on siis testauksen pohjalta toimiva.

5 Pohdinta

Opinnäytetyön toteuttamisen aikana tapahtuvan covid-19-pandemian takia toteutusta rajattiin niin, että se voidaan toteuttaa etätyöskentelyn muodossa. Opinnäytetyön alkuperäinen tavoite oli toteuttaa tietoverkkojen opetusympäristö tieto- ja viestintätekniikan insinööriopetukseen. Tietoverkkojen opetusympäristö olisi tarjonnut insinööriopiskelijoille mahdollisuuden opiskella tietoverkkoihin liittyviä käsitteitä ja konsepteja täysin etäopiskelun muodossa. Alkuperäisessä konseptissa olisi ollut liikaa työtä, joten rajaus oli tarpeellinen ilman pandemiaakin. Käytännön osuudessa toteutettiin yksi anturisolmu ja yksi klusteri niin, että ne toimivat esimerkkiratkaisuna jatkokehittäjille tulevaisuudessa. Iot-kokonaisuus jakautui kahteen suureen osa-alueeseen, anturisolmun toteutukseen ja taustajärjestelmän eli klusterin toteutukseen.

Iot-anturisolmu haluttiin toteuttaa nopeasti verrattuna Iot-taustajärjestelmään. Toteutus pidettiin yksinkertaisena vaatimusmäärittelyn mukaan. Anturisolmun toteutuksessa ilmeni kuitenkin ongelmia, jotka estivät vaatimusmäärittelyn täyden toteutuksen. Anturisolmuun valitut komponentit toimivat koekytkennässä ilman ongelmia, mutta kun piirilevy suunniteltiin ja jyrsittiin, laitteisto ei enää toiminut. Ongelmien selvittämiseen käytettiin niin paljon aikaa, että lopulta päätettiin siirtyä toteuttamaan Iot-taustajärjestelmää. Anturisolmu oli kuitenkin toimiva, vaikka se olikin vain koekytkentä.

Iot-taustajärjestelmän toteutus oli anturisolmuun verrattuna haastavampi, mutta lopputulos oli kuitenkin lähempänä vaatimusmäärittelyä kuin anturisolmu. Lopputulos täyttää tilaajan vaatimusmäärittelyn vaatimukset. Vaikka klusterin toteutus olikin onnistunut, siinä on jonkin verran jatkokehitettävää, jos sitä halutaan hyödyntää kunnolla opetuksessa.

Yksi jatkokehityksen aihe on klusterin solmujen käyttöjärjestelmien asennus ja ylläpito. Käyttöjärjestelmien käynnistäminen USB-tikuilta toimi esimerkkiratkaisussa hyvin, koska solmuja oli rajattu määrä. Solmujen määrän kasvaessa ylläpidosta voi tulla haastavaa. Vaikka USB-tikut ovat kestävämpiä kuin SD-kortit, ne eivät silti ole ideaalisia pitkäaikaiseen käyttöön. Käyttöjärjestelmät voi kenties käynnistää verkon kautta, mutta siinäkin omat haasteensa. Toinen aihe on laitteiston prosessoriarkkitehtuuri, suurin osa Docker-kuvista on kehitetty toimimaan amd64 prosessoriarkkitehtuuri pohjaisilla laitteilla. Raspberry Pi-sarjan ARM-arkkitehtuuri rajaa vaihtoehtoja Docker-kuvia valitessa. Käyttöjärjestelmiä asennettaessa on tärkeää, että asennetaan 64-bittinen käyttöjärjestelmä. Tämä laajentaa käytettävien Docker-kuvien määrää. Kolmas jatkokehityksen aihe on

laitteiston jaettu muisti. Esimerkiksi tietokanta palvelua kehittäessä piti palvelun ajaminen rajoittaa yhdelle solmulle. Jos tietokantapalvelu saisi ajaa millä tahansa solmulla, sillä ei aina välttämättä ole pääsyä samoihin tietokannan tiedostoihin, koska ne ovat tallennettu toisen solmun muistiin. Yhteinen jaettu muisti auttaisi klusteria toimimaan sulavammin ja helpottaa palveluiden kehittämistä ja ajamista.

Jos klusterilla haluaa ajaa itsekehitettyä Docker-konttia, se tulee tallentaa tällä hetkellä Docker Hub -rekisteriin. Docker swarm ei voi ajaa lokaalisti rakennettuja Docker-kuvia. Jos klusteria halutaan käyttää laajemmassa mittakaavassa, on kannattavaa asentaa klusterille yksityinen rekisteri, johon tallentaa Docker-kuvat. Tällöin ei Docker kuvia tarvitse tallentaa julkiseen Docker Hub-rekisteriin, eikä niitä tarvitse hakea internetin kautta.

Opinnäytetyössä kehitettyä esimerkkiratkaisua klusterista voi ajatella hyvänä alkuna, jonka luoma pohja toimii hyvänä lähtöpisteenä jatkokehitykselle.

6 Yhteenveto

Opinnäytetyön tavoitteena oli luoda Kajaanin Ammattikorkeakoulun käytettäväksi esimerkkiratkaisu IoT-anturiverkosta. Työ jakautui kahteen osa-alueeseen, anturisolmun toteuttamiseen ja taustajärjestelmän toteuttamiseen.

Anturisolmun toteutus pidettiin tarkoituksella nopeana ja yksinkertaisena. Toteutuksen aikana ilmeni ongelmia anturisolmun asentamisessa piirilevylle. Anturisolmun tärkein osa eli anturien lukeminen ja langaton tiedonlähetys toimivat, vaikka toteutus jäi koekytkennäksi. Tiukka aikataulu opinnäytetyön loppusuoralla esti piirilevyn ongelmien ratkaisemisen.

Taustajärjestelmän osa-alue onnistui anturisolmuun verrattuna hyvin. Järjestelmä saatiin asennettua odotettua sujuvammin, jonka jälkeen sille voitiin kehittää sisäiset toiminnot suuremmalla aikataululla. Palveluiden kehityksessä ei ilmennyt ongelmia, joista ei voinut päästä yli. Tietokantapalvelu oli suurin työ, koska InfluxDB ei ollut tuttu tietokanta. MQTT-välittäjä ja -tilaaja -palveluiden kehityksessä ei ilmennyt suuria vaikeuksia. Visuaaliset palvelut olivat hyvin helppoja ottaa käyttöön.

Opinnäytetyön kirjoituksessa suurimmat haasteet olivat teorian kirjoitus ja käytännön osuudessa ilmenevien koodipätkien järkevän esitystavan löytäminen. Opinnäytetyö saatiin suoritettua loppuun asti sovitussa aikataulussa. Jos opinnäytetyön konseptia ei olisi rajattu, sovitun aikataulun pitäminen ei olisi ollut mahdollista. Aikataulussa oli tärkeä pysyä, koska opinnäytetyö on osa suurempaa tietoverkkojen opetusympäristöprojektia. Opinnäytetyö antaa hyvän perusteen, josta rakentaa toimivan tietoverkkojen opetusympäristön tulevaisuudessa.

Lähteet

(1) What is the OSI model? [Internet]. [Viitattu 2.11.2020]. (cloudflare.com). Saatavilla: https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/

(2) TCP/IP explained simply. [Internet]. [Viitattu 2.11.2020]. (ionos.com). Saatavilla: https://www.ionos.com/digitalguide/server/know-how/introduction-to-tcpip/

(3) TCP (Transmission Control Protocol) – The transmission protocol explained [Internet]. [Viitattu 19.3.2021]. (ionos.co.uk). Saatavilla: https://www.ionos.co.uk/digitalguide/server/know-how/introduction-to-tcp/

(4) What is an IP address? And what is your IP address? [Internet]. [Viitattu 10.2.2021]. (networkworld.com). Saatavilla: https://www.networkworld.com/article/3588315/what-is-an-ip-address-and-what-is-your-ip-address.html

(5) IT Explained: IP address. [Internet]. [Viitattu 11.2.2021]. (paessler.com). Saatavilla: https://www.paessler.com/it-explained/ip-address

(6) Ultimate Subnet Cheat Sheet and Subnetting Tutorial Guide. [Internet]. [Viitattu 11.2.2021]. (dnsstuff.com). Saatavilla: https://www.dnsstuff.com/subnet-ip-subnetting-guide#what-is-an-ip-address

(7) What is a Firewall? - Definition & Explanation. [Internet]. [Viitattu 12.2.2021]. (Kaspersky.com). Saatavilla: https://www.kaspersky.com/resource-center/definitions/firewall

(8) The 5 different types of firewalls explained. [Internet]. [Viitattu 12.2.2021]. (searchsecurity.techtarget.com). Saatavilla: https://searchsecurity.techtarget.com/feature/The-five-different-types-of-firewalls

(9) What is MQTT? A practical introduction. [Internet]. [Viitattu 12.2.2021]. (opc-router.com). Saatavilla: https://www.opc-router.com/what-is-mqtt/

(10) MQTT beginner's guide. [Internet]. [Viitattu 12.2.2021]. (u-blox.com). Saatavilla: https://www.u-blox.com/en/blogs/insights/mqtt-beginners-guide.

(11) What is Iot? [Internet]. [Viitattu 19.3.2021]. (oracle.com). Saatavilla: https://www.oracle.com/internet-of-things/what-is-iot/

(12) Margaret R. Microcontroller (MCU) [Internet]. [Viitattu 26.10.2020]. (internetofthingsagenda.techtarget.com). Saatavilla: https://internetofthingsagenda.techtarget.com/definition/microcontroller

(13) Kumar. Microcontroller Basics. [Internet]. [Viitattu 26.10.2020]. (codrey.com). Saatavilla: https://www.codrey.com/learn/microcontroller-basics/

(14) Kalnoskas A. A beginner's guide to microcontrollers. [Internet]. [Viitattu 26.10.2020]. (microcontrollertips.com). Saatavilla: https://www.microcontrollertips.com/a-beginners-guide-to-microcontrollers-faq/

(15) Different types of sensors. [Internet]. [Viitattu 26.10.2020]. (electronicshub.org). Saatavilla: https://www.electronicshub.org/different-types-sensors/#What_is_a_Sensor

(16) Chakraborty J. Analog Sensors Vs. Digital Sensors. [Internet]. [Viitattu 26.10.2020]. (iot4beginners.com). Saatavilla: https://iot4beginners.com/analog-sensors-vs-digital-sensors/

(17) Campbell S. Basics of the I2C communication protocol. [Internet]. [Viitattu 23.11.2020]. (circuitbasics.com). Saatavilla: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/.

(18) What is a Cluster? An Overview of Clustering in the Cloud [internet]. [Viitattu 19.2.2020]. (capitalalone.com). Saatavilla: https://www.capitalone.com/tech/cloud/what-is-a-cluster/

(19) Cluster Computing: Definition, Architecture, and Algorithms [Internet]. [Viitattu 19.2.2021]. (esds.co.in/blog). Saatavilla: https://www.esds.co.in/blog/cluster-computing-definition-architecture-and-algorithms/

(20) What is a Raspberry Pi? [Internet]. [Viitattu 26.2.2021]. (opensource.com). Saatavilla: https://opensource.com/resources/raspberry-pi

(21) Beginner's Guide: How to Get Started With Raspberry Pi [Internet]. [Viitattu 26.2.2021]. (uk.pcmag.com). Saatavilla: https://uk.pcmag.com/raspberry-pi-3-model-b-1/122354/beginners-guide-how-to-get-started-with-raspberry-pi

(22) Operating system images [Internet]. [Viitattu 26.2.2021]. (raspberrypi.org). Saatavilla: https://www.raspberrypi.org/software/operating-systems/

(23) CONTAINERS AT GOOGLE [Internet]. [Viitattu 19.2.2021]. (cloud.google.com). Saatavilla: https://cloud.google.com/containers

(24) What is a container? [Internet]. [Viitattu 28.9.2020]. (docker.com). Saatavilla: https://www.docker.com/resources/what-container

(25) What is Docker? The spark for the container revolution [Internet]. [Viitattu 28.9.2020]. (infoworld.com). Saatavilla: https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html

(26) Docker Explained – An Introductory Guide To Docker. [Internet]. [Viitattu 19.2.2021]. (edureka.co). Saatavilla: https://www.edureka.co/blog/docker-explained/#Docker%20Compose,%20Swarm

(27) Docker Hub Quickstart [Internet]. [Viitattu 22.2.2021]. (docs.docker.com). Saatavilla: https://docs.docker.com/docker-hub/

(28) Swarm mode key concepts [Internet]. [Viitattu 28.9.2020]. (docs.docker.com). Saatavilla: https://docs.docker.com/engine/swarm/key-concepts/

(29) Docker Swarm. [Internet]. [Viitattu 28.9.2020]. (docs.docker.com). Saatavilla: https://www.sumologic.com/glossary/docker-swarm/

(30) How nodes work. [Internet]. [Viitattu 22.2.2021]. (docs.docker.com). Saatavilla: https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/

(31) WiFi kit 8 [Internet]. [Viitattu 10.3.2021]. (heltec.org). Saatavilla: https://heltec.org/project/wifi-kit-8/

(32) Digital-output relative humidity & temperature sensor/module DHT22 [Internet]. [Viitattu 10.3.2021]. (sparkfun.com). Saatavilla: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

(33) CJMCU MPL3115A2 Atmospheric Pressure Altitude Sensor [Internet]. [Viitattu 10.3.2021]. (electropeak.com). Saatavilla: https://electropeak.com/mpl3115a2-pressure-sensor-1

(34) Raspberry Pi 3 Model B [Internet]. [Viitattu 1.3.2021]. (raspberrypi.org). Saatavilla: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

(35) Docker Swarm Visualizer [Internet]. [Viitattu 4.3.2021]. (github.com). Saatavilla: https://github.com/dockersamples/docker-swarm-visualizer

(36) Time series database (TSDB) explained [Internet]. [Viitattu 8.3.2021]. (influxdata.com). Saatavilla: https://www.influxdata.com/time-series-database/

(37) Grafana [Internet]. [Viitattu 22.3.2021]. (grafana.com). Saatavilla: https://grafana.com/oss/grafana/