Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Sami Häkkinen

# MQTT-protokollan soveltuvuus testiautomaatiojärjestelmän tiedonsiirtoon

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikan tutkinto-ohjelma

Insinöörityö

12.04.2021

| | |
|---|---|
| Tekijä<br>Otsikko<br><br>Sivumäärä<br>Aika | Sami Häkkinen<br>MQTT-protokollan soveltuvuus testiautomaatiojärjestelmän tiedonsiirtoon<br>59 sivua + 1 liite<br>12.04.2021 |
| Tutkinto | insinööri (AMK) |
| Tutkinto-ohjelma | Sähkö- ja automaatiotekniikan tutkinto-ohjelma |
| Ammatillinen pääaine | Automaatiotekniikka |
| Ohjaajat | lehtori Timo Tuominen<br>senior engineer Tomi Mäentausta |

Tämän insinöörityön tarkoituksena oli tehdä selvitystyö koskien MQTT-protokollan soveltuvuutta kohdeyrityksen luotettavuuslaboratorion testiautomaatiojärjestelmän tiedonsiirtomenetelmäksi. Työn tavoitteena oli saada ymmärrys siitä, mitä nykyisen ADS-protokollan korvaaminen vaatisi, mitä MQTT-protokollan avulla olisi saavutettavissa ja mitkä protokollat ominaisuudet ovat testiautomaatiojärjestelmän tiedonsiirron osalta merkittävimpiä. Selvitystyön lisäksi tarkoituksena oli saada tehtyä käytännön osuutena tiedonsiirtoa koskeva kevyt vaatimusmäärittely sekä luoda pohja testiautomaatiojärjestelmän tiedonsiirtoprotokollan kehitysympäristölle, jonka avulla voitaisiin suorittaa tulevaisuuden protokehitystä.

Työ aloitettiin tutustumalla MQTT-protokollaan liittyvään materiaaliin. Tietoa kerättiin useista eri lähteistä ja materiaalista valittiin vain testiautomaatiojärjestelmän kannalta oleellisimmat. Työ jatkui eri MQTT-malliratkaisuihin tutustumalla. Näiden työvaiheiden pohjalta lähdettiin toteuttamaan kevyttä vaatimusmäärittelyä testiautomaatiojärjestelmälle ja tiedonsiirron kehitysympäristöä.

Lopputuloksena saatiin luotua tiedonsiirtoa koskeva vaatimusmäärittely sekä kehitettävälle testiautomaatiojärjestelmälle tiedonsiirron kehitysympäristö. Lisäksi työssä saatiin määritettyä toimenpiteet tulevaisuuden kehitystyölle. Tutkimustyön tuloksena MQTT-protokollan todettiin sopivan kohdeyrityksen testiautomaatiojärjestelmälle, sillä se täyttää järjestelmän tiedonsiirrolle asetetun vaatimusmäärittelyn, jonka lisäksi protokolla tarjoaa kattavasti hyödyllisiä ominaisuuksia, jotka listattiin työssä. MQTT toimii ADS-protokollan rinnalla hyvin, ja MQTT hyödyntää samoja komponentteja. MQTT:n käyttöönottamiseksi tarvitaan viestinvälittäjäsovellus sekä asiakassovellus tiedonsiirtoa käyttäville laitteille.

Tutkimustyössä kerättyä tietoa tullaan hyödyntämään kohdeyrityksen uusiutuvan testiautomaatiojärjestelmän tiedonsiirtoa koskevassa päätöksenteossa. Käytännön työssä luotua MQTT-testiympäristöä ja MQTT-aliohjelmaa tullaan hyödyntämään uusiutuvan testiautomaatiojärjestelmän kehitystyössä. Työssä esitettyä MQTT-protokollan toiminnan kuvausta sekä ominaisuuksien ja rajoitteiden listoja voidaan hyödyntää laajasti eri käyttökohteissa, joissa harkitaan MQTT-protokollan käyttämistä tiedonsiirtomenetelmänä.

| | |
|---|---|
| Avainsanat | MQTT, testiautomaatio, tiedonsiirto, IoT-protokolla |

| | |
|---|---|
| Author | Sami Häkkinen |
| Title | Suitability of MQTT protocol for test automation system data transfer |
| Number of Pages | 59 pages + 1 appendix |
| Date | 12 April 2021 |
| Degree | Bachelor of Engineering |
| Degree Programme | Degree Programme in Electrical and Automation Engineering |
| Professional Major | Automation Engineering |
| Instructors | Timo Tuominen, Senior Lecturer<br>Tomi Mäentausta, Senior Engineer |

The purpose of this thesis was to investigate the suitability of the MQTT protocol as a data transfer method for a test automation system in a target company's reliability laboratory. The research aim was to gain an understanding of what it would take to replace the current ADS protocol, what could be achieved with the MQTT protocol, and which protocol features are most significant for data transfer in a test automation system. In addition to the research, the aim was to make a preliminary requirement specification for data transfer and to create a basis for the development environment for the data transfer protocol of the test automation system, which could be used for future prototype development.

The work was started by getting acquainted with the material related to the MQTT protocol. Data were collected from several different sources and only the most relevant material for the test automation system was selected. The work continued by getting acquainted with different MQTT software solutions. Based on these work steps, a preliminary requirement specification for the test automation system data transfer and a development environment were started.

As a result of thesis requirement specification for data transfer and a development environment for data transfer of the test automation system was obtained. In addition, the steps for future development work were defined in the thesis. As a result of the research work, the MQTT protocol was found to be suitable for the automation system of the target company, as it meets the requirements set for the data transfer of the system, in addition to which the protocol offers useful features that are listed in thesis. MQTT works well alongside the ADS protocol and MQTT uses the same components as ADS. To implement MQTT, a broker software and an MQTT client software is needed.

The information and knowledge gathered in the research will be utilized in decision-making related to data transfer of the target company's test automation system. Created MQTT test environment and the MQTT subroutine will be utilized in the development of a new test automation system. The functional description of the MQTT protocol, and the lists of properties and constraints presented in thesis can be utilized extensively in various applications where the use of the MQTT protocol as a data transfer method is considered.

| | |
|---|---|
| Keywords | MQTT, test automation, data transfer, IoT protocols |

## Sisällys

## Lyhenteet

| | |
|---|---|
| .conf | Configuration file tai "config" file. Unix- ja Linux-järjestelmien käyttämä sovellusten asetustiedosto. |
| ADS | The Automation Device Specification protocol. Beckhoff Automationin kehittämä tiedonsiirtomenetelmä TwinCAT-järjestelmille. |
| Broker | Viestinvälittäjä. MQTT-protokollan viestinvälityksestä vastaava sovellus. |
| CA | Certificate authority. Salausvarmenteita myöntävä yhteisö. |
| Client | Asiakas. Tiedonsiirrossa palvelua käyttävä tietokone. |
| I/O | Input/Output. Ohjelmoitavien logiikoiden tulot ja lähdöt. |
| IoT | Internet of Things. Asioiden tai esineiden internet. |
| MQTT | Message Queue Telemetry Transport. Avoimeen lähdekoodin perustuva julkaisija/tilaaja-mallin mukainen tiedonsiirtoprotokolla. |
| OASIS | Organization for the Advancement of Structured Information Standards. Muun muassa avoimeen lähdekoodin ratkaisuille standardeja määrittävä virallinen yhteisö. |
| PLC | Programmable logic controller. Ohjelmoitava logiikka. |
| SQL | Structured Query Language. IBM:n kehittämä standardoitu ohjelmointi-/kyselykieli. |
| SSL | Secure Socket Layer. Sovellusten tiedonsiirtoon käytetty salausprotokolla. |
| TCP/IP | Transmission Control Protocol / Internet Protocol. Tiedonsiirrossa käytettävä protokollien pino. |

| | |
|---|---|
| TLS | Transport Layer Security. Sovellusten tiedonsiirtoon käytettävä salausprotokolla. |
| TwinCAT | Twin Control Automation Technology. Beckhoff Automationin kehittämä ohjelmointiympäristö. |

# 1 Johdanto

Tässä insinöörityössä esitetään Message Queuing Telemetry Transport -protokollan (MQTT, MQTT-protokolla) soveltuvuutta kohdeyrityksen testiautomaatiojärjestelmään koskeva selvitystyö, sekä käytännön osuutena tehdyt tiedonsiirron kevyt vaatimusmäärittely ja kehitysympäristön luominen. Työn tarpeen pohjalla on Esa Salmisen insinöörityö Luotettavuuslaboratorion tieto- ja testiautomaatiojärjestelmän vaatimusmäärittely vuodelta 2015 sekä sen jälkeen jatkettu kehitystyö koskien vaatimusmäärittelyä uudelle testiautomaatiojärjestelmälle [1]. Tämä insinöörityö on uuden testausjärjestelmän kehitystyön vaihe, jossa paneudutaan testiautomaatiojärjestelmän tiedonsiirtoon erityisesti MQTT-protokollan soveltuvuutta koskevasta näkökulmasta. Kohdeyritys on kiinnostunut tiedonsiirron vaikutuksista sekä sen suomista mahdollisuuksista testausjärjestelmään, sillä tiedonsiirto on merkittävä osa toimivaa testiautomaatiojärjestelmää sekä sen ominaisuuksia. Tiedonsiirrossa käytettävä protokolla ei saisi rajata testiautomaatiojärjestelmälle asetettuja vaatimuksia, vaan parhaimmillaan valittu protokolla jopa tukisi järjestelmälle asetettuja vaatimuksia.

Insinöörityön kohdeyritys toimii teollisuuden alalla kansainvälisenä tuotevalmistajana. Insinöörityön tarkempana kohteena on kohdeyrityksen luotettavuus- ja testausyksikkö, jonka tehtäviin kuuluu tuotetestaus. Tuotetestaus on jakautunut globaalisti kohdeyrityksen eri toimipisteille. Pääasiallisesti kaikki tuotetestaus toteutetaan automaatiolla ohjatuilla testereillä eli automaatiotestauksena. Tyypillinen testeri koostuu päävirtapiiristä, ohjelmoitavasta logiikasta ja I/O-moduuleista. Tuotetestauksesta vastaa luotettavuuslaboratorio, jonka tarkoituksena on määrittää testaamalla tuotteiden ja sitä kautta niissä käytettävien komponenttien luotettavuutta. Tuotetestauksen alle kuuluu kokonaisten markkinoille vietävien tuotteiden sekä niissä käytettävien sähköisten ja mekaanisten komponenttien elinikätestien, toiminnallisuustestien, materiaalitutkimusten suunnittelu ja suorittaminen. Lisäksi laboratorion toimintoihin kuuluu testien etenemisen tarkkailu, testidatan analysointi ja testitulosten raportointi asiakkaille.

Insinöörityön tavoitteena on selvittää, soveltuuko MQTT-protokolla kohdeyrityksen uudistuvaan testiautomaatiojärjestelmään. Nykyään yrityksen käytössä oleva tiedonsiirtoprotokolla on Beckhoffin The Automation Device Specification -protokolla (ADS-

protokolla), jonka korvaaminen yleisesti käytössä olevalla IoT-protokolalla kiinnostaa kohdeyritystä. Insinöörityön tehtävänä on selvittää, mitä nykyisen ADS-protokollan vaihtaminen MQTT-protokollaan vaatisi ja mitkä olisivat saavutettavat hyödyt.

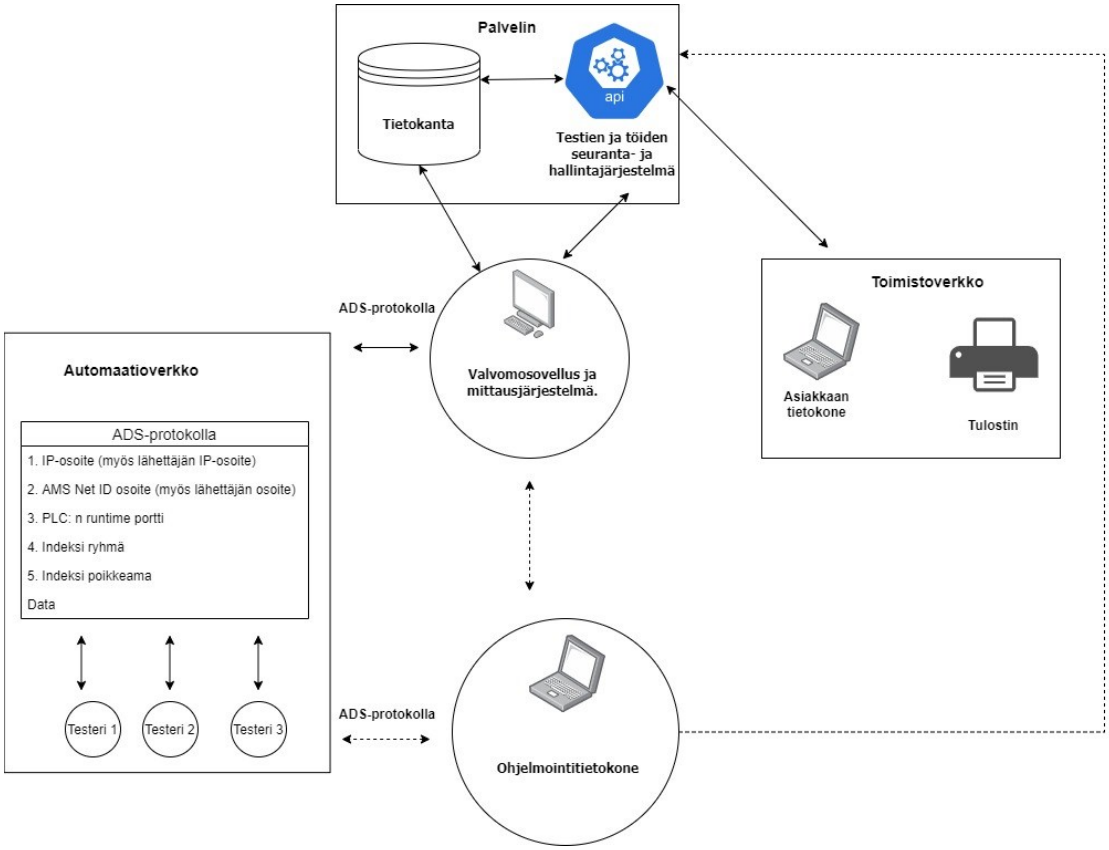
Käytännön osuutena MQTT-protokollaa koskeva selvitys tapahtuu tuottamalla vaatimusmäärittely, mikä koskee testiautomaatiojärjestelmän tiedonsiirtoprotokollaa sekä kehitysympäristö testausautomaatiojärjestelmää varten, jossa on käytössä MQTT-protokolla. Kehitysympäristön avulla voidaan tulevaisuudessa demonstroida protokollan avulla saavutettuja hyötyjä sekä suorittaa tiedonsiirron kehitystyön vaiheita kokeilualustalla ennen muutosten implementointia käytössä olevaan järjestelmään. Kokeilualustan on tarkoitus helpottaa protokollaa koskevaa testausta. Tässä insinöörityössä tuotettu vaatimusmäärittely ei saa olla ristiriidassa testiautomaatiojärjestelmää koskevan vaatimusmäärittelyn kanssa.

Toteutuskohde on teollisuusalan automaatiojärjestelmä, mikä on usein hyvinkin kriittistä tiedonsiirron nopeutta ajatellessa. Tästä asiakasyrityksen tuotetestaus on poikkeus, sillä nykyinen testiautomaatiojärjestelmä käyttää jaksottaista tiedonsiirtoa, eli dataa haetaan ohjelmoitavilta logiikoilta ja tallennetaan tietokantaan määrätyin väliajoin reaaliaikaisen tallentamisen sijaan. Tiedonsiirron nopeuden osalta MQTT täyttää nopeusvaatimukset, sillä se on itsessään hyvin kevyt/nopea kommunikointiprotokolla, minkä vuoksi insinöörityössä ei perehdytä MQTT:n vaikutukseen tiedonsiirtonopeudessa. Tiedonsiirron prototestausta nopeuden osalta kannattaa tehdä vasta, kun järjestelmä vastaa laajuudeltaan ja käytettäviltä komponenteiltaan todellista toteutusta, koska tiedonsiirron nopeuteen vaikuttaa merkittävästi pakettikoot, yhteysnopeus sekä viestinvälittäjän nopeus.

## 2 Luotettavuustestaus ja testiautomaatiojärjestelmä

Tuotesuunnittelussa luotettavuudella tarkoitetaan todennäköisyyttä, jolla laite toimii suunnitellulla tavalla ja ilman vikaantumisia, kun laitetta käytetään sille suunnitellulla tavalla sekä suunnitellun kaltaisessa ympäristössä [2, s. 1]. Luotettavuustestauksella tarkoitetaan tyypillisesti analysointimenetelmää, jonka tarkoituksena on määrittää testattavan kohteen luotettavuutta. Kohdeyrityksessä luotettavuustestaus jakautuu valmiiden tuotekokonaisuuksien ja niissä käytettävien komponenttien kesken.

Tuotetestauksen järjestelmä koostu kahdesta pääosasta: testien hallintaan tarkoitetusta käyttöliittymästä eli valvomosovelluksesta sekä selainpohjaisesta työnseuranta- ja hallintajärjestelmästä. Kuvassa 1 on esitetty tuotetestausjärjestelmä kokonaisuudessaan. Lisäksi kuvassa on esitetty ADS-protokollan viestirakennetta ja sen vaatimia tietoja yhteyden muodostamiseksi ennen varsinaiseen dataan pääsyä. Yhtenäisellä viivalla kuvataan tiedon siirron normaalia kulkureittiä ja katkonaisella viivalla on esitetty vain normaalista tilanteesta poikkeava tiedonsiirto, esimerkiksi vianselvityksessä. Työn kuvat 1, 2, 5, 8–15 ja 31 on piirretty maksuttomalla Diagrams.net-sivuston selainpohjaisella Draw.io-kaaviotyökalulla.




Kuva 1. Käytössä olevan tuotetestausjärjestelmän kuvaus [1, s.34].

Testiautomaatiojärjestelmä itsessään jakautuu kahteen osioon testien ohjaamiseen ja valvontaan tarkoitettuun käyttöliittymään eli valvomosovellukseen sekä erilliseen mittausjärjestelmään. Koko testiautomaatiotestausjärjestelmän tuottama data tallennetaan Structured Query Language -tietokantaan (SQL-tietokantaan). Datan tallentamisesta tietokantaan vastaa valvomokone, joka on yhdistetty automaatio- ja toimistoverkkoon.

Valvomotietokone hakee tallennettavan datan ohjelmoitavalta logiikalta Beckhoffin ADS-protokollaa käyttäen. Automaatioverkko toimii sille varatussa erillisessä Ethernet-verkossa toimistoverkosta erotettuna.

## 2.1 Valvomosovellus

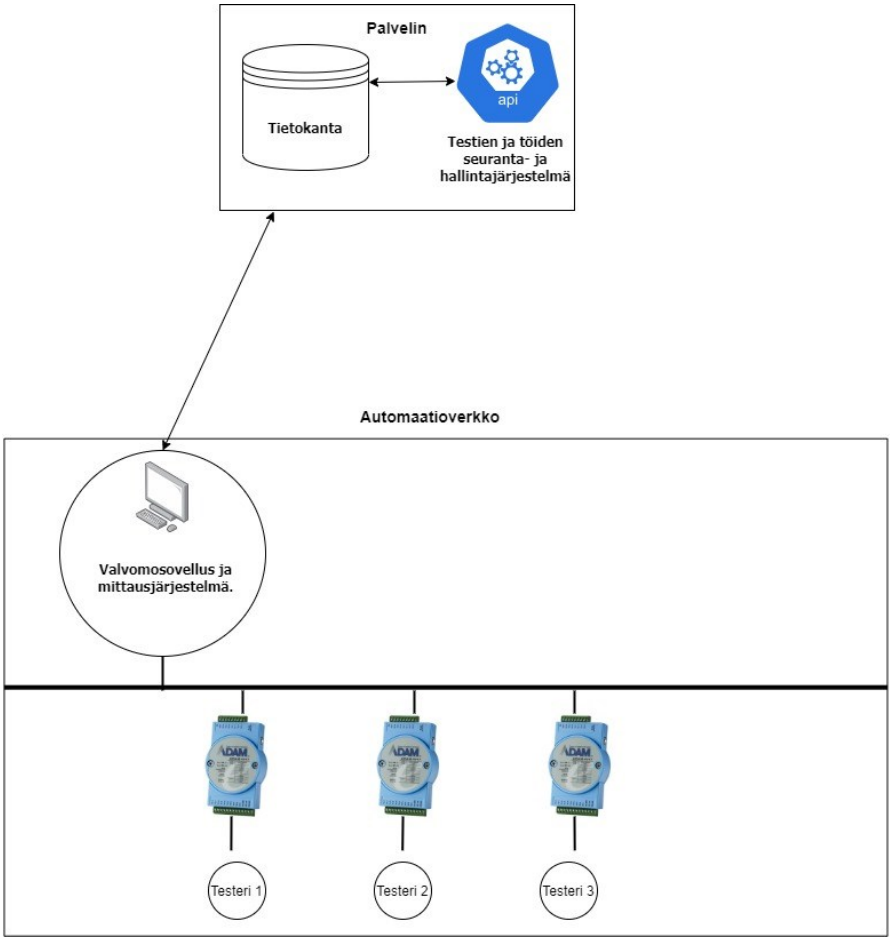
Nykyinen testiautomaatiojärjestelmä on Visual Basic .NET -kielinen valvomosovellus. Testien hallintaan tarkoitetulla käyttöliittymässä testimekaanikko voi asettaa testisyklejä, joiden perusteella ohjelmoitava logiikka (programmable logic controller, PLC) ohjaa testattavia tuotteita tai komponentteja. Käyttöliittymä mahdollistaa laboratorioteknikoiden luoda PLC-koodiin ajosyklejä ilman, että heidän tarvitsee käsitellä suoraan PLC-koodia tai ymmärtää ohjelmointia. Testipaikkakohtaisesti asetetut testisyklit koostuvat yksittäisistä testin ohjaukseen liittyvistä funktioista sekä funktioiden voimassaoloajoista. Käytännössä yksittäinen funktio ja sille asetettu aika määrittää, mitä PLC-koodin muuttujan arvoa muutetaan ja kuinka kauan asetettu arvon muutos on voimassa koodiin luodussa testisyklipohjassa. [1, s.11–12.]

Muita käyttöliittymällä asetettavia toimintoja ovat testipaikkakohtaisten sykleissä suoritettavat mittaukset ja niiden perusteella määritettävät vikaantumisehdot. Vikaantumisehdon täyttyminen pysäyttää vikaantuneen testipaikan ajon ja kirjaa siitä automaattikommentin, jossa ilmenee vian syy. Testerin tyypin mukaan vika pitää kuitata joko manuaalisesti, tai testipaikka käynnistetään automaattisesti määritetyn ajan kuluttua. Testien käynnissä olon aikana järjestelmään voidaan testipaikkakohtaisesti kirjata kommentteja tehdyistä huoltotoimenpiteistä, vikojen syiden selvittämisestä tai muista testiin liittyvistä asioista. Kaikista valvomosovelluksen käyttäjän tekemistä muutoksista testisykliin, kommenteista ja vikaantumisista tehdään aikaleimalla varustettu merkintä testeriä ja tiettyä testipaikkaa koskevaan lokiin. [1, s.12.]

## 2.2 Mittausjärjestelmä

Kohdeyrityksen käytössä oleva mittausjärjestelmä koostuu yrityksen itse kehittämästä mittausten ohjausjärjestelmästä sekä automaatiolle varattuun Ethernet-verkkoon liitetyistä Advantechin valmistamista ADAM Ethernet I/O-moduuleista. Mittausjärjestelmä on

erillinen ohjelmisto eikä ole yhteydessä valvomosovellukseen. ADAM-moduulit ovat kalibroituja mittalaitteita, joten niitä voidaan käyttää virallisissa mittauksissa. Moduuleita löytyy niin analogisina kuin digitaalisina. Lisäksi tarvittavien kanavien määrään voidaan vaikuttaa mallikohtaisesti. ADAM I/O -moduulit käyttävät tiedonsiirtoon Modbus/TCP-protokollaa, ja ne voidaan kytkeä suoraan kohdeyrityksen erotettuun automaatio Ethernet -verkkoon. Kuvassa 2 esitetyn verkkoarkkitehtuurin mukaisesti I/O-moduuleita voidaan konfiguroida sekä lukea moduulien lukemaa dataa antureilta. [1, s.35–36.]




Kuva 2. Mittausjärjestelmä [3; 4].

## 2.3 Testien ja töiden seuranta- ja hallintajärjestelmä

Selainpohjaisen testi- ja työnseurantajärjestelmän avulla projektitiimit voivat asettaa ja aikatauluttaa testeihin liittyviä työtehtäviä. Luotettavuuslaboratorion työntekijät sekä

kohdeyrityksen sisäiset asiakkaat voivat seurata tilattujen testien etenemistä, kuten testisyklien määrää, ajotunteja, testitekniikoiden kirjoittamia kommentteja huoltotoimenpiteistä sekä vikaantumisia. Järjestelmässä on erilaisia resurssienhallintatoimintoja, joiden avulla voidaan seurata ja hallita testereiden ja testipaikkojen varauksia sekä luotettavuuslaboratorion työntekijöiden käynnissä olevia projekteja. Lisäksi järjestelmän on tarkoitus toimia tehtyjen testien ja kerätyn testidatan kirjastona, jossa kerättyä dataa ja niistä tehtyjä päätelmiä voi testien aikana sekä vielä vuosien kuluttua hakea ja tarkastella.

## 2.4 Tyypillinen testeri

Tuotetestauksessa käytettävä testeri suorittaa ohjauksen ohjelmoitavan logiikan ja I/O-väylän avulla. I/O-väylän koko ja muut komponentit vaihtelevat testereittäin. Siihen vaikuttavia asioita ovat testipaikkojen määrä, testityyppi sekä se, mitä testattavasta tuotteesta halutaan valvoa tai ohjata. Testien ohjaus tapahtuu testerikohtaisesti, joko testerissä olevien painikkeiden tai Human-Machine Interface -paneelin (HMI) kautta [1, s.37]. Osa testeistä voidaan käynnistää ja pysäyttää suoraan valvomosovelluksen kautta. Testereissä käytetään pääasiallisesti Beckhoffin tuotteita. Kaikissa testereissä käytetään lähtökohtaisesti samaa PLC-koodipohjaa. Pohjakoodi koostuu PLC-koodeille tyypilliseen tapaan yhdestä pääohjelmasta sekä useista aliohjelmista.

## 2.5 Nykyinen testiautomaatiojärjestelmä

Nykyinen testiautomaatiotestausjärjestelmä on ollut käytössä perusrakenteeltaan jo vuodesta 2011 alkaen, ja se onkin toimivuutensa ja toimintavarmuutensa puolesta vastannut kohdeyrityksen tarpeita todella hyvin. Jatkuva kehitystyö järjestelmän parissa on tuonut uusia ominaisuuksia järjestelmään sekä kehittänyt sen vikasietoisuutta. Lisäksi protokolla on yksinkertaisen rakenteensa vuoksi todella nopea.

Nykyisen testiautomaatiotestausjärjestelmän huonoja puolia ovat kuitenkin heikko skaalautuvuus, yksittäisten ominaisuuksien lisäämisen vaikeus ja käytössä olevien ominaisuuksien muuttamisen hankaluus. Käytössä oleva ADS-protokollalla toteutettu tiedonsiirto vaatii Ethernet-verkon sekä valvomosovelluksen käyttöliittymän PLC:n välille,

minkä vuoksi mobiililaitteilla testisyklien asettaminen testereihin ei suoraan onnistu, vaan syklit voidaan asettaa ainoastaan valvomotietokoneelta.

### 2.6 Asiakas-palvelin-malli

Palvelin on tietoliikennettä koskeva termi, jolla tarkoitetaan palvelinohjelmiston ja sitä suorittavan tietokoneen yhdistelmää. Palvelintietokoneella suoritettavien palvelinohjelmiston tarkoituksena on jakaa haluttua palvelua ja sisältöä palvelinta käyttäville tietokoneille, joita kutsutaan asiakkaiksi. Tyypillisiä käyttökohteita palvelimelle ovat muun muassa datan tallennus, selaimien käyttöliittymät ja sovellukset sekä ohjelmistot. Yksi palvelin pystyy usein tarjoamaan useita eri palveluita, joiden käyttäjinä voi olla useita asiakkaita samanaikaisesti. Palvelin ja sen asiakkaat voivat olla joko virtuaalisia tai fyysisiä. Virtuaalisella tarkoitetaan tietokoneohjelmistoa, jota voidaan suorittaa esimerkiksi pilvessä ja fyysisellä kaikkea yksittäisestä tietokoneesta aina isoihin palvelinsaleihin asti. [5.]

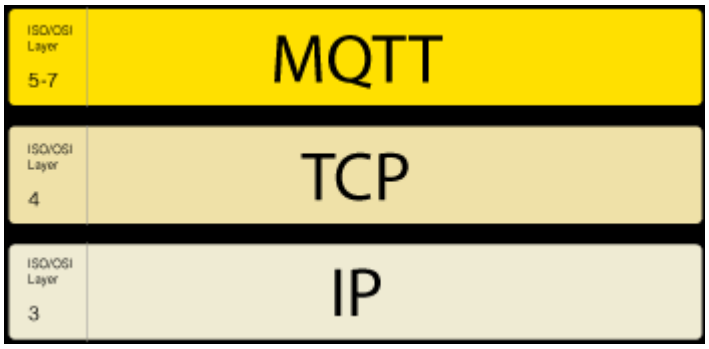
Tätä yleensä verkon ylitse tapahtuvaa asiakkaan ja palvelimen välistä tiedonsiirtomenetelmää kutsutaan asiakas-palvelin-malliksi (Client-Server model). Verkon ylitse tapahtuva tiedonsiirto mahdollistaa ohjelmistojen ja datan hajauttamisen ympäri maailmaa, eikä asiakkaan ja palvelimen tarvitse olla fyysisesti lähekkäin, mikä mahdollistaa saman datan ja ohjelmistojen reaaliaikaisen käytön eri asiakkaiden kesken ympäri maailmaa. Mallin ominaispiirteeseen kuuluu tiedonsiirron kevyenäpitäminen, mikä toteutetaan siten, että asiakkaille tarjotaan usein vain heidän tilaamansa tiedot, esimerkiksi jokin tietty selainnäkymä kerrallaan. [6.]

## 3 MQTT-protokollatutkimustyö

Tässä kirjallisessa tutkimustyössä käydään läpi MQTT-protokollan ominaisuuksia yleisesti, mutta käytännön esimerkkejä annetaan erityisesti käytännön osuuteen valitun Mosquitto-viestinvälittäjän ominaisuuksista. Mosquitto valittiin työn käytännön osuuteen avoimen lähdekoodin, Beckhoffin käytännön esimerkkien sekä suuren suosion vuoksi. Tutkimustyö koskee MQTT-protokollan versioita 3.1.1 ja 5 sekä Mosquitto-viestinvälittäjän versiota 1.6.8, joka 1.6.8 julkaistiin 28.11.2019.

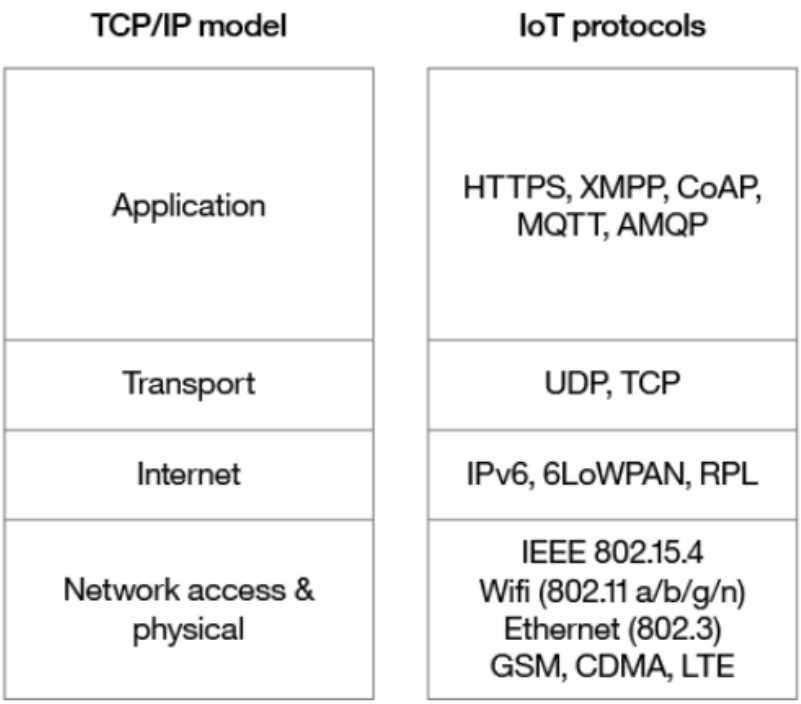
### 3.1 MQTT-yleisesti

MQTT on avoimeen lähdekoodin perustuva julkaisija/tilaaja-mallin mukainen tiedonsiirtoprotokolla, joka esitellään tarkemmin luvussa 3.2. Protokolla toimii tietoliikenne protokolla yhdistelmän Transmission Control Protocol/Internet Protocol -tiedonsiirtoprotokollan (TCP/IP) päällä kuvan 3 mukaisesti. TCP/IP-protokolla vastaa MQTT:n pakettien vastaanottamisen järjestyksessä, virheen korjauksista sekä vakauttaa protokollan toimintaa. Kuvan 3 vasemmassa reunassa on ilmoitettu protokollien sijoittuminen Open Systems Interconnection Reference Model (OSI-malli) eri kerroksille. [7, s.10–11; 8.]




Kuva 3. MQTT toimii OSI-mallin kerroksilla 5—7 [9].

Kuvassa 4 on esitetty, miten IoT-protokollat ja niiden käyttämät verkkoarkkitehtuurin ominaisuudet sijoittuvat TCP/IP-mallin eri tasoille.



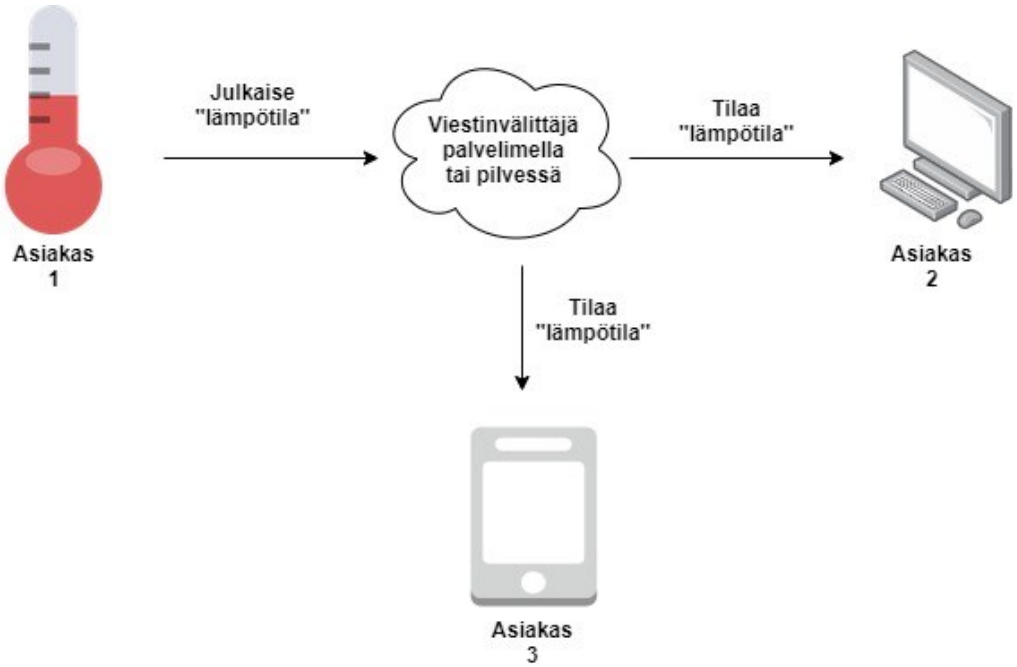

Kuva 4. MQTT toimii TCP/IP-tietoverkkoprotokolla yhdistelmän sovellustasolla [10].

MQTT-protokollan ominaisuuksia on erittäin kevyt sekä yksinkertainen rakenne. Lisäksi protokolla on koodijalanjäljeltään erittäin pieni. Protokollan viestinkuljetus on otsikko perusteista. Viestinkuljetus on agnostinen viestin sisältämän hyötykuorman suhteen, ja viestinkuljetuksen yhtenä ominaisuutena on tarpeettomien viestien poistaminen. Tarpeettomien viestien poistamista voidaan kuitenkin rajoittaa luvuissa 4.8 ja 4.9 esitetyin ominaisuuksien avulla. MQTT tarjoaa myös kattavan valikoiman erilaisia tietoturvaominaisuuksia sekä skaalausta entisestään helpottavia ominaisuuksia. Edellä mainitut ominaisuudet tekevät siitä erittäin suositun IoT-protokollan. Protokollan ominaisuudet ovat tarpeellisia käyttökohteissa, joissa kaistan leveyden ja protokollaan liitettyjen laitteiden resurssivaatimuksien minimointi on tärkeää. Tällaisia käyttöympäristöjä ovat muun muassa kommunikointi koneelta koneelle (M2M) ja esineiden Internet (Internet of Things, IoT). [11; 12; 13.]

MQTT-termiä ei varsinaisesti pidetä enää lyhenteenä vaan protokollan nimenä. Nimensä protokolla on saanut sanoista MQ Telemetry Transport. Termi MQ viittaa IBM:n MQ-väliohjelmistotuoteperheen tuotteisiin, joiden päälle MQTT rakennettiin. Silti MQTT-protokollaa käsitellessä käytetään usein termiä Message Queuing Telemetry Transport. [11.]

### 3.2 MQTT ja julkaisija-tilaaja-malli

MQTT:n käyttämä julkaisija-tilaaja-malli on hieman perinteisen asiakas-palvelin-mallin tapainen tiedonsiirtomenetelmä, jonka keskeisessä osassa on viestinvälittäjä sekä asiakkaat. Näiden kahden eri tiedonsiirtomenetelmien verkkoarkkitehtuureja nopeasti tarkasteltuna ne saattavat näyttää hyvin samanlaisilta. Käytännössä niiden välinen ero on kuitenkin merkittävä, josta merkittävimpänä on se, ettei MQTT:n käyttämä viestinvälittäjä käsittele viestien sisältämää dataa, eikä välittäjä myöskään säilytä dataa oletusasetuksilla. Tärkeä ero on myös se, että viestin julkaisija ja tilaaja eivät ole toisiinsa yhteydessä, vaan viestin välityksen näiden asiakkaiden välillä suorittaa viestinvälittäjä. Tämän vuoksi asiakkaat eivät ole tietoisia toistensa olemassaolosta, toiminnasta, IP-osoitteista tai porteista. Asiakkaiden ei tarvitse toimia yhtäaikaisesti, eivätkä ne ole riippuvaisia toisistaan viestien lähetys- ja vastaanottohetkillä. Viestien välityksen onnistumiseksi tilaajan ja vastaanottajan tarvitsee tietää ainoastaan viestivälittäjän IP-osoite, koska viestinvälittäjä vastaa viestien jakamisesta asiakkaille. [14.] Kuvassa 5 on esitetty MQTT-protokollan viestinvälitys yksinkertaisella mallilla, jossa voidaan nähdä protokollan hyödyntämä ryhmälähetys (multicast).




Kuva 5. Tyypillinen julkaisija/tilaaja-malli, jossa yhdellä aiheella on useita tilaajia.

Malli ei MQTT:n tapauksessa rajoita viestin vastaanottajien tai julkaisijoiden määrää. Yhdellä julkaisijalla voi olla useita tilaajia ja usealla julkaisijalla voi olla vain yksi asiakas. Viestien julkaisu ja tilaaminen on otsikkopohjaista, mitä esitellään tarkemmin alaluvuissa 3.7 ja 3.8. [14.]

## 3.3 MQTT-historia

MQTT-protokollan kehittämisen aloittivat kaksi insinööriä tohtori Andy Stanford-Clark ja Arlen Nipper vuonna 1999. Standord-Clark työskenteli tuolloin IBM:n alaisuudessa ja Nipper Eurotechin alaisuudessa. Protokolla kehitettiin öljyteollisuudessa käytettävien linjojen valvontaa varten. Linjojen valvontaa suoritti valvonta- ja tiedonkeruujärjestelmä (SCADA, Supervisory Control And Data Acquisition). Linjojen valvontaa varten käytössä olivat epäluotettavat satelliittiyhteydet, joten heidän tuli kehittää uusi tiedonsiirtoprotokolla, joka ominaisuuksien puolestaan sopisi epävarmoissa verkoissa käytettäväksi. [11.]

Vaatimuksiksi kehitettävälle tiedonsiirtoprotokollalle he asettivat seuraavat [11]:

- Protokollan käyttöönotto tulee olla yksinkertaista, samoin kuin sen käyttäminenkin.
- Protokollan palvelun laatutaso pitää olla määritettävissä.
- Protokollan tulee olla tietoinen istunnon jatkuvuudesta tiedonsiirron yhteydessä.
- Protokollan tulee käyttää tietoliikennekaistaa sekä protokollaa käsittelevien laitteiden resursseja mahdollisimman vähän sekä tehokkaasti.
- Protokollan tulee olla data-agnostinen, eli protokollan ei tule rajoittaa viestien hyötykuorman rakennetta.

IBM käytti MQTT-protokollaa sisäisissä projekteissaan yli kymmenen vuoden ajan, kunnes vuonna 2010 IBM julkaisi protokollan version MQTT 3.1 ilmaiseksi sekä vapaasti käytettäväksi. Protokollan julkistamisen myötä IBM antoi Eclipse Foundationin Paho projektin käyttöön MQTT-protokollan asiakas sovellukset. Neljä vuotta MQTT:n julkaisemisesta vuonna 2014 protokollan versiosta MQTT 3.1.1 tuli OASIS-standardin virallisesti standardoima tiedonsiirtoprotokolla. Vuonna 2019 OASIS hyväksyi protokollan uusinta versiota MQTT 5 koskevan määrittelyn viralliseksi standardiksi. Tällä hetkellä useat

viestinvälittäjäsovellukset tarjoavat mahdollisuuden käyttää MQTT 3.1.1- ja MQTT 5-versioita. [11.]

## 3.4 Viestinvälittäjä

Broker eli viestinvälittäjä on toinen MQTT:n keskeisistä osista, ja sen tehtävänä on toimittaa vastaanottamansa viestit eteenpäin viestin tilanneille asiakkaille asetetuin ehdoin. Broker-termistä käytetään myös englanninkielistä nimeä Server puhuttaessa MQTT-protokollasta. Viestinvälittäjä on tietokoneella suoritettava sovellus. Tietokone voi asiakas-palvelin-mallin mukaisesti olla joko yhden piirilevyn tietokone tai kokonainen palvelin. MQTT 5 -version myötä broker-ohjelmistoista löytyy myös kattavasti ominaisuuksia pilvipohjaisen viestinvälittäjän tueksi. [9; 15.] Seuraavissa luetelmissa on esitetty viestinvälittäjän tehtäviä ja ominaisuuksia.

Viestinvälittäjän pääasiallisiin tehtäviin kuuluu [12]:

- hyväksyä asiakkaat osaksi tiedonsiirtoa rajoitusten mukaisesti
- vastaanottaa ja välittää asiakkaiden julkaisemat viestit määritellyin ehdoin
- käsitellä asiakkaiden lähettämät aiheiden tilaus- ja tilauksen keskeytyspyynnöt
- estää yhteyksien luominen epäluotettaviin asiakkaisiin.

Viestinvälittäjän muita ominaisuuksia on:

- protokollan käyttämien tietoturva-asetusten toteuttaminen
- kuormituksen tasaaminen
- ketjutus muiden välittäjien kanssa siltauksen avulla
- datan säilytys.

Viestinvälittäjä sovelluksia on saatavilla laajasti niin kaupallisina kuin avoimen lähdekoodin versioinakin. Broker-ohjelmistoja löytyy eri käyttöjärjestelmille, kuten Windowsille, macOSille, Linuxille. Tunnetuimpia välittäjäsovelluksia on kaupallinen HiveMQ sekä avoimen lähdekoodin Eclipse Mosquitto. Viestinvälittäjäsovelluksista löytyy kattavasti erilaisia ominaisuuksia, joita voidaan määrittää vapaasti käyttökohteen mukaan. [16.]

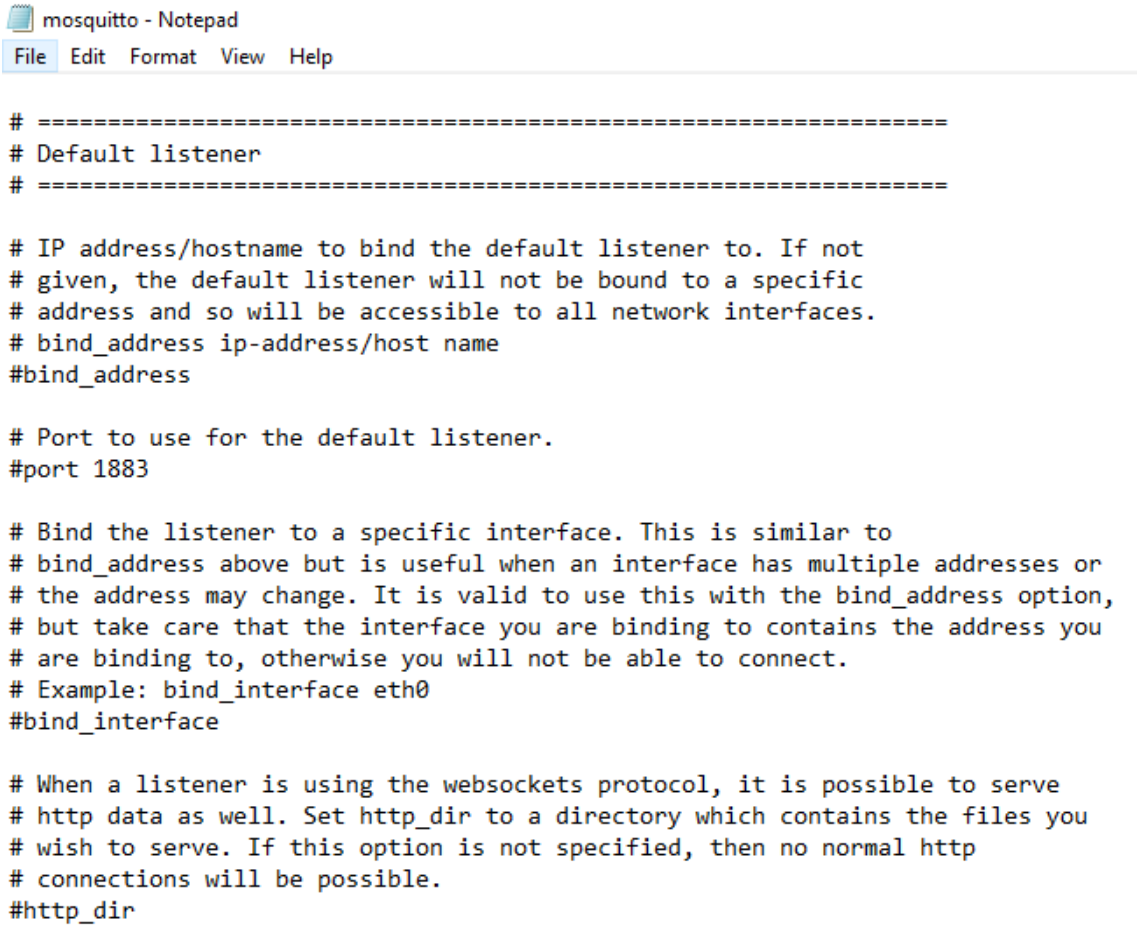
## 3.5 Asiakkaat

Client/consumer eli asiakas on toinen MQTT:n keskeisistä osista. Niiden tehtävinä on tilata aiheita, vastaanottaa tilatut viestit ja julkaista viestejä. Asiakas-termiä käytetään riippumatta siitä, julkaiseeko asiakas viestejä vai onko asiakas tilannut viestejä, mutta termiä voidaan kuitenkin täsmentää käyttämällä asiakkaasta nimiä tilaaja ja julkaisija. Yksittäinen asiakas voi kuitenkin suorittaa molempia edellä mainittuja toimintoja. MQTT-asiakas voi olla mikä tahansa laite, jonka käytössä on MQTT-kirjasto ja muodostaa yhteyden viestinvälittäjään verkon kautta. Paho-projekti sekä monet muut toimijat on mahdollistanut MQTT-asiakaskirjastojen laajan saatavuuden useille ohjelmointikielille, kuten Pythonille, C:lle, C ++:lle, C #:lle, Javalle ja JavaScriptille. MQTT:n suosion vuoksi asiakaskirjastojen lisäksi tarjolla on useita valmiita toteutusesimerkkejä eri kielillä sekä valmiita asiakassovelluksia esimerkiksi Windows-, Android- ja iOS-laitteille. [9; 15; 17.]

## 3.6 Eclipse Mosquitto

C-kielellä kirjoitettu Mosquitto on Eclipse Foundationin tarjoama avoimeen lähdekoodin perustuva ja EPL/EDL-lisensoitu viestivälittäjä, joka tukee MQTT-protokollaversioita 5.0, 3.1.1 ja 3.1. Mosquitto-projekti sisältää myös C-kielisen ohjelmakirjaston MQTT-asiakkaiden luomiseen sekä komentorivi-käyttöliittymän viestien julkaisemista ja tilaamista varten. [6.] Mosquitto on erittäin kevyt viestinvälittäjä, mikä tukee MQTT-protokollan ominaisuuksia. Mosquitton suoritustiedosto on noin 120 kilotavun suuruinen mosquitto.conf-niminen tiedosto. Suoritustiedoston on raportoitu kuluttavan noin 3 megatavun verran RAM-muistia, kun siihen on kytketty tuhat MQTT-asiakasta. Eclipse Foundation päivittää Mosquittoa usein niin OASIS Standardin MQTT-standardin muutoksien kuin Mosquitton kehitystyönkin vaiheiden osalta, mikä pitää viestinvälittäjän hyvin ajan tasalla. Viimeisen vuoden ajalta päivityksiä viestinvälittäjän suoritustiedostoon on tullut yhdeksän kappaletta. [18.]

Tiedostotyyppinä .conf-tiedostoa voidaan muokata tavallisilla tekstimuokkaintyökaluilla, kuten Windowsin Notepadilla. Koodin muokkaamiseen voidaan käyttää myös erityisesti ohjelmointikieliä tukevia muokkaimia, kuten Notepad++:aa. Kuvassa 6 on esitetty Mosquitton suoritustiedosto luettuna Notepadilla.

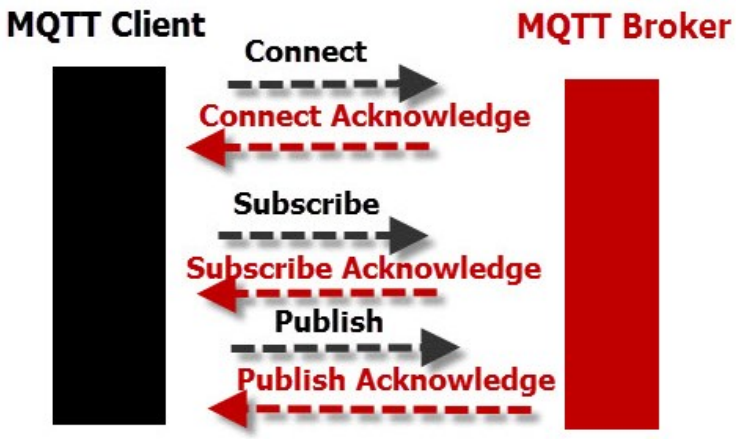



Kuva 6. Esimerkki Mosquitton muokkaamattomasta suoritustiedostosta.

Kommentit suoritustiedossa voidaan kirjoittaa merkitsemällä rivin alkuun ensimmäiseksi merkiksi #-merkin. Suoritustiedostossa ominaisuuksia on rajattu eri otsikoiden alle, otsikot on merkitty #=-merkkien sisään kuvan 6 mukaisesti. Muokkaamattomassa .conf-tiedostossa kaikki ominaisuudet on merkitty kommenteiksi, jolloin Mosquitto käyttää oletusasetuksiaan. Kaikki viestinvälittäjän ominaisuudet ovat siis käyttäjän itse määriteltävissä.

## 3.7 MQTT-protokollan viestipaketit

Tässä luvussa esitellään MQTT-protokollan viestinvälityksessä käytettyjä viestipaketteja ja niiden sisältöä. Pääasiallisesti kaikki viestinvälitys on kaksisuuntaista, eli jokaiseen lähetettyyn komentoviestiin julkaisija saa vastauksena kuittausviestin vastaanottajalta kuvan 7 mukaisesti.




Kuva 7. Vuorovaikutteinen viestinvälitys [19].

Viestipakettien ohjauselementit ovat binäärimuotoisia, eli ne muodostuvat biteistä tyypillisten merkkijonojen sijaan. Yhdistämiseen käytettävien CONNECT-pakettien sisältö, kuten asiakastunnus, otsikko, käyttäjätunnus ja salasana välitetään UTF-8-merkkijono muotoisina. MQTT on data-agnostinen, joten muu hyötykuorma on bittimuotoista dataa, jota voidaan käyttökohteen mukaan muotoilla, esimerkiksi JavaScript Object Notation (JSON) tai Extensible Markup Language (XML) -muotoon. Bittimuotoista dataa käytettäessä vastaanottajan tulee ymmärtää MQTT-viestintää, koska vastaanottajan pitää tietää tarkalleen, mitä lähetetyt bitit kuvaavat. [19; 20.]

Taulukossa 1 on esitelty MQTT-protokollan käyttämät kaikki 14 erilaista viestipakettia. Taulukko on muokattu osoitteesta https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html löytyvästä OASIS-standardin taulukosta.

Taulukko 1. MQTT:n viestipaketit [12].

| Paketin nimi | Paketin arvo | Paketin liikkuminen | Selite |
|---|---|---|---|
| Reserved | 0 | Kielletty | Varattu tulevaisuuden kehitystyötä varten |
| CONNECT | 1 | Asiakkaalta välittäjälle | Yhteyden luomispyyntö |
| CONNACK | 2 | Välittäjälle asiakkaalle | Yhteyden hyväksymispaketti |
| PUBLISH | 3 | Asiakkaalta välittäjälle tai Välittäjälle asiakkaalle | Viestin julkaisuun käytettävä paketti |
| PUBACK | 4 | Asiakkaalta välittäjälle tai Välittäjälle asiakkaalle | Julkaistun viestin kuittauspaketti (QoS 1) |
| PUBREC | 5 | Asiakkaalta välittäjälle tai Välittäjälle asiakkaalle | Julkaistun viestin kuittauspaketti (QoS 2, 1/3) |
| PUBREL | 6 | Asiakkaalta välittäjälle tai Välittäjälle asiakkaalle | Julkaistun viestin kuittauspaketti (QoS 2, 2/3) |
| PUBCOMP | 7 | Asiakkaalta välittäjälle tai Välittäjälle asiakkaalle | Julkaistun viestin kuittauspaketti (QoS 2, 3/3) |
| SUBSCRIBE | 8 | Asiakkaalta välittäjälle | Viestin tilaukseen käytettävä paketti |
| SUBACK | 9 | Välittäjälle asiakkaalle | Saapuneen tilauksen kuittaus |
| UNSUB-SCRIBE | 10 | Asiakkaalta välittäjälle | Viestin tilauksen keskeyttämiseen käytettävä paketti |
| UNSUBACK | 11 | Välittäjälle asiakkaalle | Tilauksen keskeyttämisen kuittaus |
| PINGREQ | 12 | Asiakkaalta välittäjälle | Yhteyden varmistuspyyntö |
| PINGRESP | 13 | Välittäjälle asiakkaalle | Vastaus yhteyden varmistuspyyntöön |
| DISCON-NECT | 14 | Asiakkaalta välittäjälle tai Välittäjälle asiakkaalle | Yhteyden katkaisemisilmoitus |
| AUTH | 15 | Asiakkaalta välittäjälle tai Välittäjälle asiakkaalle | Varmenteiden vaihtoon käytettävä viestipaketti |

MQTT:n viestipaketit voidaan jakaa paketteihin, joiden sisältöön asiakas voi vaikuttaa paketteihin, joiden sisältö määräytyy automaattisesti. Hyötykuormallisia paketteja ovat CONNECT, PUBLISH, SUBSCRIBE, SUBACK, UNSUBSCRIBE ja UNSUBACK. Kaikki muut MQTT:n viestipaketit ovat hyötykuormattomia paketteja, joiden sisältöön käyttäjälle ei ole mahdollisuutta vaikuttaa, koska ne ovat kuittausviestejä, jotka välittäjä ottaa automaattisesti käyttöön käyttäjän tai asiakkaan määrittämien asetusten mukaisesti. Hyötykuormalliset viestit SUBSCRIBE, SUBACK ja UNSUBSCRIBE ovat tilauksiin liittyviä viestipaketteja, joiden hyötykuormat koostuvat tilattavista otsikoista ja niiden palveluiden laatutasoista. [12; 19.]

Viestipakettien vähimmäiskoko on kaksi tavua. Tätä osaa viestipakettia kutsutaan kiinteäksi otsikoksi (fixed header field), jossa on yhden tavun pituinen ohjauskenttä (control field) ja paketin pituus -kenttä (packet length field). Paketin pituus -muuttuja on kooltaan yhden tavun mittainen, kun viestin kokonaispituus on alle 127 tavua kiinteää otsikkoa lukuun ottamatta. [12; 19.]

MQTT-protokollan erityyppiset pakettirakenteet ovat [19]:

• kiinteä otsikko (ohjauskenttä + paketin pituus kenttä) - CONNACK-paketti

• kiinteä otsikko (ohjauskenttä + paketin pituus kenttä) + muuttuva otsikko (variable length header) - PUBACK-paketti

• kiinteä otsikko (ohjauskenttä + pituus) + muuttuva otsikko + hyötykuorma (Payload) – CONNECT-paketti.

Viestinvälityksen asetusten määrityksen kannalta tärkeimmät hyötykuormalliset viestipaketeista ovat CONNECT ja PUBLISH. Asiakas lähettää paketin välittäjälle, kun asiakas luo ensimmäisen kerran yhteyden viestinvälittäjään. Kuvassa 8 on esitetty CONNECT-paketin sisältö. Asiakas lähettää CONNECT-paketin viestinvälittäjälle, kun asiakas liittyy osaksi viestinvälitystä ensimmäisen kerran. [9.]

| CONNECT-Paketti | CONNECT-Paketin esimerkkisisältö |
|---|---|
| **Asiakastunnus** | Sensori1 |
| **Puhdas istunto** | Tosi (True) |
| **Käyttäjänimi (vapaavalintainen)** | ToimistoLampotila |
| **Salasana (vapaavalintainen)** | Salasana123 |
| **Viimeinen tahto otsikko (vapaavalintainen)** | /ToimistoLampotila/ViimeinenTahto |
| **Viimeinen tahto palvelun laatutaso (vapaavalintainen)** | 2 |
| **Viimeinen tahto viesti (vapaavalintainen)** | Yhteys sensoriin on katkennut |
| **Viimeinen tahto pysyvä viesti (vapaavalintainen)** | Tosi (True) |
| **Yhteyden ylläpidon tarkistusaikaväli** | 120 |

Kuva 8. MQTT:n CONNECT-paketin attribuutit ja attribuuttien esimerkkiarvot [9].

PUBLISH-paketilla asiakas määrittää julkaistavan viestinsä otsikon sekä viestin välitystä koskevat asetukset viestinvälittäjälle kuvan 9 mukaisesti. Asiakas lähettää PUBLISH-paketin, kun asiakas haluaa aloittaa viestin julkaisun uudella otsikolla. [21.]

| Publish-Paketti | Publish-Paketin esimerkkiarvot |
|---|---|
| Viestipaketin tunnus | 1234 (arvo on 0, jos palvelun laatutaso on 0) |
| Aiheen nimi | Toimisto/Lampotila |
| Palvelun laatutaso | 1 |
| Pysyvä viesti | Epätosi (False) |
| Hyötykuorma | Lampotila: 24 |
| DUP-lippu | Epätosi (False) |

Kuva 9. MQTT:n PUBLISH-paketin attribuutit ja attribuuttien esimerkkiarvot [21].

Kuvassa 9 nähtävä duplicate-lippu (duplicate-flag, DUP-flag) on Publish-paketin attribuutti, joka saa arvon tosi, kun lähetettävä viesti on uudelleenlähetys. Attribuutti on käytössä vain alaluvussa 4.4 esiteltävissä palvelun laatutasoissa 1 ja 2. DUP-lippu on käytössä vain MQTT:n sisäisessä viestinvälityksessä eikä se ole asiakassovelluksien hyödynnettävissä. [21.]

## 3.8 Otsikko-/aihepohjainen viestintä

Julkaisija/tilaaja-mallit käyttävät viestien suodatusta viestien eteenpäin välittämiseen oikeille asiakkaille. Kolme tyypillisintä viestien suodatusmenetelmää ovat otsikko-/aihepohjainen suodatus (subject-based filtering), tyyppipohjainen suodatus (type-based filtering) ja sisältöpohjainen suodatus (content-based filtering). [14.]

MQTT-protokolla käyttää UTF-8-merkkijonomuotoista otsikko-/aihepohjaista suodatusta. Jokaisen MQTT-viestin tulee sisältää aihe, jonka pituus on oltava vähintään yksi merkki. Aihe määritetään julkaisevan asiakkaan toimesta ja se lähetetään osana PUBLISH-pakettia. Otsikon perusteella välittäjä lähettää vastaanottamansa viestit aiheen tilanneille asiakkaille. MQTT:n aiheissa käytetään tasoja, joihin jakamiseen käytetään tietokoneiden tiedosto- ja kansiorakenteissa käytettyä vinoviivaa (/) — esimerkiksi otsikolla t*oimipiste1/komponenttitesti1/tiedot* voitaisiin kohdeyrityksen tietyn toimipisteen testerin tiedot. Tasojen käyttö ei kuitenkaan ole pakollista, joten pelkkä *testeri1* on toimiva otsikko. MQTT:n otsikot ovat aakkoskoosta riippuvaisia, joten *testeri1/tiedot* ja *Testeri1/tiedot* ovat eri otsikoita. Otsikot voivat sisältää välilyöntejä. Otsikot ja niiden tasojen

lukumäärän asiakas saa määrittää itse, mutta otsikon maksimipituus määräytyy UTF-8-muotoisen merkkijonon kokonaispituuden eli 65 535 bitin mukaan. [9; 22.]

Kuvassa 10 on esitetty tyypillinen aihepohjainen viestintä, jossa asiakas julkaisee viestin valitsemallaan otsikolla ja toisen asiakkaan tulee tilata kyseinen otsikko vastaanottaakseen julkaistun viestin.


Kuva 10. Aihepohjainen viestintä.

MQTT-protokollaa käytetään tyypillisesti kohteissa, joissa on useita asiakkaita. Yksittäisten asiakaslaitteiden väliseen tiedonsiirtoon tarkalla otsikolla tilaaminen voi olla hyödyllistä, mutta kokonaisen valvomojärjestelmän tiedonsiirrossa ei ole järkevää käyttää yksittäisiä otsikoita. Villikortti (wildcard) on MQTT:n ominaisuus, minkä avulla asiakas voi tilata useita otsikoita. Villikortteja on kahdenlaisia, yksittäisen tason korvaava kortti sekä useamman tason korvaava kortti. Yksittäisen tason korvaavaa villikorttia merkitään plusmerkillä (+) ja useamman tason korvaavaa villikorttia ristikkomerkillä (#). Villikortteja ei tule käyttää julkaistavissa otsikoissa vain ainoastaan viestien tilauksissa. Alla olevissa luetelmissa on esitetty molempien villikortti tyyppien käyttäminen sekä asiakkaan vastaanottamat aiheet valitulla villikortilla. [23; 7, s.43–45.]

Asiakkaan tilatessa otsikolla *toimipiste1/+/tila* voi asiakas vastaanottaa esimerkiksi seuraavan kaltaisia otsikoita [7, s.43–45]:

- toimipiste1/testeri1/tila
- toimipiste1/testeri2/tila
- toimipiste1/ilmastointi/tila
- toimipiste1/palohalytin_jarjestelma/tila.

Plusmerkillä on korvattu yllä olevassa tilanteessa laitteen nimi. Tämän kaltaista tason korvaamista voitaisiin hyödyntää esimerkiksi kunnossapidossa, jossa on tärkeää saada tietyn toimipisteen kaikkien laitteiden tilatiedot.

Asiakkaan tilatessa otsikolla *tehdas/#*, voi asiakas vastaanottaa esimerkiksi seuraavankaltaisia otsikoita [7, s.43–45]:

- tehdas/tuotantolinja1/tila
- tehdas/tuotantolinja2/tuotteet/valmiit
- tehdas/varasto/lähtevat tuotteet
- tehdas/tuotantolinja3/huollontarve.

Ristikkomerkillä korvataan useita tasoja, jolloin asiakas vastaanottaa kaikki otsikot, jotka ovat merkin tasolla ja sen jälkeen. Toiminnanohjausjärjestelmät tai isot valvomot voisivat hyödyntää usean otsikkotason korvaamista, mutta viestin hyötykuorman sijoittaminen käyttöjärjestelmään tulisi merkitä huolellisesti, jotta voidaan varmistua datan oikeasta sijoituskohteesta. Kaikkien asiakkaiden julkaisemien viestien tilaaminen onnistuu tilaamalla otsikko #. [7, s.43–45.]

Vapaasta otsikoinnista huolimatta protokollassa on otsikointitapa, joka on useimmissa viestinvälittäjä sovelluksissa varattu viestinvälittäjän omaan käyttöön. Nämä *$SYS*-merkeillä alkavat otsikot on varattu viestinvälittäjän lokitietojen kirjaukseen, kuten *$SYS/broker/messages/received*, jonka tilaamalla asiakas saa viestin hyötykuorma viestinvälittäjän vastaanottamien viestin lukumäärän. Välittäjän lokitietoja koskevia aiheita ei voi tilata kaikkien aiheiden tilaamisessa käytetyllä pelkällä ristikkomerkillä, vaan kaikki lokitiedot asiakas voi tilata otsikolla *$SYS/#*. [12.] Mosquitto tarjoaa käyttäjilleen kattavan listan $SYS-otsikoista, joita käyttäjät voi hyödyntää viestinvälittäjän toiminnan tarkkailuun [22; 24].

# 4 MQTT-protokollan ominaisuuksia

## 4.1 Asiakastunnus

Asiakastunnus (Client Identifier, Client ID) on asiakkaan yksilöivä parametri. Asiakastunnus lähetään UTF-8-formaatissa välittäjälle pakollisena osana CONNECT-viestipakettia. Kaikkien tilauksien viestinvälitys suoritetaan asiakastunnukseen perustuen. Välittäjä lähettää viestin kaikille aiheen tilanneille asiakastunnuksille, joihin sillä on toimiva yhteys. Käyttäessä alaluvussa 4.4 esiteltäviä palvelun laatutasoja 1 ja 2 välittäjä tallentaa julkaistun viestin, jos havaitsee aiheen tilanneen asiakkaan yhteyden katkenneen ja lähettää viestin uudelleen, kun yhteys asiakkaaseen on palannut. MQTT-protokollaa koskevan standardin mukaan viestinvälittäjän tulee sallia asiakastunnukset, joiden pituus on 1—23 merkkiä ja käytettävät merkit on oltava numeroita 0—9 ja latinalaisia aakkosia pois lukien tarkkeet. Otsikot ovat aakkoskoosta riippuvaisia eli merkit *a* ja *A* tulkitaan otsikoissa eri merkeiksi. [12; 25.]

Välittäjäsovelluskohtaisesti voidaan käyttää pidempiä asiakastunnuksia tai tunnuksia, jotka sisältävät erikoismerkkejä. Lisäksi välittäjäsovellus voi sallia 0 merkin pituisen asiakkaan liittymisen, jolloin välittäjä luo satunnaisen asiakastunnuksen liittyvälle asiakkaalle, joka palautetaan asiakkaalle osana CONNACT-viestipakettia. Pysyvissä istunnoissa, alaluvuissa 5.3, 5.4 ja 5.5 esitettävien tietoturva ominaisuuksien sekä datan käsittelyn vuoksi on kuitenkin tärkeä käyttää jotakin järjestelmän kannalta loogista ja ennalta määritettyä tunnusta, kuten *Asiakas123* tai *Varasto1*. Ennalta määritetty asiakastunnus estää myös samannimisten asiakkaiden luomista ja helpottaa vikatilanteiden selvitystä. Välittäjän vastaanottaessa asiakkaan liittymispyynnön saman nimiseltä asiakastunnuksen omaavalta asiakkaalta kuin jo liitetty asiakas, välittäjä katkaisee yhteyden aiempaan asiakkaaseen. [12; 25.]

## 4.2 Yhteyden ylläpidon tarkistusaikaväli

Yhteyden ylläpidon tarkistusaikaväli (Keep Alive interval) on MQTT:n CONNECT-paketin yhteydessä asetettava ominaisuus, jonka tarkoituksena on määrittää luotavalle yhteydelle pisin mahdollinen aikaväli, jonka sisällä viestiensiirto täytyy tapahtua. Toiminnon

tarkoituksena on varmistaa, että yhteys välittäjän ja asiakkaan välillä on säilynyt. Asiakkaan tulee lähettää välittäjälle joko normaalisti julkaisemansa viesti tai erillinen yhteyden ylläpito viesti eli PINGREQ-viesti ennen tarkistusaikavälin umpeutumista. Välittäjä vastaa PINGREQ-viestiin PINGRESP-viestillä. Välittäjä valvoo yhteyden ylläpidon tarkistusaikaväliä ja katkaisee yhteyden, jos se ei ole vastaanottanut asiakkaalta kumpaakaan aiemmin mainituista viesteistä tarkistusaikavälin sisällä. Välittäjän käyttämä tarkistusaikaväli on puolitoistakertainen asetettuun arvoon nähden. Esimerkiksi asiakkaan asettaessa tarkasteluväliksi 60 sekuntia tapahtuu katkaisu 90 sekunnin jälkeen. Tarkistusaikaväli on kooltaan kaksi bittiä, ja se kuvaa tarkistusaikaväliä sekunneissa. Tarkistusaikavälin korkein mahdollinen arvo on siis 65 535 sekuntia, joka tarkoittaa 18 tuntia 12 minuuttia ja 15 sekuntia. Toiminto voidaan ottaa pois käytöstä asettamalla tarkistusaikavälin arvoksi 0, jolloin välittäjä ei enää valvo yhteyden ylläpito. [26; 12.]

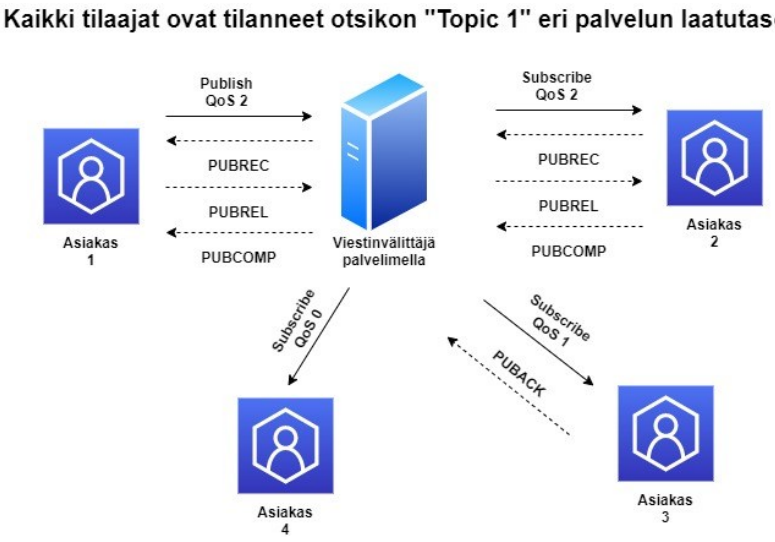
## 4.3 Pysyvä viesti

Pysyvä viesti eli Retain Message on asiakkaan määrittelemä asetus viestin lähetyksessä. Pysyvä viesti asetus lähetetään osana PUBLISH-pakettia, ja se on yksi kolmesta muuttujasta, jotka määrittävät MQTT:n viestien tallentamista. Näiden kolmen asetuksen yhteisvaikutusta on kuvattu tarkemmin taulukossa 2. Arvon ollessa tosi eli *true*, viestinvälittäjä tallentaa viimeksi julkaistun viestin sekä määritetyn palvelun laatutason, minkä jälkeen välittäjä lähettää viestin aiheen uusille tilaajille. Pysyvä viesti on tarpeellinen ominaisuus kohteissa, joissa dataa lähetetään vain harvoin, sillä uusi tilaaja ei olisi muutoin tietoinen lähettäjän tilasta ja joutuisi muuten odottamaan uusinta viestiä määritetyn ajan. Edellinen pysyvä viesti korvataan aina, kun uusi viesti julkaistaan. [27; 28; 12.]

## 4.4 Palvelun laatutaso

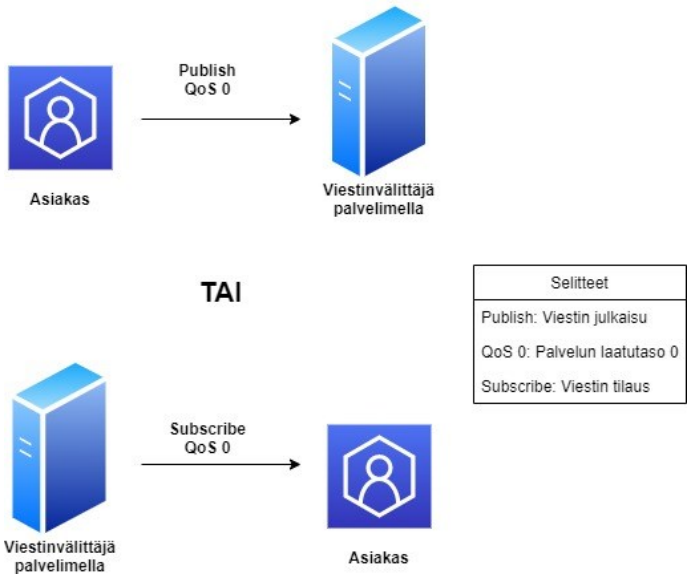
MQTT-protokollaan liittyen lyhenne QoS eli Quality of Service tarkoittaa palvelun laatutasoa. Palvelun laatutaso on viestin julkaisijan ja viestinvälittäjän sekä viestinvälittäjän ja vastaanottajan välistä viestitystä määrittäessä protokollan ominaisuus, jolla määritetään lähetetyn viestin tärkeys. Viestiä julkaistaessa ja tilatessa voidaan siis käyttää eri palvelun laatutasoa. Tämä ominaisuus mahdollistaa eri tärkeysasteiden käyttämisen erityyppisille viesteille, vastaanottajille tai lähettäjille. Viestiketjun matalin laatutaso määrittää

koko muun ketjun laatutason kuvan 11 mukaisesti. Protokolla tarjoaa kolme eri palvelun laatutasoa 0,1 ja 2. [7, s.46–48.] Kuvassa 11 asiakas 2 saa julkaistun viestin korkeimmalla tasolla, koska viestin julkaisija on käyttänyt myös korkeinta palvelun tasoa. Asiakkaat 3 ja 4 saavat viestin niiden määrittämällä palvelun laatutasolla, vaikka viesti on julkaistu alun perin korkeammalla tasolla.



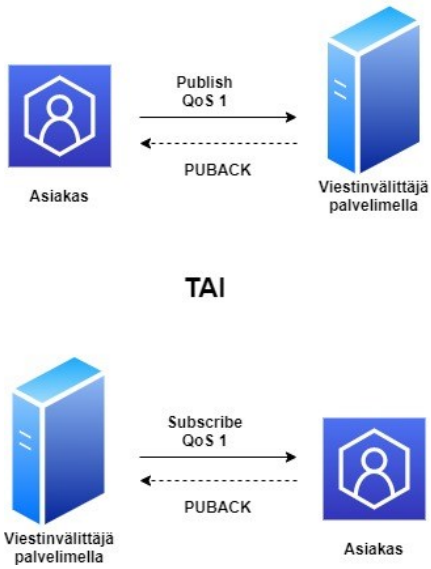

Kuva 11. Tilaajat ja julkaisijat voivat määrittää käyttämänsä palvelun laatutason [29].

Palvelun laatutaso 0 on matalin taso. Tasolla 0 julkaisija lähettää viestin eteenpäin enintään kerran, mutta minkäänlaista varmistusta viestin lähettämistä tai vastaanottamista ei ole. Lähetetty viesti ei välttämättä ikinä saavu sille tarkoitettuun kohteeseen, eikä samaa viestiä lähetetä enää uudestaan, koska julkaisija ei talleta lähetettävää viestiä. [7, s.47.] Kuvassa 12 on esitetty viestin rakenne palvelun laatutasolla 0. Matalinta tasoa voidaan käyttää kohteissa, joissa yhteydet ovat luotettavia, viestin katoamisella ei ole merkitystä tai halutaan säästää tietoliikenteessä liikkuvan datan määrässä. Tiheästi lähetettävät ja nopeasti muuttuvat datat voidaan lähettää matalimmalla palvelutasolla. [7, s.47.]



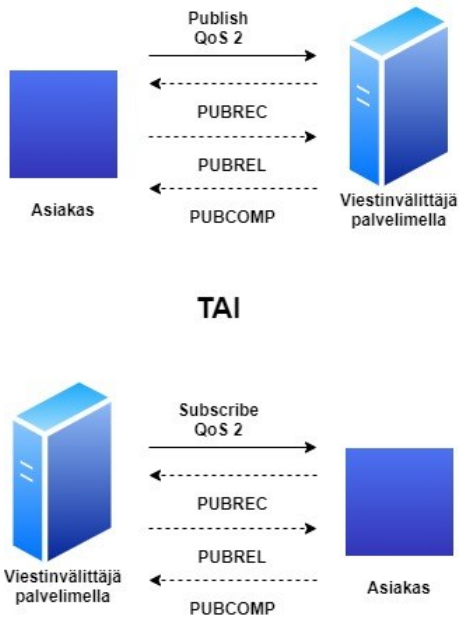

Kuva 12. Viestinvälitys palvelun laatutasolla 0 [29].

Palvelun laatutasolla 1 julkaisija lähettää viestin ainakin kerran. Tasolla 1 viestin vastaanottoja lähettää julkaisijalle kuittausviestin PUBACK kuvan 13 mukaisesti, kun se on vastaanottanut lähetetyn viestin. Sama viesti saatetaan kuitenkin lähettää uudestaan, koska julkaisija säilyttää lähetettyä viestiä niin kauan, kunnes se on saanut kuittauksen vastaanottajalta. Viesti lähetetään uudestaan, jos kuittausviesti PUBACK ei saavu määritetyssä ajassa. [5.] Kuvassa 13 on esitetty varmistusviestin käyttäminen laatutasolla 1. Tasoa 1 voidaan käyttää kohteissa, joissa viestin perille saapuminen on erityisen tärkeä, mutta kaksi kertaa saman viestin saapumisesta ei ole haittaa. [7, s.47.]




Kuva 13. Viestinvälitys palvelun laatutasolla 1 [29].

Palvelun laatutaso 2 on korkein taso. Tasolla 2 viestin lähettäminen ja vastaanottaminen kuitataan siten, että vastaanottaja saa viestin vain kerran. Viestin lähettäminen ja saapuminen on varmistettu kolmen kuittausviestin avulla kuvassa 14 esitetyn rakenteen mukaisesti. Kolme varmistusviestiä on nimiltään PUBREC, PUBREL ja PUBCOMP. Korkeinta palvelun laatutasoa voidaan käyttää kohteissa, joissa saapuneiden viestien määrällä on merkitystä. [7, s.47.]




Kuva 14. Viestin välitys palvelun laatutasolla 2 [29].

Eri palvelun laatutasot mahdollistavat protokollan viestinvälityksen määrittämisen käyttökohteen mukaan. Jos protokollan viestinvälitys on tarkoitus pitää kevyenä, käytetään matalinta tasoa, mutta mitä tärkeämpää viestin perille saapumisen varmentaminen on, sitä enemmän voidaan laatutasoa kasvattaa. [7, s.47–48.]

## 4.5 Viimeinen tahto ja testamentti

LWT eli viimeinen tahto ja testamentti on MQTT:n asiakkaan määrittämä ominaisuus, jonka avulla asiakas voi julkaista ilmoitusviestin verkkoyhteyden katketessa. Ominaisuudesta käytetään myös lyhyempiä nimityksiä viimeinen tahto (last will) ja testamentti (testament). Ominaisuus voidaan ottaa käyttöön silloin, kun asiakas yhdistetään viestinvälittäjään, eli osana CONNECT-pakettia. Viimeiseen tahtoon asiakkaan tulee määrittää kaikki normaalin viestin lähetyksen asetukset eli otsikko, hyötykuorma, laatutaso ja pysyvä viesti. Viesti lähetetään aiheen tilanneille vain silloin, kun välittäjä havaitsee yhteyden asiakkaaseen katkenneen epänormaalisti, eli kun asiakas ei ole käyttänyt DISCONNECT-viestiä. Välittäjä pitää viestin tallennettuna niin kauan, kunnes asiakas lähettää onnistuneen DISCONNECT-viestin. Viimeinen tahto ja testamentti voidaan julkaista asiakkaan normaalisti käyttämällä otsikolla, kuten ”toimisto/huone1/lampotila” tai erillisellä

viimeinen tahto -otsikolla ”huone1/anturi1/tila”. Molemmissa esimerkkitapauksissa on hyödyllistä julkaista anturin tilaa kuvaava viesti myös sen yhdistyessä välittäjään. Hyötykuorma voi olla esimerkiksi kytkentätilanteessa ”OK” ja yhteyden katkeamistilanteessa ”ERROR”. Näiden hyötykuormien avulla voidaan määrittää käyttökohteen mukaan tarvittavia toimenpiteitä, kuten hälytyksiä tai prosessin keskeytyksiä. Kuvassa 15 on esitetty CONNECT-paketin asetukset viimeisen tahdon osalta. [30; 31; 12.]

| | |
|---|---|
| **Viimeinen tahto otsikko (vapaavalintainen)** | /ToimistoLampotila/ViimeinenTahto |
| **Viimeinen tahto palvelun laatutaso (vapaavalintainen)** | 2 |
| **Viimeinen tahto viesti (vapaavalintainen)** | Yhteys sensoriin on katkennut |
| **Viimeinen tahto pysyvä viesti (vapaavalintainen)** | Tosi (True) |

Kuva 15. CONNECT-paketin sisältämät viimeisen tahdon asetukset [30].

## 4.6 Puhdas istunto

”Puhdas istunto” -lippumuuttuja (Clean Session Flag) on MQTT:n asiakkaan määrittämä ominaisuus, jonka avulla asiakas voi määrittää viestinvälittäjälle, onko luotava yhteys pysyvä vai ei. Puhdas istunto määritellään osana välittäjään yhteyden luomista käytettävää CONNECT-pakettia, jonka yhteydessä on toimitettava myös asiakastunnus (clienID), jotta ominaisuutta voidaan käyttää. Oletusasetuksena viestinvälittäjä ei tallenna välittämiänsä viestejä, mutta puhdas istunto-ominaisuuden avulla määritetään, tallennetaanko asiakkaan aiheita koskevat tilaukset ja vastaanottamattomat viestit. Niin kuin muissakin MQTT:n ominaisuuksissa viestien tallennus koskee vain viestejä, joita tilaaja ei ole vastaanottanut. Taulukossa 2 on esitetty MQTT:n viestien tallennus siihen vaikuttavien ominaisuuksien eri arvoilla. [9; 32.]

Taulukko 2. MQTT:n ominaisuuksien vaikutus viestien tallennukseen [32].

| **Puhdas istunto -muuttujan arvo** | **Pysyvä viesti -muuttujan arvo** | **Tilauksen palvelun laatutaso** | **Julkaisun palvelun laatutaso** | **Julkaistut viestit saapuvat perille?** |
|---|---|---|---|---|
| Tosi | Epätosi | 0 | 0 | Ei |
| Tosi | Epätosi | 0 | 1 tai 2 | Ei |
| Tosi | Epätosi | 1 tai 2 | 0 | Ei |
| Tosi | Epätosi | 1 tai 2 | 1 tai 2 | Ei |
| Epätosi | Epätosi | 0 | 0 | Ei |
| Epätosi | Epätosi | 0 | 1 tai 2 | Ei |
| Epätosi | Epätosi | 1 tai 2 | 0 | Ei |
| Epätosi | Epätosi | 1 tai 2 | 1 tai 2 | Kyllä – Kaikki lähetyt viestit. |
| Tosi | Tosi | 0 | 0 | Kyllä, mutta vain viimeksi lähetetty viesti. |
| Tosi | Tosi | 0 | 1 tai 2 | Kyllä, mutta vain viimeksi lähetetty viesti. |
| Tosi | Tosi | 1 tai 2 | 0 | Kyllä, mutta vain viimeksi lähetetty viesti. |
| Tosi | Tosi | 1 tai 2 | 1 tai 2 | Kyllä, mutta vain viimeksi lähetetty viesti. |
| Epätosi | Tosi | 0 | 0 | Kyllä, mutta vain viimeksi lähetetty viesti. |
| Epätosi | Tosi | 0 | 1 tai 2 | Kyllä, mutta vain viimeksi lähetetty viesti. |
| Epätosi | Tosi | 1 tai 2 | 0 | Kyllä, mutta vain viimeksi lähetetty viesti. |
| Epätosi | Tosi | 1 tai 2 | 1 tai 2 | Kyllä – Kaikki lähetyt viestit. |

Puhdas istunto -lipun arvon ollessa tosi eli *true,* välittäjä ei tallenna asiakkaasta mitään tietoja. Arvon ollessa epätosi eli *false*, pysyvä viesti -muuttujan ollessa tosi ja julkaisun sekä tilauksen palvelun laatutason ollessa vähintään 1 välittäjä tallentaa asiakkaan tilaukset ja viestit, joita asiakas ei ole vahvistanut saaneensa. Oletusasetuksena Mosquitto-välittäjä tallentaa enintään ensimmäiset sata viestiä, mutta välittäjän ’’max_queued_messages 100’’ -asetuksen lukuarvoa muuttamalla voidaan vaihtaa tallennettujen viestien lukumäärää. Toinen viestien pituutta rajoittava asetus on *max_queued_bytes 0*. Oletusarvona on 0 eli viestin enimmäismäärää ei ole rajoitettu bittitasolla. [9; 32.]

### 4.7 Käyttäjän ominaisuudet

Käyttäjän ominaisuudet eli User Attributes on MQTT:n version 5 ominaisuus, jonka asiakas lähettää välittäjälle tai toiselle asiakkaalle UTF-8-merkkijonona osana normaaleja viestejä viestityypin mukaan. Käyttäjäominaisuudet voidaan liittää lähes kaikkiin viestipaketteihin, paitsi PINGREQ- ja PINGRESP-paketteihin, mukaan lukien vahvistusviestit PUBREL ja PUBCOMP. Käyttäjäominaisuuksien määrää ei ole rajoitettu, vaan niiden määrä rajautuu viestin kokonaiskoon rajoituksen mukaan. [33; 34.]

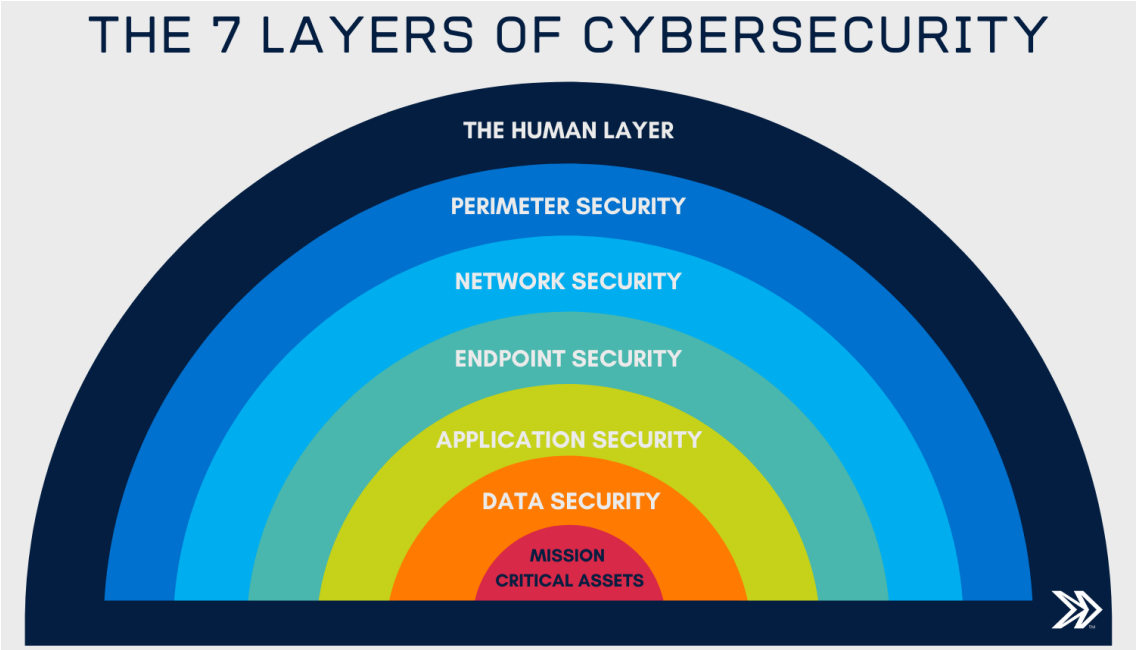
User Attributes on tärkeä ominaisuus ja laajentaa MQTT:n käyttökohteita huomattavasti. Ominaisuus voi sisältää esimerkiksi viestien numeroinnin, aikaleiman, viestin kulun reititystiedot, tiedoston nimen ja käytetyn tiedostomuodon, kuten XML, JSON, sekä eri pakatut tiedostomuodot. MQTT-protokollaa käytetään usein järjestelmissä, joissa asiakaslaitteet ovat tyypiltään hyvin erilaisia, minkä vuoksi viestien hyötykuorman rakenne, käsittely ja käyttö vaihtelevat suuresti. Käyttäjäominaisuudet tarjoavat ikään kuin toisen otsikko- ja asiakastunnusyhdistelmän, jonka mukaan lähetettyä dataa voidaan käyttää eri käyttökohteissa eri tavoin ja lajitella entistä tarkemmin. Lisäksi ominaisuus mahdollistaa viestin julkaisijan jäljittämisen, jota protokolla ei ole aiemmin tukenut. [33; 34.]

## 5 Protokollan tietoturvaominaisuudet

Tietoturva ja erityisesti sähköisen tiedonsiirron kyberturva on tärkeässä osassa järjestelmien käytettävyyttä ja luotettavuutta. Tietoturvaa tarkastellessa käytetään usein termejä

luottamuksellisuus (confidentiality), eheys (integrity) ja saatavuus (availability). Yhdessä nämä termit muodostavat CIA-mallin. Luottamuksellisuudella tarkoitetaan tietoon käsiksi pääsemisen estämistä siten, että tietoon pääsee käsiksi vain määritetyt käyttäjät. Luottamus toteutetaan usein tunnistautumisella kirjautumisten yhteydessä, pääsyrajauksilla ja salausalgoritmeilla. Eheys tarkoittaa sitä, että saapuva tieto ei ole muuttunut matkalla väärien käyttäjien toimesta tai jos tieto on muuttunut matkalla, se havaitaan ja siihen reagoidaan. Eheys voidaan varmentaa luottamuksen tapaan salausalgoritmeilla, jotta tieto ei ole luettavissa tai muokattavasti ilman oikeita avaimia. Saatavuus tarkoittaa, että tieto tai palvelu on käyttäjien saatavilla tarvittaessa, eikä tiedon saatavuus saa aiheuttaa tarpeettoman paljon työtä tiedon käyttäjälle. Saatavuutta voidaan parantaa tallentamisella, varmuuskopioinnilla ja käyttäjämäärärajoituksilla. [35, s-33–37; 36, s.29–33.]

Tietoturva on sisällöltään todella laaja ja jakautuu useaan tasoon kuvassa 16 esitetyllä tavalla. Tässä insinöörityössä keskitytään MQTT-protokollan tietoturvaan, eli sovellus- ja tiedonsiirtotason turvallisuuteen. Protokollaa käyttöönotettaessa on kuitenkin erityisen tärkeä varmistaa, että ylemmän sekä alemman tietoturvatason suojaukset ovat kunnossa. Hyvän tietoturva saavuttaminen vaatii kaikkien tietoturvatasojen toimivuutta ja ylläpitoa. [37.]



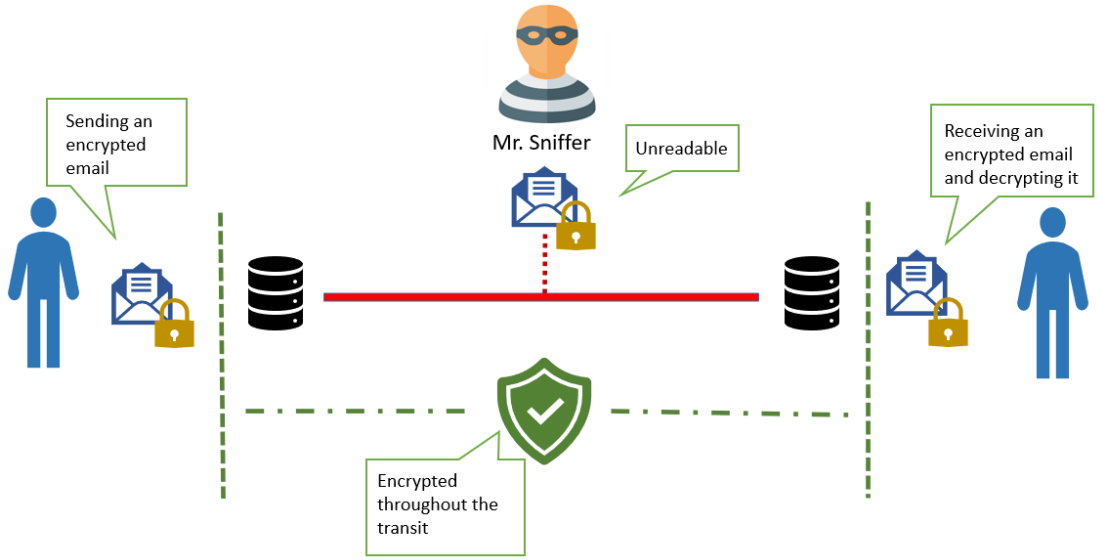

Kuva 16. Tietoturvan tasot [38].

MQTT-protokolla käyttää TCP/IP-protokollaa, eikä siirrettävä data ole oletusasetuksena salattua. Protokolla käyttää salaamattomaan liikenteeseen porttia 1883. Protokolla tarjoaa kuitenkin kattavat ominaisuudet tietoturvan parantamiseen. Näistä ominaisuuksista tärkeimmät ovat välitettävän tiedon salaaminen, pääsynhallintaluettelo, asiakastunnus, sekä käyttäjä- ja salasanasuojaus. [39.]

Työstä on rajattu pois verkkoyhteyksien tunnelointi MQTT-protokollan kanssa usein käytetyn virtuaaliset erillisverkkojen (VPN, Virtual Private Network.) avulla, sillä yrityksen tämänhetkinen verkkoarkkitehtuuri ratkaisu on tietoturva näkökulmasta riittävä myös MQTT-protokollan käyttöön. On kuitenkin hyvä muistaa, että uudelle järjestelmälle tulee tehdä tietoturvan riskienkäsittelysuunnitelma sekä tietoturvan riskianalyysi ennen käyttöönottoa. Kohdeyrityksen automaatioverkko on segmentoitu erilleen muusta verkosta, sijoitettu palomuurien sisään ja yhteydet esimerkiksi tietokantaan on salattu. Lisäksi yrityksen tietoturva on verkko- ja laitetasolla hyvin ylläpidettyä ja valvottua mikrotuen puolesta.

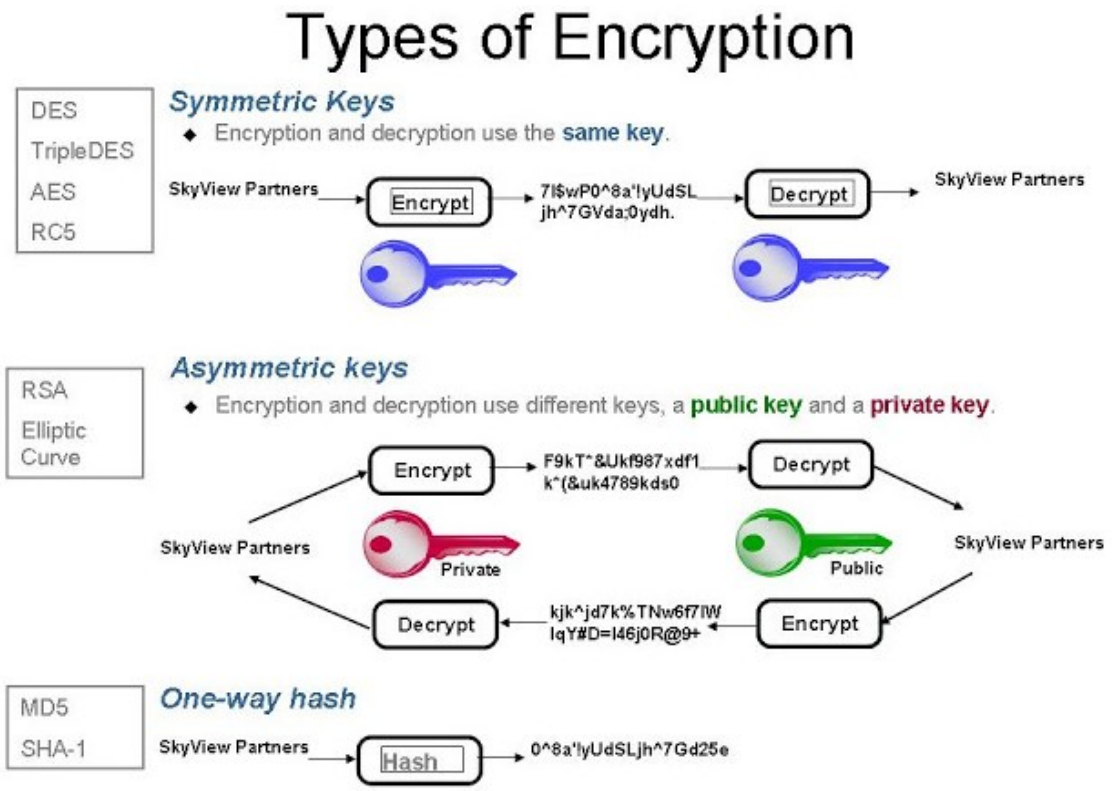
## 5.1 TLS/SSL-salaaminen

Salausmenetelmien eli symmetrisien ja asymmetristen salauksien tarkoituksena on taata verkon ylitse tapahtuvan viestinnän tietoturvallisuus. Salauksen avulla voidaan taata viestinnän luottamuksellisuus ja eheys. Kuvassa 17 on esitetty salausmenetelmien perustarkoitus eli varmistaa, että lähetettyä viestiä on mahdoton lukea tai muuttaa matkalla ulkopuolisen toimesta. Viesti kirjoitetaan salattuun muotoon eli tiivisteeseen avaimella ja salaukseen purkamiseen tarvitaan purkamiseen soveltuva avain, ilman oikeaa salattua avainta viesti on muodossa, jossa sisältö ei ole tulkittavassa muodossa. [40, s.15–16; 41.]




Kuva 17. Salausmenetelmien perustarkoitus [42].

Asymmetriset menetelmät eli julkisen avaimen menetelmät on esitelty tarkemmin alaluvussa 5.2. Julkisten avainten hyvä puoli on se, että jokaisella asiakkaalla on omat avaimensa, jolloin voidaan varmistua siitä, että saapunut viesti on saapunut juuri tietyltä asiakkaalta. Symmetrisissä menetelmissä salauksen muodostamiseen käytetään yhteistä salaista avainta. Yhteisen salausavaimen käytön huono puoli on se, ettei sitä voida hyödyntää tietyn viestin asiakkaan varmentamiseen, sillä kaikki viestit on kirjattu samalla avaimella. Salausmenetelmästä riippumatta avainten luominen ja todentaminen tapahtuu usein täysin automaattisesti osana yhteyksien luomista, eikä käyttäjä huomaa tätä käytännössä ollenkaan, koska menetelmien viiveetkin ovat hyvin pienet. Kuvassa 18 on esitetty molempien menetelmien toiminta viestin salaamisessa. [38; 40, s.6; 43, s.20–22.]



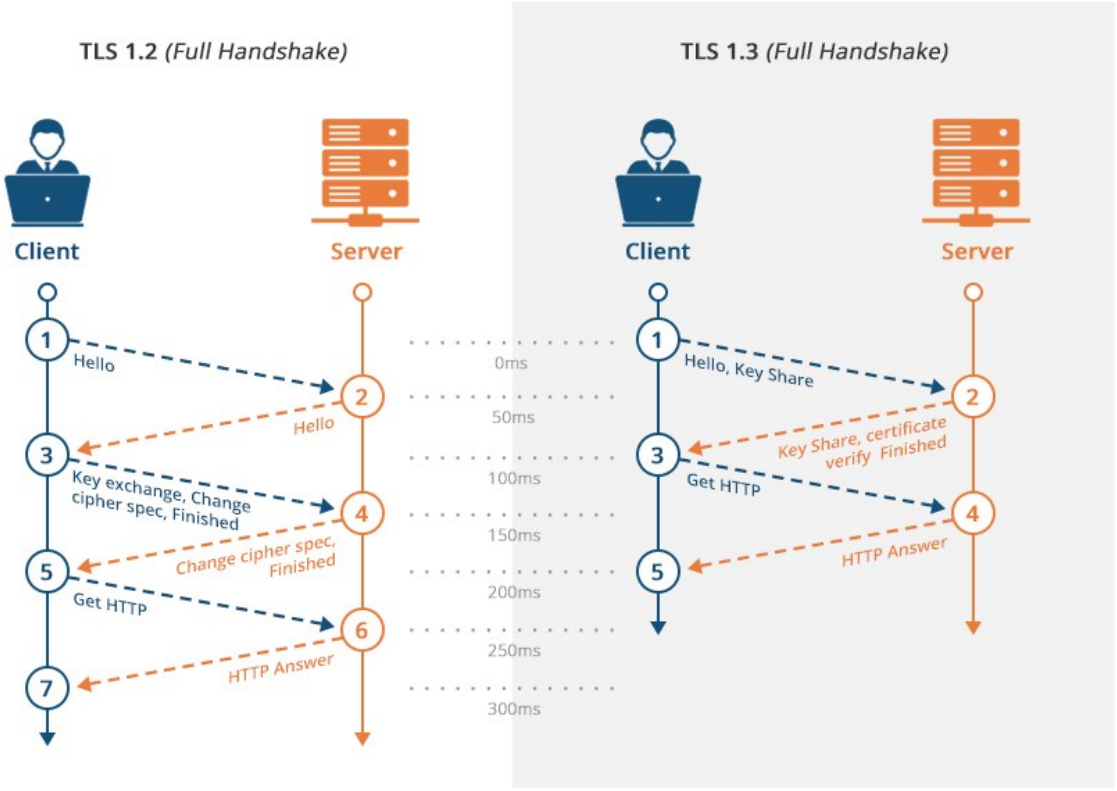

Kuva 18. Symmetrinen ja asymmetrinen salausmenetelmä sekä kryptograafinen tiivistefunktio Secure Hash Algorithm [44].

MQTT hyödyntää salaus- ja varmennusmenetelmänään TCP/IP-protokollan Transmission Layer Security -salausprotokollaa (TLS-protokolla), jonka aiempi versio tunnetaan Secure Socket Layer -protokollana (SSL-protokolla). TLS on asymmetristä salausmenetelmää hyödyntävä protokolla. TLS on IP-verkkojen ja monien sovellusten verkkoliikenteen päästä päähän suojaamiseen käytetty salausprotokolla. Lisäksi TLS-protokollaan käytetään datan muuttumattomuuden todentamiseen. Tunnetuin protokollan hyödyntämiskohde on hypertekstin siirtoprotokollan (HTTP, Hypertext Transfer Protocol) salaaminen TLS-protokollalla, josta käytetään termiä Hypertext Transfer Protocol Secure (HTTPS), jota käytetään internetsivujen ja asiakkaan viestinnän salaamiseen. [40, s.25–26.]

## 5.2 TLS-kättely

TLS on sovellustasolla toimiva asiakas-palvelinmallin mukainen protokolla. Mallia hyödynnetään kättelyn (handshake), salausalgoritmien ja salausavainten vaihdon

määrittelemiseen [40, s.25–26]. Kuvassa 19 on kuvattu TLS-protokollan uusimpien versioiden 1.2 ja 1.3 kättely.




Kuva 19. TLS-kättely protokollan versioilla 1.2 ja 1.3 [45].

MQTT:n tapauksessa kättelyn aikana viestinvälittäjä ja asiakas autentikoivat toisensa sekä muodostavat käytettävän salausavaimen. Kättelyn tarkoituksena on luoda salattu yhteys suojausprotokollaa käyttävien asiakkaiden välillä. Kättelyn molempien osapuolien tulee käyttää salausalgoritmia, jota molemmat tukevat, joka sovitaan myös kättelyn aikana. Salausalgoritmien sopimisen yhteydessä viestinvälittäjä toimittaa käyttämänsä sertifikaatin. Sertifikaatti sisältää viestinvälittäjän salauksessa käyttämän julkisen avaimen. Tästä pakettikokonaisuudesta käytetään varmenne nimitystä. Käytettävät varmenteet voivat sijaita joko suoraan viestinvälittäjän ja asiakkaan kovalevyillä tai ne voidaan hakea jostakin verkossa sijaitsevasta tallennustilasta. [40, s.25–26; 46; 47.]

Asiakkaan ja viestinvälittäjän yhteys autentikoidaan vertaamalla, että molemmat osapuolet käyttävät samaa varmennetta. Kun molempien käyttäjien on todettu käyttävän samaa varmennetta, asiakas luo alustavan salausavaimen, jonka asiakas salaa julkisella

avaimella. Paketti alustavan salausavaimen kanssa lähetään viestinvälittäjälle, joka purkaa paketin viestinvälittäjän omalla salaisella avaimella. Tässä vaiheessa salaus on jo varmennettu niin, ettei ulkopuolinen osapuoli pysty lukemaan lähetettyjä paketteja, sillä paketit puretaan käyttäjien omilla salaisilla avaimilla julkisen avaimen sijaan. Alustavan salausavaimen avulla luodaan varsinaiset salausavaimet, joilla toteutetaan varsinaisen viestinnän salaaminen. [46; 47.]

Salaukseen käytettävät sertifikaatit ja avaimet voidaan ostaa kaupallisina versioina tai ne voidaan luoda maksuttomasti itse siihen tarkoitetuilla ohjelmistoilla, kuten OpenSSL:llä. MQTT-protokollan kanssa käytettäviä SSL-sertifikaatteja ovat domain-varmennetut (Domain Validated, DV) sertifikaatit ja laajennetusti varmennetut (Extended Validation, EV) sertifikaatit. Käytetyimpiä sertifikaatteja MQTT-protokollan kanssa ovat kuitenkin domain-varmennetut X.509-standardin mukaiset sertifikaatit. Salaukseen käytettävä varmennekokonaisuus sisältää salaukseen käytettävän avaimen sekä identiteettitiedot. Identiteettitiedot sisältävät esimerkiksi suojattavan kohteen nimen sekä useita sertifikaatin hakijaa tai luojaa koskevia tietoja, joita ovat esimerkiksi nimi, osoite, yhteystiedot. [47; 48.] Sertifikaattien ja avaimien käyttäminen osana MQTT-viestinvälitystä esitellään luvussa 6.5.

## 5.3 Asiakkaan autentikointi

Salausmenetelmillä voidaan salaamisen lisäksi varmistua viestien eheydestä sekä asiakkaan sekä viestinvälittäjän autentikoinnista. Eheyden ja käyttäjän todentamiseen voidaan hyödyntää samoja tai samankaltaisia avaimia kuin itse salaukseen. Sertifikaatit ja avaimet voidaan TLS-salauksen tapaan hankkia joko kaupallisina versioina tai luoda itse. Hyvin suojatussa järjestelmässä jokainen asiakas käyttää omaa autentikointivarmenneettaan, jolloin voi olla käytännöllisempää luoda varmenteet itse, varsinkin jos liitettyjen asiakkaiden määrä on suuri. MQTT-protokollan käyttäessä TLS-salausta ei asiakkaan tarvitse kuitenkaan todentaa itseään. Liittyvän asiakkaan autentikointi on kuitenkin tärkeä osa turvallista järjestelmää. [47; 48.]

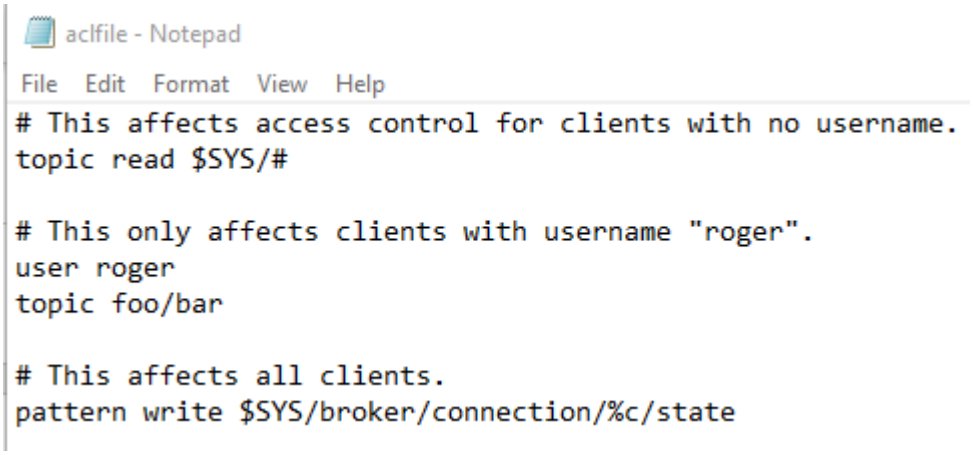
Yksittäisten asiakkaiden autentikoimiseen käytettävien varmenteiden luominen manuaalisesti on kuitenkin työlästä, jos asiakasmäärä ja asiakkaiden vaihtuvuus on suurta, joten muilta osin kattavan tietoturvan katsotaan usein riittävän. Tähän MQTT tarjoaa

asiakasvarmenteiden käyttämisen. Toiminto määritellään viestinvälittäjän suoritustiedostossa komennoilla *allow_anonymous false* ja *require_certificate true*. [49.] Komentojen sijoittaminen viestinvälittäjän suoritustiedostoon esitellään tarkemmin luvussa 6.4.

## 5.4 Pääsynvalvontaluettelo

Palveluiden käyttäjien oikeuksien hallitsemiseen on käytössä erilaisia pääsynhallintaratkaisuja. Käytetyimpiä pääsynhallintaratkaisuja ovat pääsynvalvontaluettelo (Access Control List, ACL) ja roolipohjaiset käyttöoikeudet (Role Based Access Control, RBAC). Käyttäjien oikeuksia voidaan rajata esimerkiksi tiedostojen luku- tai kirjoitusoikeuksien osalta. Pääsynvalvontaluettelossa on listattuna palvelun osia, joiden käyttämistä halutaan rajoittaa. Rajoitetun osan yhteydessä mainitaan poikkeukset, eli käyttäjät, jotka voivat poiketa rajoituksesta. Roolipohjaiset käyttöoikeudet ovat pääsynvalvontaluetteloa käytännöllisempi tapa rajata käyttäjien oikeuksia. RBAC määrittää jokaiselle käyttäjälle rooli-muuttujan. Rooleja voi olla useita ja jokaisella roolilla voi olla omat oikeutensa. Tyypillisiä rooleja ovat ylläpitäjä, asiantuntija ja asiakas. [50.]

MQTT käyttää käyttöoikeuksien rajoittamiseen pääsynvalvontaluetteloa. Kaikkia rajattuja otsikoita ei luetteloon tarvitse kuitenkaan luetella erikseen, vaan rajaamiseen voidaan käyttää alaluvussa 3.8 esiteltyä villikorttitoimintoa. Kuvassa 20 on esimerkki Mosquitton ACL-tiedostosta, joka on erillinen .example-tekstitiedosto mosquitton asennuskansiossa. Tiedostosta esitellään kolme erilaista tapaa tehdä käyttäjärajoituksia: kaikkia asiakkaita ilman käyttäjänimeä koskeva rajoitus, käyttäjänimi kohtainen rajoitus sekä kaikkia asiakkaita koskeva rajoitus. [51.]



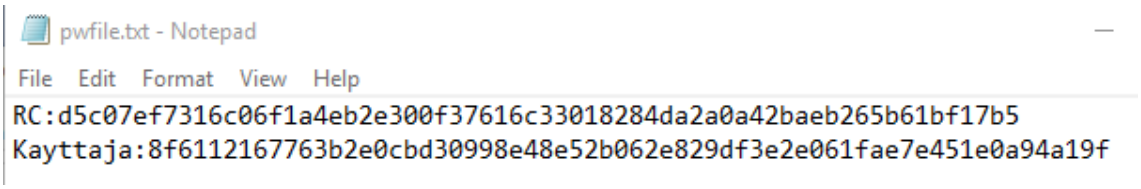

Kuva 20. Mosquitton asennuskansion pääsynvalvontaluettelo esimerkki.

Ensimmäinen esimerkki eli kaikkia asiakkaita ilman käyttäjänimeä koskeva rajoitus antaa kaikille asiakkaille, jotka toimivat ilman käyttäjänimeä (username), oikeuden tilata otsikon *$SYS/#* ja vastaanottaa sitä koskevat viestit. Toinen esimerkkirajoitus koskee käyttäjänimellisiä (username) asiakkaita. Esimerkissä vain Roger-niminen käyttäjä voi julkaista ja tilata aiheen *foo/bar*. Rajoitteen tarkentamatta jättäminen tarkoittaa samaa komentoa kuin readwrite-komento. Kolmannessa esimerkissä esitetään tapa rajoittaa kaikkia asiakkaita, jotka käyttävät käyttäjänimeä tai kaikkia asiakkaita, jotka käyttävät asiakastunnusta. Merkinnällä *%c* viitataan Clien ID -ominaisuuteen. Esimerkissä asiakas, joka käyttää asiakastunnusta voi julkaista vain asiakastunnuksensa mukaisen aiheen, asiakas nimeltä *sensori* voi tehdä julkaisun vain aiheeseen *$SYS/broker/state/**sensori**/state*. Rajoitus voitaisiin osoittaa kaikille käyttäjänimeä käyttäville asiakkailla merkinnällä *%u*, jolloin rajoitus pitäisi kirjata tiedostoon muodossa *$SYS/broker/state/%u/state*. Rajoituksista ei ilmoiteta tilauksien tai julkaisujen yhteydessä asiakkaille, joten ainut tapa huomata rajoitus on tilata asiakkaan julkaisuyritys ja tarkistaa, ilmestyykö asiakkaan julkaisu. [51.]

ACL-toiminto voidaan ottaa käyttöön Mosquitto:n .conf-suoritustiedostossa komennolla *acl_file c:\mosquitto\aclfile.example*. Komento löytyy Default authentication and topic access control -nimisen otsikon alta. Tekstitiedostoon voi tehdä tarvittavia muokkauksia vapaasti, mutta rajoitteiden käyttämiseen kannattaa luoda jokin toimintamalli. Tuotantotehtaan käyttäjärajoitusten toimintamallissa sensorit voisivat julkaista vain oman nimensä mukaisiin aiheisiin, kun taas järjestelmän valvomosovellus voisi tilata kaikkia aiheita ja tehdä muutoksia laitteiden ohjauksia koskeviin aiheisiin. [51.]

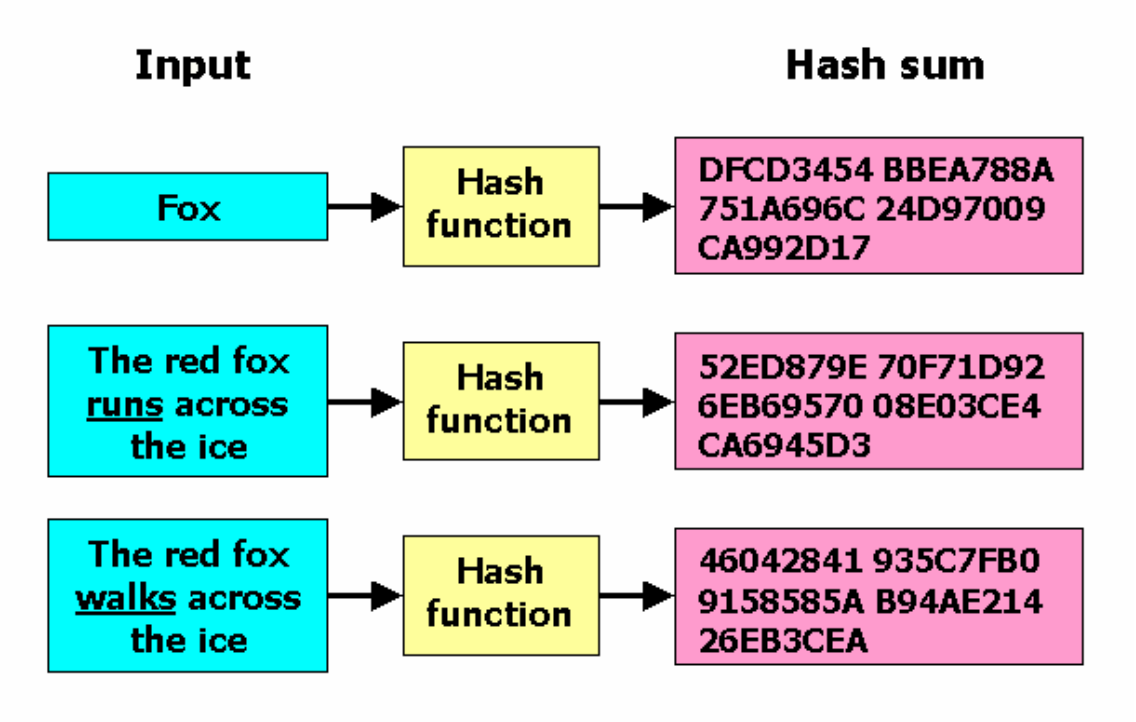
## 5.5 Käyttäjä- ja salasanasuojaus

MQTT viestinvälittäjät sisältävät käyttäjä- ja salasanasuojaus ominaisuuden. Ominaisuuden ollessa käytössä asiakkaan tulee lähettää välittäjälle käyttäjätunnus sekä salasana CONNECT-paketin mukana. Jos asiakas ei toimita vaadittuja parametrejä, liittymispyyntö evätään. Käyttäjätunnus ja salasana on määritetty välittäjälle aina pareina. Yhdelle välittäjälle voi olla tallennettuna useita eri käyttäjätunnus- ja salasanapareja. Käyttäjä- ja salasanatiedot välitetään oletuksena salaamattomassa muodossa, joten viestinvälitys kannattaa salata TLS/SSL-salauksella kirjautumistietojen suojaamiseksi. [52.] Kuvassa 21 on esitetty Mosquitton salasanatiedosto.




Kuva 21. Mosquitto-viestinvälittäjän salasanatiedosto.

Mosquitto-viestinvälittäjälle suojaus voidaan ottaa käyttöön .conf-tiedostossa käyttämällä komentoja *allow_anonymous false* ja *password_file C:\Program Files\mosquitto\pwfile.txt*. Komennoissa esitetty hakemisto viittaa kuvan 21 kaltaiseen käyttäjä- ja salasanatiedostoon. Mosquitto tarjoaa mahdollisuuden salata tiedoston salasanat myös säilytystä varten Hash-funktiolla. Funktiolle syötetään tulona salattava merkkijono selkeässä muodossa, jolloin lähtönä saadaan merkkijono salatussa muodossa kuvan 22 mukaisesti. [52.]




Kuva 22. Tekstin salaaminen Hash-funktiolla [53].

Salasanatiedosto voidaan ottaa käyttöön luomalla txt-tiedosto, johon kirjataan käyttäjätunnus ja sille salasanapari. Salasana voidaan salata komennolla *mosquitto_passwd -U pwfile.txt*. Mosquitto tarjoaa myös komentoja, joilla tiedosto ja salaaminen voidaan luoda suoraan komennoilla. [52.]

## 5.6 Asiakastunnus etuliite

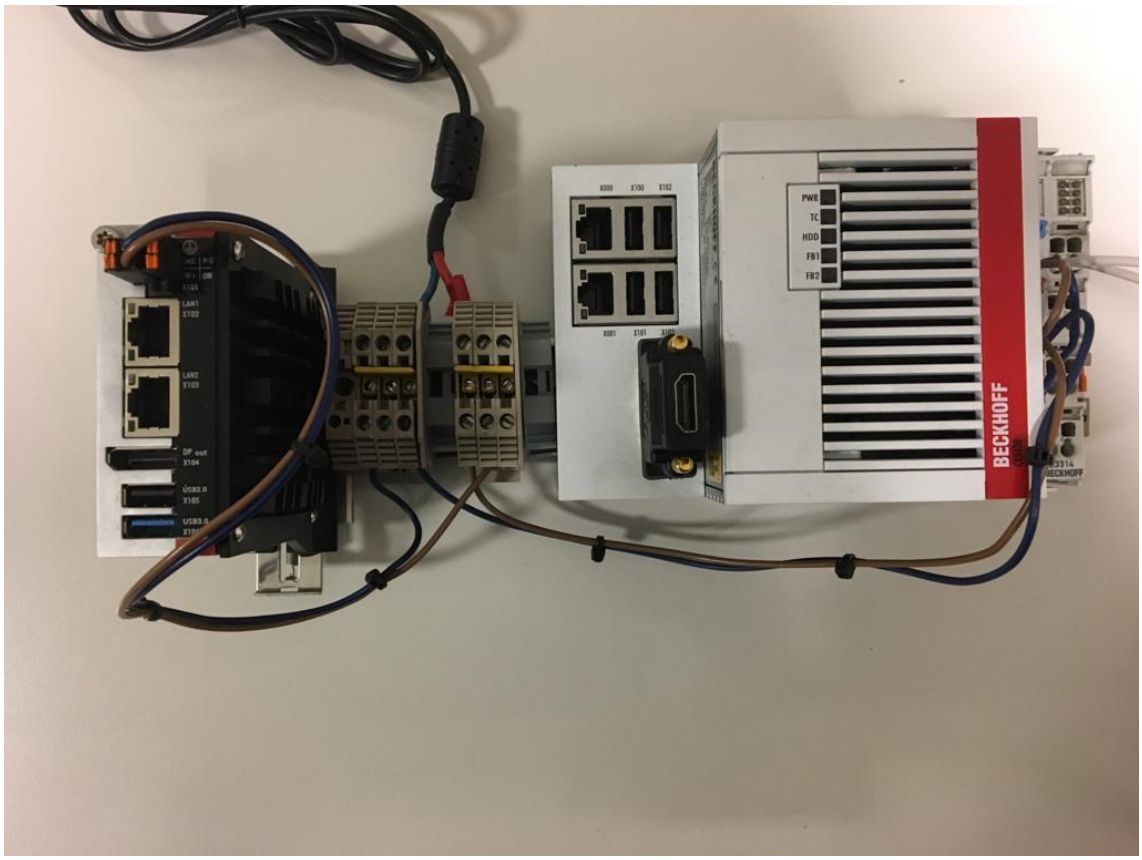
Luvussa 4.1 kuvatulla tavalla, jokaisella MQTT:n asiakkaalla on asiakastunnus. Asiakastunnusetuliite on protokollan tietoturvaominaisuus, jolla välittäjä voi vaati liittyvältä asiakkaalta halutuilla merkeillä alkavaa asiakastunnusta. Tietoturvaominaisuus voidaan ottaa käyttöön Mosquitto:n .conf-suoritustiedostossa kirjoittamalla *komennon clientid_prefixes* jälkeen haluttu etuliite. Asetuksella *clientid_prefixes suojattu-* täytyisi liittyvän asiakkaan asiakastunnus olla esimerkiksi *suojattu-sensori1*. [39; 54.]

# 6 Käytännön osuuden toteutus

Tässä luvussa käydään läpi käytännön osuutena tehtyjä kehitysympäristön luomisen työvaiheita. Käytännön toteutuksena syntyvän kehitysympäristön on tarkoitus demonstroida testeri, joka julkaisee lämpötilatietoa sekä tilaa lämpötilan asetustiedon. Tarkoituksena on demonstroida testerin tuottaman datan siirtämistä testeriltä testiautomaatiojärjestelmään sekä mahdollisuutta kontrolloida testeriä järjestelmällä käyttäen MQTT-protokollaa. Toimintoja otettiin käyttöön asteittain siten, että jokaisen uuden välittäjää tai asiakasta koskevan toiminnon käyttöönoton jälkeen asiakkaan ja välittäjän välinen yhteyden toiminta testattiin. Tässä luvussa käytännön työn vaiheet on kuitenkin esitetty peräkkäin ilman erillistä mainintaa käyttöönotettujen toimintojen testaamisesta. Opinnäytetyössä esitettävät koodiesimerkit ovat kommentoimattomia tilan säästämisen ja kuvien selkeyttämisen vuoksi, mutta liitteessä 1 on esitetty TwinCAT 3 -ohjelmistolla tehty aliohjelma kokonaisuutena.

## 6.1 Fyysinen käyttöympäristö

Fyysinen kehitysympäristö koostuu kolmesta pääkomponentista: reitittimestä, jota käytetään erotetun automaatioverkon demonstroimiseen sekä esitetty kuvassa 23 vasemmalta oikealle esitetyt komponentit: teollisuustietokone (Industrial PC, IPC) Beckhoff C6015-0010 ja ohjelmoitava logiikka (programmable logic controller, PLC) Beckhoff CX5130. Ohjelmoitavaan logiikkaan on kiinnitetty analoginen termopareille tarkoitettu tulomoduuli EL3314. Muita kehitysympäristön oleellisia komponentteja on 24VDC:n virtalähde ja K-tyypin termopari.




Kuva 23. Kehitysympäristön suoritinlaitteet

Teollisuustietokone toimii kehitysympäristön palvelimena/pilvipalveluna, jossa MQTT-viestinvälittäjä- ja asiakassovelluksia suoritetaan. Ohjelmoitava logiikka kuvaa testiautomaation testeriä, joka lähettää lämpötilatietoja sekä tilaa asetustietoja ja asiakassovellustietoja.

## 6.2 Node-RED:n käyttöönotto

Node-RED on tietovuo-ohjelmointiin (flow-based programming) perustuva IBM:n Emerging Technology Services -tiimin vuonna 2013 kehittämä avoimen lähdekoodin kevyt visuaalinen ohjelmointityökalu. Node-RED kehitettiin erityisesti MQTT-protokollan viestinnän selkeyttämistä varten, mutta nykyään Node-RED on laajalti käytössä eri IoT-protokollien ohjelmointiin. Tietovuo-ohjelmoinnin kehitti 1970-luvulla J. Paul Morrison. Tietovuo-ohjelmointi on toiminnaltaan hyvin samankaltainen kuin toimilohko-ohjelmointi (Function Block Diagram, FBD). Node-RED ohjelmointityökalun laatikoita kutsutaan

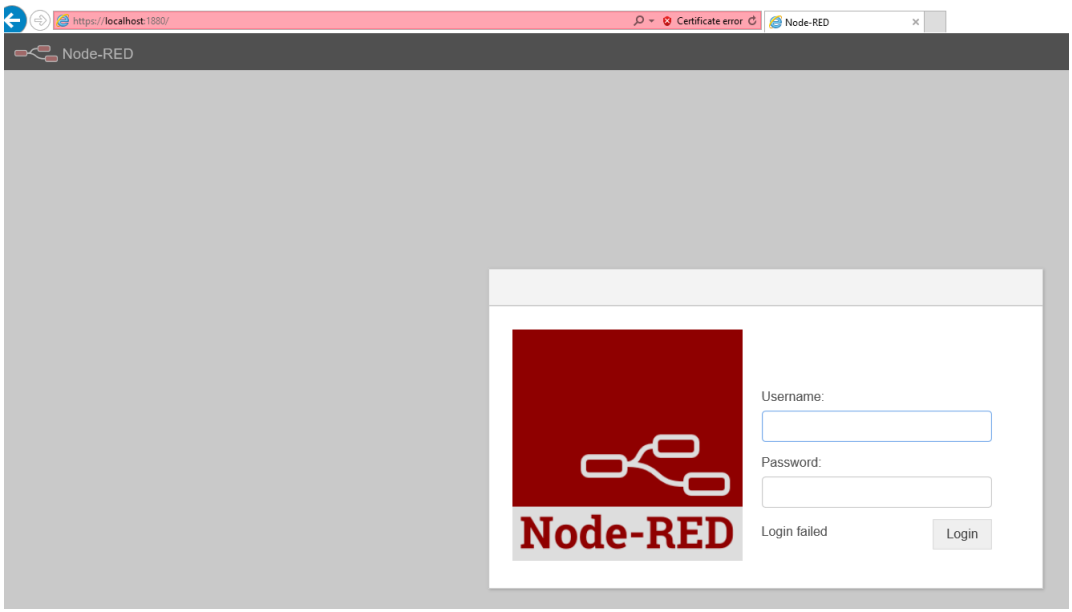
solmuiksi (node). Yksittäinen solmu voi sisältää useita rivejä koodia, jolloin pitkätkin ohjelmat voidaan koota yksittäisistä solmuista, jotka johdotetaan toisiinsa. Yksittäinen node koostuu .json-, .js- ja .html-tiedostoista. Solmujen muodostamaa ohjelmakokonaisuutta kutsutaan vuoksi, jotka ovat tiedostomuodoltaan JSON-tiedostoja. Node-RED-ohjelmointi sekä ohjelman suorittaminen toteutetaan Web-palvelin sovelluksella. Tyypillisessä ratkaisussa Node-RED-ohjelmointiympäristöä suoritetaan palvelimella tai pilvessä. [55; 56; 57.]

Node-RED käyttää suoritustiedostonaan JavaScript-tiedostoa, joka on toiminnaltaan hyvin samankaltainen kuin Mosquitto-viestinvälittäjän suoritustiedosto. Suoritustiedosto on käyttäjän vapaasti muokattavissa ja pakollisia asetuksia on vain muutamia. Tässä insinöörityössä haluttiin suojata myös Node-RED-käyttöympäristö, joten suojausta varten luotiin tarvittavat varmenteet sekä käyttäjä- ja salasanaparit. Varmenteet luotiin OpenSSL-sovelluksella ja salasana kryptattiin Node-REDin omalla komennolla. Näiden työvaiheiden jälkeen suoritustiedostossa otettiin käyttöön tarvittavat komennot. Esimerkkikoodissa 1 on esitetty valmis suoritustiedosto käyttöönotettujen toimintojen osalta.

```
var fs = require("fs");

adminAuth: {
        type: "credentials",
        users: [{
            username: "Admin",
            password:"$2b$08$QbsTtIA07X8UpxoeT5JR3uN1/c0A",
            permissions: "*"
        },
            {
            username: "Kayttaja",
            password: "$2b$08$XwJJA7QIYeKiRrOa07il..vtoL/",
            permissions: "read"
        }]
    },

httpNodeAuth: {user:"Admin",pass:"$2b$08$HKEPrWwPcpFMme"},

httpStaticAuth: {user:"Admin",pass:"$2b$08$jIr4unAGtPWri"},

https: {
     key: fs.readFileSync('c:/Users/Administrator/.node-red/nodecerts/node-
key.pem'),
     cert: fs.readFileSync('c:/Users/Administrator/.node-red/nodecerts/node-
cert.pem')

requireHttps: true,
```

Esimerkkikoodi 1. Ote suojatun Node-REDin käyttöympäristön suoritustiedostosta.

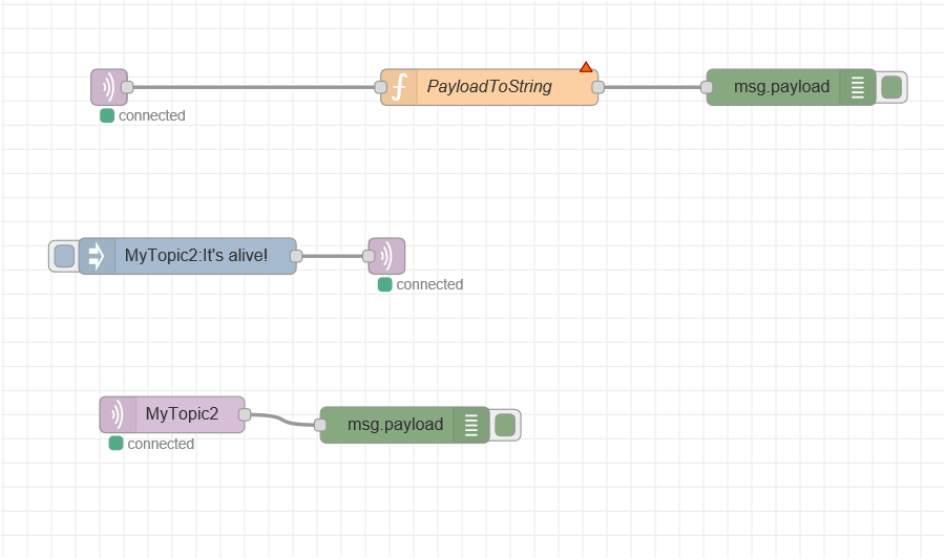
Käyttöönotettujen toimintojen jälkeen Node-RED vaatii kirjautuvalta asiakkaalta käyttäjätunnuksen, salasanan ja käytössä on salattu HTTPS-yhteys kuvan 24 mukaisesti.




Kuva 24. Node-RED-käyttöliittymän web-sivu käyttöönotetulla suojauksella.

Vuon luominen

Vuo luodaan Node-REDin visuaalisessa ohjelmointiympäristössä. Käyttäjä voi halutessaan luoda omat solmut, mutta Node-RED tarjoaa käyttäjilleen yli 225 000 solmua sisältävän kirjaston. Kirjastosta löytyvät tarvittavat solmut MQTT-aiheiden julkaisuun ja tilaamiseen, joten ne tarvitsee vain hakea kirjastosta ja raahata ohjelmointiympäristöön. Solmuista löytyvät kaikki tarvittavat ominaisuudet kuten varmenteiden ja salasanojen asettamiseen suojattua MQTT-tiedonsiirtoa varten. Solmujen yhdistämisen jälkeen voidaan luotu vuo ottaa käyttöön (Deploy). Kuvassa 25 on esitetty valmis vuo, jossa on yksi solmuyhdistelmä aiheen julkaisua varten ja toinen solmuyhdistelmä aiheen tilaamista varten. [55; 56; 57.]

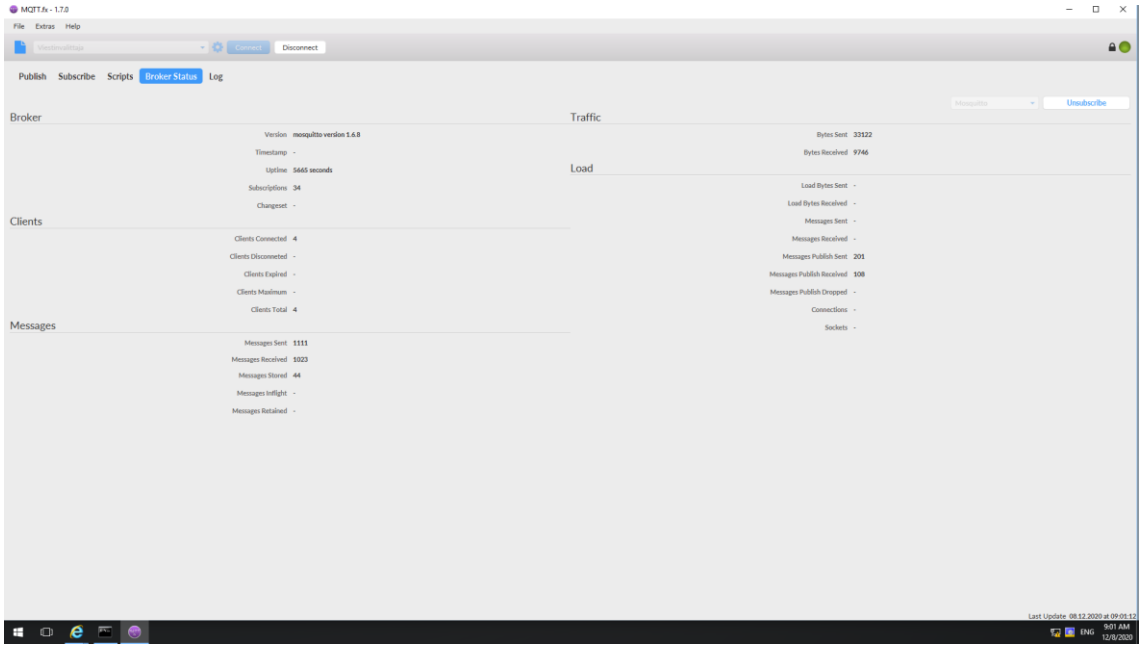



Kuva 25. Valmis Node-RED-vuo MQTT-viestien julkaisua ja tilausta varten.

Onnistunut yhteyden luominen MQTT-viestinvälittäjään ilmaistaan vihreillä connected-muuttujilla MQTT-solmujen yhteydessä. Solmut, joissa on liitäntä vain oikealla puolella, ovat lähtöjä (output). Solmut, joiden liitäntä on vain vasemmalla puolella, ovat tuloja (input), kun taas solmut, joiden läpi yhteys kulkee ovat datan käsittelyyn tarkoitettuja solmuja.

## 6.3 MQTT.fx:n käyttöönotto

MQTT.fx on JavaFX-pohjainen graafinen MQTT-viestien julkaisu- ja tilaustyökalu. Työkalu on luotu Eclipse Pahon MQTT Java-kirjastolla. MQTT.fx on ladattavissa maksuttomasti, ja se on erittäin helppo ottaa käyttöön. [58.] Käyttäjän tarvitsee täyttää vain viestinvälittäjän vaatimukset CONNECT-paketin osalta, jonka jälkeen yhteys viestivälittäjään voidaan luoda. Samalla CONNECT-paketilla voidaan tehdä useita tilauksia ja julkaisuja. Kuvassa x on esitetty työkalun perusnäkymä yhteyden muodostamisen jälkeen.




Kuva 26. MQTT.fx-työkalun perusnäkymä, kun se on yhteydessä viestinvälittäjään.

Työkalu helpottaa huomattavasti muun muassa viestinvälittäjään liitettyjen asiakkaiden ja julkaistujen viestin määrään tarkastelua, joita voidaan tarkastella Broker Status -välilehdellä. Lisäksi Log-välilehdellä voidaan tarkastella viestinvälittäjän lokitietoja MQTT-viestimuodossa.

## 6.4 Eclipse Mosquitto -viestinvälittäjän käyttöönotto

Tässä aliluvussa esitellään pelkästään Mosquitto-viestinvälittäjän käyttöönotto. Viestinvälittäjän tarkempia ominaisuuksia käytiin läpi tarkemmin aliluvussa 3.6. Esimerkkikoodissa 2 on esitetty viestinvälittäjän suoritustiedosto käyttöönotettujen komentojen osalta.

```
port 8883
cafile C:\Program Files\mosquitto\certs\ca.crt
keyfile C:\Program Files\mosquitto\certs\server.key
certfile C:\Program Files\mosquitto\certs\server.crt
tls_version tlsv1.3
require_certificate true
allow_anonymous false
password_file C:\Program Files\mosquitto\pwfile.txt
```

Esimerkkikoodi 2. Ote Eclipse Mosquitton suoritustiedostosta, jossa on otettu käyttöön tietoturvatoimintoja.

Esimerkkikoodilla 2 toimiva Mosquitto-viestinvälittäjä vaatii liittyvää asiakasta autentikoimaan itsensä asiakasvarmenteilla sekä käyttäjätunnus- ja salasanaparilla. Lisäksi asiakkaan ja viestinvälittäjän yhteys on TLS-salattu Certificate authority -varmenteella (CA-varmenne). Asiakasvarmenteet sekä CA-varmenne on luotu OpenSSL-sovelluksella.

## 6.5 PLC-koodin luominen TwinCAT 3:lla

PLC-koodi luotiin Beckhoffin TwinCAT 3 -ohjelmistolla, joka on luotu Windowsin Visual Studion ohjelmiston pohjalle. Tässä luvussa esitettävä esimerkkikoodi pohjautuu Beckhoffin ja PLCCoder-sivuston malliratkaisuihin [59; 60]. MQTT-aliohjelma luotiin tyhjään projektiin, ja se toimii projektin ainoa ohjelma. PLC-koodi on kirjoitettu Structured Text -kielellä (ST-kieli) hyödyntäen Beckhoffin tarjoamaa MQTT-asiakaskirjastoa nimeltä Tc3_IotBase.

### MQTT-aliohjelman muuttujat

Kuvassa 27 on esitelty MQTT-aliohjelmassa käytettävät muuttujat. Kaikki aliohjelmassa käytetyt muuttujat ovat ohjelman sisäisiä, eikä globaaleja muuttujia käytetä. Suurin osa MQTT-protokollaan liittyvistä muuttujista käyttää String-merkkijonomuotoa.

```
PROGRAM MAIN
VAR
    (*Viestin julkaisun parametrit*)
    fbMqttClient: FB_IotMqttClient;
    TopicToPublish : STRING(255) := 'Temperature';
    MessageToPublish : STRING(255);
    fbSendMessageIntervalTimer : TON := (PT:=T#1S);

    Thermocouple AT %I* : INT;

    (*Viestin tilauksen parametrit*)
    fbMessageQueue: FB_IotMqttMessageQueue;
    fbMessage: FB_IotMqttMessage;
    Subscribed:BOOL;
    TopicToSubscribe : STRING(255) := 'SetTemperature';
    ReceivingTopic:STRING(255);
    ReceivingData:STRING(255);
END_VAR
```

Kuva 27. MQTT-aliohjelman muuttujat.

Ensimmäisessä muuttujia koskevassa luvussa ohjelmaan on tuotu TwinCAT 3:n MQTT-kirjasto. Lisäksi luvussa määritetään julkaisuun liittyvät muuttujat, joita ovat julkaistavan viestin otsikko, julkaistava viesti sekä viestin julkaisun aikaväli. Lisäksi muuttujissa on termoparilta arvon saava muuttuja, joka on linkitetty PLC:n tulokorttiin sekä viestin julkaisun aikaväli, joka on yksi sekunti. Toisessa muuttujia koskevassa luvussa on määritetty viestin tilauksen muuttujia, joita ovat viestijono, MQTT-viesti toimilohko, tilattavan viestin otsikko ja hyötykuorma.

CONNECT-paketti

Kuvassa 28 on esitelty aliohjelman käyttämän MQTT:n CONNECT-paketin sisältö. Paketin sisältö määritetään vain kerran ohjelman ensimmäisellä suorituskerralla. CONNECT-paketille pakollisia muuttujia ovat ainoastaan viestinvälittäjän nimi ja portti. Salatun liikenteen porttina käytetään porttia 8883. Viestinvälittäjän nimen pitää vastata sertifikaatissa määritettyä nimeä. Tässä opinnäytetyössä nimenä on käytetty IP-osoitetta.

```
(*CONNECT-Paketin parametrien asettaminen*)
IF _TaskInfo[GETCURTASKINDEXEX()].FirstCycle THEN
    fbMqttClient.stTLS.sCA := 'C:\Users\Administrator\Desktop\certs\ca.crt';
    fbMqttClient.stTLS.sCert := 'C:\Users\Administrator\Desktop\certs\client.crt';
    fbMqttClient.stTLS.sKeyFile := 'C:\Users\Administrator\Desktop\certs\client.key';
    fbMqttClient.stTLS.sVersion := 'tlsv1.3';
    fbMqttClient.sUserName:= 'RC';
    fbMqttClient.sUserPassword:= '[redacted]';
    fbMqttClient.sHostName := '[redacted]';
    fbMqttClient.nHostPort := 8883;
    fbMqttClient.sTopicPrefix := '';
    fbMqttClient.sClientId := 'Lampotilasensori';
    fbMqttClient.ipMessageQueue := fbMessageQueue;
END_IF
```

Kuva 28. CONNECT-paketin parametrit

Käyttöönotettujen tietoturva ominaisuuksien vuoksi asiakkaan tulee toimittaa CONNECT-paketin yhteydessä sertifikaatit ja avain sekä käyttäjätunnus ja salasana. Asiakastunnuksen käyttäminen helpottaa julkaistujen viestien jäljittämistä.

Aiheen julkaiseminen

Yhteyden muodostamiseen käytetään ”fbMqttClient.Execute(bConnect)”-komentoa. Komento tulee suorittaa syklisesti jokaisella ohjelmakierrolla yhteyden ylläpitämiseksi. Kuvassa 29 on esitetty viestin julkaisuun käytettävän PUBLISH-paketin sisältö.

```
(*Yhteyden muodostaminen*)
fbMqttClient.Execute(bConnect := TRUE);

(*Viestin julkaisu, PUBLISH-paketti*)
IF fbMqttClient.bConnected THEN
    fbSendMessageIntervalTimer(IN:=TRUE);
    IF fbSendMessageIntervalTimer.Q THEN
        fbSendMessageIntervalTimer(IN:=FALSE);
        MessageToPublish := CONCAT('Lampotila: ',REAL_TO_STRING(Thermocouple / 10.0));

        fbMqttClient.Publish(sTopic:= TopicToPublish,
            pPayload:= ADR(MessageToPublish),
            nPayloadSize:= LEN2(ADR(MessageToPublish))+1,
            eQoS:= TcIotMqttQos.AtMostOnceDelivery,
            bRetain:= FALSE,
            bQueue:= FALSE);
    END_IF
END_IF
```

Kuva 29. Aiheen julkaisemiseen käytetty koodi.

Viestin julkaisemiseksi Publish-paketin pakollisia muuttujia ovat viestin otsikko, hyötykuorma, nPayloadsize ja palvelunlaatutaso. Paketissa on määritetty julkaistavan viestin hyötykuorma, joka on *Lampotila: lämpötila-arvo/10.0*. Termoparilta mitattua lämpötila-arvo jaetaan kymmenellä skaalauksen vuoksi. bRetain-muuttuja on tärkeä asettaa todeksi, jos lähetetty viesti halutaan säilyttää. Luodussa aliohjelmassa julkaistava viesti lähetetään palvelun laatutasolla 0, eli viesti lähetetään viestinvälittäjälle ainoastaan kerran.

Aiheen tilaaminen

Aiheen tilaaminen on jaettu kahteen osioon MQTT-aliohjelmassa varsinaiseen tilauksen suorittavaan osioon sekä tilattujen viestien jonon tarkastelun suorittavaan osioon. Kuvassa 30 on esitelty PLC-koodi aiheen tilaamisen osalta.

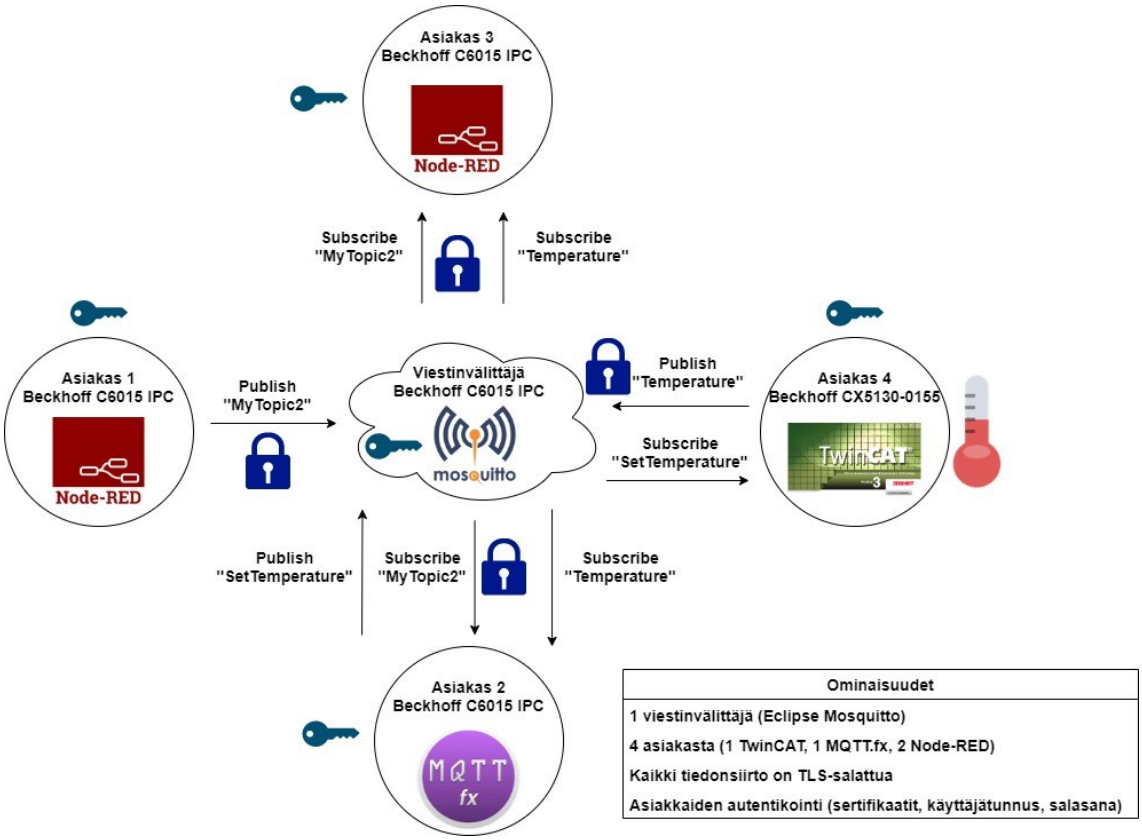
```
(*Viestin tilaus, SUBSCRIBE-paketti*)
IF fbMqttClient.bConnected THEN
    IF NOT Subscribed THEN
        Subscribed := fbMqttClient.Subscribe(sTopic:=TopicToSubscribe,
        eQoS:=TcIotMqttQos.AtMostOnceDelivery);
    END_IF
END_IF

 (*Viesti jonon tarkastelu*)
IF fbMessageQueue.nQueuedMessages > 0 THEN
    IF fbMessageQueue.Dequeue(fbMessage:=fbMessage) THEN
        fbMessage.GetTopic(pTopic:=ADR(ReceivingTopic), nTopicSize:=SIZEOF(ReceivingTopic) );
        fbMessage.GetPayload(pPayload:=ADR(ReceivingData), nPayloadSize:=SIZEOF(ReceivingData), bSetNullTermination:=FALSE);
    END_IF
END_IF
```

Kuva 30. Aiheen tilaamiseen käytetty koodi.

Luodussa aliohjelmassa julkaistu viesti tilataan palvelun laatutasolla 0, eli viestinvälittäjä lähettää viestin tilaajalle ainoastaan kerran. Toimilohkon fbMessageQueue parametri cMaxEntriesInMqttMessageQueue määrittää viestijonoon tallennettavien viestien määrä oletuksena tuhanteen viestiin. Kyseistä parametria voidaan muuttaa tarvittaessa. Oletuksena yhden viestin maksimikoko on 100 kilotavua ja koko viestijonon 1000 kilotavua. Toiminnaltaan toimilohko on First In First out (FIFO) periaatteen mukainen, eli ensimmäisenä saapunut viesti luetaan jonosta ensimmäisenä. Jonon ylikirjoitus toiminnon toimintatapa voidaan määrittää ”bOverwriteOldestEntry”-muuttujalla, jos muuttujan arvo asetetaan arvoon tosi *TRUE* vanhin viesti hävitetään jonosta ensimmäisenä. Mikäli muuttujan arvo on epätosi *FALSE*, uusin viesti hävitetään ensiksi. [61.]

## 6.6 Kehitysympäristö toiminnassa

Kuvassa 31 on kuvattu insinöörityössä luodun kehitysympäristön toiminta. Keskellä kuvaa on teollisuustietokoneella toimiva Mosquitto-viestinvälittäjä. Viestinvälittäjän vasemmalla puolella ja ylä- ja alapuolella on teollisuustietokoneella toimivat Node-RED ja MQTT.fx-asiakassovellukset. Oikealla puolella viestinvälittäjää on testeri-PLC, joka mittaa huonelämpötilaa.




Kuva 31. Insinöörityössä luodun kehitysympäristön järjestelmäarkkitehtuuri [57; 58; 61; 62].

Tärkeimmät osat syntynyttä kehitysympäristöä on viestinvälittäjä sekä testeri Asiakas 4. Mosquitto-suoritustiedoston mukaisesti kaikki viestintä on TLS-salattua, ja viestinvälittäjä vaati kirjautuneilta asiakkaita todentamaan itsensä sertifikaatilla sekä käyttäjätunnus- ja salasanaparilla. Asiakas 4 julkaisee mittaamansa lämpötiladataa otsikolla *Temperature*, minkä lisäksi testeri tilaa Asiakkaan 2 julkaisevan lämpötilan asetustiedon *SetTemperature*. Testerin julkaisevan lämpötiladatan tilaa asiakkaat 2 ja 3. Asiakas 1 julkaisee tekstimerkkijonon NodeREDin graafisen käyttöliittymän kytkintä painattaessa. Tämän merkkijonon tilaa Asiakas 3 ja Asiakas 2.

# 7 MQTT-protokollan soveltuvuus testiautomaatiojärjestelmään

Tässä luvussa käydään läpi tutkimustyössä selvitettyjä ominaisuuksia ja niiden hyödyntämistä kohdeyrityksen testiautomaatiojärjestelmässä. Protokollan soveltuvuutta testiautomaatiojärjestelmään pohtiessa ei siihen liittyviä varsinaisia ominaisuuksia enää

toisteta, vaan niihin viitataan tämän insinöörityön lukua merkitsevällä numerolla, jossa itse ominaisuuden hyödyt ja mahdolliset haitat on esitelty.

## 7.1 Vaatimusmäärittely

Tässä alaluvussa esitetyt vaatimukset testiautomaatiojärjestelmälle on määritetty yhdessä asiakasyrityksen asiantuntijoiden kanssa sekä Salmisen opinnäytetyössä esitetyn vaatimusmäärittelyn pohjalta [1]. Testiautomaatiojärjestelmään valittavan protokollan arvioimiseksi luotiin kevyt vaatimusmäärittely, jonka tarkoituksena on varmentaa, että valittu protokolla täyttäisi vähintään nykyisen ADS-protokollan ominaisuudet. Lisäksi vaatimusmäärittelyyn kirjattiin ne uuden järjestelmän ominaisuudet, jotka protokollan vaihdoksella voitaisiin ja haluttaisiin saavuttaa. Taulukossa 3 on esitelty luotu vaatimismäärittely automaatiotestausjärjestelmän tiedonsiirrolle. Taulukon 3 luku sarakkeeseen on merkitty insinöörityön luku, jossa vaatimuksen täyttävää ominaisuutta käsitelty. Luku sarakkeen avulla ominaisuutta koskevaan teoriaan on tarvittaessa helpompi palata.

Taulukko 3. Vaatimusmäärittely automaatiotestausjärjestelmän tiedonsiirrolle

| **Vaatimus** | **MQTT** | **Luku** |
|---|---|---|
| Tietoturva vähintään nykyisellä tasolla | ✔ | 5 |
| Saatavuus, luotettavuus ja eheys vähintään nykyisellä tasolla | ✔ | 5 |
| Yhteensopivuus Beckhoffin laitteistojen kanssa | ✔ | 3.4&3.5 |
| Järjestelmää käyttävät asiakkaat pitää olla jäljitettävissä | ✔ | 4.1&4.7 |
| Järjestelmään tallennettu data pitää olla jäljitettävissä | ✔ | 4.1&4.7 |
| Asiakkaiden pitää olla mahdollista lähettää dataa aikamääreisiin perustuen | ✔ | 6.5 |
| Protokollan pitää mahdollistaa skaalattavan järjestelmän ylläpitäminen | ✔ | 3.2 |
| Valvomosovellusta pitää olla mahdollista käyttää henkilökohtaisilla tietokoneilla | ✔ | 3.4 |
| Protokolla pitää olla liitettävissä pilvipalveluihin ja niiden tietokantoihin | ✔ | 3.4&3.5 |

## 7.2 MQTT-protokollan tarjoamia uusia ominaisuuksia

Automaatiojärjestelmän tiedonsiirrolle asetettujen vaatimusten täyttämisen lisäksi MQTT-protokolla tarjoaa runsaasti uusia ominaisuuksia. Tässä luvussa esitellään MQTT:n automaatiojärjestelmän kannalta tärkeimpiä saavutettavia hyötyjä.

### Tietoturva

MQTT-protokollan tietoturva voidaan taata kattavasti luvussa 5 esitetyin keinoin. MQTT itsessään tarjoaa mahdollisuuden pääsynvalvontaluettelon ja käyttäjä- ja salasanasuojauksen sekä asiakasetuliitteen käyttöön, joiden lisäksi TCP/IP-pinon päällä toimiminen mahdollistaa TLS/SSL-salauksen sekä asiakasvarmenteiden käyttämisen. Nämä kaikki tietoturvaominaisuudet tuovat kattavan määrän erilaisia suojausyhdistelmiä, joista käyttöön voidaan ottaa vain tarvittavat. Tietoturvaominaisuudet mahdollistavat matalamman tietoturvatason omaavien laitteiden, kuten mobiililaitteiden liittämisen osaksi testausautomaatiojärjestelmää. Lisäksi tiedonsiirtoprotokollan tukemat tietoturvaominaisuudet tarjoavat tarpeellisia lisäsuojausominaisuuksia perinteisten erotettujen automaatioverkkojen ja VPN-yhteyksien rinnalle.

### Toimivuus ADS-protokollan rinnalla/kanssa

Monien muiden IoT-protokollien tapaan MQTT ei rajoita protokollan rinnalla toimivia protokollia tai väyliä. MQTT-protokolla hyödyntää samoja komponentteja muiden protokollien tapaan kuten verkkolaitteita sekä tietokoneita ja pilvipalveluita, mutta käyttää niitä omalla tavallaan, eikä sen toimintaan vaikuta rinnalla toimivat protokollat. Viestinvälittäjä sovellusta voidaan suorittaa palvelin tietokoneella muiden tarjottavien palvelujen ohella, ja MQTT-aliohjelma voi toimia muiden aliohjelmien rinnalla osana PLC-koodia. Tämä on tärkeä ominaisuus, joka helpottaa varsinkin uuden protokollan kehitystyötä sekä käyttöönottoa, sillä MQTT-protokollaan perustuvaa automaatiojärjestelmää voidaan kehittää nykyisen järjestelmän rinnalla eikä käyttöönoton aikanakaan koko järjestelmää tarvitse välttämättä pysäyttää.

### Laiteriippumattomuus

MQTT-protokolla ei aseta protokolla käyttäville laitteille juurikaan vaatimuksia. Ainoat vaatimukset laitteille on mahdollisuus liittää laite Internet-verkkoon ja tuki jollekin tunnetulle ohjelmointikielelle, joita on esitetty tarkemmin alaluvuissa 3.4 ja 3.5. Laitteistoriippumattomuus yhdessä kattavan tietoturvan kanssa mahdollistaa useiden erityyppisten asiakaslaitteiden käyttämisen. MQTT-protokolla mahdollistaa henkilökohtaisten kannettavien tietokoneiden ja muiden mobiililaitteiden käyttämisen yhtenä automaatiojärjestelmän osana.

Data-agnostisuus

MQTT on hyötykuormansa suhteen data-agnostinen, jonka vuoksi hyötykuormana voidaan lähettää kattavasti eri tiedostotyyppejä ja -muotoja. Tämä on hyödyllinen ominaisuus testiautomaatiojärjestelmälle, jossa eri datamuotoja on aina yksittäisistä boolean-muuttujista kokonaisiin tietueisiin (struct). Tämän ominaisuuden tarve korostuu erityisesti käyttäessä MQTT-protokollaa koko tuotetestausjärjestelmän tiedonsiirtonaprotokollana. Lisäksi MQTT-protokolla tarjoaa runsaasti muita ominaisuuksia, jotka on koottu taulukkoon 4. Taulukon 4 luku sarakkeeseen on merkitty insinöörityön luku, jossa ominaisuutta käsitelty. Luku sarakkeen avulla ominaisuutta koskevaan teoriaan on tarvittaessa helpompi palata.

Taulukko 4. Testiautomaatiojärjestelmälle hyödyllisiä MQTT:n ominaisuuksia.

| **Ominaisuus** | **Luku** |
|---|---|
| Palvelunlaatutaso | 4.4 |
| Mahdollisuus muuttaa tiedonsiirron toimintaa automaatiotiimin toimesta | 3.6 |
| Käyttäjien pääsyn rajaus | 5.4 |
| Käyttäjien ominaisuuksien rajattavuus | 5 |
| Voidaan tehdä käyttäjä kohtaisia muutoskirjauksia | 5 |
| Mahdollisuus parantaa nykyistä tietoturvan tasoa protokollan avulla | 5 |
| Lähetettyjen tietojen jäljitettävyys kätevästi - esimerkiksi nimen perusteella | 4.7 |
| Asiakkaat jäljitettävissä kätevästi - esimerkiksi nimen perusteella | 3.4,3.5&3.6 |
| Protokollan standardointi | 3.3 |
| Standardien avoimuus | 3.3 |
| IP-osoite riippumattomuus | 3.2 |
| Beckhoff ADS-AmsNetId riippumattomuus | 3.2 |
| Yhteensopivuus muiden protokollien kanssa | 3.4&3.5 |
| Eheyden, saatavuuden ja luotettavuuden ylläpito/varmistaminen | 5 |
| Mahdollisuus jaksottomaan tiedonsiirtoon – julkaistaan vain uudet tapahtumat | 3.8&4.2 |
| Voidaan julkaista jo tapahtuneet tapahtumat uusille asiakkaille | 4.4 |

## 7.3 MQTT-protokollan rajoitteet

Tässä luvussa on esitetty MQTT-protokollan rajoitteet testiautomaatiojärjestelmän osalta.

### MQTT-viestien bittimuotoinen data

MQTT-viestit sisältävät bittimuotoista dataa alaluvun 3.7 mukaisesti. Vaikka hyötykuorman datamuoto voidaan vaihtaa esimerkiksi JSON-muotoon, käyttää MQTT silti bittidatamuotoa ohjauselementeissään. MQTT-viestien datamuodon vuoksi lähettäjän tulee ymmärtää MQTT-viestipakettien rakenne, jotta protokollan käyttö onnistuu ja on hyödyllistä. Lisäksi SUBSCRIBE-paketin muodostaminen ja lähettäminen vaatii tilaavalta asiakkaalta MQTT-protokollan käyttöä. Datan tallentaminen suoraan tietokantaan ole mahdollista MQTT-protokollalla, vaan määritetyn MQTT-asiakkaan tulee suorittaa datamuodon muuttaminen tietokannalle sopivaan muotoon ja suorittaa itse tallentaminen. Monet pilvipalvelut tarjoavat kuitenkin tuen MQTT-protokollalle, jolloin varsinaista MQTT-asiakas laitetta ei tarvita vaan pilvipalvelussa on itsessään erillinen asiakassovellus suorittamassa pilvessä sijaitsevaan tietokantaan tallentamisen. Lisäksi fyysisen palvelintietokoneen käyttötarkoituksen mukaan voidaan palvelintietokonetta käyttää halutessa asiakkaana pilvipalveluratkaisun tapaisesti, jolloin erillistä asiakastietokonetta datamuodon muuttamiseen ei tarvita.

### Raskas TLS-salaus

Alaluvuissa 5.1 ja 5.2 esiteltiin MQTT-protokollan käyttävän TLS-salausta, joka on hyvä sovellustason salausmenetelmä tietoturvaa ajatellen, mutta TLS-kättelyn ja salaukseen tarvittavan laskentatehon vuoksi, se on viestinnän keveyteen pyrkivään MQTT-protokollaan verrattuna raskas salausmenetelmä. Kohdeyrityksen testiautomaatiojärjestelmän tämänhetkiseen tai suunniteltuun järjestelmään salauksen vaativuudella ei ole merkitystä, sillä MQTT-protokollan TCP/IP-pinon päällä toimiminen ja raskas salausmenetelmä estävät lähinnä vain kauko-ohjain-tyyppisen reaaliaikaisen ohjauksen. Tämä

rajoite on kuitenkin hyvä tiedostaa protokollan valintaa koskevassa päätöksenteossa ja tulevaisuuden päivitettävyyttä määrittäessä.

Ohjelmistokirjastot vain TwinCAT 3 -ohjelmistolle

Tämän insinöörityön kirjoitushetkellä Beckhoff tarjoaa virallisen MQTT-kirjaston Tc3_IotBase, joka on nimensä mukaisesti ainoastaan TwinCAT 3 -ohjelmistolle, jolla koodit voidaan konfiguroida ainoastaan TwinCAT 3:a tukeville ohjelmoitaville logiikoille. Tämä rajoite tarkoittaa sitä, että MQTT-protokollaa hyödyntääkseen tarvitsisi kaikkien testiautomaatiojärjestelmän logiikoiden olla TwinCAT 3 -tuellisia. Toistaiseksi sama rajoite on myös muissa IoT-protokollista, kuten HTTPS/REST-protokollassa.

Eclipse Mosquitto -viestinvälittäjän tekstimuotoinen suoritustiedosto

Eclipse Mosquitto -viestinvälittäjän tekstimuotoinen suoritustiedosto on selkeä ja helppokäyttöinen, mutta se on työläs tapa muokata MQTT-asiakkaita koskevia asetuksia kohteissa, joissa asiakasmäärät ovat suuria. Villikorttien avulla voidaan kuitenkin vähentää muokattavien asetusten määrää.

## 7.4 Kehitystyön jatkuminen

Kehitystyötä tullaan jatkamaan MQTT-protokollan käyttöönoton ja insinöörityössä tuotetun kehitysympäristön parissa. Insinöörityön yhtenä käytännön osuuden vaiheena oli tuottaa tiedonsiirtoa koskevan kehitystyön suuntaa antava etenemismalli insinöörityön päättymisen jälkeiselle ajalle. Kehitystyölle määritettiin kolme päävaihetta sekä kolme päävaihetta tukevaa vaihetta. Tukevat vaiheet eivät kuitenkaan ole merkitykseltään vähempiarvoisia, vaan ne ovat ominaisuuksia, jotka ovat toimenpiteinä rajatumpia ja mahdollista suorittaa mikrotuen toimesta.

Kehitystyön jatkamisen päätavoitteena on saada MQTT-protokollan kehitysympäristö vastaamaan oikeaa käyttötapaa. Ensimmäisenä kehitystyön päävaiheena on jättää insinöörityössä käytetyt asiakassovellukset tukemaan kehitysympäristön monipuolisuutta ja virheen jäljitettävyyttä varsinaiseen ohjaamisen sijaan. Insinöörityössä käytetyt asiakassovellukset Node-RED ja MQTT.fx ovat hyviä sovelluksia useiden asiakkaiden

julkaisujen ja tilausten demonstrointiin sekä viestinvälittäjän lokihistorian tarkkailuun, mutta varsinaisessa testauskäytössä ei ole mielekästä käyttää erillistä sovellusta tilauksien tekemiseen. Testauskäytössä testerikohtaisten aiheiden tilaus ja ohjauskäskyjen julkaisu tulee olemaan osa varsinaista valvomosovellusta nykyisen järjestelmän tapaan, minkä vuoksi olisi hyvä kehittää tätä vastaava käyttöliittymä. Uutta valvomosovellusta ei tarvitse kehitysympäristöä varten luoda, vaan palvelinta demonstroivalle tietokoneelle voidaan luoda käyttöliittymä TwinCAT 3-ohjelmistolla kuvastamaan lopullisen ratkaisun valvomosovellusta.

Toinen päävaihe seuraa vahvasti ensimmäisen päävaiheen mallia, jossa asiakassovellukset on tarkoitus korvata todellista käyttöä vastaavilla komponenteilla. Toisen päävaiheen tarkoituksena on korvata asiakassovelluksilla luodut tilaajat ja julkaisijat jo olemassa olevilla testereillä. Opinnäytetyön käytännön osuutena kirjoitettu MQTT-aliohjelma on tarkoitus konfiguroida osaksi valittujen testerien lähdekoodia. MQTT-aliohjelmaa voidaan käyttää normaalin testauksen ohella, vaikka testerit käyttävätkin vielä nykyistä ADS-protokollaa.

Kolmantena päävaiheena on tarkoitus luoda ja ottaa käyttöön erikseen kehitystyötä varten tarkoitettu SQL-tietokanta. Tietokanta on keskeisessä osassa kohdeyrityksen tuotetestaus prosessia, joten on tärkeää päästä ottamaan tietokanta käyttöön jo varhaisessa vaiheessa järjestelmäkehitystä. Tietokantaan tallennetaan kehitysympäristöön liitettyjen testereiden tuottamaa dataa. MQTT-protokolla käyttää binääridataa, joten dataa ei voida lähettää suoraan tietokantaa viestinvälittäjän toimesta, vaan data pitää muuntaa tietokantaan soveltuvaan muotoon. Tietokantaan lähetettävä data tullaan muuntamaan JSON-tiedostomuotoon PLC-koodiin ohjelmoitavalla aliohjelmalla. JSON-muotoinen data voidaan sitten lähettää tietokantaan sellaisenaan.

Suunniteltuja tukevia työvaiheita ovat muun muassa MQTT:n käyttäjärajoitusten luominen ja testaaminen, TSL-salatun liikenteen salauksen testaaminen, esimerkiksi MQTT-spy-ohjelmistolla, sekä käytössä olevan laajemman kokonaisuuden tiedonsiirron nopeuden testaaminen, esimerkiksi MQTT.fx- tai WireShark-ohjelmistoilla.

# 8 Yhteenveto

Insinöörityön teoriaosuudessa saatiin selville, miten MQTT-protokolla toimii ja listattua ne ominaisuudet, jotka olisivat hyödyllisimpiä kohdeyrityksen testiautomaatiojärjestelmän kannalta. MQTT-protokollan soveltuvuutta kohdeyrityksen testiautomaatiojärjestelmän tiedonsiirtomenetelmäksi voidaan arvioida insinöörityössä luodun kevyen järjestelmän tiedonsiirtoa koskevan vaatimusmäärittelyn sekä protokollan ominaisuuksien ja rajoitteiden pohjalta.

MQTT-protokolla täyttää kaikki testiautomaatiojärjestelmän tiedonsiirrolle asetetut vaatimukset, minkä vuoksi protokollan voidaan todeta soveltuvan uudistuvan järjestelmän tiedonsiirtomenetelmäksi. Vaatimusten täyttämisen lisäksi MQTT-protokolla tarjoaa laajan kirjon testiautomaatiojärjestelmälle hyödyllisiä ominaisuuksia, mitä voidaan pitää tärkeänä asiana tiedonsiirtoprotokollaa valittaessa, koska protokollan laajat ominaisuudet vähentävät tarvittavien ominaisuuksien luontia valvomosovelluksiin ja lähdekoodeihin.

Tavoitteiden mukaisesti työssä saatiin selville, mitä kohdeyrityksen käyttämän ADS-protokollan vaihtaminen MQTT-protokollaan vaatisi. MQTT-protokollaa voidaan käyttää ADS-protokollan rinnalla, koska MQTT hyödyntää samoja komponentteja. MQTT-protokollan käyttöönottamiseksi tiedonsiirtojärjestelmä tarvitsee joko pilvessä tai palvelimella suoritettavan viestinvälittäjäsovelluksen sekä asiakaslaitteille asiakassovelluksen. Ohjelmoitaville logiikoille asiakassovellus voidaan ohjelmoida valmiiden MQTT-kirjastojen avulla tai käyttää suljetumpien laitteiden tapaan valmiita asiakassovelluksia. Lisäksi käyttöön tarvitsee ottaa käyttökohteesta riippuen tarvittavat tietoturvaominaisuudet ja hankkia tarvittavat varmenteet. MQTT-protokollan käyttöönoton jälkeen voidaan ADS-protokolla jättää pois käytöstä, jolloin voidaan minimoida tiedonsiirtomenetelmän vaihdosta aiheutuvan seisokin pituus.

Kohdeyrityksen testiautomaatiojärjestelmän kannalta MQTT-protokollassa ei ole merkittäviä rajoitteita, mutta protokolla soveltuu parhaiten älykkääseen M2M-viestintään, minkä vuoksi se ei sovi ainoaksi tiedonsiirtoprotokollaksi kohdeyrityksen testiautomaatiojärjestelmään. MQTT-protokollaa käytettäessä mittausdata tulisi edelleen hakea nykyään käytössä olevien mittausmoduulien tukemalla Modbus/TCP-protokollalla. Datan

tietokantaan tallentamisesta huolehtisi palvelimen tai pilvipalvelun sovellus, joka muuttaisi datan tietokannalle sopivaan muotoon.

Tutkimustyössä kerättyä ja koottua tietoa tullaan hyödyntämään kohdeyrityksen uusiutuvan testiautomaatiojärjestelmän tiedonsiirtoa koskevassa päätöksenteossa. Työssä esitettyä MQTT-protokollan toiminnankuvausta sekä sen perusteelta tehtyjä ominaisuuksien ja rajoitteiden listoja voidaan hyödyntää laajasti eri kohteissa, joissa harkitaan MQTT-protokollan käyttämistä tiedonsiirtonamenetelmänä.

Käytännön työnä tehtyä MQTT-testiympäristöä ja lähdekoodiin liitettävää MQTT-aliohjelmaan tullaan hyödyntämään uusiutuvan testiautomaatiojärjestelmän kehitysympäristössä tehtävässä kehitystyössä. Insinöörityön tuloksena saatiin määritettyä myös alkaneen kehitystyön jatkovaiheet. Kattavan materiaalin lisäksi kohdeyritys sai käytännönosaamista MQTT-protokollan käyttöönotosta, joka tulee olemaan hyödyllistä, mikäli uuden testiautomaationjärjestelmän tiedonsiirtoprotokollaksi valikoituu MQTT.

## Lähteet


1 Salminen, Esa. 2015. Luotettavuuslaboratorion tieto- ja testiautomaatiojärjestelmän vaatimusmäärittely. Insinöörityö. Metropolia Ammattikorkeakoulu. Theseus-tietokanta. Luettavissa osoitteessa <https://www.theseus.fi/handle/10024/99002>. Luettu 3.9.2020.

2 O'Connor, Patrick D.T. & Kleyner, Andre. 2012. Practical reliability engineering. United Kingdom: Wiley.

3 Advantech ADAM-6000 Ethernet I/O Modules. 2014. Verkkoaineisto. Mouser Electronics, Inc. <https://www.mouser.fi/new/advantech/advantech-adam-6000-ethernet-modules/>. Päivitetty 9.11.2020. Luettu 10.11.2020.

4 ADAM-6017. 2017. Verkkoaineisto. Advantech Co., Ltd. <https://select.advantech.com/wise-iot-sensing-devices/files/2017/01/ADAM-6017_B-300x300.png>. Luettu 10.11.2020.

5 Mikä on palvelin?. 2020. Verkkoaineisto. Suomen Hostingpalvelu Oy. <https://www.hostingpalvelu.fi/ohjeet/yleiset-ohjeet/mika-on-palvelin/>. Luettu 11.11.2020.

6 Web-palvelimen toiminta. 2019. Verkkoaineisto. Helsingin yliopiston Agile Education Research -tutkimusryhmä. <https://web-palvelinohjelmointi-19.mooc.fi/osa-1/2-web-palvelimen-toiminta>. Luettu 3.9.2020.

7 Hillar, Gastón C.. 2017. MQTT Essentials – A Lightweight IoT Protocol. United Kingdom:Packt Publishing Ltd.

8 Cope, Steve. 2019. How MQTT Works -Beginners Guide. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-works/>. Päivitetty 12.2.2021. Luettu 4.9.2020.

9 Client, Broker / Server and Connection Establishment - MQTT Essentials: Part 3. 2019. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/>. Luettu 13.11.2020.

10 Connecting all the things in the Internet of Things. 2017. Verkkoaineisto. IBM, Anna Gerber, Jim Romeo. <https://developer.ibm.com/technologies/iot/articles/iot-lp101-connectivity-network-protocols/>. Päivitetty 31.1.2020. Luettu 12.11.2020.

11 Introducing the MQTT Protocol - MQTT Essentials: Part 1. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>. Luettu 12.11.2020.

12 MQTT Version 5.0 OASIS Standard. 2019. Verkkoaineisto. OASIS. <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>. Luettu 3.9.2020.

13 Cope, Steve. Beginners Guide To The MQTT Protocol. 2016.Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt/>. Luettu 15.11.2020.

14 Publish & Subscribe - MQTT Essentials: Part 2. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/>. Luettu 15.11.2020.

15 Getting Started with MQTT. 2020. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/how-to-get-started-with-mqtt/>. Luettu 15.11.2020

16 Cope, Steve. MQTT Brokers and Cloud Hosting Guide. 2018. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-hosting-brokers-and-servers/>. Luettu 15.11.2020

17 Eclipse Paho, MQTT Client Comparison. 2019. Verkkoaineisto. Eclipse Foundation. <https://www.eclipse.org/paho/index.php?page=downloads.php>. Luettu 16.11.2020.

18 Eclipse Mosquitto. 2020. Verkkoaineisto. Eclipse Foundation. <https://projects.eclipse.org/projects/iot.mosquitto>. Päivitetty 11.1.2021. Luettu 16.11.2020.

19 Cope, Steve. Understanding the MQTT Protocol Packet Structure. 2017. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/>. Päivitetty 28.1.2021. Luettu 19.11.2020.

20 Craggs, Ian. 2015. Why doesn’t MQTT have a payload format?. Verkkoaineisto. Model Based Testing. <https://modelbasedtesting.co.uk/2015/12/06/why-doesnt-mqtt-have-a-payload-format/>. Luettu 19.11.2020.

21 MQTT Publish, Subscribe & Unsubscribe - MQTT Essentials: Part 4. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/>. Luettu 16.11.2020.

22 Cope, Steve. Understanding MQTT Topics. 2018. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/understanding-mqtt-topics/>. Päivitetty 2.10.2020. Luettu 20.11.2020.

23 MQTT Topics & Best Practices - MQTT Essentials: Part 5. 2019. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/>. Luettu 16.11.2020.

24 Light, Roger. Mosquitto man page. 2014. Verkkoaineisto. Mosquitto. <https://mosquitto.org/man/mosquitto-8.html>. Luettu 20.11.2020.

25 Cope, Steve. Paho Python MQTT Client Objects. 2020. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/client-objects-python-mqtt/>. Päivitetty 13.3.2020. Luettu 20.11.2020.

26 Keep Alive and Client Take-Over - MQTT Essentials Part 10. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-10-alive-client-take-over/>. Luettu 17.11.2020.

27 Retained Messages - MQTT Essentials: Part 8. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-8-retained-messages/>. Luettu 17.11.2020.

28 Cope, Steve. MQTT Retained Messages Explained. 2018. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-retained-messages-example/>. Päivitetty 20.3.2020. Luettu 17.11.2020.

29 Publish MQTT Messages and Subscribe to Message Topics. Verkkoaineisto. MathWorks. <https://se.mathworks.com/help/supportpkg/raspberrypi/ref/publish-and-subscribe-to-mqtt-messages.html>. Luettu 19.11.2020.

30 Last Will and Testament - MQTT Essentials: Part 9. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/>. Luettu 18.11.2020.

31 Cope, Steve. Paho Python MQTT Client Objects. 2019. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-last-will-example/>. Päivitetty 19.2.2021. Luettu 18.11.2020.

32 Cope, Steve. MQTT Clean Sessions and QOS Examples. 2019. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-clean-sessions-example/>. Päivitetty 20.3.2021. Luettu 18.11.2020.

33 User Properties - MQTT 5 Essentials Part 6. 2019. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt5-essentials-part6-user-properties/>. Luettu 19.11.2021.

34 Cope, Steve. Examining MQTTv5 User Properties. 2019. Steve's internet Guide - verkkosivusto. <http://www.steves-internet-guide.com/examining-mqttv5-user-properties/>. Päivitetty 4.2.2021. Luettu 19.11.2021.

35 Johdanto kyberturvallisuuteen. 2019. STEK Sähköturvallisuuden edistämiskeskus, Elisa Santa Monica Oy. Kyberturvallisuuskurssin sisäinen dokumentti. Luettu 15.12.2021.

36 Järvinen, Petteri. 2003. Salausmenetelmät. Porvoo: Docendo Finland Oy.

37 HiveMQ - MQTT Security Fundamentals. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals/>. Luettu 15.12.2020.

38 Climer, Siobhan & Khan, Mishaal. What Are The 7 Layers Of Security? A Cybersecurity Report. 2020. Verkkoaineisto. Mindsight, Siobhan Climer and Mishaal Khan. <https://gomindsight.com/insights/blog/what-are-the-7-layers-of-security/>. Päivitetty 14.7.2020. Luettu 15.12.2020.

39 Cope, Steve. Introduction to MQTT Security Mechanisms. 2020. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-security-mechanisms/>. Päivitetty 12.2.2021. Luettu 16.12.2020.

40 Halunen, Kimmo; Karinsalo, Anni; Lehtonen, Sami; Suomalainen, Jani; Vallivaara Visa. Sähköisen viestinnän salaus- ja suojausmenetelmät. 2018. Verkkoaineisto. Helsinki: Liikenne- ja viestintäministeriö. <https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/160614/LVM_02_2018_Sahkoisen_viestinnan%20salaus_ja_suojaus.pdf>. Luettu 14.12.2020.

41 What is Transport Layer Security Protocol?. 2020. Verkkoaineisto. SolarWinds MSP. <https://www.solarwindsmsp.com/blog/how-does-tls-work>. Luettu 14.12.2020.

42 Send Secure Email With Office 365 Message Encryption. 2019. Verkkoaineisto. University of Massachusetts Medical School (UMMS) Information Technology. <https://www.umassmed.edu/it/security/send-secure-email-with-msome/>. Luettu 21.12.2020.

43 Luku II: Kryptografian perusteita. 2014. Verkkoaineisto. Helsingin yliopisto, Matemaattis-luonnontieteellinen tiedekunta, Department of Computer Science. <https://www.cs.helsinki.fi/u/karvi/perusteet-luku2_14.pdf>. Luettu 18.12.2021.

44 Catfish. Encryption. 2015. Verkkoaineisto. TrickSter. <http://computer-trickster.blogspot.com/2015/11/encryption.html>. Luettu 21.12.2020.

45 TLS 1.3 Protocol Released – Move Ahead to Advanced Security and Privacy. Verkkoaineisto. ssl2buy.com. <https://www.ssl2buy.com/wiki/tls-1-3-protocol-released-move-ahead-to-advanced-security-and-privacy>. Luettu 21.12.2020.

46 TLS/SSL - MQTT Security Fundamentals. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/>. Luettu 16.12.2021.

47 Cope, Steve. SSL and SSL Certificates Explained For Beginners. 2018. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/ssl-certificates-explained/>. Päivitetty 2.1.2021. Luettu 16.12.2021.

48 X509 Client Certificate Authentication - MQTT Security Fundamentals. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-security-fundamentals-x509-client-certificate-authentication/>. Luettu 18.12.2021.

49 Cope, Steve. Creating and Using Client Certificates with MQTT and Mosquitto. 2020. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/creating-and-using-client-certificates-with-mqtt-and-mosquitto/>. Päivitetty 2.1.2021. Luettu 17.12.2021.

50 Authorization - MQTT Security Fundamentals. 2015. Verkkoaineisto. HiveMQ. <https://www.hivemq.com/blog/mqtt-security-fundamentals-authorization/>. Luettu 17.12.2021.

51 Cope, Steve. Mosquitto ACL -Configuring and Testing MQTT Topic Restrictions. 2019. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/topic-restriction-mosquitto-configuration/>. Päivitetty 12.3.2021. Luettu 17.12.2021.

52 Cope, Steve. Mosquitto Username and Password Authentication -Configuration and Testing. 2016. Verkkoaineisto. Steve's internet Guide -verkkosivusto. <http://www.steves-internet-guide.com/mqtt-username-password-example/>. Päivitetty 28.8.2021. Luettu 16.12.2021.

53 Lowery, Jeff M.. MD5 vs SHA-1 vs SHA-2 - Which is the Most Secure Encryption Hash and How to Check Them. 2020.Verkkosivusto. freeCodeCamp. <https://www.freecodecamp.org/news/md5-vs-sha-1-vs-sha-2-which-is-the-most-secure-encryption-hash-and-how-to-check-them/>. Luettu 22.12.2020.

54 mosquitto.conf man page. Verkkoaineisto. Roger Light. <https://mosquitto.org/man/mosquitto-conf-5.html>. Luettu 21.12.2020.

55 Node-RED. Verkkoaineisto. OpenJS Foundation & Node-RED. <https://node-red.org/>. Luettu 22.12.2020.

56 Creating your first node. Verkkoaineisto. OpenJS Foundation & Node-RED. <https://nodered.org/docs/creating-nodes/first-node>. Luettu 23.12.2020.

57 About. Verkkoaineisto. OpenJS Foundation & Node-RED. <https://node-red.org/about/>. Luettu 23.12.2020.

58 Welcome to the home of MQTT.fx. Verkkoaineisto. MQTT.fx. <https://mqttfx.jensd.de/>. Luettu 23.12.2020.

59 IotMqttSampleUsingQueue Verkkoaineisto. Beckhoff Information System. <https://infosys.beckhoff.com/english.php?content=../content/1033/tf6701_tc3_iot_communication_mqtt/45035999665460363.html&id=>. Luettu 23.12.2020.

60 Barteling, Gerhard. TwinCAT and MQTT – Part 1 Getting started. 2020. PLC-coder.com -verkkosivusto. <https://www.plccoder.com/twincat-and-mqtt-part-1/>. Luettu 23.12.2020.

61 FB_IotMqttMessageQueue. Verkkoaineisto. Beckhoff Information System. <https://infosys.beckhoff.com/english.php?content=../content/1033/tf6701_tc3_iot_communication_mqtt/54043198920564747.html&id=2787512762305093350>. Luettu 23.12.2020.

62 An open source MQTT broker. Verkkoaineisto. Eclipse Mosquitto™. <https://mosquitto.org/>. Luettu 23.12.2020.

**MQTT-aliohjelma [59;60].**

MAIN*

```
PROGRAM MAIN
VAR
    (*Viestin julkaisun parametrit*)
    fbMqttClient: FB_IotMqttClient; //TwinCAT 3:n MQTT-asiakaskirjasto TC3_IotBase:n toimintalohko.
    TopicToPublish : STRING(255) := 'Temperature'; //Julkaistavan viestin otsikko
    MessageToPublish : STRING(255); //Viestin hyötykuorma.
    fbSendMessageIntervalTimer : TON := (PT:=T#1S); //Viestin julkaisuväli

    Thermocouple AT %I* : INT;  //Analogiseen input-moduulin kanavaan linkitetty muuttuja, joka saa arvonsa K-tyypin termoparilta.

    (*Viestin tilauksen parametrit*)
    fbMessageQueue: FB_IotMqttMessageQueue; //Viestijono, johon julkaistut viestit voidaan tallentaa
    fbMessage: FB_IotMqttMessage;
    Subscribed:BOOL; //Tilauksen statusta kuvaava muuttuja
    TopicToSubscribe : STRING(255) := 'SetTemperature'; // Tilatun viestin otsikko
    ReceivingTopic:STRING(255); //Vastaanotetun viestin otsikko
    ReceivingData:STRING(255); //Vastaanotetun viestin hyötykuorma
END_VAR
```

```
(*CONNECT-Paketin parametrien asettaminen*)
IF _TaskInfo[GETCURTASKINDEXEX()].FirstCycle THEN
    fbMqttClient.stTls.sCA := 'C:\Users\Administrator\Desktop\certs\ca.crt'; //CA-salausvarmenne
    fbMqttClient.stTls.sCert := 'C:\Users\Administrator\Desktop\certs\client.crt'; //Asiakkaan autentikointivarmenne
    fbMqttClient.stTls.sKeyFile := 'C:\Users\Administrator\Desktop\certs\client.key'; //Asiakkaan autentikointivarmenteiden kirjaamiseen käytettävä avain
    fbMqttClient.stTls.sVersion := 'tlsv1.3'; // tls eri versiot : 'tlsv1' tai 'tlsv1.1' tai 'tlsv1.2' tai 'tlsv1.3'
    fbMqttClient.sUserName:= 'RC'; // Asiakkaan käyttäjänimi
    fbMqttClient.sUserPassword:= '[redacted]'; //Asiakkaan salasana
    fbMqttClient.sHostName := '[redacted]'; //Valittäjän nimi
    fbMqttClient.nHostPort := 8883; //MQTT:n käyttämä portti, 8883 on suojatun liikenteen portti.
    fbMqttClient.sTopicPrefix := ''; //Otsikon etuliite
    fbMqttClient.sClientId := 'Tampotilasensori'; //Asiakastunnus
    fbMqttClient.ipMessageQueue := fbMessageQueue; //Viestijono, johon julkaistut viestit tallennetaan
END_IF

(*Yhteyden muodostaminen*)
fbMqttClient.Execute(bConnect := TRUE);
```

**MQTT-aliohjelma [59;60].**

```
18
19          (*Viestin julkaisu, PUBLISH-paketti*)
20      IF fbMqttClient.bConnected THEN
21          fbSendMessageIntervalTimer(IN:=TRUE);
22          IF fbSendMessageIntervalTimer.Q THEN
23              fbSendMessageIntervalTimer(IN:=FALSE);
24              MessageToPublish := CONCAT('Lampotila: ',REAL_TO_STRING(Thermocouple / 10.0));
25
26              fbMqttClient.Publish(sTopic:= TopicToPublish,
27                                   pPayload:= ADR(MessageToPublish),
28                                   nPayloadSize:= LEN2(ADR(MessageToPublish))+1,
29                                   eQoS:= TcIotMqttQos.AtMostOnceDelivery,
30                                   bRetain:= FALSE,
31                                   bQueue:= FALSE);
32          END_IF
33      END_IF
34
35      (*Viestin tilaus, SUBSCRIBE-paketti*)
36      IF fbMqttClient.bConnected THEN
37          IF NOT Subscribed THEN
38              Subscribed := fbMqttClient.Subscribe(sTopic:=TopicToSubscribe,
39              eQoS:=TcIotMqttQos.AtMostOnceDelivery);
40          END_IF
41      END_IF
42
43      (*Viesti jonon tarkastelu*)
44      IF fbMessageQueue.nQueuedMessages > 0 THEN
45          IF fbMessageQueue.Dequeue(fbMessage:=fbMessage) THEN
46              fbMessage.GetTopic(pTopic:=ADR(ReceivingTopic), nTopicSize:=SIZEOF(ReceivingTopic) );
47              fbMessage.GetPayload(pPayload:=ADR(ReceivingData), nPayloadSize:=SIZEOF(ReceivingData), bSetNullTermination:=FALSE);
48          END_IF
49      END_IF
```