# Selainpohjaisen valvomosovelluksen soveltuvuus testiautomaatiojärjestelmään

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulun opinnäytetyö

Sähkö- ja automaatiotekniikka, insinööri (AMK)

Kevät 2023

Teemu Anttila

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Sähkö- ja automaatiotekniikka, insinööri (AMK) Tiivistelmä
Tekijä Teemu Anttila Vuosi 2023
Työn nimi Selainpohjaisen valvomosovelluksen soveltuvuus testiautomaatiojärjestelmään
Ohjaajat Lehtori Juha Sarkula (HAMK), Insinööri Tomi Mäentausta (Kohdeyritys)

---

Tässä opinnäytetyössä selvitetään testiautomaatiojärjestelmän valvomosovelluksen soveltuvuutta selainpohjaisena ohjelmistona. Tämänhetkinen valvomosovellus on paikallisesti valvomotietokoneelle asennettu ohjelmisto. Kohdeyrityksen tahtona on uudistaa nykyistä testiautomaatiojärjestelmää, mikä tuottaa tarpeen selvittää onko valvomosovellus toteutettavissa selainpohjaisena ratkaisuna.

Kohdeyrityksenä toimii suomalainen teollisuuden laitevalmistaja. Soveltuvuusselvitys toteutettiin kohdeyrityksen testaus- ja luotettavuusyksikössä.

Työssä tutustutaan kohdeyrityksen luotettavuustestaukseen ja testiautomaatiojärjestelmään. Tutkitaan ohjelmistomalleja ja käsitellään soveltuvuusselvityksessä käytettyä tiedonsiirtoprotokollaa.

Soveltuvuusselvityksen lopputuloksena voidaan todeta selainpohjaisen valvomosovelluksen olevan mahdollinen tulevaisuuden ratkaisu kohdeyrityksessä korvaamaan nykyinen paikallisesti asennettu ohjelmisto. Jatkokehitykselle ja uusille tutkimuksille on kuitenkin vielä tarvetta, sillä soveltuvuusselvityksessä tuotettu ratkaisu ei täytä kohdeyrityksen lopullisen tuotteen vaatimuksia.

Tämän opinnäytetyön aineistoa ja soveltuvuusselvityksessä luotua ratkaisua tullaan hyödyntämään kohdeyrityksen uuden automaatiojärjestelmän suunnittelussa ja tulevissa jatkotutkimuksissa. Kohdeyritys oli tyytyväinen saatuun lopputulokseen.

Avainsanat Valvomosovellus, käyttöliittymä, luotettavuustestaus, selainpohjainen ohjelmisto, MQTT
Sivut 33 sivua ja liitteitä 2 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Electrical and automation technology, engineer (Bachelor) Abstract
Author Teemu Anttila Year 2023
Subject Applicability of a browser-based control application to a test automation system
Supervisors Lecturer Juha Sarkula (HAMK), Engineer Tomi Mäentausta (Target Company)

---

In this thesis, the applicability of the control application of the test automation system as a browser-based software is investigated. The current control application in the target company is a locally installed software on the control computer. The target company wants to renew the current test automation system, which creates the need to find out if the control application can be implemented as a browser-based solution.

The target company is a Finnish industrial equipment manufacturer. The suitability assessment was carried out in the testing and reliability unit of the target company.

The work introduces the target company's reliability testing and test automation system. Software models are studied, and the data transfer protocol used in the suitability assessment is discussed.

As a result of the suitability assessment, it can be stated that a browser-based control application is a possible future solution in the target company to replace the current locally installed software. However, there is still a need for further development and new studies, because the solution produced in the suitability assessment does not meet the requirements of the target company's final product.

The material of this engineering work and the solution created in the feasibility study will be used in the design of the target company's new automation system and in future follow-up studies. The target company was satisfied with the result.

Keywords Control application, user interface, reliability, browser-based application, MQTT
Pages 33 pages and appendices 2 pages

## Sisällys

## Liitteet

# 1 Johdanto

Tässä opinnäytetyössä käsitellään selainpohjaisen valvomosovelluksen soveltuvuutta valvomosovellukseksi luotettavuustestauksen testiautomaatiojärjestelmään. Työn tarpeenmukaisuus periytyy kohdeyrityksen sisäisen kehityksen tarpeesta. Työ on luonteeltaan konstruktiivinen tutkimus, jossa selvitetään selainpohjaisen valvomosovelluksen soveltuvuutta testiautomaatiojärjestelmään luomalla ratkaisu, jolla toimivuus saadaan testattua. Kirjallisuuskatsauksessa on keskitytty soveltuvuusselvityksessä käytettyyn ohjelmistomuotoon ja kommunikaatiorajapintaan. Työn tarkoitus on selvittää, onko selainpohjaisen valvomosovelluksen käyttö ylipäätään mahdollista ja toisaalta, mitkä sen edut olisivat nykyiseen paikallisesti asennettuun valvomosovellukseen verrattuna. Lisäksi tarkoitus on selvittää voisiko MQTT-protokolla toimia valvomosovelluksen ja ohjelmoitavien logiikoiden kommunikaatiorajapintana.

Kohdeyrityksenä toimii suomalainen teollisuuden laitevalmistaja. Kohdeyritys kehittää, tuottaa ja huoltaa valmistamansa laitteet, ja niiden erinäiset komponentit. Soveltuvuusselvitys suoritettiin kohdeyrityksen testaus- ja luotettavuusyksikössä. Automaation rooli testausyksikössä on suuri, sillä iso osa testattavista laitteista ja komponenteista ovat testattavissa testiautomaatiojärjestelmän avulla, siten että ihmisen panosta tarvitaan ainoastaan testin alustamiseen, huoltoon ja purkamiseen. Nykytilaltaan kohdeyrityksessä on testiautomaatiojärjestelmä vuodelta 2011, ja sen vaatimusmäärittelyä ja kehitystä on tutkittu Esa Salmisen insinöörityössä vuodelta 2015: ”Luotettavuuslaboratorion tieto- ja testiautomaatiojärjestelmän vaatimusmäärittely” (Salminen, 2015). Ajankohtaisesti nykyinen järjestelmä toimii ja täyttää tehtävänsä, joten välttämätöntä uudistuksen tarvetta ei ole. Tekniikan alalla jatkuva kehitys on kuitenkin tarpeen ja täten kohdeyrityksessä on päätetty tutkia ja selvittää vaihtoehtoisia menetelmiä automaatiojärjestelmän päivitykseen, jotta vääjäämättömän päivityksen tullessa eteen, kohdeyrityksessä olisi valmiiksi tarpeeksi tietoa päivitetystä tai korvaavasta järjestelmästä.

Valvomosovelluksen selainpohjaista ratkaisua työssä selvitetään, koska tämänhetkinen paikallisesti asennettu on yleisesti ottaen vanhentunut tapa. Paikallisesti asennetun sovelluksen ollessa asennettuna vain tietyille valvomokoneille on käyttäjien lähdettävä

työpisteeltään valvomokoneille, tai otettava niihin etäyhteys omalta työkoneeltaan. Etäyhteyden muodostamisessa on aina se riski, että joku toinen käyttää valvomokonetta juuri sillä hetkellä. Selainpohjainen ratkaisu helpottaisi olennaisesti huoltotöitä, sillä huoltotöiden aikana valvomotietokoneille meneminen on haastavaa ja aikaa vievää.

# 2 Luotettavuustestaus ja testiautomaatio kohdeyrityksessä

Luotettavuus on todennäköisyys sille, että tuote suorittaa vaaditut toiminnon ilman vikaantumista tuotteelle tarkoitetuissa olosuhteissa tuotteelle suunnitellun eliniän ajan. Luotettavuus on tärkeä osa nykypäivän teollisuutta, sillä usein pettävä tuote takuuajan puitteissa koituu kalliiksi valmistajalle ja vaivalloiseksi käyttäjälle. Heikosti kestävät tuotteet saattavat aiheuttaa valmistajalle maineen menetystä. (O'Connor & Kleyner, 2012)

Luotettavuustestauksessa hyvä testiautomaatiojärjestelmä mahdollistaa laitteiden ja komponenttien testauksen automaattisesti mahdollisimman vähäisellä ihmisen panostuksella. Tässä osiossa käydään läpi kohdeyrityksessä suoritettavaa luotettavuustestausta ja esitellään kohdeyrityksen testiautomaatiojärjestelmää.

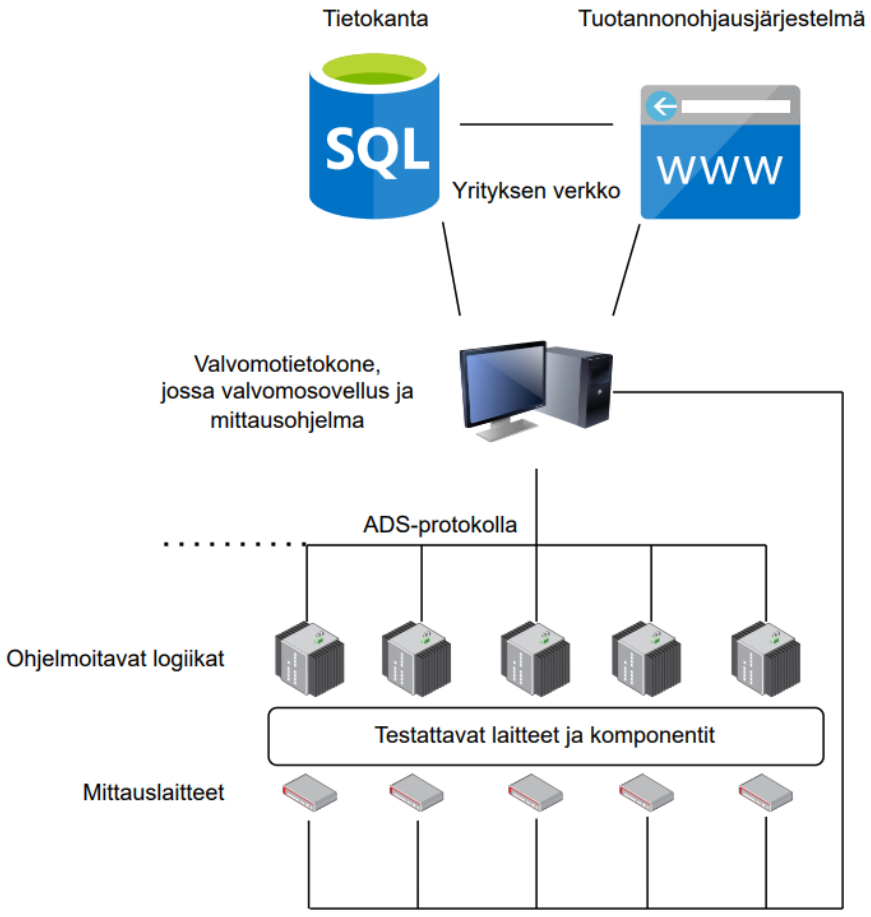
## 2.1 Luotettavuustestaus kohdeyrityksessä

Kohdeyrityksessä luotettavuustestausta suoritetaan laitekokonaisuuksille sekä niiden erillisille komponenteille. Luotettavuustestausta suoritetaan laitteille ja komponenteille niiden koko elinkaaren ajalta kehitysvaiheesta aina käytöstä poistoon. Tämän lisäksi kohdeyrityksessä selvitetään odottamattomia vikaantumisia ja pyritään löytämään niihin johtaneet syyt, jotta vastaavilta vikaantumisilta kyetään jatkossa välttymään. Luotettavuustestausta suoritetaan myös komponenteille, jotka tulevat korvaamaan valmistuksessa tai kentällä käytössä olevien laitteiden alkuperäisosia. Näin toimitaan, koska kohdeyritys haluaa varmistua siitä, että korvaavat komponentit täyttävät samat vaatimukset kuin alkuperäiset komponentitkin.

## 2.2 Testiautomaatiojärjestelmä kohdeyrityksessä

Testiautomaatiojärjestelmä kohdeyrityksessä on vuodelta 2011. Vuosien varrella sen eri osat ovat saaneet päivityksiä, mutta toimintaperiaate on pysynyt samana. Testiautomaatiojärjestelmä koostuu tuotannonohjausjärjestelmästä, valvomosovelluksesta, ohjelmoitavista logiikoista ja mittausjärjestelmästä. Seuraavaksi esitellään kohdeyrityksen testiautomaatiojärjestelmän eri osat. Kuvassa 1 on piirrettynä yksinkertaistettu kaavio kohdeyrityksen testiautomaatiojärjestelmästä ja sen verkkoarkkitehtuurista. Kaavio on piirretty ilmaisella Diagrams.net -sivuston selainpohjaisella Draw.io -ohjelmistolla.

Kuva 1. Kohdeyrityksen testiautomaatiojärjestelmä



### 2.2.1 Tuotannonohjausjärjestelmä

Tuotannonohjausjärjestelmästä on selainpohjainen ohjelmisto, jonka kautta kaikki testityöt ja tehtävät luodaan sekä hallinnoidaan. Työlle ja tehtäville luodaan aloitusaika ja suunnitellut lopetusajat, laaditaan testisuunnitelma ja asetetaan vastuuhenkilöt tuotannonohjausjärjestelmässä. Työtä luodessa tietokannasta varataan testidataa varten tallennustilaa. Seuraavaksi käsiteltävässä valvomosovelluksessa on mahdollista määritellä testin kulkua vasta, kun tuotannonohjausjärjestelmässä työ ja tehtävät ovat luotuja. Testin ollessa käynnissä sen kulkua on mahdollista seurata tuotannonohjausjärjestelmästä. Työ on suljettava tuotannonohjausjärjestelmän kautta, kun testityön kaikki tehtävät ja testit ovat suoritettuja. (Salminen, 2015, ss. 11-12)

### 2.2.2 Valvomosovellus

Nykyinen valvomosovellus on Visual Basic .NET-kielellä ohjelmoitu paikallisesti valvomotietokoneelle asennettava ohjelmisto. Kommunikaatio rajapintana valvomosovelluksen ja ohjelmoitavien logiikoiden (PLC) välillä toimii TwinCAT:in ADS (The Automation Device Specification protocol). Valvomosovelluksen tehtävänä on toimia rajapintana mekaanikon ja ohjelmoitavan logiikan välillä. Valvomosovelluksella luodaan testisyklit ja asetetaan vikaantumisehdot ajettaville testipaikoille. Graafisen käyttöliittymän ansiosta tämä onnistuu mekaanikolta ilman, että hänen on tunnettava ohjelmoitavien logiikoiden ohjelmointia. (Salminen, 2015, ss. 11-12)

Valvomosovelluksesta voidaan seurata testin kulkua. Käyttöliittymästä näkee testin tilan, eli onko pysäytetty, ajossa tai vialla. Vikatilanteessa sovellukseen saadaan selkokielinen viesti vian syystä. Sovelluksesta näkee myös ajettujen syklien määrän, ajetun ajoajan ja mahdolliset ennalta asetetut ajettavat syklimäärät. Testiä asettaessa automaattiajoon on pyynti lähetettävä sovellukselta ohjelmoitavalle logiikalle, sillä testialue on tarkistettava ennen kuin testin saa laittaa ajoon vaaratilanteiden välttämiseksi. Automaattiajoon laitto tapahtuu testerin hallintapaneelilta, eli alueelta, mikä mahdollistaa testin turvallisen käynnistyksen.

Kaikki testeistä saatu data kirjoitetaan tietokantaan. Tietokantaan kirjoituksen suorittaa kohdeyrityksen verkkoon liitetty valvomotietokone, jolle valvomosovellus on asennettu. Tietokanta on Structured Query Language -tietokanta (SQL). (Häkkinen, 2021, ss. 2-4)

### 2.2.3 Ohjelmoitavat logiikat

Ohjelmoitavat logiikat kohdeyrityksessä ovat Beckhoffin tuottamia logiikoita. Ohjelmointikielenä logiikoissa on sekä vanhempaa TwinCAT 2 - että uudempaa TwinCAT 3 -ohjelmointiympäristöä, riippuen laitteistosta. Logiikoiden tehtävä on saada edellä mainitulta valvomosovellukselta viesti sisältäen testisyklin. Logiikka pilkkoo viestin sekvensseihin ja muodostaa niistä ajettavan syklin ja lähettää valvomotietokoneelle testidataa sykleistä ja ajoajasta. Yksi ohjelmoitava logiikka ohjaa yleisesti ottaen useampaa testipaikkaa aina kahdesta toiseen kymmeneen.

### 2.2.4 Mittausjärjestelmä

Mittausjärjestelmä koostuu Advantechin valmistamista ADAM Ethernet I/O-moduuleista ja kohdeyrityksessä kehitetystä mittausohjelmistosta. Mittausohjelmisto toimii valvomotietokoneella erillään valvomosovelluksesta. ADAM-moduulit kytketään automaatioverkkoon Ethernet-kaapelilla ja laitteeseen/komponenttiin tarvittavin johdotuksin. ADAM-moduuleiden käyttö on sallittua virallisissa mittauksissa. (Häkkinen, 2021, ss. 4-5; Salminen, 2015, ss. 35-36)

## 2.3 Kohdeyrityksen testiautomaatiojärjestelmän tulevaisuus

Kohdeyrityksessä on vuodesta 2019 lähtien suunniteltu tulevaa testiautomaatiojärjestelmää. Tuotannonohjausjärjestelmän ja valvomosovelluksen yhdistelmälle on laadittu yhteinen vaatimusmäärittely. Tässä työssä käsitellään selainpohjaisen valvomosovelluksen soveltuvuutta vain siinä mielessä, onko kyseinen konsepti edes mahdollinen. Kun katsotaan alla olevaa taulukkoa (Taulukko 1) nähdään tarkemmat vaatimukset selainpohjaiselle valvomosovellukselle, jotka kohdeyrityksessä on määritelty aikaisemmin. Taulukosta on karsittu osiot, jotka koskevat tuotannonohjausjärjestelmää tai muita osia.

Taulukko 1. Valvomosovelluksen selainpohjaisen käyttöliittymän vaatimusmäärittelmä

| Päätoiminto | Kategoria | Toiminto | Prioriteetti |
|---|---|---|---|
| Tekniset vaatimukset | Ympäristö vaatimukset | Erillinen tuotanto- ja testiversio | 1 |
| | | Kohdeyrityksen palvelin | 1 |
| | | Single sign on (SSO). Kohdeyrityksen tunnuksilla kirjautuminen | 1 |
| | Tietosuojaa koskevat vaatimukset | Kehittäjä ja tuottaja hyväksyvät kohdeyrityksen tietosuojasopimuksen | 1 |
| | Aika ja alueasetukset | Englanti on kohdeyrityksen toimintakieli | 1 |
| | | Eurooppalainen numero- ja päiväysjärjestelmä | 1 |
| | | Aika on näyttävä sen mukaan missä testeri on. Nykyinen näyttää UTC +2, mikä ei ole käytännöllinen kaikkialla. | 1 |
| | | Kaiken on oltava synkronoitua oikean ajan kanssa, sillä luotettavuustestaus perustuu oikeaan aikaan. | 1 |
| | Suoritusvaatimukset | Oltava vähintään yhtä nopea käyttöliittymä, kuin nykyisessä paikallisesti asennetussa | 1 |
| | Läpäisykykyä koskevat vaatimukset | Testitietojen tallennus 10 min välein, käyttäjän toimesta tai odottamattoman tapahtuman kohdalla | 1 |
| | Siirrettävyysvaatimukset | Käytettävyys ympäri maailman, missä kohdeyrityksen toimipisteitä sijaitsee | 1 |
| | Palauttamisvaatimukset | Verkkosivulta ei pysty poistamaan mitään pysyvästi. Kohdeyrityksen IT-tuen on kyettävä palauttamaan tiedostoja | 1 |
| Käytettävyys | Valvomosovellus | Graafinen esitys testisyklistä, sykliä luodessa | 2 |
| | Mittaukset | Graafinen esitys mittauksista | 2 |
| | | Sisäänrakennettu kaavionäkymä mittauksia varten (sekä automaattiset että manuaaliset) | 2 |
| | Käyttöliittymä | Selainpohjainen, ilman ylimääräisiä asennuksia käyttäjän tietokoneelle | 1 |
| | | Yksinkertainen ja helppo käytettävyys. Rajattu määrä informaatiota kerralla näkyvillä. | 1 |
| | | Laite: Työkannettava/Työpöytäkone, eri selaimet (IE, Chrome, Firefox, Safari, Opera, Edge jne.) | 1 |
| | | Laite: Matkapuhelin (Android, iOS) | 2 |
| | | Laite: Tabletti/iPad | 2 |
| | Haku ja suodatus toiminnot | Vastaava kuin Googlella tai Outlookissa | 1 |

| | | | |
|---|---|---|---|
| | Testidata | Mahdollisuus saada ladattua raakadata | 1 |
| Turvallisuus | Työturvallisuus | Testien käynnistäminen ei voi olla mahdollista, ilman lupaa sovellukselta, eikä ilman suoraa näköyhteyttä testitilaan | 1 |
| Tietoturva | Tietoturva | Rooli perusteiset käyttöoikeudet | 1 |
| | | Näkymän mahdollinen muokkaus roolista/testistä riippuen | 1 |
| Ylläpidettävyys | Ylläpidettävyys ja laajennusvaatimukset | Mahdollisuus laajentaa ohjelmistoa | 1 |
| | | Mahdollisuus laajentaa testijärjestelmiä ja laitteistoa itsenäisesti kohdeyrityksen puolelta | 1 |
| | | Tuottaja tarjoaa koulutukset tuotteelle | 1 |
| | | Järjestelmän on lähetettävä viestiä (puhelin/sähköposti), mikäli ohjelmoitavassa logiikassa ilmenee ongelmia tai testi epäonnistuu muusta syystä | 1 |
| Käyttöoikeudet | Käyttöoikeudet | Käyttäjät voidaan ryhmitellä, jotta kaikki näkevät vain heille tarpeelliset tiedot ja voivat tehdä vain heille tarkoitetuilla testipaikoilla | 1 |
| | | Käyttöoikeusluokat (Tarkkailija, Käyttäjä, Järjestelmänvalvoja) | 1 |
| | | Mahdollisuus rajattuun käyttöoikeuteen, joka on muokattavissa siten, että oikeus nähdä ainoastaan tietty osa järjestelmää | 1 |

# 3 Ohjelmistomallit

Tämä kappale pohjautuu asiantuntijahaastatteluun. Asiantuntijana haastateltiin Arcadan ammattikorkeakoulusta 2018 valmistunutta tietotekniikan insinööriä, joka työskentelee Stora Ensolla ammattinimikkeellä ”Experienced Service Desk Specialist”. Asiantuntija on antanut näkemyksensä paikallisesti asennettujen, selainpohjaisten sekä hybridi mallisten ohjelmistojen hyödyistä ja haitoista. (Henkilökohtainen tiedonanto, 23.11.2022)

Haastattelu suoritettiin Microsoft Teams -puheluna. Asiantuntija oli saanut kysymykset haltuunsa viikkoa ennen haastattelua, joka järjestettiin 23.11.2022. Haastattelukielenä toimi ruotsi, sillä asiantuntija on ruotsinkielinen. Haastattelun tuotos on kohdeyritykselle arvokas aineisto testiautomaatiojärjestelmän uudistamisessa, sillä luotettavan tiedon löytäminen erilaisten ohjelmistojen vertailusta on todettu haastavaksi.

## 3.1 Paikallisesti asennettu ohjelmisto

Paikallisesti asennettu ohjelmisto tarkoittaa järjestelmään asennettua ohjelmistoa, joka suoritetaan paikallisesti järjestelmässä eikä ulkopuolisella palvelimella. Kohdeyrityksessä suositeltu tapa asentaa ohjelmistoja on asentaa ne valmiiksi paketoituina Microsoftin Software Centeristä. Microsoft Software Center on organisaatioiden IT-järjestelmävalvojien työkalu ohjelmistojen, ohjelmistojen päivityksien ja käyttöjärjestelmän päivityksiin tarkoitettu ohjelmisto (Microsoft, 2022c).

Asiantuntijan mukaan paikallisesti asennetun ohjelmiston hyötynä on käyttäjien ja/tai järjestelmänvalvojien mahdollisuus hallita ohjelman asennuksia ja päivityksiä, sekä ohjelman ylläpito on helpommin suunniteltavissa. Lisäksi ohjelmiston etuna voidaan pitää sen mahdollista käyttöä ilman internet-yhteyttä. Vakaata paikallista ohjelmistoa ei ole aina välttämätöntä päivittää uusimpaan versioon, jos ensin halutaan arvioida lopputulosta, joka päivityksestä seuraisi. (Henkilökohtainen tiedonanto, 23.11.2022)

## 3.2 Selainpohjainen ohjelmisto

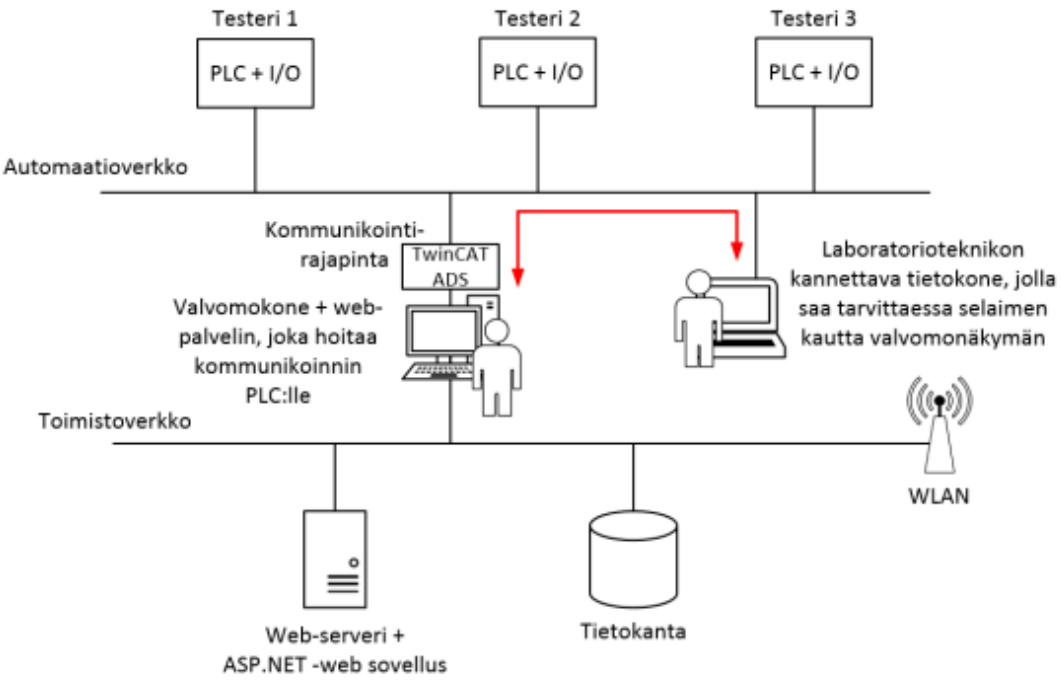
Selainpohjainen ohjelmisto on palvelimelle asennettu ohjelmisto, jonka käyttöliittymään päästään käsiksi selaimen kautta. Selainpohjainen ohjelmisto vaatii internet-yhteyden ja ohjelmistoa tukevan selaimen, jotta ohjelmiston käyttö onnistuu. Selainpohjaisten ohjelmistojen etuina yleisesti ovat niiden helppokäyttöisyys sekä tietokoneen pienet tehovaatimukset paikallisesti asennettuihin ohjelmistoihin verrattuna. Selainpohjaisen ohjelmiston tarvittavat päivitykset voidaan tehdä käyttäjiltä huomaamatta. (Warren, 2021)

Asiantuntija kertoo selainpohjaisen ratkaisun tuovat huomattavasti enemmän joustavuutta paikallisesti asennettuun ohjelmistoon verrattuna. Ohjelmiston asennusta laitteelle ei tarvita, vaan korkeintaan selainlaajennus. Ohjelmistoa voidaan periaatteessa käyttää, miltä tahansa laitteelta, missä on ohjelmistoa tukeva verkkoselain. Päivityksiä ei tarvitse työntää käyttäjille erikseen, koska ohjelmiston käyttöliittymä toimii palvelinta vasten. Selainpohjaisessa ratkaisussa loppukäyttäjillä on aina käytössä sama viimeisin versio ohjelmistosta. (Henkilökohtainen tiedonanto, 23.11.2022)

## 3.3 Hybridimalli

Esa Salmisen insinöörityössä vuodelta 2015: ”Luotettavuuslaboratorion tieto- ja testiautomaatiojärjestelmän vaatimusmäärittely”, käsitellään hybridimallista ratkaisua, jossa paikallinen ja selainpohjainen valvomosovellus toimisivat rinnakkain (Salminen, 2015, ss. 33-34). Kuvassa 2 on Esa Salmisen esittämä ehdotus valvomosovelluksen hybridimallista. Asiantuntija oli haastattelussa sitä mieltä, että hybridi ratkaisu toimii parhaiten vain silloin, kun tehdään samoin, miten Microsoft teki Microsoft Teams-yhdistelmätyökalunsa kanssa. Eli ohjelma olisi niin kutsuttu ”web container”. Kahden eri version hallinnoiminen on yhtä ohjelmistoa haastavampaa, ja vaarana on saada molempien ratkaisujen huonoimmat puolet ilman etuja. (Henkilökohtainen tiedonanto, 23.11.2022)

Kuva 2. Valvomosovelluksen hybridimalli (Salminen, 2015, s. 34)




## 3.4 Ohjelmistomuodon valinta

Ohjelmistonmuodon valinta on tehtävä tarpeen mukaan. Kohdeyrityksessä on toimiva paikallisesti asennettu ohjelmisto, joka täyttää vaatimukset. Asiantuntija kuitenkin

haastattelussaan totesi, että kohdeyrityksen tarpeisiin selainpohjainen ratkaisu olisi parempi joustavuutensa takia (Henkilökohtainen tiedonanto, 23.11.2022). Kohdeyrityksessä tuotannonohjausjärjestelmä on selainpohjainen ratkaisu, ja valvomosovelluksen liittäminen osaksi sitä olisi mahdollista selainpohjaisella ohjelmistolla, mutta ei nykyisellä paikallisesti asennetulla.

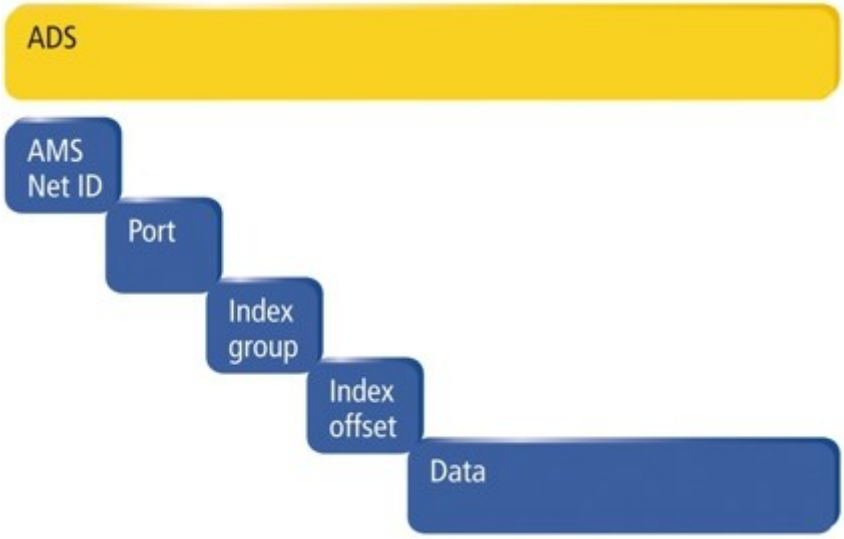
# 4 Kommunikaatiorajapinta

Kohdeyrityksessä valvomosovelluksen ja testereiden väliseen tiedonsiirtoon käytetty kommunikaatiorajapinta on Beckhoffin kehittämä ”The Automation Device Specification” -protokolla (ADS). MQTT:n (Message Queue Telemetry Transport) soveltuvuutta ADS:n korvaavaksi tiedonsiirto protokollaksi on käsitelty Sami Häkkisen insinöörityössä vuodelta 2021: ”MQTT-protokollan soveltuvuus testiautomaatiojärjestelmän tiedonsiirtoon” (Häkkinen, 2021). Tässä osiossa verrataan näiden kahden edellä mainitun kommunikaatio rajapinnan välisiä eroja, ja perustellaan, miksi tämän työn soveltuvuusselvitykseen valittiin MQTT-protokolla ADS-protokollan sijaan. Osion kohdassa 4.3 olevaa tutkimustietoa saatiin henkilökohtaisella kokemuksella kohdeyrityksessä, ennen soveltuvuusselvityksen aloittamista.

## 4.1 ADS-protokolla

Beckhoffin kehittämä ADS-protokolla on TwinCAT-ohjelmiston sisäinen tiedonsiirtotaso. ADS-protokolla on käytettävissä TCP/IP:n (Transmission Control Protocol / Internet Protocol) päällä, mikäli tahdotaan kommunikoida tietokoneiden tai muiden vastaavien laitteiden kanssa. ADS-protokollan toimilohkot sijaitsevat TwinCAT:in Tc2_System.lib kirjastossa. ADS kommunikaatio koostuu viidestä osasta AMSNetId:stä, portti numerosta, indeksi ryhmästä, indeksi offset:istä ja datasta, kuten kuvassa 3 on havainnollistettu. (Beckhoff Information System, n.d.-a)

Kuva 3. Beckhoffin ADS kommunikaatio (Beckhoff Information System, n.d.-a)




### 4.1.1 TCP/IP-protokolla

TCP/IP-protokollaa (Transmission Control Protocol / Internet Protocol) käytetään kahden eri laitteen tiedonsiirron välillä. Tämä tarkoittaa sitä, että vaikka järjestelmässä olisi kuinka monta laitetta, niin tiedonsiirto tapahtuu aina vain kahden laitteen välillä. TCP/IP-protokollassa TCP vastaa siirrettävän tiedon paketoinnista osiin ennen lähetystä ja tiedon purkamisesta oikeassa järjestyksessä sen päästyä perille. IP vastaa pakettien reitistä ja perille saapumisesta. (Klusaitė, 2022)

TCP/IP-protokolla koostuu neljästä kerroksesta. Ylimmäinen kerros eli sovelluskerros on sovelluksien sisältämät protokollat kuten TwinCAT:in käyttämä ADS. Seuraavana kerroksena on kuljetuskerros, joka sisältää TCP- ja UPD-protokollat (User Datagram Protocol). Kuljetuskerros vastaa tiedon pilkkomisesta viestin lähtiessä ja kokoamisesta viestin saapuessa perille. Kuljetuskerroksen alla on verkkokerros, joka vastaa pakettien liikkumisesta verkossa. IP-protokolla on osa verkkokerrosta. Alimpana on siirtokerros, minkä tarkoitus on varmistaa tiedon saapuminen oikeaan osoitteeseen. (Klusaitė, 2022)

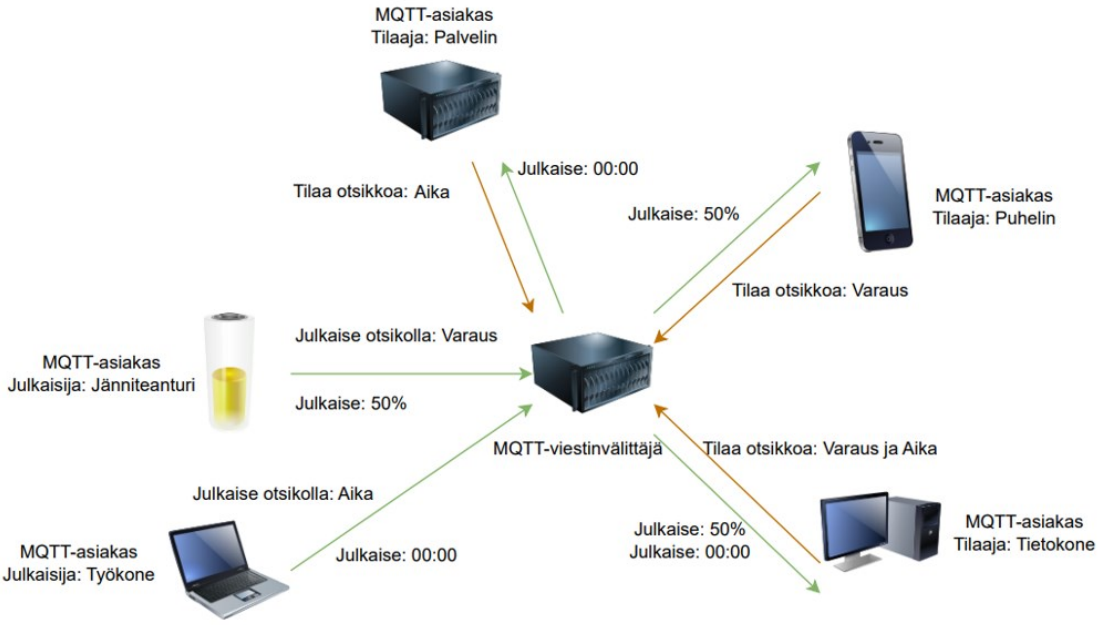
### 4.1.2 Kohdeyrityksessä toimiva ADS kommunikaatiorajapinta

Kohdeyrityksessä ohjelmoitavat logiikat ovat reititetty valvomotietokoneisiin ADS-protokollaa käyttäen. Ohjelmoitavilta logiikoilta ja valvomotietokoneilta löytyy TwinCAT-ohjelmointiympäristö. Valvomotietokoneiden ja ohjelmoitavien logiikoiden välinen yhteys on luotu reitittämällä valvomotietokoneelta yhteys ohjelmoitaviin logiikoihin, joihin kyseisen valvomotietokoneen on oltava yhteydessä. Ohjelmoitavat logiikat eivät ole yhdistettyinä toisiinsa, ja ovat täten tietämättömiä toisistaan testiautomaatioverkossa.

## 4.2 MQTT-protokolla

Message Queue Telemetry Transport -protokolla (MQTT) on OASIS standardin mukainen kevyt Internet of Things (IoT) tiedonsiirtoprotokolla. MQTT-protokolla perustuu julkaisija/tilaaja – malliin (publish/subscribe). MQTT-protokolla toimii ADS:n kaltaisesti TCP/IP-protokollan päällä. Perustoiminnaltaan MQTT viestinnässä MQTT-viestinvälittäjään on yhteydessä useampi MQTT asiakas. Asiakkaat ovat, joko julkaisijoita (Publisher), tilaajia (Subscriber) tai molempia. Julkaisija julkaisee otsikolla viestejä viestinvälittäjälle, josta kyseistä otsikkoa tilaavat tilaajat saavat julkaistun viestin. MQTT-viestinvälittäjään voi olla yhteydessä useampi asiakas. Kuvassa 4 on hahmoteltu MQTT-viestinnän periaatetta piirrettynä ilmaisella Diagrams.net-sivuston selainpohjaisella Draw.io ohjelmistolla. (MQTT, n.d.)

Kuva 4. MQTT-viestintä




## 4.3 MQTT-protokollan valinta soveltuvuusselvitykseen

MQTT-protokollan valintaan soveltuvuusselvityksessä vaikutti useampi asia. Ensinnäkin insinöörityössä (Häkkinen, 2021, s. 58) todetaan MQTT-protokollan soveltuvan testiautomaatiojärjestelmän viestintäprotokollaksi. Täten kohdeyrityksessä vallitsi tahtotila laajentaa jo soveltuvan kommunikaatiorajapinnan tutkimista testiautomaatiojärjestelmässä. Täten kohdeyrityksessä vallitsi tahtotila laajentaa jo soveltuvan kommunikaatiorajapinnan tutkimista testiautomaatiojärjestelmässä.

Toiseksi ADS-protokollassa on kohdeyrityksessä huomattu erinäisiä pieniä ongelmia, kuten laitteiden välisen yhteyden reitittämisessä. Esimerkiksi ADS protokollaa käytettäessä valvomokoneena ja liitetty toimilaite (ohjelmoitava logiikka tai teollisuustietokone) saattaa hävittää yhteyden toisiinsa esimerkiksi päivitetyn koodin lataamisen yhteydessä, jolloin yhteys on muodostettava uudelleen valvomokoneelta laitteelle teknikon toimesta. MQTT-protokollaa käyttäessä edellä mainittua ongelmaa ei synny, koska laitteiden välillä ei tarvitse olla erikseen muodostettua yhteyttä.

Kolmanneksi helppokäyttöisyys ja tuki. MQTT-protokolla on helppokäyttöinen siinä mielessä, että viestinvälittäjän asentamiseen tietokoneelle ja viestinvälittäjän asetusten määrittämiseen on paljon dokumentaatiota saatavilla internetistä. Tämän lisäksi Beckhoff on luonut TwinCAT 3:seen toimilohkot MQTT-protokollaa varten (Beckhoff Information System, n.d.-c). MQTT-protokolla soveltuu erilaisten laitteiden väliseen kommunikaatioon. ADS-protokollan kanssa asiat ovat toisin, sillä ADS-protokolla on Beckhoffin kehittämä TwinCAT-ohjelmiston sisäinen tiedonsiirtotaso ja täten toimii ainoastaan laitteiden kanssa, joissa on TwinCAT-ohjelmisto asennettuna. Tuen saanti ongelmatilanteissa on yksinkertaista lähettämällä sähköpostia tai soittamalla suoraan Beckhoffin tekniseen tukeen (Beckhoff, n.d.). Ongelmana kuitenkin on teknisen tuen saatavuus Beckhoffin teknisen tuen asiointiajan ulkopuolella, sekä ADS-protokollasta muuten internetistä löytyvän niukan dokumentaation vuoksi.

# 5 Soveltuvuusselvitys

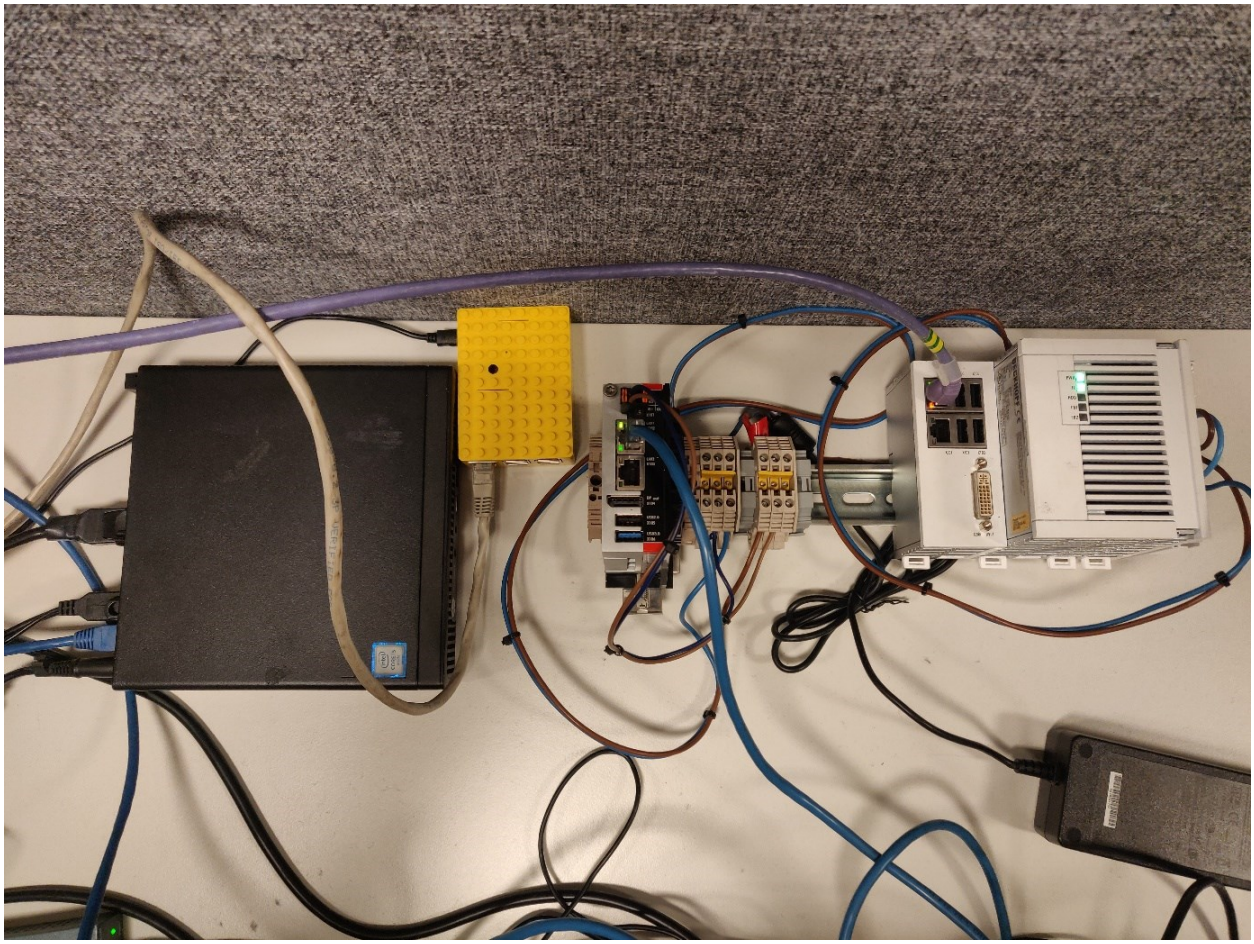
Selvityksen tavoitteena oli luoda selainpohjainen käyttöliittymä, joka kykenee viestittämään sisäisessä verkossa MQTT-viestinvälittäjälle ja tilaamaan välittäjältä haluttuja viestejä. Toteutus oli malliltaan soveltuvuusselvitys (Proof of Concept) ja täten toteutusta ei voida pitää valmiina tuotteena käyttöönottoon kohdeyrityksessä. Soveltuvuusselvityksessä rakennetun ympäristön oli tarkoitus kuvastaa automaatiojärjestelmää pienoiskoossa ilman tuotannonohjausjärjestelmää ja ilman kirjoitusta tietokantaa. Soveltuvuusselvitys toteutettiin kolmessa eri käytännön vaiheessa. Tässä luvussa käydään läpi toteutukseen käytetyt laitteet, suunniteltu kokonaisuus ja toteutuksen eri vaiheet.

Kohdan 3.4 huomioiden perusteella soveltuvuusselvityksessä päädyttiin täysin selainpohjaiseen ratkaisuun. Selainpohjainen ratkaisu on paikallisesti asennettua nykyaikaisempi ja mahdollinen liittäminen tulevaisuudessa tuotannonohjausjärjestelmään on mahdollista.

## 5.1 Ympäristön luomiseen tarvitut laitteet

Fyysisiä laitteita ympäristön luomiseen tarvittiin HP:n mini tietokone (HP ProDesk 600 G2 mini PC), pienoistietokone Raspberry Pi 3, Beckhoffin teollisuustietokoneesta IPC C6015-0010 (Industrial PC) ja Beckhoffin ohjelmoitavasta logiikasta (programmable logic controller) PLC CX5130-0155. Edellä mainitut laitteet ovat näkyvillä kuvassa 5. Toimintaympäristö on sisäisessä verkossa TP-Linkin 5-porttisen kytkimen välityksellä (TL-SG105) (Kuva 6). Lisäksi ympäristö vaati 24 VDC virtalähteen ohjelmoitavalle logiikalle ja teollisuustietokoneelle riviliittimien välityksellä kytkettynä, sekä kolme näyttöä, jotta viestiliikennettä kyettiin seuraamaan samanaikaisesti kaikilta tarvittavilta laitteilta.

Kuva 5. Ympäristön laitteisto vasemmalta oikealle HP mini PC, Raspberry Pi 3, IPC C6015-0010 ja PLC CX5130-0155

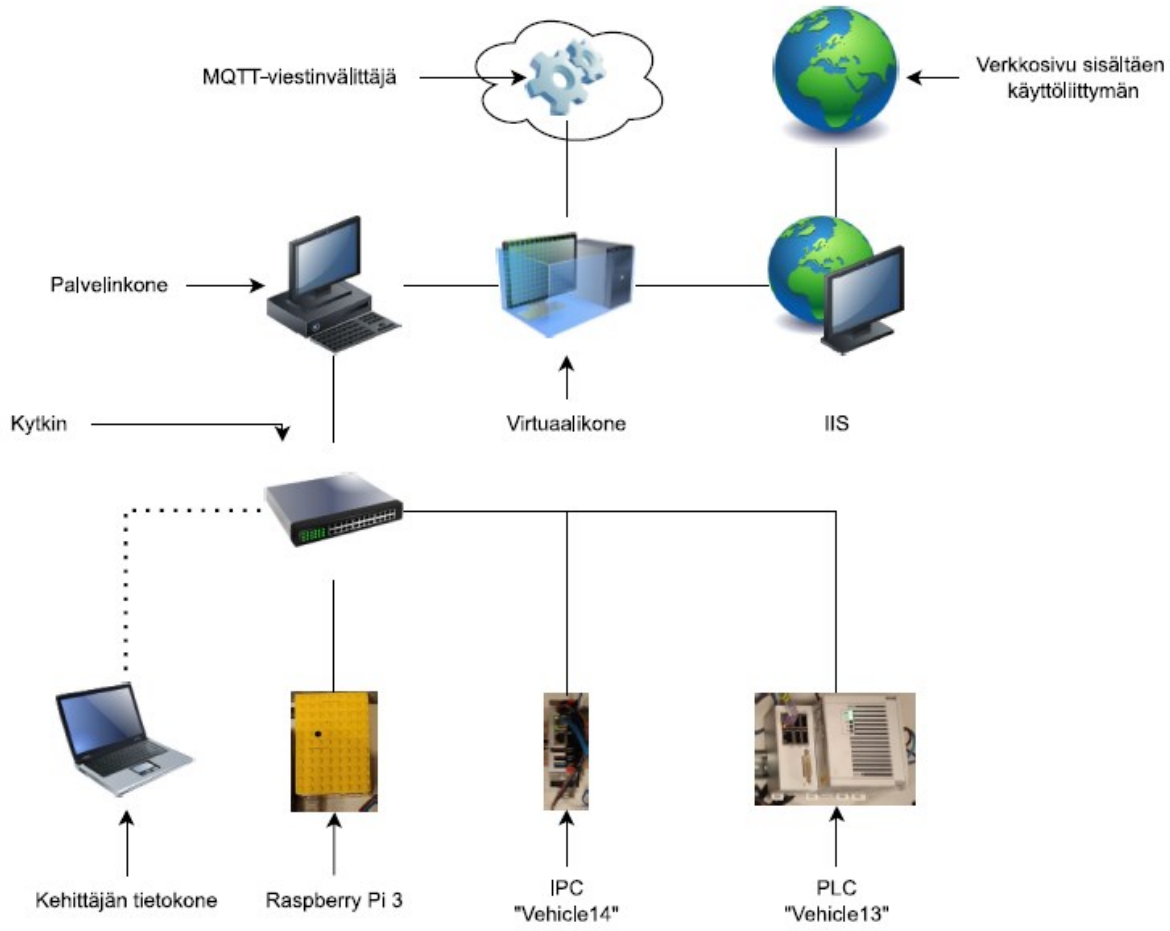


Kuva 6. TP-Link TL-SG105 verkkokytkin


## 5.2 Kokonaisuuden suunnittelu

Kun kaikki tarvittavat laitteet oli saatu pöydälle, suunniteltiin järjestelmä. HP:n mini tietokone saatiin kohdeyrityksen atk-tuelta ja sisälsi täten valmiiksi asennetun Windows 10-käyttöjärjestelmän ja oli liitettynä kohdeyrityksen toimialueeseen. Täten mini tietokoneelle olisi asennettava, jokin virtuaalikone käyttöjärjestelmällä, jossa verkkosivua kyetään isännöimään ja jossa MQTT-viestinvälittäjä sijaitsee. PLC ja IPC saisivat toimia liitettyinä laitteina tunnisteilla Vehicle13 ja Vehicle14. Raspberry Pi 3 tuotiin projektiin täysin omasta mielenkiinnostani testata Linux ympäristössä MQTT-viestintää, sillä kaikki muut toteutuksen osat pyörivät Windows ympäristössä. Laitteet kytkeytyvät yhteen TP-Linkin kytkimen kautta, mihin on kytkettävissä kehittäjän tietokone järjestelmän testausta varten ja koodien lataamiseksi ohjelmoitavalle logiikalle ja teollisuustietokoneelle. Kuvassa 7 suunnitelmasta piirretty kaavio. Kaavio on piirretty ilmaisella Diagrams.net-sivuston selainpohjaisella Draw.io ohjelmistolla.

Kuva 7. Toteutuksen suunniteltu ja toteutunut kaavio



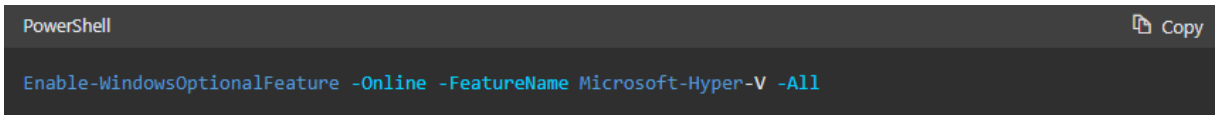

## 5.3 Selvityksen ensimmäinen vaihe

Ensimmäinen vaihe koostui useasta osa-alueesta. Palvelinkoneelle (HP:n mini tietokone) oli asennettava virtuaalikone käyttöjärjestelmällä ja määriteltävä IIS (Internet Information Services) verkkosivua varten. MQTT-viestinvälittäjä oli asennettava virtuaalikoneen käyttöjärjestelmään ja sen toimivuus testattava. Raspberry Pi 3 -pienoistietokoneelle oli hankittava käyttöjärjestelmä ja asennettava MQTT-viestinvälittäjä. Ohjelmoitavan logiikan ja teollisuustietokoneen TwinCAT-projektit oli luotava ja testattava niiden ja MQTT-viestinvälittäjän välinen viestintä.

### 5.3.1 Palvelinkoneen ja virtuaalikoneen käyttöönotto

HP mini tietokone saapui suoraan kohdeyrityksen atk-tuelta valmiiksi asennettuna Windows 10 -käyttöjärjestelmällisenä tietokoneena. Kyseessä oli kuitenkin toimialueeseen liitetty tietokone, joten paikalliset järjestelmänvalvoja oikeudet oli koneelle haettava. Oikeuksien ollessa kunnossa aktivoitiin Microsoft Hyper-V tietokoneeseen, jotta virtuaalikone saatiin luotua. Microsoft Hyper-V aktivoidaan avaamalla PowerShell järjestelmän valvojalla ja ajamalla kuvassa 8 näkymä teksti. Microsoft Hyper-V:tä ei tarvitse erikseen ladata, sillä se on valinnainen ominaisuus Windowsissa.

Kuva 8. PowerShell komento Microsoft Hyper-V:n asentamiseksi (Microsoft, 2022b)

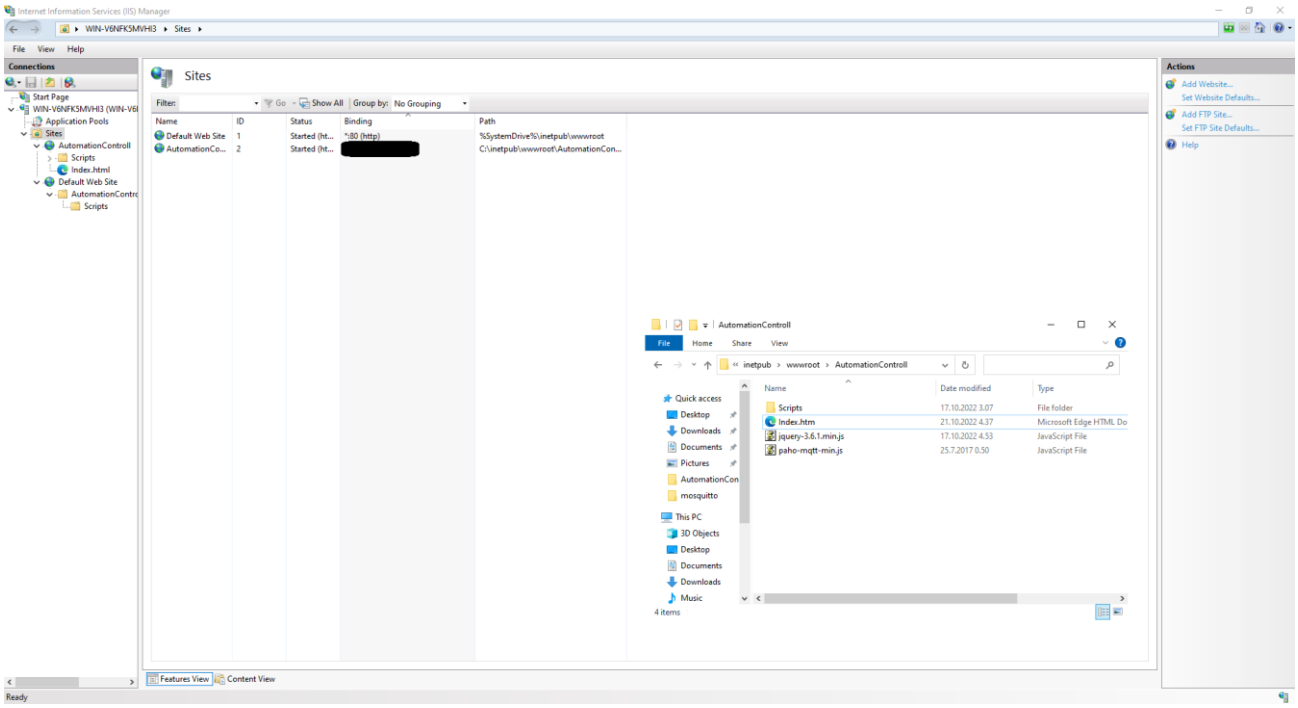



Virtuaalikoneen käyttöjärjestelmäksi valikoitui Microsoft Server 2022 evaluation-palvelin käyttöjärjestelmä. Käyttöjärjestelmän lisenssi oli määräaikainen työn luonteen vuoksi. Käyttöjärjestelmä on ladattavissa Microsoftin sivuilta (Microsoft, n.d.). Virtuaalikoneen määrittäminen ja käyttöjärjestelmän asennus onnistui moitteetta seuraten Microsoftin luomia asennusohjeita (Microsoft, 2022a).

### 5.3.2 ”Internet Information Services” -ohjelmiston käyttöönotto

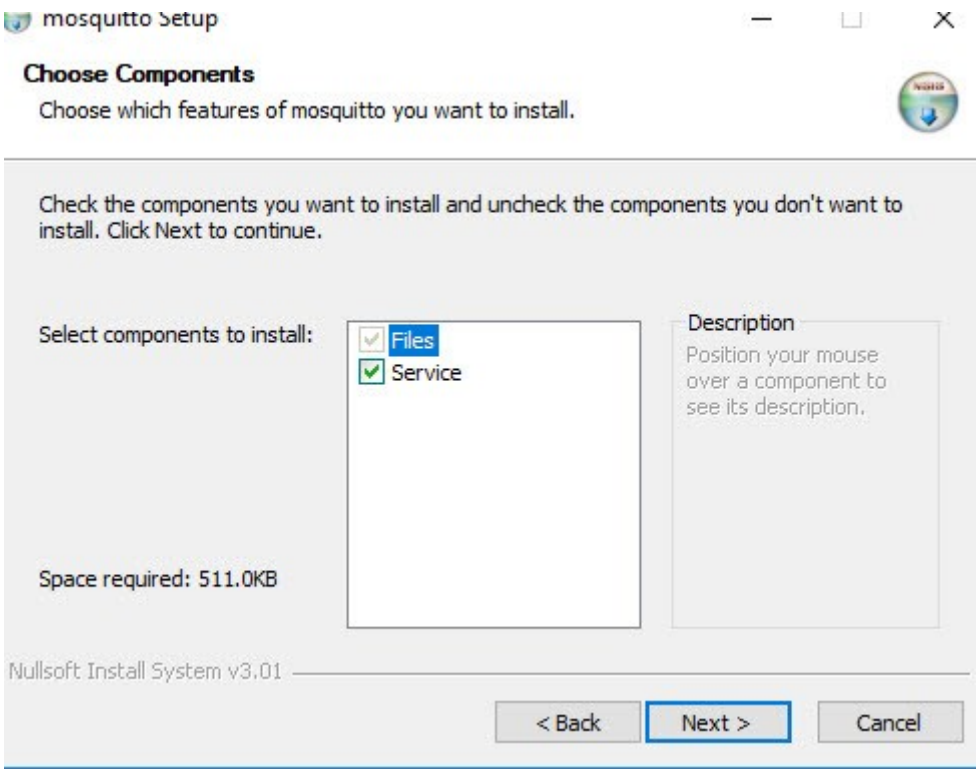
Selainpohjaista ohjelmistoa varten oli luotava virtuaalikoneella sijaitsevalle palvelimelle verkkosivu. Tämä tehtiin käyttämällä Microsoftin omaa IIS-palvelinohjelmistoa. Verkkosivun luominen käy avaamalla palvelimella Internet Information Services (IIS) Manager, hiiren oikealla painetaan ”Sites”-kohtaa ja lisätään verkkosivu. Verkkosivun määrittämisessä sivulle määritetään nimi, hakemiston sijainti ja ”binding” (eli tyyppi, IP osoite, portti ja mahdollinen ”host name”). Kuvassa 9 on toteutuksen valmiiksi määritetty IIS ja verkkosivun hakemistokansio.

Kuva 9. Valmis IIS ja verkkosivun hakemisto




### 5.3.3 MQTT-viestinvälittäjän käyttöönotto

MQTT-viestinvälittäjän käyttöönotto suoritettiin, sekä virtuaalikoneelle Windows-ympäristöön että Raspberry pi:lle Linux-ympäristöön. Linux-ympäristössä tehty testi oli vain ja ainoastaan allekirjoittajan mielenkiinnosta, sillä kohdeyrityksessä kaikki pyörii Windows-ympäristössä. Eclipse Mosquitto on Eclipse Foundationin kehittämä avoimen lähdekoodin viestinvälittäjä, joka tukee MQTT-protokollaa (Eclipse Foundation, n.d.-b). Windows-ympäristöön asennettava Mosquitto on ladattavissa suoraan Eclipse Foundationin sivuilta (Eclipse Foundation, n.d.-a). Asennus tapahtuu muuten suoraviivaisesti, kunhan muistaa kuvan 10 kohdassa valita ”Services”-kohdan asennettavaksi, jotta Mosquitto asennetaan palveluna. Näin Mosquitto saadaan käynnistymään automaattisesti tietokoneen uudelleen käynnistyessä.



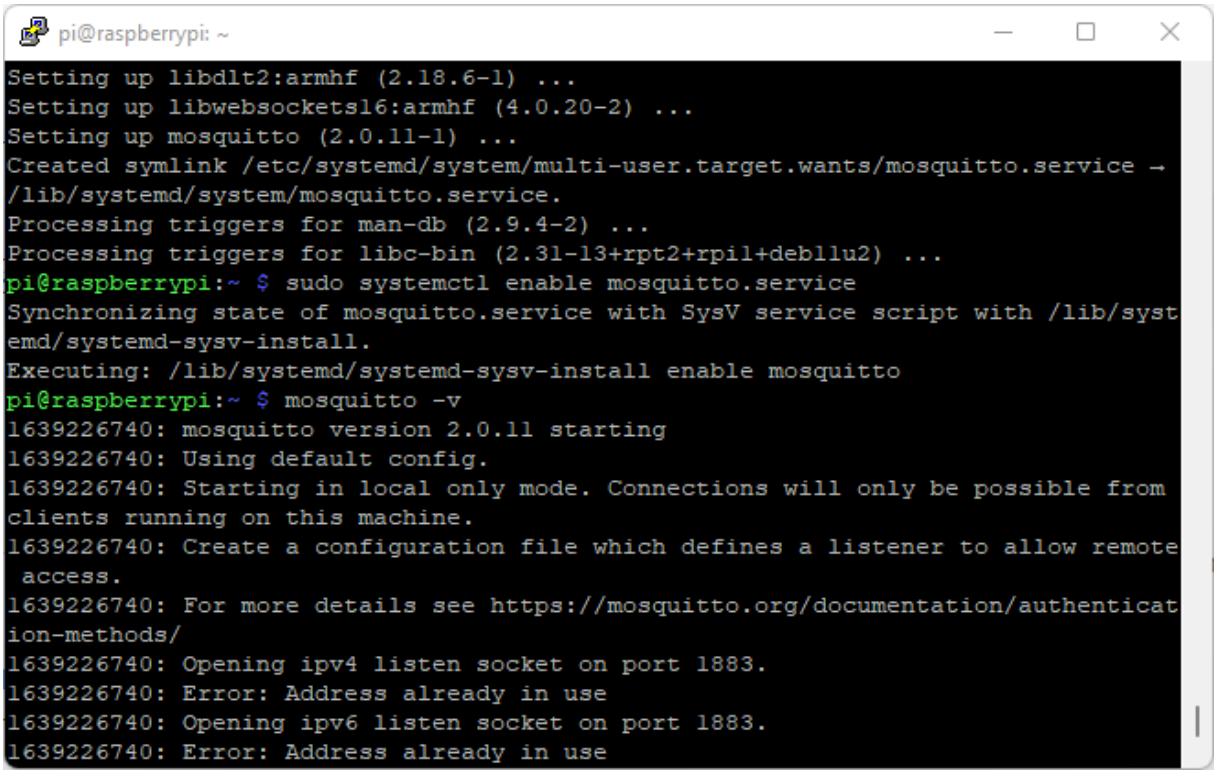

Kuva 10. Mosquitton asennus palveluna (Cope, How to Install The Mosquitto MQTT Broker on Windows, n.d.)

Mosquitton asennus Raspberry pi:lle oli lähes yhtä suoraviivaista. Ensinnäkin Raspberry Pi:lle on asennettava MicroSD-kortille käyttöjärjestelmä Raspberry Pi:n omilta sivuilta (Raspberry Pi, n.d.). Käyttöjärjestelmän asennuksen jälkeen, kun Raspberry Pi on käynnissä, avataan käyttöjärjestelmässä konsoli, johon kirjoitetaan seuraavat komennot järjestyksessä:

- ”sudo apt update && sudo apt upgrade” (Järjestelmän päivitys, saattaa kestää jonkin aikaa)
- ”sudo apt install -y mosquitto mosquitto-clients” (Mosquitton asennus)
- ” sudo systemctl enable mosquitto.service” (Jotta Mosquitto käynnistyy aina uudestaan Raspberry Pi:n käynnistyessä)
- ” mosquitto -v” (Mosquitton testaus)

Mikäli kaikki onnistui, näyttää konsoli kuvan 11 mukaiselta. (Random Nerd Tuotrials, 2022)

Kuva 11. Raspberry Pi MQTT testaus (Random Nerd Tuotrials, 2022)




Käyttöönoton viimeisessä vaiheessa molemmissa ympäristöissä on lisättävä Mosquitto.conf-tiedostoon seuraavat virkkeet:

- ”listener 1883” (Jotta kyetään kuuntelemaan porttia 1883)
- ” allow_anonymous true” (Jotta ei tarvita autentikointia viestintään)

Mosquitto.conf-tiedosto löytyy Mosquitton juurikansiosta.

### 5.3.4 Ohjelmoitavan logiikan ja teollisuustietokoneen TwinCAT-projektien käyttöönotto

Ohjelmoitavan logiikan ja teollisuustietokoneen koodit oli luotu Beckhoffin TwinCAT 3 -ohjelmistolla. Tähän työhön oli käytetty koodia, joka oli muokattu Beckhoffin omien esimerkkien mukaan, PLCCoder-sivuston esimerkistä, että Sami Häkkisen insinöörityön (Häkkinen, 2021) tuottaman toimivan koodin pohjalta. Projektissa ei ollut muita ohjelmia ja se oli kirjoitettu ST-kielellä (Structure Text), käyttäen Beckhoffin luomaa Tc3_IotBase-

kirjastoa. Projektin koodi löytyy liitteenä 1. (Barteling, 2020; Beckhoff Information System, n.d.-b; Häkkinen, 2021, ss. 46-50)

Projektin muuttujat ovat nähtävissä kuvassa 12. Muuttujat olivat sisäisiä, sillä globaaleja muuttujia ei tässä soveltuvuusselvityksessä tarvittu. Muuttujien avulla määritetään julkaisun ja tilauksen parametrit kuvan 12 vihreiden kommenttien mukaisesti. Teollisuustietokoneen muuttujat olivat samat kuin ohjelmoitavalla logiikalla, paitsi ”TopicToPublish”-muuttuja oli Vehicle14 ja ”TopicToSubscribe”-muuttuja oli Vehicle14DriveCommand.

Kuva 12. Ohjelmoitavan logiikan muuttujat

```
PROGRAM MAIN
VAR
    (*Viestin julkaisun parametrit*)
    fbMqttClient: FB_IotMqttClient; //TwinCAT 3:n MQTT-asiakaskirjasto TC3_IotBase:n toimintalohko.
    TopicToPublish : STRING(255) := 'Vehicle13'; //Julkaistavan viestin otsikko
    MessageToPublish : STRING(255); //Viestin hyötykuorma.
    fbSendMessageIntervalTimer : TON := (PT:=T#30S); //Viestin julkaisuväli

    (*Viestin tilauksen parametrit*)
    fbMessageQueue: FB_IotMqttMessageQueue; //Viestijono, johon julkaistut viestit voidaan tallentaa
    fbMessage: FB_IotMqttMessage;
    Subscribed:BOOL; //Tilauksen statusta kuvaava muuttuja
    TopicToSubscribe : STRING(255) := 'Vehicle13DriveCommand'; // Tilatun viestin otsikko
    ReceivingTopic:STRING(255); //Vastaanotetun viestin otsikko
    ReceivingData:STRING(255); //Vastaanotetun viestin hyötykuorma

    Cycle   : int := 0;
END_VAR
```

Kuva 13 osoittaa kuinka kerran suoritettavan MQTT:n CONNECT-paketin parametrin määritettiin. Pakollisia määritettäviä olivat välittäjän nimi ja portti. Välittäjän nimenä toimi laitteen IP osoite ja porttina 1883, koska soveltuvuusselvityksessä käytettiin salaamatonta yhteyttä.

Kuva 13. CONNECT-paketin määritys

```
(*CONNECT-Paketin parametrien asettaminen*)
IF _TaskInfo[GETCURTASKINDEXEX()].FirstCycle THEN
    fbMqttClient.sHostName := [redacted]; //Välittäjän nimi
    fbMqttClient.nHostPort := 1883; //MQTT:n käyttämä portti.
    fbMqttClient.sTopicPrefix := ''; //Otsikon etuliite
    fbMqttClient.sClientId := 'Vehicle13'; //Asiakastunnus
    fbMqttClient.ipMessageQueue := fbMessageQueue; //Viestijono, johon julkaistut viestit tallennetaan
END_IF

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

Kuvassa 14 oli koodin viestintään liittyvät toimilohkot. PUBLISH-paketin tarkoituksena oli julkaista syklitietoja MQTT-viestinvälittäjälle. SUBSCRIBE-paketti tarkoituksena oli tilata MQTT-viestinvälittäjältä ennalta muuttujissa määritellyn Vehicle13DriveCommandin mukaisella otsikolla viestejä. Viestijonon tarkastelulla kyettiin testaus vaiheessa tarkastelemaan vastaanotettua viestiä.

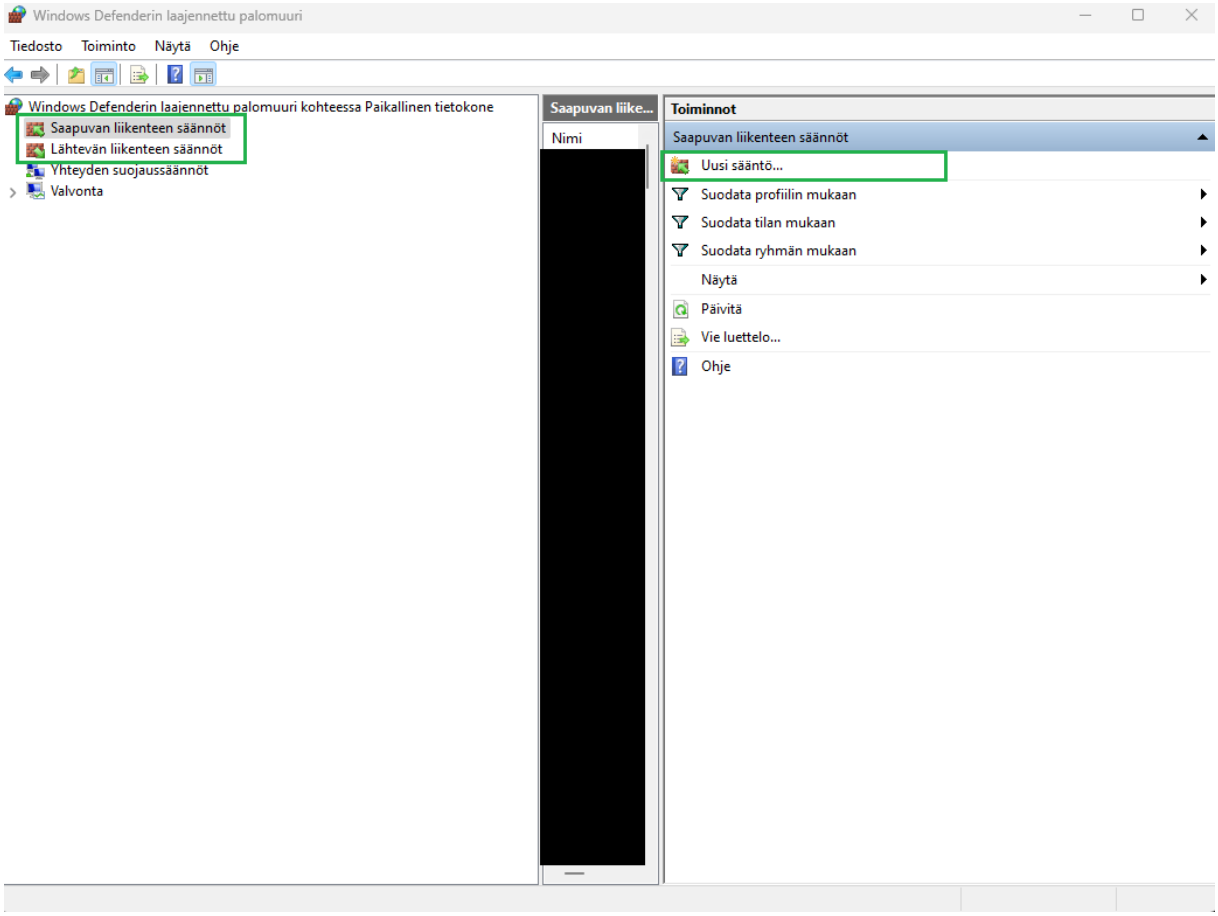
Kuva 14. PUBLISH-paketti, SUBSCRIBE-paketti ja viestijonon tarkastelu

```
(*Viestin julkaisu, PUBLISH-paketti*)
IF fbMqttClient.bConnected THEN
    fbSendMessageIntervalTimer(IN:=TRUE);
    IF fbSendMessageIntervalTimer.Q THEN
        fbSendMessageIntervalTimer(IN:=FALSE);
        MessageToPublish := CONCAT('Cycle count: ' , INT_TO_STRING (Cycle));

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

(*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
```

### 5.3.5 MQTT-viestinvälittäjän testaus

Testaus oli ensimmäinen vaihe, jossa vastaan tuli ongelmia. Viesti ei välittynyt mistään suunnasta mihinkään. Tässä päästään MQTT:n olennaisimpaan ongelmaan, sillä apua ei oikeastaan internetistä saanut eikä työympäristöstä löytynyt apua ongelmaan. Ongelma löytyi lopulta Windows Defender palomuurista. Windowsilla on avattava Windows Defender palomuuri ohjauspaneelista ja valittava sieltä lisäasetukset, joka vaatii järjestelmänvalvojan oikeudet. Avautuvassa ”Windows Defenderin laajennettu palomuuri” -ikkunassa on luotava neljä uutta sääntöä, kaksi saapuvan liikenteen sääntöä ja kaksi lähtevän liikenteen sääntöä. Kuvassa 15 Windows Defenderin laajennettu palomuuri.

Kuva 15. Windows Defenderin laajennettu palomuuri

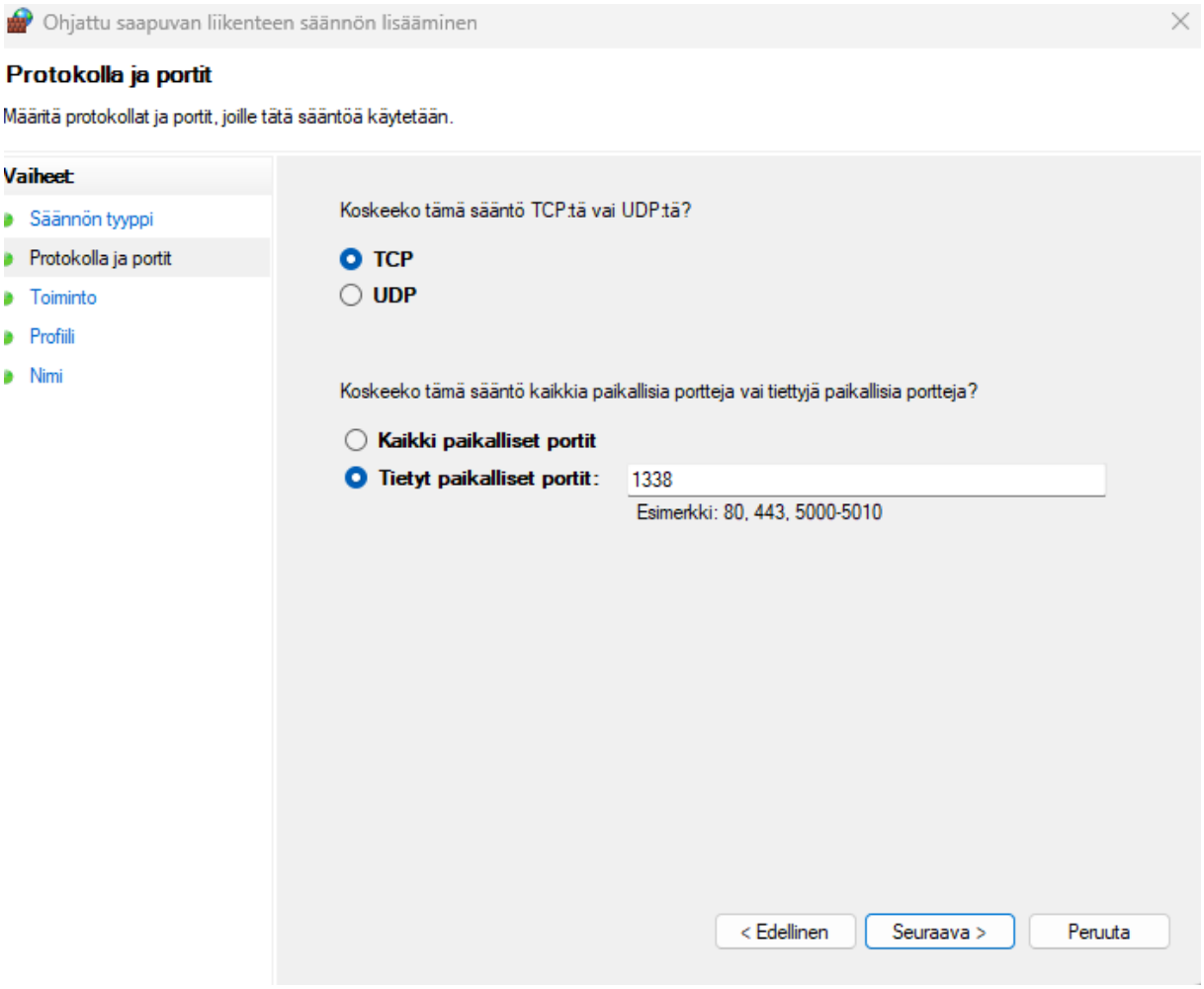



Uudet säännöt ovat seuraavat:

- Kaikissa neljässä valitaan ensimmäisenä ”Säännön tyyppi”-kohdassa ”Portti”.

- ”Protokolla ja portit”-kohdassa valitaan TCP kerran saapuvan ja kerran lähtevän liikenteen sääntöihin. Vastaavasti valitaan UDP. ”Tietyt paikalliset portit:”-kohtaan asetetaan portti 1883. Kuvassa 16 yksi säännöistä.
- ”Toiminto”-kohdassa sallitaan yhteys.
- ”Profiili”-kohdassa valitaan, mitä verkkoja sääntö koskee (toteutuksessa pidettiin kaikki varmistaaksemme toimivuuden).
- ”Nimi”-kohdassa vapaavalintaisesti nimet säännöille. Toteutuksessa kaikki säännöt olivat Mosquitto nimellä.

Kuva 16. Saapuvan liikenteen TCP portin määritys




Palomuuriasetuksien jälkeen toimivuus saatiin testattua. Testaus suoritettiin lähettämällä syklitietoja ohjelmoitavalta logiikalta ja teollisuustietokoneelta samalla, kun virtuaalikoneen MQTT-viestinvälittäjää seurattiin. Raspberry Pi kuunteli virtuaalikoneen liikennettä.

Huomioitavaa tässä kaikessa on se, että Raspberry Pi:lle ei tarvinnut tehdä minkäänlaisia lisätoimintoja MQTT-viestinvälittäjän toimivuuden kannalta. Yhtä havaittua ongelmaa ei pystytty korjaamaan, nimittäin testauksessa oli käytössä myös viestinvälittäjä kohdeyrityksen työkoneella. Tähän työkoneeseen viestin välittäminen ei onnistunut toiselta vastaavalta työkoneelta. Tämä ei haitannut tämän työn etenemistä, mutta tämä on syytä ottaa huomioon jatkokehityksessä.

## 5.4 Selvityksen toinen vaihe

Selvityksen toisessa vaiheessa oli määrä luoda verkkosivu, jolla MQTT-viestinvälittäjään saadaan kommunikoitua. IIS määriteltiin ensimmäisessä vaiheessa, jotta toisessa vaiheessa päästiin suoraan aloittamaan verkkosivun luomista. Toisen vaiheen isoin ongelma oli, että verkkosivujen luomisesta kokemus oli melkein kymmenen vuotta vanhaa ja täten alku menikin w3schools.com-sivuston kautta HTML, CSS ja JavaScriptiä opetellessa (W3Schools, n.d.).

### 5.4.1 Verkkosivun luominen

Verkkosivu luotiin vain soveltuvuusselvitystä varten, eikä sen tarkoitus ollut vastata lopullisen mahdollisen tuotteen vaatimuksiin, koska tuotteen toimivuudesta ei ollut käsitystä, eikä haluttu tuhlata resursseja mahdollisesti turhaan työhön.

Verkkosivun luomiseen löytyi pohja ”Steve’s Internet Guide”-sivustolta (Cope, 2022). Tätä työtä varten lähdekoodia oli kuitenkin muokattava. Lähdekoodiin tehtiin muokkauksia poistaen kaikki, mitä ei toteutuksessa tarvittu ja lisättiin tai muutettiin kohtia, jotta verkkosivu sopisi paremmin toteutettavaan soveltuvuusselvitykseen.

Soveltuvuusselvitystä tehdessä tahtotila oli, että työ tehdään itsenäisessä verkossa yrityksen ja automaatioverkon ulkopuolella ilman yhteyttä internettiin. Tämä tarkoitti sitä, että verkkosivun käyttämät kirjastot oli ladattava virtuaalikoneen verkkosivun hakemiston juureen. Tässä työssä oli käytössä kaksi kirjastoa, joista ensimmäinen helpotti JavaScriptin kirjoittamista ja toinen sisälsi MQTT:n viestintää varten tarvitut toiminnot. Muuten

verkkosivu on täysin HTML-ohjelmointikielellä kirjoitettu verkkosivu sisältäen JavaScriptiä ja MQTT-toiminnot. Verkkosivun koodi on kokonaisuudessaan tämän työn liitteessä 2.

### 5.4.2 Verkkosivun toimintaperiaate

Soveltuvuusselvitykseen luotu verkkosivu voidaan toiminnallisuuksien mukaan jakaa neljään osaan. Ylin osa koostuu palkista, joka näyttää yhteyden tilan viestinvälittäjään ollen vaalean sininen, mikäli yhteyttä ei ole sisältäen tekstin ”Not Connected” ja vihreä mikäli yhteys on muodostettu sisältäen tekstin ”Connected”. Palkin alla on tiputusvalikossa yksi ainoa palvelin osoite, koska työssä ei ollut muita palvelimia käytössä. Samoin seuraavassa tiputusvalikossa on vain yksi portti vaihtoehtona. Ensimmäisen osan päättää ”Connect”-painike, jolla yhteys viestinvälittäjään muodostetaan. Toinen osa sisältää ”Subscribe Topic”-syöttö ikkunan, johon voidaan kirjoittaa tilattava otsikko. Tilaus luodaan klikkaamalla ”Subscribe”-painiketta. Tilauksia on mahdollista tehdä useille otsikoille yksi kerrallaan. Kolmas osuus on taas viestin lähettämistä varten tarkoitettu osuus. ”Message”-kohtaan kirjoitetaan haluttu viesti ja ”Publish Topic”-kohtaan otsikko. Viesti lähtee vasta kun ”Submit”-painiketta klikataan. Neljäs osuus on ”Messages:” eli verkkosivun näyttämät viestit. Viestit tulevat keltaisella pohjalla. Viestejä saapuu tilatuilta otsikoilta ja virhe viestit ovat saatavilla, jos esimerkiksi yhteyttä ei ole viestinvälittäjään niin sivu automaattisesti ilmoittaa, että tilaaminen ja lähettäminen on mahdotonta klikatessa ”Subscribe”- tai ”Submit”-painikkeita. Kuvassa 17 on kuvattu tilanne, jossa on yritetty lähettää viestiä yhteyden ollessa katki.

Kuva 17. Verkkosivu ja virheilmoitus


### 5.4.3 Verkkosivun testaus

Verkkosivun testauksessa kokeiltiin aivan ensimmäisenä avata verkkosivu paikallisesti virtuaalikoneella, missä verkkosivua ylläpidettiin ja jossa viestinvälittäjä sijaitsi. Yhteys viestinvälittäjään saatiin ja viestien tilaaminen ja lähettäminen onnistui moitteetta. Tämän jälkeen testaus suoritettiin vielä kahdella erillisellä työkoneella, jotka liitettiin erilliseen verkkoon, avattiin verkkosivu ja ensimmäisellä tietokoneella tilattiin viestiä otsikolla, johon toinen tietokone lähetti viestiä. Kaiken tämän toimiessa voitiin todeta verkkosivun olevan soveltuvuusselvityksen tarpeet täyttävä ja toimiva.

## 5.5 Selvityksen kolmas vaihe

Käytännön osuuden kolmannessa ja viimeisessä vaiheessa oli tarkoitus testata kokonaisuus ja todeta konsepti toimivaksi. Tähän mennessä ensimmäisessä ja toisessa vaiheessa suoritetut testit olivat olleet viestinvälittäjälle lähetettyjä viestejä. Kolmannessa vaiheessa Raspberry Pi asetettiin kuuntelemaan viestinvälittäjää, jotta testiä voitiin helposti seurata erilliseltä näytöltä.

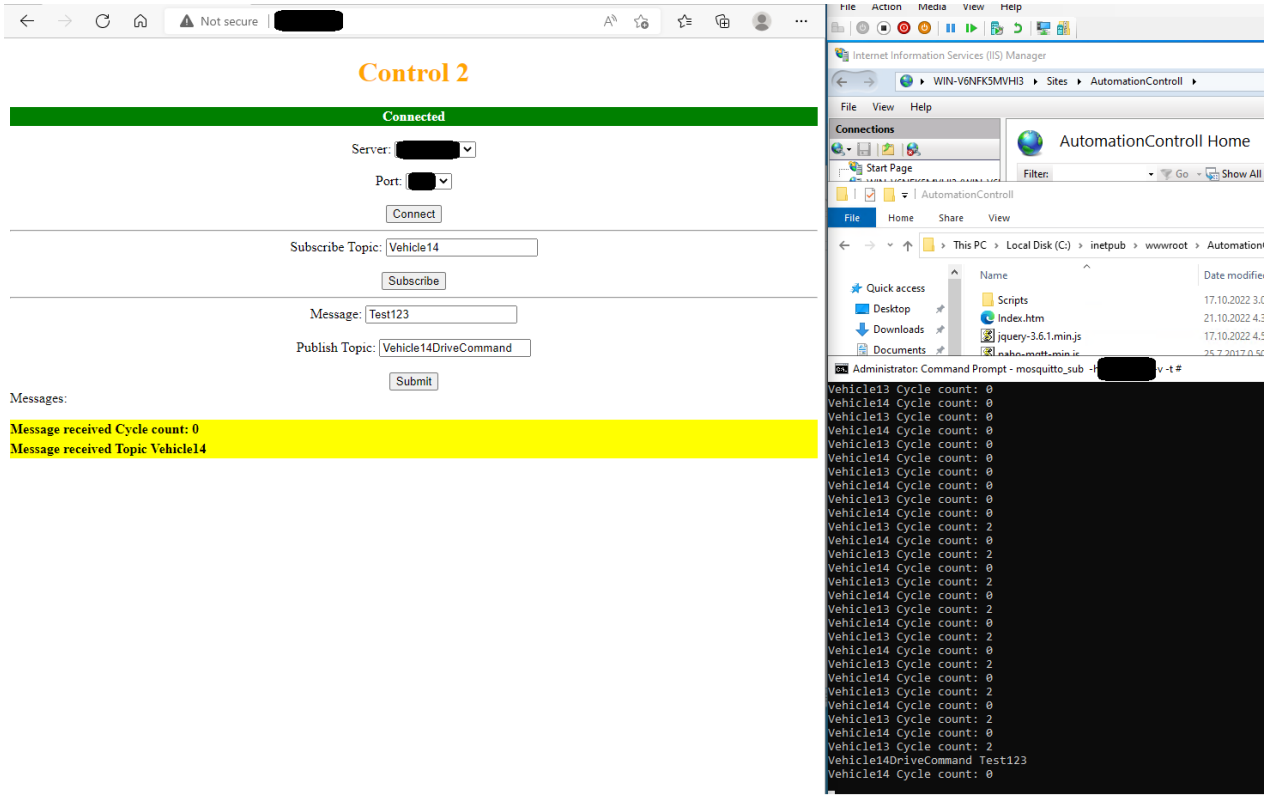
### 5.5.1 Testin kulku

Ajatuksena oli suorittaa testi siten, että käyttäjä avaa toisessa vaiheessa luodun verkkosivun koneellaan ollessaan kytkettynä testilaitteiden sisäisessä verkossa. Yhdistää palvelimeen, tilaa viestit otsikoilta Vehicle14 sekä lähettää viestit molemmille liitetyille laitteille otsikolla Vehicle13DriveCommand ja Vehicle14DriveCommand. Odotusarvona oli, että ohjelmoitava logiikka saa Vehicle13DriveCommand otsikolla lähetetyn viestin ja teollisuustietokone saa Vehicle14DriveCommand otsikolla lähetetyt viestit. Verkkosivun odotettiin näyttävän Vehicle14 otsikolla tulevat viestit teollisuustietokoneelta. Viestinvälittäjälle odotettiin ilmaantuvan kaikki viestit ja täten lukemisen helpottamiseksi ohjelmoitavan logiikan syklimuuttuja pakotettiin arvoon 2 testauksen ajaksi. Viestinvälittäjää seurattiin Raspberry Pi pienoistietokoneelta jatkuvana, sillä virtuaalikoneella toimivaa viestinvälittäjää kyettiin seuraamaan astetta hankalammin reaaliajassa.

### 5.5.2 Tulos

Testin tuloksena saatiin odotettu onnistunut viestinlähetys verkkosivulta teollisuustietokoneelle ja ohjelmoitavalle logiikalle. Verkkosivulle saatiin myös onnistuneesti tilatut syklitiedot teollisuustietokoneelta, elikkä Vehicle14:sta. Ohjelmoitava logiikka viestitti omat syklitietonsa vain viestinvälittäjälle asti. Kuvassa 18 on lopputuloksesta saatu kuvankaappaus, toki tässä vaiheessa verkkosivu avattu palvelinkoneelta, jonka päällä virtuaalikone pyörii. Lähetetty viesti otsikolla Vehicle14DriveCommand oli Test123. Tätä ennen testi suoritettiin erillisillä työkoneella saaden sama onnistunut lopputulos. Kuvassa 19 on teollisuustietokoneen TwinCAT-koodista sisään kirjautuneena luettuna sille lähetetty viesti, joka näkyy myös aikaisemmassa kuvassa 18.

Kuva 18. Vasemmalla luotu verkkosivu ja oikealla virtuaalikone ja MQTT-viestinvälittäjä




Kuva 19. Teollisuustietokoneen (Vehicle14) saama viesti

```
36
37    (*Viesti jonon tarkastelu*)
38  IF fbMessageQueue.nQueuedMessages 0 > 0 THEN
39      IF fbMessageQueue.Dequeue(fbMessage:=fbMessage) THEN
40          fbMessage.GetTopic(pTopic:=ADR(ReceivingTopic 'Vehicle14D' ), nTopicSize:=SIZEOF(ReceivingTopic 'Vehicle14D' ) );
41          fbMessage.GetPayload(pPayload:=ADR(ReceivingData 'Test123' ), nPayloadSize:=SIZEOF(ReceivingData 'Test123' ), bSetNullTermination:=FALSE);
42      END_IF
43  END_IFRETURN
```

Raspberry Pi pienoistietokoneelta seurattuna MQTT-viestinvälittäjä näytti siellä täsmälleen samalta kuin edellä mainitussa kuvassa. Saadun tuloksen valossa voitiin todeta konsepti toimivaksi, sillä lähetettävät viestit kulkivat verkkosivulta viestinvälittäjän kautta kohdelaitteille ja tilatulla otsikolla saatiin luettua verkkosivulta halutun laitteen viestit.

# 6 Jatkokehitys ja lisätutkimukset

Soveltuvuusselityksessä saadut tulokset olivat positiivisia, mutta kehityskohteita jatkoon kuitenkin löytyy. Erityisesti selainpohjaisen valvomosovelluksen kehittämistä olisi hyvä jatkaa

niin, että se vastaisi käytettävyydeltään valvomosovellusta. Lisäksi kohdeyrityksen tämänhetkinen automaatiokoodi vaatii kehitystä. Mielenkiintoa herätti myös pienoistietokone Raspberry Pi:n mutkaton toimivuus, joten Linux-ympäristön lisätutkimukset saattaisivat olla kohdeyrityksessä paikallaan. Tässä luvussa esitettävät suositukset ovat allekirjoittaneen henkilökohtaisia suosituksia, jotka perustuvat selvitystyön tuloksiin.

## 6.1 Valvomosovelluksen selainpohjaisen käyttöliittymän tuottaminen

Kuten kohdassa 2.3 olevasta selainpohjaisen valvomosovelluksen vaatimusmäärittelystä voidaan huomata sen olevan erittäin laaja ja laajenee entisestään, kun siihen lisätään tuotannonohjausjärjestelmän vaatimat osat. Kohdeyrityksellä tai ainakaan osastolla, jossa soveltuvuusselvitys tehtiin, ei ole minkäänlaista käytännön mahdollisuutta rakentaa kokonaisuutta itse. Täten suositeltavaa on, että kohdeyritys ostaa työn, joko sisäisesti toiselta osastolta kohdeyrityksessä tai sitten ulkopuoliselta toimijalta. Ulkopuolinen toimija olisi allekirjoittaneen suositus, sillä kohdeyritykseltä ei löydy näin laajaan työhön resursseja, eikä sen vuoksi ole järkevää investoida resursseihin, joista on sitten työn valmistuttua luovuttava. Onko sitten ulkopuolinen toimija, jokin tunnettu isompi kehitystalo vai olisiko kannattavampaa suorittaa työ jonkun korkeakoulun tutkimus-, kehitys- ja innovaatiotyönä.

## 6.2 Ohjelmoitavien logiikoiden ohjelmat

Testiautomaatiojärjestelmään kohdistuvaa jatkokehitystä on ainakin siihen liitettyjen ohjelmoitavien logiikoiden osalta koodin muokkaaminen. Nykyisessä tilassaan laitteiden kommunikaatiorajapintana on ADS-protokolla, mutta mikään ei estä tekemästä järjestelmässä sisäistä testiä, siten että liitetään oikeasti testattava laite ohjelmoitavaan logiikkaan, joka käyttää kommunikaatio rajapintanaan MQTT-protokollaa. Tämä tietenkin vaatii ensin määrittelyn siitä, millainen tulevaisuuden viesti ohjelmoitavalle logiikalle tulee olemaan, ja kuinka se parsitaan muodostamaan syklitiedot. Tämä vaihe saattaisi edesauttaa valvomosovelluksen selainpohjaisen käyttöliittymän tarvittavien ja haluttujen toimintojen määrittelemistä ennen varsinaista tuotetta. Samaan aikaan olisi mahdollista havaita, mikä ei ole mahdollista ja mitä muita rajoitteita järjestelmässä voisi olla. Lisähuomiona on mainittava, että ainakaan toistaiseksi MQTT-kirjastoa ei ole saatavissa TwinCAT 2 -

ohjelmointiympäristöön vaan on käytettävä ohjelmoitavaa logiikkaa, joka on varustettu TwinCAT 3:lla.

### 6.3 Linux-ympäristö

Linux-ympäristö ei ole kohdeyrityksen IT puolen hyväksymä siinä mielessä, että sen voisi yhdistää kohdeyrityksen omaan verkkoon, mutta testiautomaatiojärjestelmän automaatioverkko ja siihen liitetyt on erotettu kohdeyrityksen sisäisestä. Testiautomaatioverkkoon liitetyt laitteet ovat niin kutsuttuja osaston omia laitteita, mitä ei kohdeyrityksen käytäntöjen vuoksi ole sallittua liittää kohdeyrityksen sisäiseen verkkoon. Näen suuren syyn tutkia Linux pohjaista järjestelmää osana tulevaa automaatiojärjestelmää, sillä tässä työssä huomattiin sen helppokäyttöisyys Microsoft Windows-järjestelmään verrattuna. Kohdeyrityksen käytäntöjen vuoksi Linux-järjestelmä ei kuitenkaan voisi olla pääkäyttöjärjestelmä, mutta voisi erittäin hyvin toimia tukevana osana, varajärjestelmänä tai valvontajärjestelmänä.

## 7 Päätelmät ja yhteenveto

Tässä opinnäytetyössä suoritettiin soveltuvuusselvitys kohdeyrityksen automaatiojärjestelmän valvomosovelluksen toteuttamisesta selainpohjaisena ohjelmistona. Lopputuloksena voidaan todeta konsepti toimivaksi, edellyttäen kommunikaatiorajapinnan olevan ADS-protokollan sijaan MQTT-protokolla. Soveltuvuusselvitykselle asetettuihin tavoitteisiin päästiin. Jatkokehitettävää ja tutkittavaa selainpohjaista ratkaisua varten riittää, jotta kohdeyrityksellä olisi mahdollisimman hyvät lähtökohdat aloittaa järjestelmän uusiminen.

Työn kirjallisuuskatsauksessa saatiin selvitettyä asiantuntijahaastattelun perusteella, että selainpohjainen valvomosovellus sopisi paremmin kohdeyrityksen tarpeisiin. Selainpohjaisen valvomosovelluksen etuja paikallisesti asennettuun verrattuna ei kuitenkaan tässä työssä päästy käytännönasteella testaamaan. Soveltuvuusselvityksen lopputuloksesta voitiin kuitenkin todeta selainpohjaisen valvomosovelluksen olevan ylipäätään mahdollinen ja että

MQTT-protokolla on toimiva kommunikaatiorajapinta selainpohjaisen ohjelmiston ja ohjelmoitavan logiikan välillä.

Tiedonsiirto-/kommunikaatiorajapinnan vaihtamiseen ADS-protokollaa joustavampaan vaihtoehtoon on kohdeyrityksessä syytä paneutua, ja muistaa ettei MQTT-protokolla ole suinkaan ainoa vaihtoehto, vaikka olikin tässä työssä käytössä. Samassa valvomosovelluksen uudistaminen paikallisesti asennetusta ohjelmistosta selainpohjaiseen ratkaisuun on mietittävä huolella, sillä ohjelmistomallin muuttamisen on tultava selvästä tarpeesta. Nykyinen toimiva järjestelmä on ollut käytössä vuodesta 2011, eikä toistaiseksi vaadi välitöntä uudistusta. Syytä on kuitenkin jatkaa korvaavan järjestelmän tutkintaa, sillä järjestelmän uusimisen tullessa ajankohtaiseksi, on suunnitelman korvaavasta järjestelmästä oltava valmis. Järjestelmää uudistaessa ei ole syytä vaihtaa vain yhtä osaa vaan pyrkiä kerralla uudistamaan iso osa järjestelmästä, jotta ei jäädä tilanteeseen, jossa vanhaa ja uutta yritetään käyttää rinnakkain. Järjestelmän uudistaminen vaatii kaiken lisäksi resursseja, aikaa ja rahaa. Mikään ei myöskään takaa, että uudistettu järjestelmä toimisi vanhaa paremmin ja voi aiheuttaa käyttäjissä vastarintaa, mikäli uudistuksesta koituu käyttäjille lisätyötä tai ylimääräistä vaivaa.

Tämän opinnäytetyön aineistoa ja soveltuvuusselvityksessä luotua ratkaisua tullaan hyödyntämään kohdeyrityksen uuden automaatiojärjestelmän suunnittelussa ja tulevissa jatkotutkimuksissa. Tuloksena kohdeyritys sai lisätietoa MQTT-protokollasta ja sen käytettävyys eroista Windows ja Linux-ympäristöissä, ohjelmistomalleista, palvelimista ja verkkosivujen luomisesta. Saatu lisätieto ja osaaminen tulee olemaan merkittävässä roolissa, mikäli kohdeyrityksessä päädytään valitsemaan selainpohjainen ratkaisu uudeksi valvomosovellukseksi.

## Lähteet


Barteling, G. (8.3.2020). *TwinCAT and MQTT – Part 1 Getting started*. https://www.plccoder.com/twincat-and-mqtt-part-1/

Beckhoff Information System. (n.d.-a). *ADS-Communication*. https://infosys.beckhoff.com/english.php?content=../content/1033/cx8190_hw/5091854987.html&id=

Beckhoff Information System. (n.d.-b). *IotMqttSampleUsingQueue*. https://infosys.beckhoff.com/english.php?content=../content/1033/tf6701_tc3_iot_communication_mqtt/45035999665460363.html&id=

Beckhoff Information System. (n.d.-c). *TF6701 | TwinCAT 3 IoT Communication MQTT*. https://infosys.beckhoff.com/english.php?content=../content/1033/tf6701_tc3_iot_communication_mqtt/3518541195.html&id=

Beckhoff. (n.d.). *Tukipalvelumme*. https://www.beckhoff.com/fi-fi/support/our-support-services/

Cope, S. (31.8.2022). *Using The JavaScript MQTT Client With Websockets*. http://www.steves-internet-guide.com/using-javascript-mqtt-client-websockets/

Cope, S. (n.d.). *How to Install The Mosquitto MQTT Broker on Windows*. http://www.steves-internet-guide.com/install-mosquitto-broker/

Eclipse Foundation. (n.d.-a). *Download*. https://mosquitto.org/download/

Eclipse Foundation. (n.d.-b). *Eclipse Mosquitto™*. https://mosquitto.org/

Häkkinen, S. (2021). *MQTT-protokollan soveltuvuus testiautomaatiojärjestelmän tiedonsiirtoon.* [Insinöörityö, Metropolia Ammattikorkeakoulu]. https://urn.fi/URN:NBN:fi:amk-202104275959

Klusaitė, L. (9.3.2022). *TCP IP -mikä se on, mihin sitä tarvitaan ja mitä se tekee?* https://nordvpn.com/fi/blog/tcp-ip-protokolla/

Microsoft. (26.4.2022a). *Create a Virtual Machine with Hyper-V on Windows 10 Creators Update*. Haettu 27.12.2022 osoitteesta https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/quick-create-virtual-machine

Microsoft. (26.4.2022b). *Install Hyper-V on Windows 10*. Haettu 27.12.2022 osoitteesta https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v

Microsoft. (19.12.2022c). *Software Center user guide*. Haettu 27.12.2022 osoitteesta https://learn.microsoft.com/en-us/mem/configmgr/core/understand/software-center

Microsoft. (n.d.). *Evaluate Windows Server 2022*. https://info.microsoft.com/ww-landing-windows-server-2022.html

MQTT. (n.d.). *MQTT: The Standard for IoT Messaging*. https://mqtt.org/

O'Connor, P. D. & Kleyner, A. (2012). *Practical Reliability Engineering.* Wiley.

Random Nerd Tuotrials. (10.6.2022). *Install Mosquitto MQTT Broker on Raspberry Pi*. Haettu 27.12.2022 osoitteesta https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/

Raspberry Pi. (n.d.). *Raspberry Pi OS*. https://www.raspberrypi.com/software/

Salminen, E. (2015). *Luotettavuuslaboratorion tieto- ja testiautomaatiojärjestelmän vaatimusmäärittely.* [Insinöörityö, Metropolia Ammattikorkeakoulu]. https://urn.fi/URN:NBN:fi:amk-2015110515985

W3Schools. (n.d.). *Learn to Code*. Haettu 27.12.2022 osoitteesta https://www.w3schools.com/

Warren, G. (25.6.2021). *What Are Browser-Based Tools and Applications?* https://www.lifewire.com/what-are-browser-based-tools-2377407

**Liite 1: MQTT-TwinCAT koodi (Barteling, 2020; Beckhoff Information System, n.d.-b; Häkkinen, 2021, ss. 46-50)**

```
PROGRAM MAIN
VAR
    (*Viestin julkaisun parametrit*)
    fbMqttClient: FB_IotMqttClient; //TwinCAT 3:n MQTT-asiakaskirjasto TC3_IotBase:n toimintalohko.
    TopicToPublish : STRING(255) := 'Vehicle13'; //Julkaistavan viestin otsikko
    MessageToPublish : STRING(255); //Viestin hyötykuorma.
    fbSendMessageIntervalTimer : TON := (PT:=T#30S); //Viestin julkaisuväli

    (*Viestin tilauksen parametrit*)
    fbMessageQueue: FB_IotMqttMessageQueue; //Viestijono, johon julkaistut viestit voidaan tallentaa
    fbMessage: FB_IotMqttMessage;
    Subscribed:BOOL; //Tilauksen statusta kuvaava muuttuja
    TopicToSubscribe : STRING(255) := 'Vehicle13DriveCommand'; // Tilatun viestin otsikko
    ReceivingTopic:STRING(255); //Vastaanotetun viestin otsikko
    ReceivingData:STRING(255); //Vastaanotetun viestin hyötykuorma

    Cycle   : int := 0;
END_VAR
```

```
(*CONNECT-Paketin parametrien asettaminen*)
IF _TaskInfo[GETCURTASKINDEXEX()].FirstCycle THEN
    fbMqttClient.sHostName := [redacted]; //Välittäjän nimi
    fbMqttClient.nHostPort := 1883; //MQTT:n käyttämä portti.
    fbMqttClient.sTopicPrefix := ''; //Otsikon etuliite
    fbMqttClient.sClientId := 'Vehicle13'; //Asiakastunnus
    fbMqttClient.ipMessageQueue := fbMessageQueue; //Viestijono, johon julkaistut viestit tallennetaan
END_IF

(*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('Cycle count: ' , INT_TO_STRING (Cycle));

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

(*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
```

**Liite 2: Verkkosivun HTML-koodi (Cope, 2022)**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<style>
#messages
{
background-color:yellow;
font-size:3;
font-weight:bold;
line-height:140%;
}
#status
{
background-color:lightblue;
font-size:4;
font-weight:bold;
color:white;
line-height:140%;
}
#boxes
{
width: 350px;
border: 15px lightblue;
padding: 20px;
}



</style>
  <head>
    <title>Control 2</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
       <script src="paho-mqtt-min.js" type="text/javascript"></script>
       <script type = "text/javascript"
          src = "jquery-3.6.1.min.js"></script>
    <script type = "text/javascript">


       function onConnectionLost(){
       console.log("connection lost");
       document.getElementById("status").innerHTML = "Connection Lost";
       document.getElementById("messages").innerHTML ="Connection Lost";
       connected_flag=0;
       }
       function onFailure(message) {
             console.log("Failed");
             document.getElementById("messages").innerHTML = "Connection Failed- Retrying";
        setTimeout(MQTTconnect, reconnectTimeout);
        }
             function onMessageArrived(r_message){
             out_msg="Message received "+r_message.payloadString+"<br>";
             out_msg=out_msg+"Message received Topic "+r_message.destinationName;
             //console.log("Message received ",r_message.payloadString);
             console.log(out_msg);
             document.getElementById("messages").innerHTML =out_msg;
             }
       function onConnected(recon,url){
       console.log(" in onConnected " +reconn);
       }
       function onConnect() {
       document.getElementById("messages").innerHTML ="Connected to "+host +" on port "+port;
       connected_flag=1
       document.getElementById("status").innerHTML = "Connected";
       document.getElementById("status").style.backgroundColor = 'Green';
       console.log("on Connect "+connected_flag);
         }
```

```
function MQTTconnect() {
   document.getElementById("messages").innerHTML ="";
   var s = document.forms["connform"]["server"].value;
   var p = document.forms["connform"]["port"].value;
   if (p!="")
   {
   console.log("ports");
          port=parseInt(p);
          console.log("port" +port);
          }
   if (s!="")
   {
          host=s;
          console.log("host");
          }
   console.log("connecting to "+ host +" "+ port);
   var x=Math.floor(Math.random() * 10000);
   var cname="orderform-"+x;
   mqtt = new Paho.MQTT.Client(host,port,cname);
   var options = {
    timeout: 3,
          onSuccess: onConnect,
          onFailure: onFailure,

 };

    mqtt.onConnectionLost = onConnectionLost;
    mqtt.onMessageArrived = onMessageArrived;

   mqtt.connect(options);
   return false;


   }
   function sub_topics(){
          document.getElementById("messages").innerHTML ="";
          if (connected_flag==0){
          out_msg="<b>Not Connected so can't subscribe</b>"
          console.log(out_msg);
          document.getElementById("messages").innerHTML = out_msg;
          return false;
          }
   var stopic= document.forms["subs"]["Stopic"].value;
   console.log("Subscribing to topic ="+stopic);
document.getElementById("messages").innerHTML ="Subscribed to " +stopic;
   mqtt.subscribe(stopic);
   return false;
   }
   function send_message(){
          document.getElementById("messages").innerHTML ="";
          if (connected_flag==0){
          out_msg="<b>Not Connected so can't send</b>"
          console.log(out_msg);
          document.getElementById("messages").innerHTML = out_msg;
          return false;
          }
          var msg = document.forms["smessage"]["message"].value;
          console.log(msg);

          var topic = document.forms["smessage"]["Ptopic"].value;
          message = new Paho.MQTT.Message(msg);
          if (topic=="")
                 message.destinationName = "test-topic"
          else
                 message.destinationName = topic;
          mqtt.send(message);
          return false;
   }

</script>
```

```
  </head>
  <body>
    <h1 align="center" style="color:orange">Control 2</h1>

           <script type = "text/javascript">
//ll

</script>
       <script>
       var connected_flag=0
       var mqtt;
    var reconnectTimeout = 2000;


       </script>

<div id="status" align="center">Connection Status: Not Connected</div>
</div>
<br>
        <form align="center" name="connform" action="" onsubmit="return MQTTconnect()">

        <label for="server">Server:</label>
        <select name="server" id="server">
            <option value="[redacted]">[redacted]</option>
        </select><br><br>

        <label for="port">Port:</label>
        <select name="port" id="port">
            <option value="[redacted]">[redacted]</option>
        </select><br><br>

<input type="submit" value="Connect">
</form>
<hr>
<div align="center">
<form name="subs" action="" onsubmit="return sub_topics()">
Subscribe Topic:   <input type="text" name="Stopic"><br><br>

<input type="submit" value="Subscribe">
</form>
</div>
<hr>
<form align="center" name="smessage" action="" onsubmit="return send_message()">

Message: <input type="text" name="message"><br><br>
Publish Topic:   <input type="text" name="Ptopic"><br><br>
<input type="submit" value="Submit">
</form>
Messages:<p id="messages"></p>

  </body>
</html>
```