LAAJENNUS PUNNITUKSEN PILVIPALVELUUN

Lahti Precision Oy (Proof of Concept)

LAHDEN AMMATTIKORKEAKOULU
Insinööri (AMK)
Kone- ja tuotantotekniikan koulutus
Mekatroniikka
Syksy 2018
Markus Järvinen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Järvinen, Markus	Opinnäytetyö, AMK	Syksy 2018
	Sivumäärä	
	51	

Työn nimi

Laajennus punnituksen pilvipalveluun

Lahti Precision Oy (Proof of Concept)

Tutkinto

Insinööri (AMK)

Tiivistelmä

Opinnäytetyön tarkoituksena oli tehdä Proof of Concept -malli laajennuksesta Lahti Precision Oy:n punnituksen pilvipalveluun. Aihe liittyi Lahti Precision Oy:n uuden mScales-tuotteen kehitystyöhön. Laajennus punnituksen pilvipalveluun kattaa fyysisiä kenttälaitteita, yhdistävän päälaitteen ja mScales-palvelun. Tarkoituksena oli todentaa konseptointia laitteiden liittämisestä pilvipalveluun. Yhdistävältä päälaitteelta vaaditaan kykyä yhdistää teollisuuden komponentit pilvipalveluun. Tämä tarkoittaa usean erilaisen laitteen yhdistämistä päälaitteeseen, joka yhdistyy mScales-palveluun.

Teoriaosuus käsittelee tiedonsiirron ja laitteiden yhteyskäytäntöjä. Työ aloitettiin ideointivaiheella, jossa pohdittiin sopivia laiteratkaisuja laajennuksen tekemiseen. Tavoitteena oli löytää sopiva laite, johon voitiin yhdistää erilaisia teollisia komponentteja ja pilvipalvelu. Ideointivaiheen jälkeen edettiin testausvaiheeseen, jonka aikana testattiin valittua laitetta siten, että voitiin todeta sen ominaisuudet riittäviksi laajennuksen Proof of Concept -mallin tekemiseen. Viimeisessä vaiheessa eli toteutusvaiheessa luotiin valittujen komponenttien avulla Proof of Concept oikeasta tilanteesta. Opinnäytetyön toteuttamiseen käytettiin IOT2040-yhdyskäytävää ja Node-RED-ohjelmointityökalua.

Laajennuksen konsepti saatiin todennettua simuloimalla oikeaa tilannetta. Opinnäytetyön avulla saatiin Lahti Precision Oy:n kehitystyötä eteenpäin ja todennettua yhdyskäytävän ja Node-RED-työkalun toimintaa. Tulosten perusteella tiedetään paremmin etenemisprosessi teollisen internetin sovelluksen kehittämiselle. Kehitystyö laajennuksen parissa jatkuu.

Asiasanat

IoT, kehitystyö, Node-RED, Proof of Concept, teollinen internet

Abstract

Author(s)	Type of publication	Published
Järvinen, Markus	Bachelor's thesis	Autumn 2018
	Number of pages	
	51	

Title of publication

Extension to the cloud service of weighing

Lahti Precision Oy (Proof of Concept)

Name of Degree

Bachelor of Engineering

Abstract

The subject of the Bachelor's thesis was to produce a Proof of Concept -model of the extension, which was made for the cloud service of weighing. The subject is a part of development of Lahti Precision Ltd. This thesis deals with industrial devices, the main device and the mScales-cloudservice. The main purpose was to proof the concept of extension. The main device of the extension must be able to connect different industrial components to cloud service.

The thesis contains theory of data transfer and device protocols. The practical part of the thesis was defided into three steps. The first step is the planning point, where the different devices are studied. The main purpose of this point is to find the right device with sufficient properties to perform the extension. After planning, the device was tested. The purpose of the testing point is to prove that the device will be able to connect the field devices to the cloud service. Finally, a demo application for the Proof of Concept was created. IOT2040-gateway was used as the main device and the program was created with Node-RED-tool.

The concept of the extension was proved with the application, which simulated the real application. The thesis proofed that a working application could be created with the IOT2040-gateway and Node-RED. Knowledge of process for developing the industrial network applications was improved. The development of the extension will continue.

Keywords

IoT, development, Node-RED, Proof of Concept, industrial network

SISÄLLYS

LYHENTEET JA MÄÄRITELMÄT

Alivirtaus	Tarkoittaa Node-RED-työkalussa subflow -ominaisuutta.
COTP	Yhdeltä käyttäjältä toiselle käyttäjälle tietopaketteja lähettävä protokolla.
GET	HTTP-protokollan metodi, jota käytetään tiedon hakemisessa web-sivulta.
IoT	Muodostuu sanoista Internet of Things, joka tarkoittaa esineiden internetiä.
OSI-malli	Kuvaus tiedonsiirron protokollien yhdistelmästä seitsemässä kerroksessa. Jokainen kerros käyttää yhtä alemman kerroksen palvelua ja tarjoaa palvelua yhden kerroksen ylöspäin.
POST	HTTP-protokollan metodi, jota käytetään tiedon lähettämiseen web-sivulle.
Proof of Concept	Menetelmä osoittaa jokin idea, malli tai konsepti toteuttamiskelpoiseksi.
Protokolla	Protokolla eli yhteyskäytäntö tarkoittaa kommunikointitapaa kahden eri järjestelmän välillä.
Solmu	Tarkoittaa Node-RED-työkalussa yksittäistä lohkoa (node), jolla on yksilölliset ominaisuudet.
Virtaus	Tarkoittaa Node-RED-työkalussa flow-ominaisuutta, joka on käytännössä yksi ohjelmakokonaisuus käsittäen alivirtauksia ja solmuja.

1 JOHDANTO

Opinnäytetyön aiheena on laajennuksen tekeminen punnituksen pilvipalveluun Proof of Concept -työnä. Tarkoituksena on etsiä ominaisuuksiltaan oikea laite, jonka avulla laajennus voidaan suorittaa. Sopivan laitteen löytyessä testataan sitä ja toteutetaan Proof of Concept, joka simuloi oikeaa tilannetta. Opinnäytetyötä tarjottiin Lahden ammattikorkeakoulun sivulla, ja aiheessa kiinnosti teollisten laitteiden ja pilvipalvelun kohtaaminen. Olen opinnoissani myös erityisen kiinnostunut automaatio- ja ohjelmistosuunnittelusta, joten opinnäytetyön aihe kuulosti sopivalta. Opinnäytetyö tehdään Lahti Precision Oy:lle. Opinnäytetyö aloitettiin joulukuussa 2017 ja sen toiminnallisen osuuden valmistumisen ajankohdaksi sovittiin vuoden 2018 toukokuu. Opinnäytetyötä tehdään Lahti Precision Oy:n tiloissa ja se on osa mScales-pilvipalvelun kehitystyötä.

Opinnäytetyö jaoteltiin kolmeen vaiheeseen: ideointi-, testaus- ja toteutusvaiheeseen. Ideointivaiheessa etsitään sopivaa laitetta, jolla voidaan yhdistää teollisia komponentteja ja pilvipalvelu toisiinsa. Teoreettisessa osuudessa kerrotaan tietoja tutkituista laitteista ja niiden ominaisuuksista. Testausvaiheessa tavoitteena on todentaa laitteen ominaisuudet riittäviksi, testaamalla erilaisten laitteiden yhdistämistä sillä. Testausvaiheen teoreettisessa osuudessa on sisältöä käytetyistä työkaluista ja eri laitteiden yhdistämisestä päätelaitteeseen. Toteutusvaihe on työn viimeinen vaihe, jossa todennetaan konsepti simuloimalla oikeaa sovellusta. Opinnäytetyön laajennuksen laitteisiin kuuluu ainakin etälukija, Lahti Precision Oy:n vaakapääte ja ohjelmoitava logiikka.

Opinnäytetyössä hyödynnän Lahden ammattikorkeakoulussa oppimiani taitoja automaatio- ja ohjelmistosuunnittelussa. Lähteisiin kuuluu alan kirjallisuutta ja aiheeseen liittyviä Internet-lähteitä. Työssä käsitellään erilaisia protokollia, Proof of Concept -mallia, punnitusta ja kehitystyötä. Opinnäytetyön tavoitteena on löytää oikeat työkalut ja päätelaite Proof of Concept -mallin toteutusta varten sekä saada simuloitua oikeaa tilannetta.

2 TOIMEKSIANTAJA

2.1 Yleistä

Lahti Precision Oy on Pohjoismaiden johtava asiantuntija punnitus- ja annostusteknologian alalla. Yritys toimittaa pääosin laastitehtaita ja teollisia punnitusratkaisuja. (Lahti Precision Oy, 2016d.) Lahti Precision Oy:n tarjontaan kuuluvat muun muassa vaa'at, lamellikuljettimet ja sekoituslaitteet prosessiteollisuudelle, punnituskomponentit, punnitusautomaatio ja punnitustiedon hallintajärjestelmät (Lahti Precision Oy 2016e). Yrityksen päätoimipiste sijaitsee Lahden Sopenkorvessa. Lahti Precision Oy:n liikevaihto oli 21 miljoonaa euroa vuonna 2016 ja se työllisti 140 ihmistä. (Kauppalehti 2016.)

2.2 Historia

Tehtailija P. Kuivalainen perusti konepajayrityksen Lahteen vuonna 1908. Yritys korjasi koneita ja valmisti rautasänkyjä. Varsinainen historia punnitusteknologialle sai alkunsa vuonna 1914, kun rautasänkyjen tuotanto korvattiin vaakojen valmistuksella ja yritys kehitti osaamistaan punnitusteknologian alueella. Uusi yhtiö perustettiin, jonka nimeksi tuli Lahden Vaaka Oy. (Lahti Precision Oy 2016a.)

Lahden Vaaka Oy valmisti ensin vaakoja kauppoihin ja kotitalouksiin, mutta 1930-luvulla yritys alkoi kehittämään uusia vaakatyyppejä teollisuus- ja varastokäyttöön sekä kuormavaakoja ajoneuvojen punnitsemiseen. 1960-luvulla Lahden Vaaka Oy panosti tuotekehitykseen ja yritys kehitti sähköisen koodianturin, jonka ensimmäinen toimitus tehtiin vuonna 1964. Yhtiön oma elektroniikkatehdas aloitti toimintansa vuonna 1975, kun sähköiset vaa'at ja automaatio yleistyivät. (Lahti Precision Oy 2016a.)

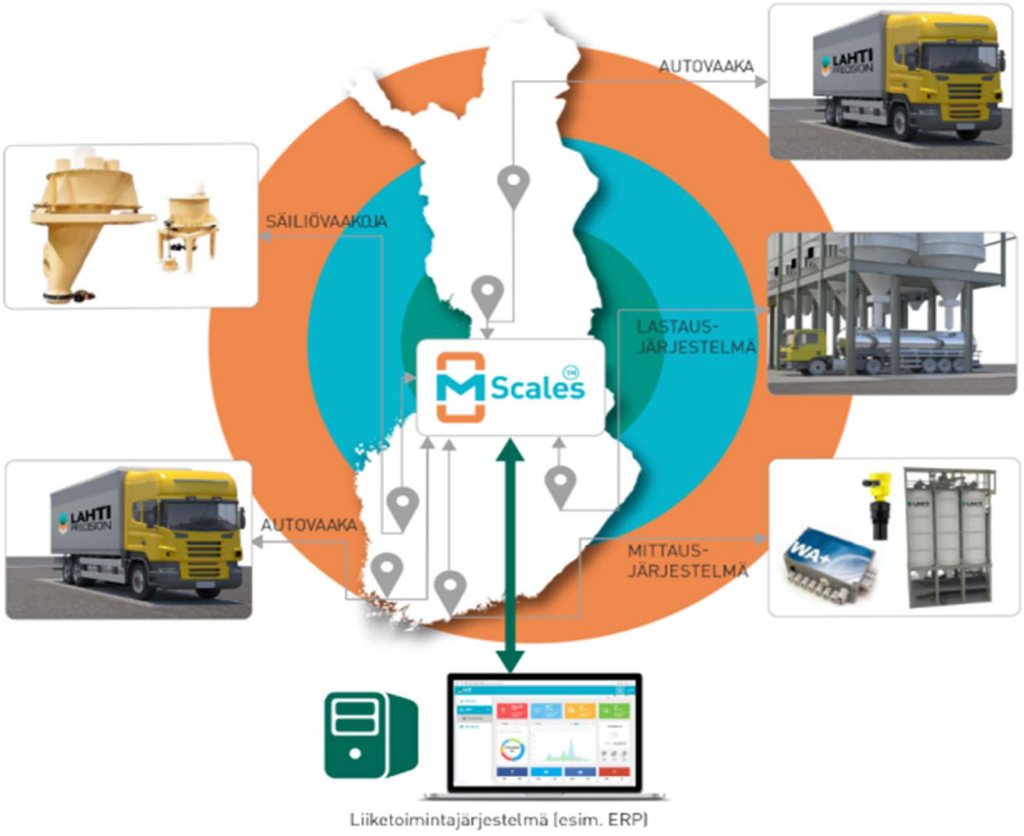
Yrityksen asema prosessikokonaisuuksien ja laitosten toimittajana vahvistui, kun se kansainvälistyi 1980-luvulla. Ulkomailla oli kysyntää punnitus- ja annostusjärjestelmille, irtomateriaalin käsittelylle ja näiden automaation osaamiselle. Yritys rakensi mittaus- ja materiaalilaboratorion. Yrityksen toimintaa jatkettiin vuoteen 2007 asti nimellä Raute Precision, kunnes se muutettiin Lahti Precision Oy:ksi. Yrityksen sisälle muodostui kolme eri liiketoiminta-aluetta: laasti- ja lasiteollisuus sekä teollinen punnitus. (Lahti Precision Oy 2016a.)

2.3 mScales-verkkopalvelu

mScales-verkkopalvelu on Lahti Precision Oy:n tuotekehityksen tulos. Verkkopalvelu tuo punnituspalvelut nykyaikaan yhdistämällä yrityksen vaa'at ja liiketoimintajärjestelmä yhdeksi kokonaisuudeksi (kuva 1). (Lahti Precision Oy 2017.)

mScales™-palvelun tarjoamat mahdollisuudet ovat rajattomat. Se mahdollistaa nopeampia ja tehokkaampia prosesseja varastointiin, punnitukseen ja annostukseen sekä materiaalien käsittelyyn ja logistiikkaan. (Lahti Precision Oy 2017.)

mScales-verkkopalvelu tarjoaa mobiilikäyttöliittymät vaa'oille sekä turvallisen ja helpon liittymisen liiketoimintajärjestelmiin. Punnitukseen liittyvä tietojen hallinta ja raportointi tehdään palvelun avulla keskitetysti. Punnitustapahtuma on tehokkaampi ja kustannuksiltaan edullisempi. mScales tarjoaa myös säästöä ja varmuutta, koska perinteiset oheislaitteet korvautuisivat verkkopalvelun tarjoamilla käyttöliittymillä. Perinteisiin oheislaitteisiin kuuluu muun muassa kirjoittimia, etätunnistimia, PC-tietokone ja kuljettajaterminaali. Kiintolevyrikkojen ja ukkosvahinkojen aiheuttamat riskit sekä huollon tarve vähenevät. mScales-verkkopalvelun tyypillisiä käyttökohteita ovat esimerkiksi jäteasemat, lastausasemat, biovoimalaitokset, teollisuuslaitosten portit ja sorakuopat. (Lahti Precision Oy 2017.)



Kuva 1. mScales-verkkopalvelu siirtää punnitusten käsittelyn internetin verkkopalveluun (Lahti Precision Oy 2016c)

3 TEOLLINEN INTERNET

> *Teollinen internet on esineiden, koneiden, tietokoneiden ja ihmisten internet, joka mahdollistaa älykkäät teolliset operaatiot käyttäen kehittynyttä data-analytiikkaa muutoksiin johtavien liiketoiminnallisten tulosten aikaansaamiseksi. Se ilmentää sitä, kuinka globaali teollinen ekosysteemi, kehittynyt tietojenkäsittely, kehittynyt valmistus, kaikkialle leviävä anturointi ja kaikkialle ulottuvat verkot yhdistyvät. (Collin & Saarelainen 2016, 34.)*

Teollinen internet käsittää teollisuuteen tarkoitettujen ethernet-pohjaisten laitteiden ja liityntöjen kirjon. Tähän joukkoon kuuluu myös langattomat PAN/WPAN- ja WLAN-verkkolaitteet ja niissä kytkettyinä olevat tietoa lähettävät anturit. Teollinen internet perustuu toimistoverkoissa käytettäviin ethernet-standardin protokolliin. (Collin & Saarelainen 2016 167.)

Ympäristö vaatii teollisen internetin komponenteilta vaihtelevien ja rajujen olosuhteiden vaihtelua. Teollisessa ympäristössä voi olla voimakkaita lämpötilan vaihteluita, likaa, tärinää ja sähköisiä häiriöitä. Lisäksi jotkin komponentit voivat olla sijoitettuina koko vuoden ulkona, jolloin niiltä vaaditaan kosteuden ja pakkasen sietoa. Teollisuudessa automaatiolaitteet verkotetaan muun muassa kytkimillä, reitittimillä ja mediamuuntimilla. Teollinen internet sisältää analytiikka ja käyttäjien analysointia. Tästä esimerkkinä itseohjautuvat tehtaat, jotka optimoivat energiankäyttöänsä tai minivoivat ylijäämää tuotantoprosessissa. (Collin & Saarelainen 2016, 168.) Teollisen internetin mahdollistavat esineiden internetin älykkäät laitteet ja esineet. Esineiden internet taas viittaa fyysisiin laitteisiin sekä infrastruktuuriin, joilla on yhteys internetiin. Näillä komponenteilla on kyky tunnistaa itsensä muiden laitteiden joukosta. (Tieto Oyj 2018.)

Punnituksen pilvipalvelun laajennuksen teossa yhdistetään päätelaitteen avulla teollinen internet osaksi esineiden internetiä. Päätelaite käsittelee kaikki sinne lähetettävät tiedot, minkä jälkeen se lähettää ne eteenpäin verkkopalvelulle. Asiakas käyttää näitä teollisen internetin tietoja tietokoneellaan tai älypuhelimellaan, jolla hän on liittynyt verkkopalveluun. Tarkoituksena on todentaa teollisen internetin ja verkkopalvelun yhdistämistä ja mahdollisesti tulevaisuudessa liittää monimuotoinen laitevalikoima verkkopalvelun käyttöön.

4 PROOF OF CONCEPT -MALLI

Teollisen tai esineiden internetin sovellutuksen ideoinnin jälkeen kriittinen ensiaskel on toteuttaa Proof of Concept eli lyhennettynä PoC. Nimensä mukaisesti tarkoituksena on todentaa sovellutuksen ideoiden konseptia. Valikoituja ideoita koetellaan niin, että maksimoidaan mahdollisen pilottihankkeen onnistumisen todennäköisyys ja minimoidaan riskit. Proof of Concept -malli on enemmän valmistelua kuin todellista testaamista. PoC-mallia tehdessä suunnitellaan ja pohditaan idean tarpeita, haluttua lopputulosta ja arvioidaan käytettävissä olevaa tietoa. PoC-mallilla on vähintään kolme perustavoitetta, jotka ovat toteuttamiskelpoisuus, teknisten haasteiden varhainen tunnistaminen ja onnistumisen keston arviointi. (Collin & Saarelainen 2016, 283-284.)

Toteuttamiskelpoisuus on ensimmäinen tavoite PoC-mallissa, joka täytyy saavuttaa ennen muita tavoitteita. Ensimmäinen perustavoite vaatii asiantuntijuutta ja testilaitteita. Toteuttamiskelpoisuutta tutkitaan esimerkiksi virtualisoiduissa hiekkalaatikkoympäristöissä, joissa voidaan simuloida eri laitteiden toimivuutta halutulla tavalla. Toinen perustavoite on teknisten haasteiden varhainen tunnistaminen. Teknisiä haasteita selvitetään ennen mahdollista pilottihankkeeseen etenemistä, jolloin projektiin liittyvät asiantuntijat saavat arvokasta tietoa siitä, mihin hankkeessa täytyy varautua. PoC-mallissa siirrytään kolmannen perustavoitteen selvittämiseen, kun toinen perustavoite on saavutettu. Kolmannessa perustavoitteessa on saatava käsitys siitä, kuinka nopeasti mahdollisen tulevaisuuden pilottihankkeen voi viedä loppuun asti. Proof of Concept -mallin avulla yrityksen teknisen alueen työntekijät saavat paremman käsityksen työmäärästä pilottihankkeessa. (Collin & Saarelainen 2016, 283-284.)

PoC-malli on kustannuksiltaan halvimmillaan muutamien tuhansien eurojen arvoinen. PoC-mallin luonne riippuu täysin yrityksen tavoitteista ja tuotteistettavasta asiasta. (Collin & Saarelainen 2016, 283-284.) Laajennuksen tekeminen pilvipalveluun toteutetaan Proof of Concept -mallina. PoC-malleista oli hyvin vähän tietoa, mutta Teollinen internet -kirjassa käsiteltiin yleispätevästi asiaa kahden sivun verran. Englanninkielisiä lähteitä aiheesta löytyi, mutta niissä käsiteltiin lähinnä mekaanisia prototyyppejä, jolloin vertailukelpoista tietoa ei löytynyt.

5 LAAJENNUKSEN VAIHEET

5.1 Ideointivaihe

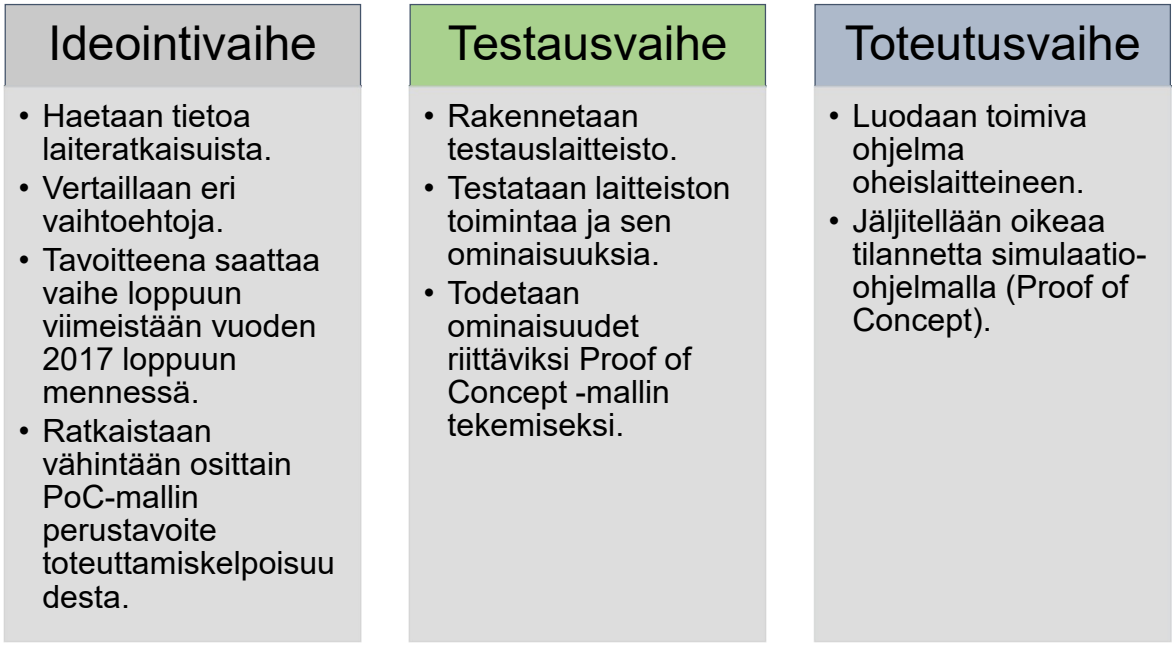
Ideointivaiheessa aloitetaan erilaisten vaihtoehtojen etsiminen laajennuksen tekemiseen. Toteutuksessa pyritään hyödyntämään valmiita komponentteja, ja PLC-laitteina käytetään Siemens- tai Beckhoff-yrityksen tuotteita. Ideointivaiheessa tutkitaan erilaisia laitevaihtoehtoja, joiden ominaisuudet riittävät laajennuksen tekemiseen. Tällaisia vaihtoehtoja ovat sellaiset laitteet, joilla voidaan yhdistää useita erilaisia kenttälaitteita verkkopalveluun. Yhdistettäviin kenttälaitteisiin kuuluu ohjelmoitava logiikka, Lahti Precision Oy:n autovaakapääte ja etälukija. Laitteeseen luotaisiin yksilölliset rajapinnat eri protokollia käyttäville laitteille. Eri laitteilta saapuvat viestit lähetetään verkkopalveluun sopivassa viestimuodossa. Ideointivaiheen lopussa valitaan kaikista eri vaihtoehdoista sopivin opinnäytetyötä varten. Ideointivaiheen jälkeen Proof of Concept -mallin ensimmäinen perustavoite on osittain ratkaistu (kuvio 1). Ideointivaiheeseen oli varattu aikaa vuoden 2017 joulukuuhun saakka.

5.2 Testausvaihe

Testausvaihe aloitetaan ideointivaiheen jälkeen. Tarkoituksena on saada käyttöön erilaisia testikomponentteja, joiden avulla voidaan todentaa mahdollisen ratkaisun toimivuutta. Testikomponenteille luodaan toimiva simulaatioympäristö, jossa simuloidaan ratkaisun toimivuutta ja tutkitaan mahdollisia teknisiä haasteita. Testausvaiheessa on tavoitteena ohjata erilaisia laitteita niiden omilla protokollilla ja tunnistaa tekniset haasteet. Ideana olisi kartoittaa valitun ratkaisun ominaisuudet niin hyvin, että voidaan jo todeta, että konseptin todennus tulee onnistumaan (kuvio 1). Tämän takia testausvaiheessa täytyy alustavasti jo yrittää hahmotella eri asioiden toimivuutta siten, että toteutusvaiheeseen voidaan edetä itsevarmana.

5.3 Toteutusvaihe

Toteutusvaihe aloitetaan, jos testausvaiheen tavoite saavutetaan. Toteutusvaiheessa todennetaan laajennuksen toimivuus rakentamalla simulaatio-ohjelma käytännön sovelluksesta (kuvio 1). Luodaan ohjelman kulku, joka vastaa oikeaa tilannetta. Toteutusvaiheen tavoitteena on mallintaa oikeaa sovellusta testiympäristössä ja todeta testiympäristöön tehdyn sovelluksen avulla laajennuksen toimivuus. Tavoitteena on myös kartoittaa mahdollisia tulevaisuuden mahdollisuuksia ja haasteita. Toteutusvaiheen alussa voidaan määrittää aika mahdollisen pilottihankkeen toteuttamiselle.



Kuvio 1. Opinnäytetyön toiminnallisuuden osuuden kolme vaihetta

6 IDEOINTIVAIHE

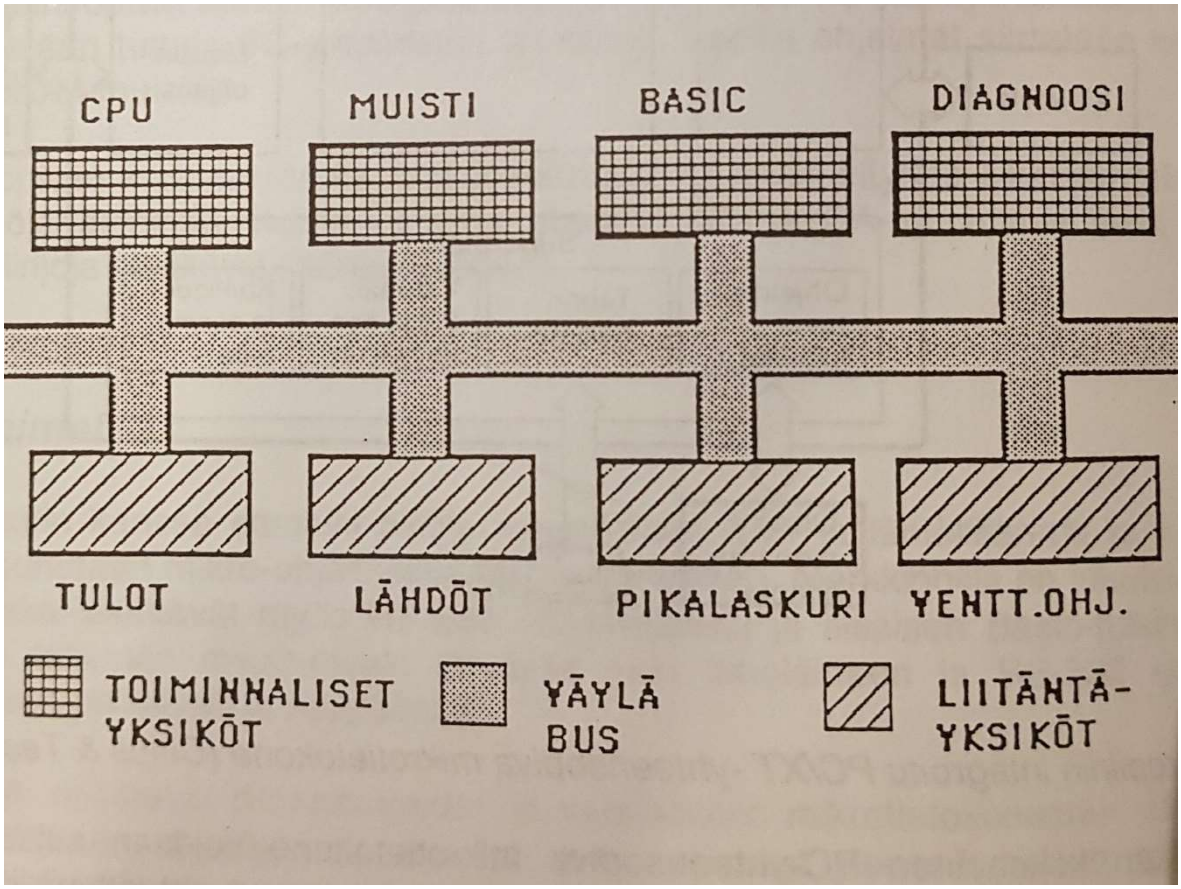
6.1 Ohjelmoitavan logiikan hyödyntäminen laajennuksessa

Ohjelmoitavan logiikan hyödyntämistä opinnäytetyössä pohdittiin ensimmäisenä, koska niihin voidaan liittää monipuolisesti erilaisia laitteita. Uusimmissa logiikoissa on myös mahdollista käyttää erilaisia protokollia, joiden avulla voidaan liittyä ylempiin järjestelmiin. Ohjelmoitavan logiikan avulla pitäisi olla mahdollista liittää laitteita verkkopalvelun käyttöön. Tässä luvussa tutkitaan Siemensin ohjelmoitavien logiikoiden tarjontaa.

6.1.1 Ohjelmoitava logiikka

Ohjelmoitava logiikka eli PLC (Programmable Logic Controller) on teollisuudessa käytettävä ohjauslaite. Logiikkoja käytetään hyväksi esimerkiksi kappaletavaratuotannon automatisoinnissa. Logiikkaan liittyy usein tulo- ja lähtöportteja. Tuloportteihin kytketään ohjaavia komponentteja, esimerkiksi antureita tai painonappeja, ja lähtöportteihin kytketään ohjattavia laitteita. Ohjelmoitavan logiikan sisällä on ohjelmoitava muisti, joka sisältää esimerkiksi funktioita, ajastimia ja laskureita, joiden avulla ohjataan konetta tai prosessia. (Airila 1999, 40-42.)

Toiminnalliset yksiköt (kuva 2) käsittelevät tietoa, joka liikkuu väylällä. CPU eli keskusyksikkö on mikroprosessori, joka usein jaetaan bitti- ja sanaprosessoriksi. Bittiprosessori käsittelee tehokkaasti yhden bitin tietoja kuten tulojen ja lähtöjen tilatietoja. Yhden bitin tiedolla tarkoitetaan esimerkiksi anturin signaalijohdossa kulkevaa jännitettä, joka määrää bitin tilaan yksi ylittäessään tietyn rajan. Jännitteetön tila on nolla. Sanaprosessori käsittelee sanoja, jotka ovat 16 bitin pituisia. Sanankäsittelyyn liittyy käskyjä, joiden osoiteosassa on yksi tai useampi sana. Tällaiset toiminnot ovat esimerkiksi laskutoimitukset, vertailut, ajastintoiminnot ja laskuritoiminnot. CPU voi lukea ohjelmamuistissa olevia käskyjä suoraan tai käyttämällä aliprosessoria. Ohjelmoitavan logiikan muita toiminnallisia yksiköitä ovat muun muassa BASIC-yksikkö, joka on tarkoitettu tietokonetoimintojen toteutukseen sekä diagnoosiyksikkö ohjattavan toiminnon diagnoosiin. (Airila 1999, 40-42.)



Kuva 2. Logiikan väylän yksiköitä (Airila 1999, 40)

Ohjelmoitavan logiikan liityntäyksiköt välittävät tietoa ohjattavan toiminnan ja väylän välillä. Esimerkiksi tulo- ja lähtöyksiköt on tarkoitettu kaksitilaisten toimilaitteiden ohjaukseen tai lukemiseen. Tuloyksikköön voidaan liittää kaksitilainen anturi. Logiikoihin voidaan kytkeä myös analogiatulojen ja lähtöjen ohjaukseen tarkoitettuja kortteja. Logiikan pikalaskuriyksikkö soveltuu pulssianturin lukemiseen. Pulssianturi saattaa syöttää logiikalle jopa useamman tuhat pulssia moottorin kierrosta kohti. Pulssilla tarkoitetaan tässä tapauksessa jännityksellisten ja jännitteettömien tilojen nopeaa vaihtelua. Pikalaskuriyksiköllä ohitetaan logiikan hidas ohjelmakierto, joka ei olisi ideaalinen pulssianturin tyyppiselle toimilaitteelle. Venttiilinohjausyksiköt ovat tarkoitettu ohjaaman sellaisia toimilaitteita, joilla on suurempi lähtöteho. Normaalilähtöjä käytetään ohjaamaan pienempi tehoisia toimilaitteita. (Airila 1999, 40-42.)

Ohjelmoitavan logiikan ohjelma muodostuu käskyryhmistä, joista jokainen vastaa itsenäistä toimintaa. Aloituskäsky aloittaa ohjelmakierron ja se erottaa ryhmän edellisestä ryhmästä. Aloituskäskyssä on ehto ja seuraavissa käskyissä on muut tarvittavat ehdot. (Airila 1999, 40-42.) Ohjelmoitavan logiikan ohjelmassa on ohjelmakierto, jolloin se kiertää koko ohjelman aina kerralla. Uusimmissa logiikoissa on myös erilaisia modulaarisia yksiköitä, joita käyttäjä voi itse ohjelmoida ohjelman sisälle, mutta perusidea on pysynyt samana.

Logiikoissa on myös erilaisia ohjelmointitapoja ja -kieliä. Logiikan ohjelmointikieliä ovat esimerkiksi IEC 61131-3 -standardin mukaiset kielet: FBD (Function Block Diagram), LD (Ladder Diagram), ST (Structure Text), IL (Instruction List) ja SFC (Sequential Function Chart). (Real Time Automation 2018).

6.1.2 OPC UA -standardi

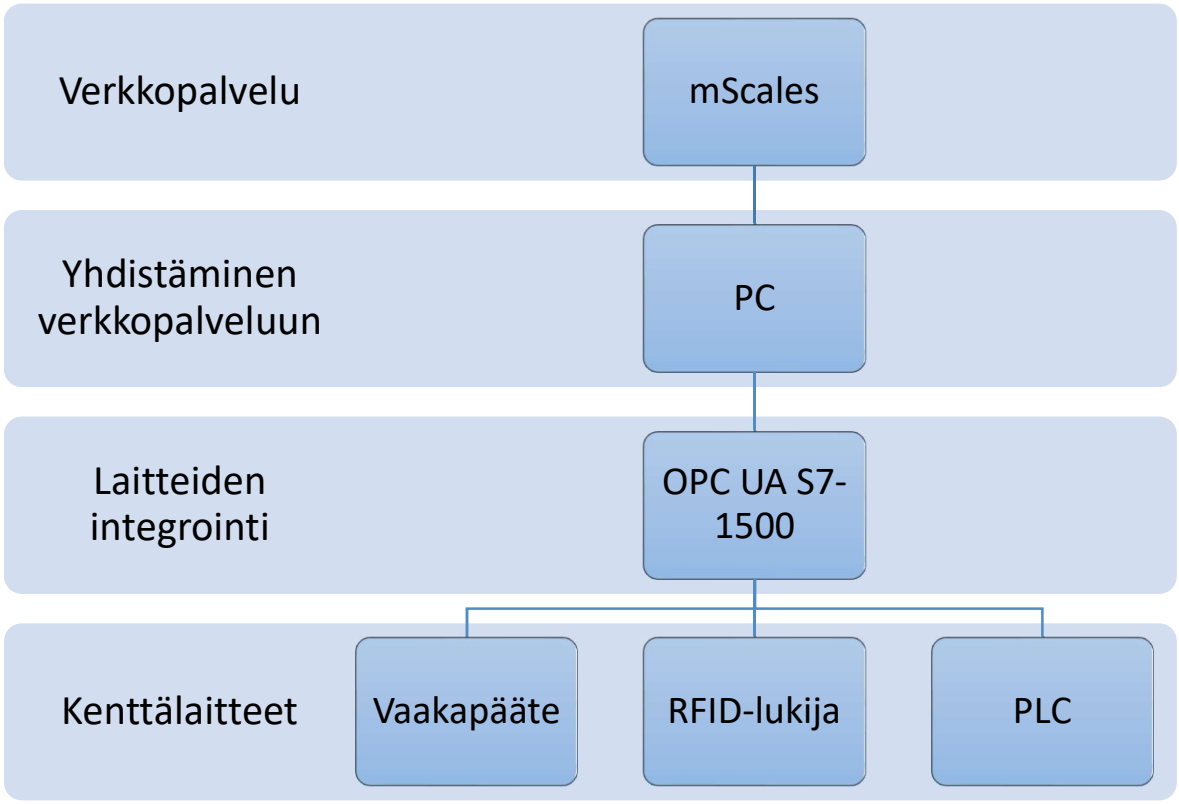
OPC on standardi luotettavalle ja turvalliselle tiedonsiirrolle teollisuusautomaation alalla ja muilla teollisuudenaloilla. Se takaa saumattoman tiedonsiirron eri hankkijoiden laitteiden välillä. OPC-säätiö on vastuussa OPC-standardin kehittämisestä ja ylläpidosta. OPC-standardi on sarja teknisiä vaatimuksia, joiden luomisessa ovat olleet mukana teollisuuden hankkijat, loppukäyttäjät ja ohjelmistokehittäjät. Standardin vaatimukset määrittävät palvelimien ja asiakkaiden sekä kahden palvelimen välisen käyttöliittymän. Se tarjoaa pääsyn ajantasaiseen tietoon, hälytyksien ja tapahtumien tarkkailuun, historiallisiin tapahtumiin ja muihin sovelluksiin. (OPC Foundation 2018.)

OPC-standardi julkaistiin vuonna 1996 ja sen tarkoituksena oli tiivistää ohjelmoitavien logiikoiden ominaisprotokollia yhdeksi standardisoiduksi käyttöliittymäksi. Käyttöliittymän tarkoituksena oli sallia HMI- (Human-Machine Interface) ja valvomo-ohjelmistojen (SCADA) vuorovaikutus jonkin yhdistävän laitteen/ohjelmiston välillä, joka muuttaisi tavalliset OPC-komennot laitekohtaisiksi komennoiksi ja päinvastoin. (OPC Foundation 2018.) OPC-standardin kehitys alkoi siitä, kun teollisuudessa alkoi yleistymään erilaiset ohjelmoitavat logiikat. Logiikoiden, paneelien ja teollisuus-PC:n välille tehtiin usein erilaisia valvomo-ohjelmia ja käyttöliittymiä. OPC-standardi yksinkertaisti ja vähensi työn määrää siinä.

Turvallisuuteen ja tiedon muotoiluun tuli haasteita, kun tuotantojärjestelmien käyttöön alkoi ilmestyä palvelukeskeisiä arkkitehtuureja (OPC Foundation 2018). Palvelukeskeisessä arkkitehtuurissa suunnitellaan järjestelmien eri toiminnot ja prosessit toimimaan itsenäisinä, avoimina ja joustavina palveluina. Näitä erilaisia palveluita tulisi päästä käyttämään standardisoitujen käyttöliittymien avulla. (Barry & Associates Inc 2018.) OPC-säätiö kehitti standardinsa vastaamaan palvelukeskeisen arkkitehtuurin vaatimuksia ja se samalla tuotti ominaisuuksiltaan laajan arkkitehtuurin, joka on laajennettava ja skaalautuva (OPC Foundation 2018).

6.1.3 SIMATIC OPC UA S7-1500

Siemens on lisännyt OPC UA -standardin uusimpaan logiikkaperheeseen. S7-1500 logiikkaperheessä OPC UA mahdollistaa logiikan ulkopuolisille laitteille standardisoidun tavan ottaa yhteyttä S7-1500 logiikkaan. Siemens tarjoaa tällä hetkellä vain OPC UA palvelimen ohjelmoitavan logiikan ominaisuutena. Tämä tarkoittaa, että logiikka ei toimi kommunikaatiossa ensimmäisenä viestijänä. OPC UA -palvelin valitaan aktiiviseksi TIA Portal -ohjelmointityökalusta. (Siemens AG 2018a.)



Kuvio 2. OPC UA S7-1500 logiikan rooli laajennuksessa

Ideointivaiheen ensimmäinen vaihtoehto on OPC UA :n hyödyntäminen mScales-laajennuksessa. OPC UA -palvelin otettaisiin käyttöön S7-1500 logiikassa, jolloin logiikan ulkopuolinen laite voisi luoda yhteyden logiikan kanssa. Tämä tarkoittaisi sitä, että ohjelmoitava logiikka toimisi liitännässä päälaitteena (kuvio 2), johon yhdistettäisiin kaikki muut laitteet. Siemens tarjoaa tällä hetkellä vain OPC UA -palvelimen mahdollisuuden logiikkaan, jolloin logiikka ei itse ottaisi yhteyttä verkkopalveluun vaan päinvastoin. Vuoden 2018 aikana logiikkaan on tulossa lisäominaisuus ottaa käyttöön OPC UA -standardi myös asiakkaana. Opinnäytetyössä on tavoitteena löytää laite, jonka avulla voidaan ottaa yhteys verkkopalveluun, jolloin verkkopalveluun ei tarvitse määritellä useita eri laitteita erikseen. Logiikan käyttäminen yhteyden muodostamiseen ei myöskään ole ideaalisin tilanne,

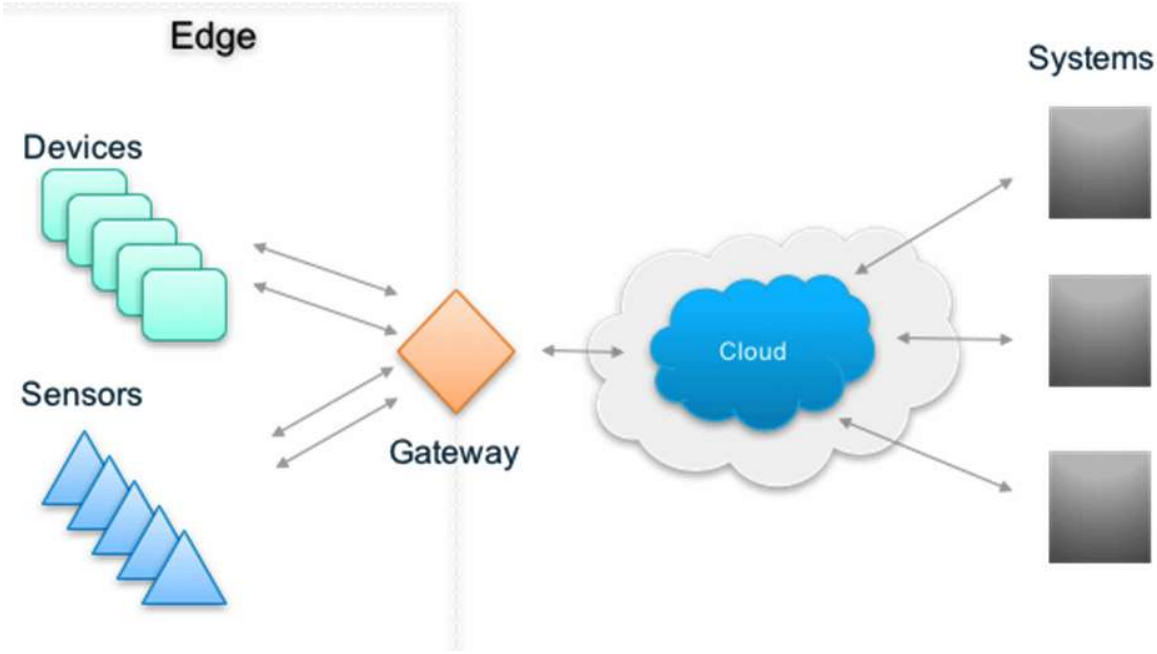
koska logiikka on pääsääntöisesti suunniteltu ohjauslaitteeksi. Tällaisessa ratkaisussa joudutaan myös käyttämään tietokonetta verkkopalvelun ja logiikan yhteydessä, mikä ei varsinaisesti tue alkuperäistä ideaa kevyemmästä laitekokoonpanosta.

6.2 Yhdyskäytävän hyödyntäminen laajennuksessa

Opinnäytetyössä tutkittiin yhdyskäytäviä toisena laiteryhmänä. Siemens oli kehittänyt IOT2000-sarjan yhdyskäytäviä, joiden ominaisuudet sopivat opinnäytetyön laitevaatimuksiin. Verkkopalvelun ja fyysisten laitteiden liittämiseen yhdyskäytävät vaikuttivat joustavammilta ja sopivimmilta ohjelmoitavaan logiikkaan nähden.

6.2.1 Yhdyskäytävä

Yhdyskäytävällä yhdistetään kaksi verkkoa toisiinsa eri protokollien avulla. Sillä yhdistetään toisiinsa sellaisia verkkoja, jotka eivät ole keskenään samankaltaiset. Yleinen yhdyskäytävä sovellutus on reititin, joka liittää yksityisen verkon internetiin. Yhdyskäytäviä on monenlaisia, joilla on erilaiset toimintaperiaatteet ja tehtävät. (TechTarget 2018.) Tässä opinnäytetyössä keskitytään IoT-yhdyskäytäviin, joilla voidaan esimerkiksi käsitellä ja lähettää eteenpäin anturilta tulevaa tietoa (kuva 3). Etuna tällaisessa laitteessa on se, että erilaisia protokollia käyttävien laitteiden tiedot voidaan käsitellä joustavasti laitteessa.



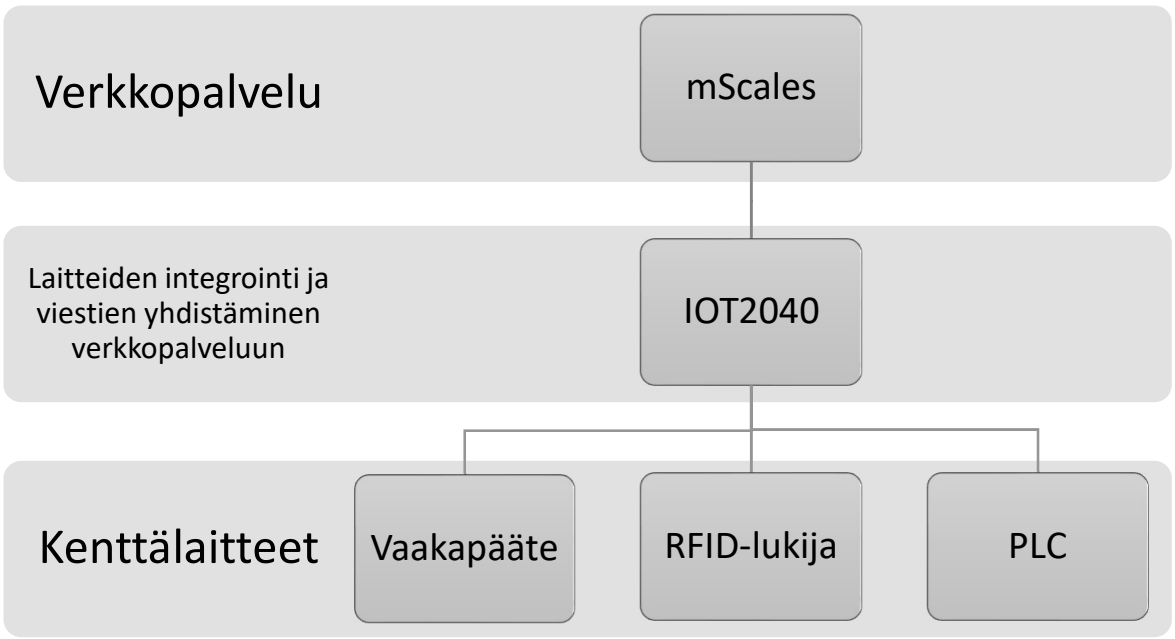
Kuva 3. IoT-yhdyskäytävän sijoitus järjestelmässä (TechTarget 2018)

6.2.2 SIMATIC IOT2000

SIMATIC IOT2000 on luotettava avoin alusta tiedon keräämiseen, muuntamiseen ja lähettämiseen teollisessa ympäristössä. Tuoteperheen ideaalisin tarkoitus on toimia yhdyskäytävänä tuotannon ja pilvipalvelun tai yrityksen tietoliikennetason välillä (kuvio 3). Tuoteperheen avoimuus tarkoittaa sitä, että se tukee monia erilaisia protokollia sekä korkean tason ohjelmointikieliä. (Siemens AG 2018b.)

SIMATIC IOT2000 tuoteperheeseen kuuluu kaksi laitetta: IOT2020 ja IOT2040. IOT2040 on tuoteperheen uusin malli, joten se olisi mahdollinen laiteratkaisu mScales-laajennukseen. IOT2040 -laitteessa on monta erilaista liityntämahdollisuutta:

- kaksi ethernet-porttia
- kaksi RS232/485-liityntää
- USB-portti
- miniUSB-portti.



Kuvio 3. Gateway -laitteen rooli laajennuksessa

Lisäksi laitetta voidaan laajentaa Arduino- ja miniPCIe-korteilla. Laitetta myydään myös kahtena erilaisena mallina. Ensimmäinen malli toimii Siemens-yrityksen oman pilvipalvelun, MindSpheren kautta. Tätä mallia ei tarvitse itse ohjelmoida, mutta silloin täytyy ottaa käyttöön MindSphere-pilvipalvelu ja luoda sinne omat käyttöliittymät. Toinen malli on täysin tyhjä alusta, jolloin käyttäjän täytyy itse ottaa laite käyttöön ja ohjelmoida laitteeseen

tarvittavia ominaisuuksia. (Siemens AG 2018b.) Käyttäjä saa Siemensin verkkosivuilta ladattua vedostiedoston laitteeseen, johon on esiasennettu erilaisia ohjelmia, kuten esimerkiksi npm- ja Node-RED-työkalu.

6.3 Ideointivaiheen lopputulos

Lahti Precision Oy oli ennen opinnäytetyön alkamista tehnyt taustatutkimusta laajennuksen laitevaihtoehdoista. Opinnäytetyön alussa määriteltiin, että tutkitaan laiteratkaisua Siemens- ja Beckhoff-yritysten tarjonnasta. Opinnäytetyön alussa oli jo muutama lupaava laite, joiden ominaisuuksista täytyi ottaa selvää. Työssä tutkittiin laajasti Siemensin S7-1500 -tuoteperheen ohjelmoitavaa logiikkaa, jossa oli OPC UA -liityntämahdollisuus. Ongelmana oli se, että logiikasta ei löytynyt vielä mahdollisuutta nimetä sitä OPC UA -asiakkaaksi, vaan ainoana mahdollisuutena oli OPC UA -palvelimen käyttäminen. Lisäksi Siemensin ohjelmoitavan logiikan käyttäminen voisi rajoittaa tai sulkea erilaisia vaihtoehtoja pois laajennuksen teosta. Beckhoffin tarjontaa ei opinnäytetyössä tutkittu, koska päätimme hankkia Siemensin IOT2040-laitteen testikäyttöön. Olimme joulukuun aikana tutkinut Siemensin yhdyskäytävää, jonka ominaisuudet vaikuttivat sopivilta laajennuksen tekoon. Se on halpa ja soveltuisi kenttälaitteiden integrointiin hyvin. IOT2040 on luotettava avoin alusta, jolloin laajennuksen vaihtoehdot ja mahdollisuudet eivät välttämättä rajoitu laitteen ominaisuuksiin.

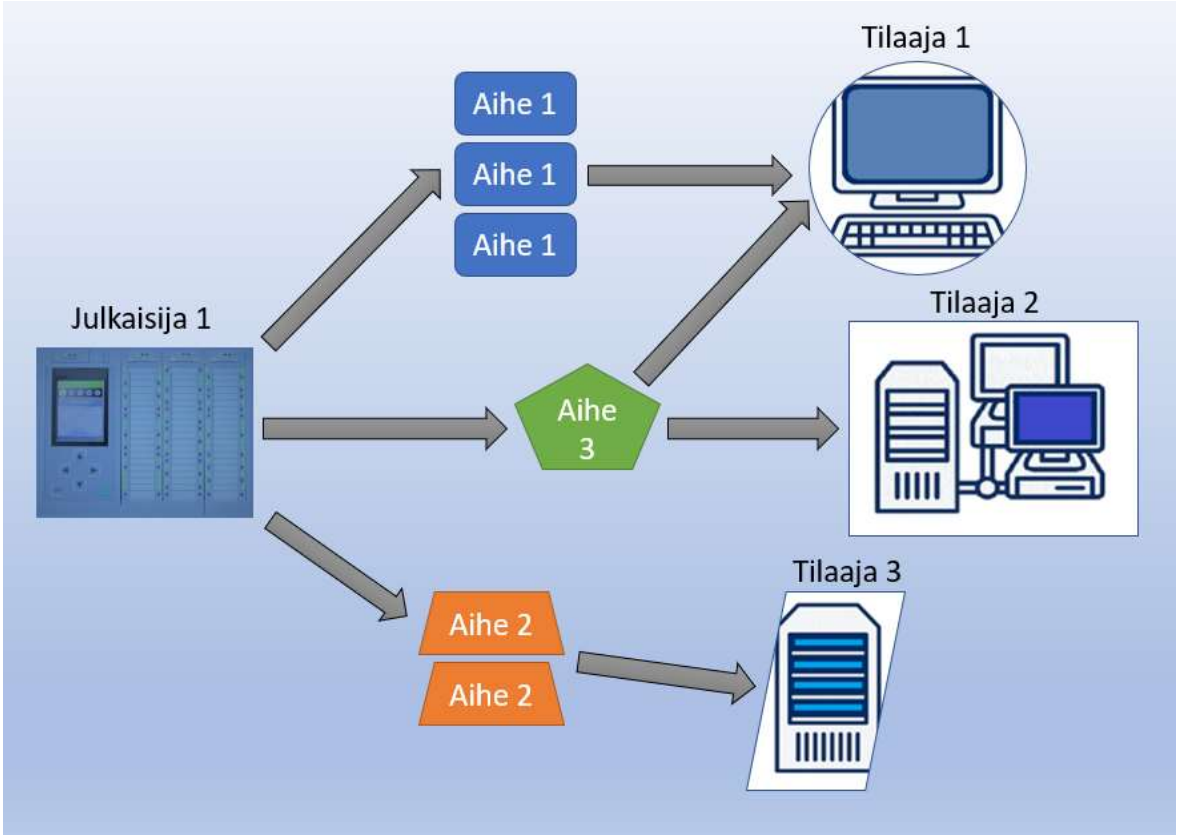
7 STANDARDIT JA YHTEYSKÄYTÄNNÖT

7.1 Tiedonsiirron yhteyskäytännöt

Tässä luvussa käsitellään opinnäytetyön tiedonsiirron yhteyskäytäntöjä. Tiedonsiirron protokollista käsitellään lähinnä tiedonsiirron rakenne ja protokollan roolit. Opinnäytetyön kannalta luvussa käsitellään kaksi tärkeintä tiedonsiirron protokollaa. Tarkoituksena on esitellä molemmat protokollat, joiden toimintaperiaatteiden avulla on opinnäytetyön sovelluksen rakentamista suunniteltu.

7.1.1 Julkaisija-tilaaja

Tiedonsiirrossa yhteyskäytännöt ryhmitellään niiden tiedonsiirron periaatteen mukaan. Julkaisija-tilaaja-malliin perustuvat protokollat ovat yleistymässä. Tietoa tuottavat elimet järjestelmässä toimivat tässä mallissa julkaisijan roolissa, mikä tarkoittaa sitä, että tietolähteet ”julkaisevat” määrättyyn aiheeseen liittyvää tietoa. Lähetetty tieto kulkee kohti tietoliikenteen keskitettyä pistettä, josta se ohjataan tiedon tilaajalle. (kuva 4) Tässä mallissa tilaajan roolissa toimii esimerkiksi tietokanta, liiketoiminta- tai teollinen sovellus tai jokin muu päätepiste. Tilaajat tilaavat tietystä aihealueesta julkaistavaa tietoa. Järjestelmän sama piste voi toimia niin julkaisijana kuin tilaajana. (Collin & Saarelainen 2016, 184.) Tässä protokollassa julkaisijat ovat sellaisia laitteita, jotka tuottavat järjestelmään tietoa. Tällaisessa mallissa esimerkiksi anturit julkaisevat havaittua tietoa ja lähettävät tiedon johonkin järjestelmän pisteeseen, josta se osataan ohjata tilaajalle. S7-1500 -tuoteperheen OPC UA toimii juuri julkaisija-tilaaja-mallilla. Logiikan rooli on olla julkaisija, jolla on tilaajia. Se julkaisee määritettyä tietoa tilaajille.

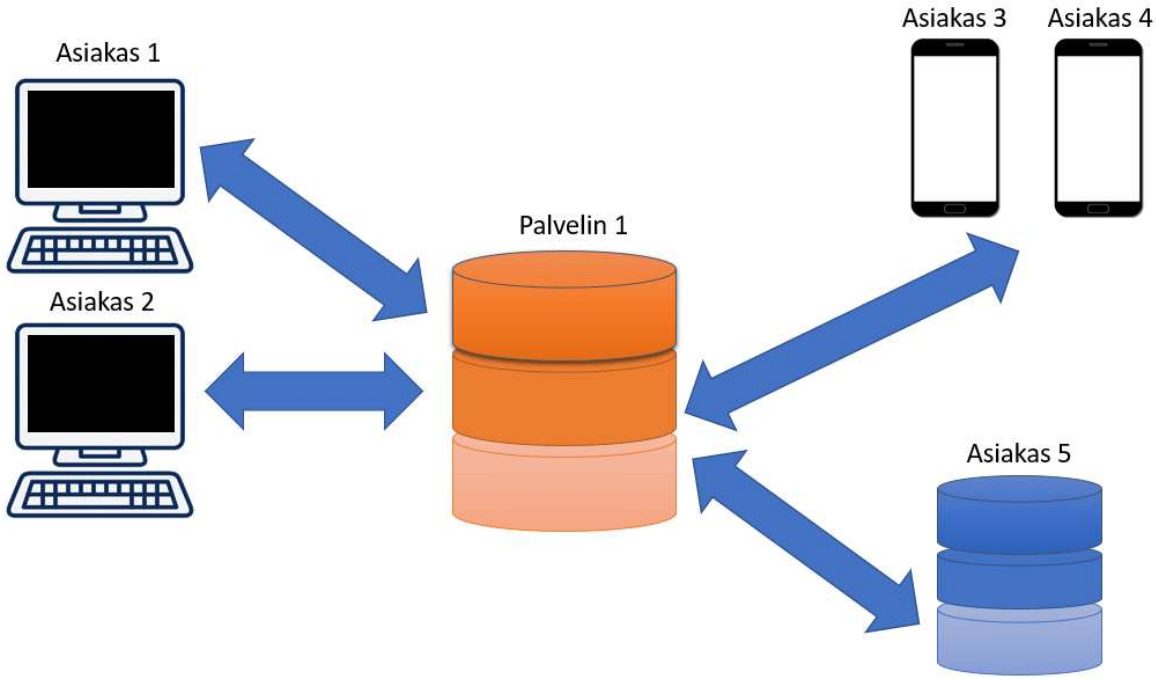


Kuva 4. Julkaisija-tilaaja -mallin rakenne

7.1.2 Asiakas-palvelin

Toinen tiedonsiirron yhteyskäytännöistä perustuu asiakas-palvelin-tyyppisesti kyselyihin. Palvelin tuottaa lähetettävää tietoa ainoastaan silloin, kun asiakas sitä kyselee. Tässä mallissa kuormitetaan vähemmän verkkoa, koska mitään tietoa ei lähetetä vain varmuuden vuoksi vaan aina, kun joku sitä pyytää. Asiakkaan ja palvelimen roolit voivat vaihdella paljonkin. Teollisessa internetissä yleensä palvelimen roolissa on älykäs päätelaite antureineen. Päätelaite kerää antureilta tietoa ja mahdollisesti myös käsittelee sen, minkä jälkeen se lähettää esimerkiksi puskuristaan tiedot asiakkaaksi kytketylle järjestelmälle. (Collin & Saarelainen 2016, 185.) Asiakas-palvelin-mallissa on myös mahdollisuus, että palvelimena toimii esimerkiksi verkkopalvelu. Laajennuksen teossa sovittiin yhteisesti siitä, että verkkopalvelu toimii palvelimena, jolla on asiakkaita. Asiakkaat ovat verkkopalveluun liitettäviä järjestelmiä ja näiden järjestelmien vastuulle jää yhteydenpidon aloittaminen verkkopalveluun. Tällöin verkkopalveluun ei tarvitse erikseen määritellä jokaista laitetta IP-osoitteineen, vaan laite ilmoittaa verkkopalvelulle olemassaolostaan.

Kuvassa 5 on esitelty asiakas-palvelin-mallinen tiedonsiirron rakenne. Kuvan keskipisteessä on mallinnettu järjestelmän palvelin, johon kaikki muut järjestelmän osat ovat liittyneet. Tällaisessa mallissa palvelin on järjestelmän lopullinen piste. Opinnäytetyössä koko järjestelmä toimii tällaisella tavalla, jolloin luonnollisesti PoC-mallin suunnittelussa vaatimus huomioidaan.



Kuva 5. Asiakas-palvelin -mallin rakenne

7.2 Laajennuksen yhteyskäytännöt

Tietoliikenteessä ja yleisesti tiedon siirtämisessä antureilta päätelaitteille ja tietokantoihin yhteyskäytännöt ja standardit ovat erittäin tärkeitä. Niiden avulla tieto välittyy järjestelmien ja ohjelmien välillä. Standardit ovat tarkoitettu laitteiden helppokäyttöiseen liittämiseen eri järjestelmiin. (Collin & Saarelainen 2016, 181.) Käytännössä standardit on suunniteltu yhteiseen käyttöön, jolloin eri laitevalmistajat voivat yhdistää laitteensa toisiin järjestelmiin. Protokollat ja viestintästandardit ovat eri laitteiden ja järjestelmien tapoja olla yhteydessä keskenään. Tämä luku käsittelee erilaisia yhteyskäytäntöjä. Lukuun on lisätty sellaisia protokollia, joita tullaan käyttämään testaus- ja todennusvaiheessa. Luku sisältää pintapuolista tietoa eri protokollien toimintaperiaatteista ja viestirakenteista.

7.2.1 S7comm

S7comm on Siemensin oma protokolla, joka toimii Siemens S7-300/400 tuoteperheen logiikoiden välillä. Protokollaa käytetään logiikkaohjelmoinnissa tiedon vaihdossa eri

logiikoiden välillä sekä tiedon välityksessä logiikan ja SCADA-järjestelmän välillä. S7comm -protokollan tieto välitetään tietosisältönä, joka koostuu COTP-tietopaketeista. Yhteyskäytännön ensimmäinen tavu on 0x32, joka toimii protokollan tunnisteena. (WIRESHARK 2016.)

S7comm yhteyden kokonaisuus on esitetty OSI-mallin avulla (kuva 6). Yhteyden luominen S7-logiikkaan vaatii kolme vaihetta. Ensimmäisessä vaiheessa yhdistetään logiikka TCP-protokollan avulla porttiin 102. Tässä vaiheessa käytetään logiikan IP-osoitetta yhdistämiseen. Toisessa vaiheessa luodaan yhteys OSI-kerrokseen. Tässä vaiheessa yhdistetään kuljetuskerros ja yhteysjaksokerros kahden tavun pituisella osoitteella. Ensimmäinen tavu osoitteessa kertoo kommunikaation lajin. Toinen tavu kertoo logiikan telinenumeron ja korttipaikan numeron. Logiikkaa ohjelmoitaessa määritellään telinenumero ja korttipaikan numero. Korttipaikan numero ohjelmoidaan bittien 0-3 avulla ja telinenumero bittien 4-7 avulla. Kolmannessa vaiheessa luodaan yhteys S7comm kerroksiin, joita ovat esitystapakerros ja sovelluskerros (kuva 6). (WIRESHARK 2016.)

NUMERO	kerros	TARKOITUS
7	sovellus	itse sovellukset
6	esitystapa	yhteinen datan esitysmuoto (esim. miten esitetään rivinvaihto)
5	yhteysjakso	yhteyksien hallinta ja synkronointi
4	kuljetus	sovellusten osoittaminen ja virheiden korjaus
3	verkko	verkkojen yhdistäminen ja siirto verkkojen välillä
2	siirto	bittien siirtäminen yhden verkon sisällä
1	fyysinen	fyysiset liitännät

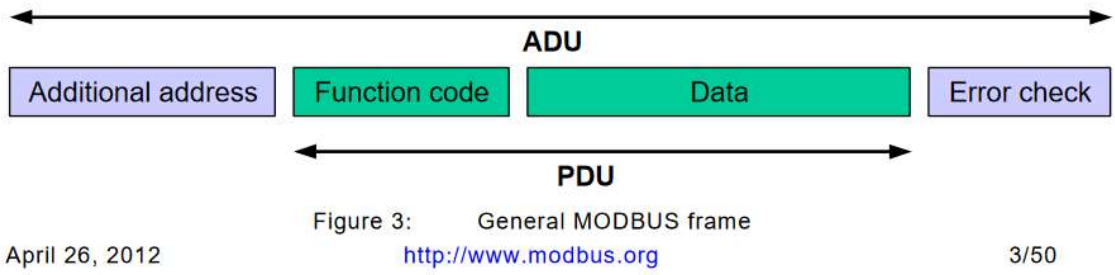
Kuva 6. OSI-mallin kerrokset ja niiden kuvaukset (Internetix 2018)

Valitsemassani lähteessä, joka kertoo S7comm-protokollasta, mainitaan ainoastaan S7-300/400 tuoteperheen logiikoiden kommunikointi kyseisen protokollan avulla. Tutustuin aiheeseen enemmän ja löysin Snap7-kommunikaatioalustan. Snap7 on avoimen lähdekoodin alusta, jolla kommunikoidaan S7-tuoteperheen logiikoiden kanssa S7comm-protokollan avulla. Snap7 verkkosivuilla mainitaan yhteyden muodostus S7comm-protokollalla myös S7-1200/1500- ja LOGO-logiikoihin. S7comm-protokollaa tullaan käyttämään yhteyden muodostamiseen gateway-laitteen ja Siemens-logiikan välillä. Logiikan erilaisia portteja voidaan lukea ja kirjoittaa gateway-laitteella.

7.2.2 Modbus

Modbus on ollut vuosikymmeniä teollisuusautomaation yleinen standardi. Modbus-standardi on avoin, ilmainen ja helposti hyödynnettävissä teollisessa verkossa tai logiikoissa. Modbus oli ensin vain sarjaliikennemuotoisen viestinnän yhteyskäytäntö. Modbus/TCP on protokollan ethernet-versio. Protokollaa käytetään esimerkiksi yhdistämään lämpöä tai kosteutta mittaavat anturit valvovaan tietokoneeseen. Modbus:n suurin hyöty on se, että viestintä on tilatonta. Tilattomuus tekee protokollasta hyvin häiriösietoisen. Huono puoli protokollassa on se, ettei siinä ole normaalisti minkäänlaista tietoturvaa. (Collin & Saarelainen 2016, 186.)

Modbus-protokollaa käytetään isäntä-orja-mallin sovelluksissa. Sen käyttötarkoituksiin kuuluvat esimerkiksi laitteiden valvonta ja ohjelmointi, yhteydenpito älykkäiden laitteiden ja antureiden välillä sekä kenttälaitteiden valvonta PC:n tai HMI-paneelien avulla. (Modbus Organization 2018b.) Sarjaliikennemuotoinen modbus-protokolla sijoittuu OSI-mallissa fyysiseen-, siirto- ja sovelluskerrokseen. Ethernet-muotoinen modbus-versio sijoittuu fyysisen-, siirto- ja sovelluskerroksen lisäksi yhteysjakso- ja esitystapakerrokseen (kuva 6). Yhteys eri väylissä tai verkoissa olevien laitteiden välillä tarjotaan asiakas-palvelin-mallin mukaisesti. Modbus on pyynnön ja vastauksen protokolla ja se tarjoaa palveluja tietyn toimintokoodin mukaisesti. (Modbus Organization 2018c.)



Kuva 7. Protokollan tietoyksikkö (PDU) ja sovelluksen tietoyksikkö (ADU) (Modbus Organization 2018a)

Modbus-protokolla määrittää yksinkertaisen protokollan tietoyksikön, joka on itsenäinen muista yhteyden tasoista. Modbus-protokollan laajentaminen eri väyliin tai verkkoihin voi lisätä uusia kenttiä sovelluksen tietoyksikköön. (Modbus Organization 2018a.) Sovelluksen tietoyksikkö (kuva 7) nimensä mukaisesti sisältää kokonaisen modbus-sovelluksen tietokentät. Sovelluksen tietoyksikkö sisältää protokollan tietoyksikön kentät, jotka ovat jokaisessa samaa protokollaa käyttävässä sovelluksessa samanlaiset. Sovelluksen tietoyksikön ajatuksena on tarkentaa sovelluskohdetta. Vuorovaikutuksen käynnistänyt asiakas muodostaa sovelluksen tietoyksikön. Toimintokoodi osoittaa palvelimelle suoritettavan

toiminnon. Modbus-sovelluksen protokolla perustelee asiakkaan käynnistämän pyynnön muodon. Protokollan tietoyksikön toimintokoodikenttä on ohjelmoitu yhteen tavuun. Hyväksytyt arvot toimintokoodiin ovat numerot 1-255, vaikka numerot 128-255 ovat varattu poikkeustapauksiin. Toimintokoodi osoittaa suoritettavan toiminnon asiakkaan lähettämässä viestissä palvelinlaitteelle. Tietokenttä sisältää sovelluskohtaista lisätietoa, joka yleensä liittyy toimintokoodiin. Asiakkaan lähettämässä viestissä palvelinlaitteelle, tietokenttä määrittää palvelinlaitteelle esimerkiksi erillisen osoitteen, rekisteriosoitteen, käsiteltävien asioiden määrän tai tietotavujen todellisen määrän kentässä. Tietokenttä voi tietyissä tapauksissa olla olematon, jolloin palvelin ei vaadi lisätietoja viestissä. Tällöin toimintokoodi ainoastaan määrittää suoritettavan toiminnon. (Modbus Organization 2018a.)

Virheettömässä modbus-yhteydessä asiakkaan lähettämän toimintokoodia ja tietokenttää vastaan palvelimen vastaus sisältää asiakkaan lähettämän toimintokoodin ja pyydetyn tietokentän. Virhetilanteessa vastauskenttä sisältää virhetilanteen toimintokoodin ja tietokentän. Virhetilanteen vastauksella palvelin ja asiakas voivat päätellä virheen laadun ja aloittaa tarvittavat toimenpiteet. (Modbus Organization 2018a.)

Funktiokoodi	Kuvaus
01	Coils luku
02	Discrete Inputs luku
03	Holding Registers luku
04	Input Registers luku
05	Single Coil kirjoitus
06	Single Register kirjoitus
15	Multiple Coil kirjoitus
16	Multiple Registers kirjoitus

Kuva 8. Modbus-yhteyskäytännön yleisimpiä toimintokoodeja (FläktWoods 2016)

Modbus-protokolla on suhteellisen monimuotoinen, mutta yksinkertaisuudessaan voidaan todeta, että yhteydenpito suoritetaan pääosin toimintokoodien varassa. Tarkastellessa yleisimpiä toimintokoodeja (kuva 8) voidaan todeta, että eri toimintokoodit jo kertovat tapahtumasta paljon. Oletetaan, että haluttaisiin esimerkiksi kirjoittaa yhteen rekisteriin jokin arvo. Yleisimpien toimintokoodien (kuva 8) perusteella toimintokoodiksi saadaan heksadesimaali lukuna 0x06. Luku määrittää palvelinlaitteelle toiminnon: kirjoitetaan yhteen rekisteriin. Toimintokoodin lisäksi tietokenttään sisällytettäisiin myös rekisterin osoite ja kirjoitettava arvo. Kirjoitetaan rekisteriin 0x100 arvo 0x01. Riippuen sovelluksesta, sovelluksen tietoyksikkö sisältäisi ainakin luvut 0x06, 0x100, 0x01. Tämän yksinkertaisen esimerkin johdosta voitaisiin olettaa, että palvelin kirjoittaisi 0x06 toimintokoodilla rekisteriin 0x100

luvun 0x01 ja vastaisi asiakkaalle tapahtuman jälkeen toimintokoodin ja tietokentän. Luettaessa rekistereistä arvoja, kuvan 8 perusteella käytetään toimintokoodia 0x03. Tällä toimintokoodilla voidaan lukea useita rekistereitä, jolloin luettaessa esimerkiksi kahdeksan rekisteriä alkaen rekisteristä 0x200, sisältäisi sovelluksen tietoyksikkö ainakin luvut 0x03, 0x200 ja 0x08. Lahti Precision Oy:n WA-805 -vaa'an ja gateway-laitteen yhteydenpitoon käytetään Modbus/TCP -protokollaa ja yleisimpiin yhteydenpitotoimintoihin tarvitaan toimintokoodeja 0x03 ja 0x06.

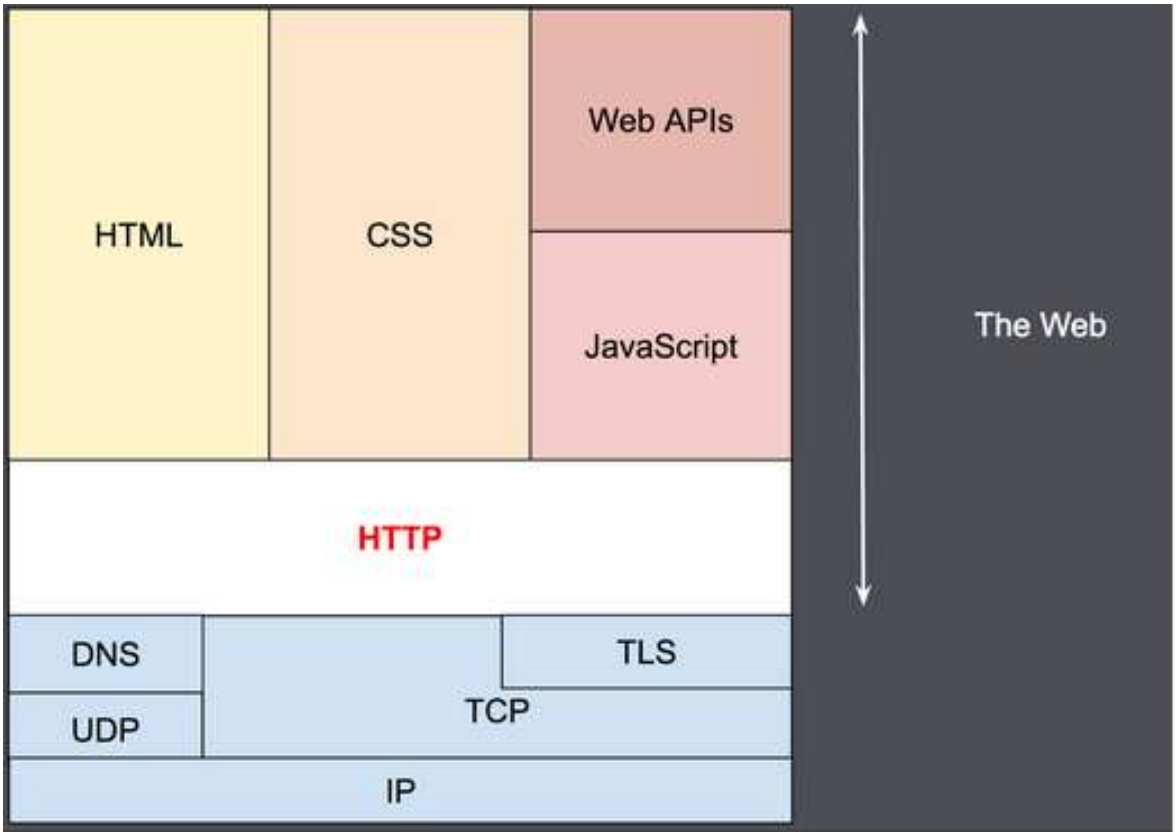
7.2.3 OSDP

OSDP-yhteyskäytäntö (Open Supervised Device Protocol) on turvallisuusalan yhdistyksen SIA:n kehittämä kulunvalvonnan viestintästandardi. Standardin avulla parannetaan kulunvalvonnan ja turvallisuustuotteiden yhteensopivuutta. Verrattuna alhaisen turvallisuuden omaaviin yhteyskäytäntöihin, OSDP tarjoaa korkeampaa turvallisuutta, monikäyttöisyyttä ja helppokäyttöisyyttä. OSDP-protokollassa korkeamman turvallisuuden luo muun muassa AES-128 salaus ja kytkennän valvonta. Monikäyttöisyyttä tuo muun muassa monimuotoiset toiminnot, kuten kuvien tai tekstien syöttäminen, virheiden valvonta sekä usean laitteen toiminta samassa väylässä. Helppokäyttöisyys ilmenee käyttäjälähtöisellä laitteen toiminnalla. Lisäksi standardin luvataan olevan yhteensopiva useiden eri valmistajien laitteilla. (Security Industry Association 2018.)

Käytännössä OSDP-protokollan toiminta perustuu viestipaketteihin, joita lähetetään päätelaitteelle. Päätelaite vastaa tietyllä viestipaketilla saapuneeseen viestiin. Viestipaketit koostuvat tavuista, joiden määrä ja laatu vaihtelee riippuen viestin lajista. Jokaisella viestipaketin tavulla on myös oma toiminto, jolloin yhdessä ne muodostavat tiettyjä toimintoja, kuten esimerkiksi kuvan näyttämistä tai laitteen lähdön päälle kytkemistä. Tietyt tavut kertovat esimerkiksi aloitusnumeron, laitteen osoitteen ja viestipaketin koon. Protokollan rakenne on hyvin monimuotoinen ja sen vuoksi se tarjoaa suuren määrän erilaisia toimintoja. OSDP-protokollaa käytetään opinnäytetyössä etälukijan ja päätelaitteen välisessä yhteydenpidossa.

7.2.4 HTTP

HTTP on yhteyskäytäntö, joka sallii erilaisten resurssien noutamisen. Se on minkä tahansa verkossa tehtävän tiedonsiirron perustus. HTTP on asiakas-palvelin-mallin protokolla, jossa yleensä pyyntöjä suorittaa Internet-selain. Asiakkaat ja palvelimet viestivät lähettämällä yksittäisiä viestejä. Asiakkaan lähettämiä viestejä kutsutaan pyynnöiksi ja palvelimen lähettämiä viestejä kutsutaan vastauksiksi. (Mozilla 2018a.)



Kuva 9. HTTP -protokollan sijoittuminen verkossa (Mozilla 2018b)

1990-luvun alussa suunniteltu HTTP on laajennettava protokolla, joka on kehittynyt ajan myötä. Se on OSI-mallin sovelluskerroksen yhteyskäytäntö, joka lähetetään TCP:n tai TLS-suojatun TCP:n välityksellä. Laajennettavuuden takia HTTP-protokollaa ei käytetä vain hypertekstin noutamiseen, vaan myös kuvien ja videoiden noutamiseen sekä sisällön lähettämiseen palvelimille (kuva 9). (Mozilla 2018a.) HTTP-protokollassa on määritelty erilaisia pyyntömetodeja, joista jokainen määrittää tietyn toimenpiteen, joka suoritetaan annettuun lähteeseen. Tällaisia metodeja ovat esimerkiksi GET, HEAD, POST, PUT ja DELETE -metodit. Opinnäytetyön kannalta tärkeimmät metodit ovat POST ja GET -metodit. GET -metodia pitäisi käyttää ainoastaan tiedon hakuun. POST toimittaa tietynlaisen lähteen toimitusosoitteeseen. (Mozilla 2018b.) HTTP-protokollaa tullaan käyttämään gateway-laitteen ja verkkopalvelun välillä välittämään esimerkiksi komentoja ja niiden vastauksia.

8 TESTAUSVAIHE

8.1 Alkutilanne

Tammikuussa 2018 aloitettiin gateway-laitteen testausvaihe. Standardit ja yhteyskäytännöt -luvussa käsittelin erilaisia tiedonsiirron standardeja ja protokollia, joita laajennuksessa on tarkoituksena testata. Tiedonsiirron protokollista asiakas-palvelin-malli liittyy gateway-laitteeseen. Testausvaiheen tavoitteena on todentaa IOT2040-laitteen ominaisuudet riittäviksi työn jatkamiseen ja tunnistaa teknisiä haasteita. Testauksen käytetään Node-RED-ohjelmointityökalua. Tutkitaan Node-RED-työkalun soveltuvuutta yhdistämään kenttälaitteita verkkopalveluun. Tarkoituksena on todentaa laitteiden yhdistäminen siten, että löydetään käytetystä työkalusta keinot kommunikoida kenttälaitteiden avulla. Testataan yhdyskäytävän ja yksittäisten kenttälaitteiden yhteyksien toimivuus.

8.2 Virtauspohjainen ohjelmointi (FBP)

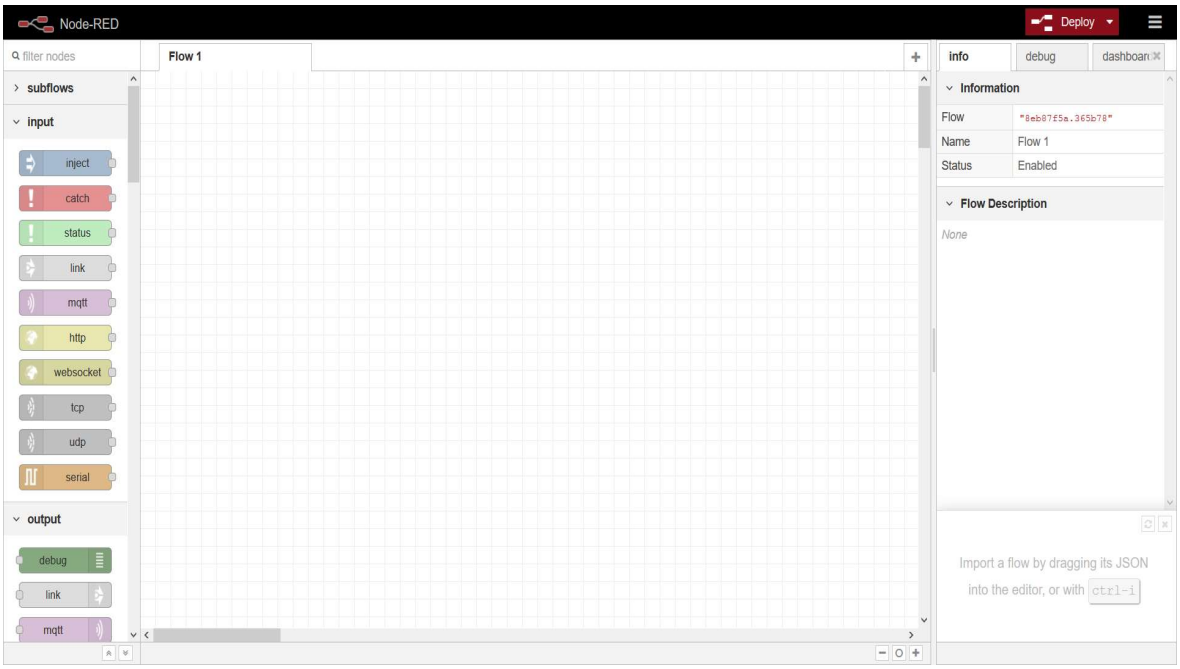
J. Paul Morrison kehitti virtauspohjaisen ohjelmoinnin (Flow-Based Programming) 1970-luvulla. Virtaustyyppisessä ohjelmoinnissa ohjelman käyttäytymistä kuvataan laatikoiden verkkona tai solmuina. Eri solmujen tehtävät on määritelty ennalta. Solmuihin syötetään tietoa, jonka se käsittelee ja lähettää eteenpäin. Solmujen välille syntyy verkko, jota pitkin tieto liikkuu. Virtauspohjainen ohjelmointi on erittäin visuaalinen ja helppopääsyinen ohjelmointitapa. Ohjelmassa olevaa ongelmaa on helpompi yrittää ratkaista seuraamalla solmuja, vaikka ei ymmärtäisi solmujen sisällöstä mitään. (JS Foundation 2017a.)

8.3 Node-RED

Node-RED on virtauspohjainen työkalu (flow-based programming tool). Sen alkuperäinen kehittäjä on IBM-yrityksen Emerging Technology -osaston Nick O'Leary ja Dave Conway-Jones. Ohjelma perustuu solmuihin ja niiden välillä käytävien tapahtumapohjaisten viestien toimintaan. Node-RED-työkalun kehitys aloitettiin vuonna 2013 sivuprojektin muodossa. Tällä hetkellä Node-RED on osa JS Foundation -säätiötä. (JS Foundation 2017a.) Node-RED-työkalu on Siemensin valmistamassa vedostiedostossa valmiina, mutta muutoin se voidaan asentaa esimerkiksi npm-työkalun avulla komentoriviltä.

8.3.1 Yleistä

Node-RED on ohjelmointityökalu, jolla liitetään yhteen laitteistoja, ohjelmointirajapintoja ja verkkopalveluja uusilla tavoilla. Työkalu tarjoaa selainpohjaisen muokkaimen (kuva 10), joka tarjoaa helpon tavan luoda virtauksia erilaisilla solmuilla. Ohjelma otetaan käyttöön yhdellä napin painalluksella. JavaScript-ohjelmointikielen toimintoja voi luoda muokkaimen sisällä monipuolisessa tekstimuokkaimessa. Sisäänrakennettu kirjasto sallii hyödyllisien toimintojen, mallien ja virtauksien talletuksen. Node-RED-työkalu on rakennettu Node.js -ajoympäristön päälle, joka tarjoaa kevyen ajoajan Node.js -ympäristön tapahtumakohtaisen ja estottoman mallin takia. Noden pakettien tietolähteessä on yli 225000 moduulia, joiden avulla Node-RED-työkaluun voi asentaa helposti uusia toimintoja. (JS Foundation 2017b.)



Kuva 10. Node-RED-ohjelmointityökalu verkkoselaimessa

Node-RED-ohjelmointityökalu ei ole minulle ennestään tuttu. Virtauspohjainen ohjelmointitapa ei myöskään ollut minulle ennestään tuttu, vaikka olen ohjelmoitavien logiikoiden parissa tutustunut FBD-ohjelmointikieleen (Function Block Diagram). FBD-kieli vaikuttaa visuaalisesti hieman samankaltaiselta kuin FBP-kieli (Flow-Based Programming).

8.3.2 Alivirtaukset

Alivirtaukset ovat solmuja, joiden sisälle voi rakentaa samanlailla virtauksia, kuin normaalissa näkymässä. Päänäkymään ilmestyy alivirtauksen lohko, kun se on luotu. Niitä voidaan luoda päävalikon “Subflows: Create Subflow” -valinnasta. (A Medium Corporation

2018.) Alivirtaukset ovat hyödyllinen tapa luoda modulaarisia yksiköitä, joiden sisältöä voidaan käyttää useissa eri tilanteissa. Ohjelmaa voidaan selventää alivirtauksien avulla, koska niihin voi ryhmitellä samaan aiheeseen liittyviä toimintoja. Opinnäytetyössä tehtävissä ohjelmissa on laaja valikoima erilaisia toimintoja, joita voi jaotella alivirtauksiksi.

8.3.3 Solmut

Solmut ovat virtaukseen vedettäviä lohkoja, joilla on tietyt toimintaperiaatteet. Solmuilla luotu ohjelma tulee käyttöön, kun muokattu virtaus otetaan käyttöön. Solmut voivat vastaanottaa ja lähettää viestejä virtauksen ollessa käynnissä. Solmut poistuvat käytöstä, kun uusi virtaus otetaan käyttöön. Ne koostuvat yleensä tiedostoparista; JavaScript tiedostosta, joka määrittää solmun käyttäytymisen sekä html tiedostosta, joka määrittää solmun asetukset ja ohjeen. Solmu sisältää myös JSON tiedoston, joka pakkaa solmun kaikki tiedostot yhdeksi npm-moduuliksi. (JS Foundation 2017c.)

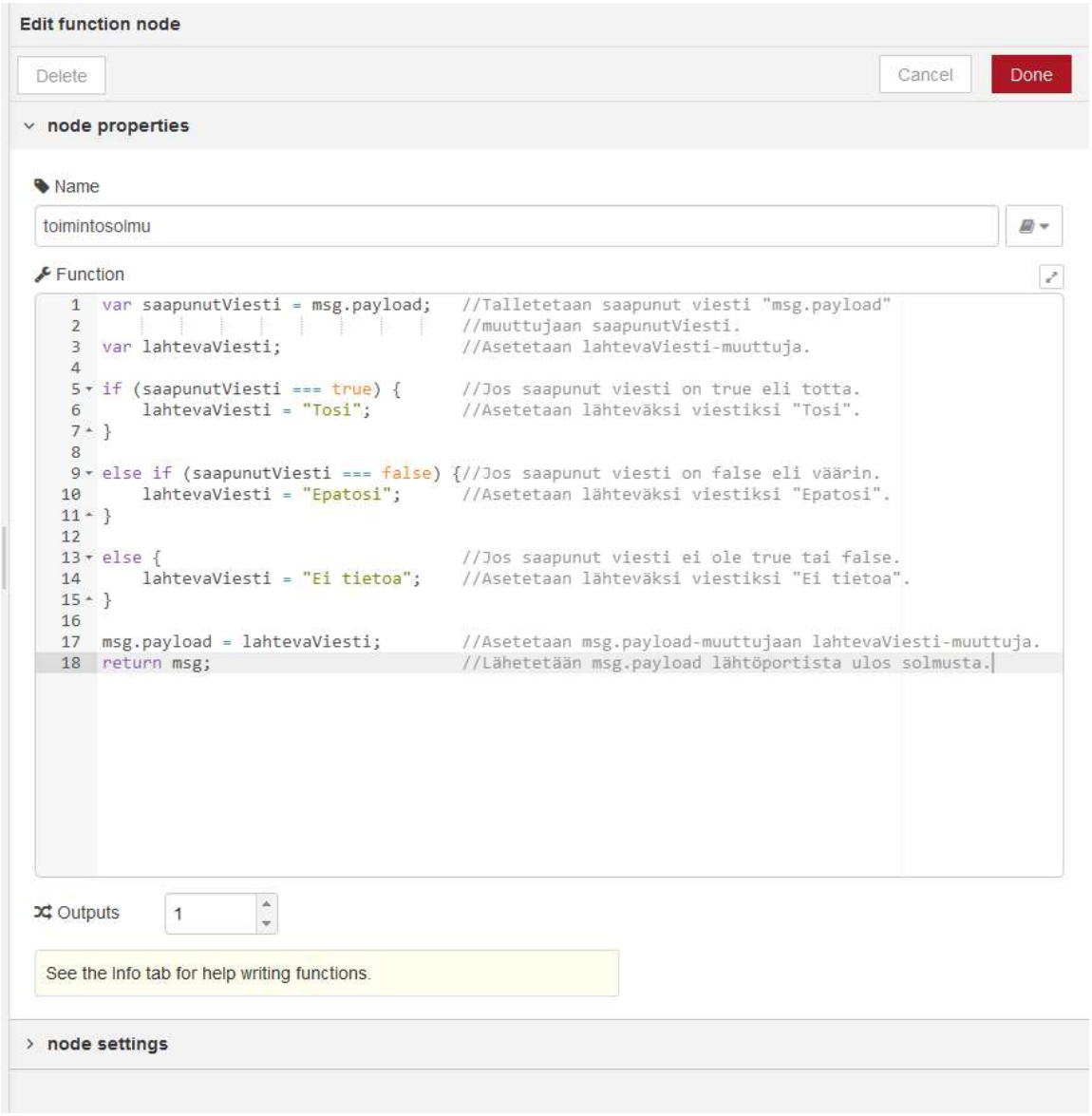
Node-RED sisältää valmiiksi jo useita erilaisia solmuja. Oletussolmuihin sisältyy muun muassa seuraavia toimintoja:

- tulo- ja lähtö
- ajastin
- http- ja mqtt-yhteydet
- virheilmoitukset
- tiedoston käsittely
- kommentointi
- tiedostomuotojen käsittely
- vapaamuotoisen toiminnon ohjelmointi.

Node-RED-työkaluun voidaan asentaa kirjastoja, jos oletussolmut eivät riitä ominaisuuksiltaan ohjelman tekemiseen. Tällainen tilanne voi tulla vastaan, kun halutaan käyttää jotain erikoisempaa protokollaa, kuten esimerkiksi modbus-protokollaa. Oletussolmut eivät sisällä solmuja, joiden avulla luodaan yhteys modbus-protokollalla. Uusia solmuja lisätään työkaluun ”Manage palette” -toiminnon avulla tai asentamalla ne suoraan komentoriviltä npm-työkalun avulla.

8.3.4 Toimintosolmu

Toimintosolmuun voi kirjoittaa JavaScript-kielellä vapaamuotoista ohjelmaa. Mielestäni toimintosolmu luo joustavuutta ja monipuolisuutta ohjelmien tekemisessä. Jokainen toimintosolmu vastaanottaa yhteen tuloon viestin. Ohjelma käy toimintosolmun sisältämän ohjelman läpi, kun viesti on saapunut solmun tuloporttiin. Solmulla on yksi tai useampi lähtöportti, johon voidaan ohjata viestejä. Yksittäiseen solmuun saapuvia ja lähteviä viestejä kutsutaan ohjelmassa msg.payload-muuttujalla. Kuvassa 11 nähdään tekemäni esimerkkiohjelma, joka on yksinkertainen ehtolause. Tämän solmun läpi kulkevat viestit laukaisevat toimintosolmun ohjelman, jolloin solmun lähtöportti palauttaa viestin, jos viestin arvo on ollut ”true”. Toimintosolmulla voidaan tehdä tällaisten yksinkertaisten ohjelmien lisäksi vaikka monimutkaisempiakin ohjelmia, joissa ohjelman koko on suuri.

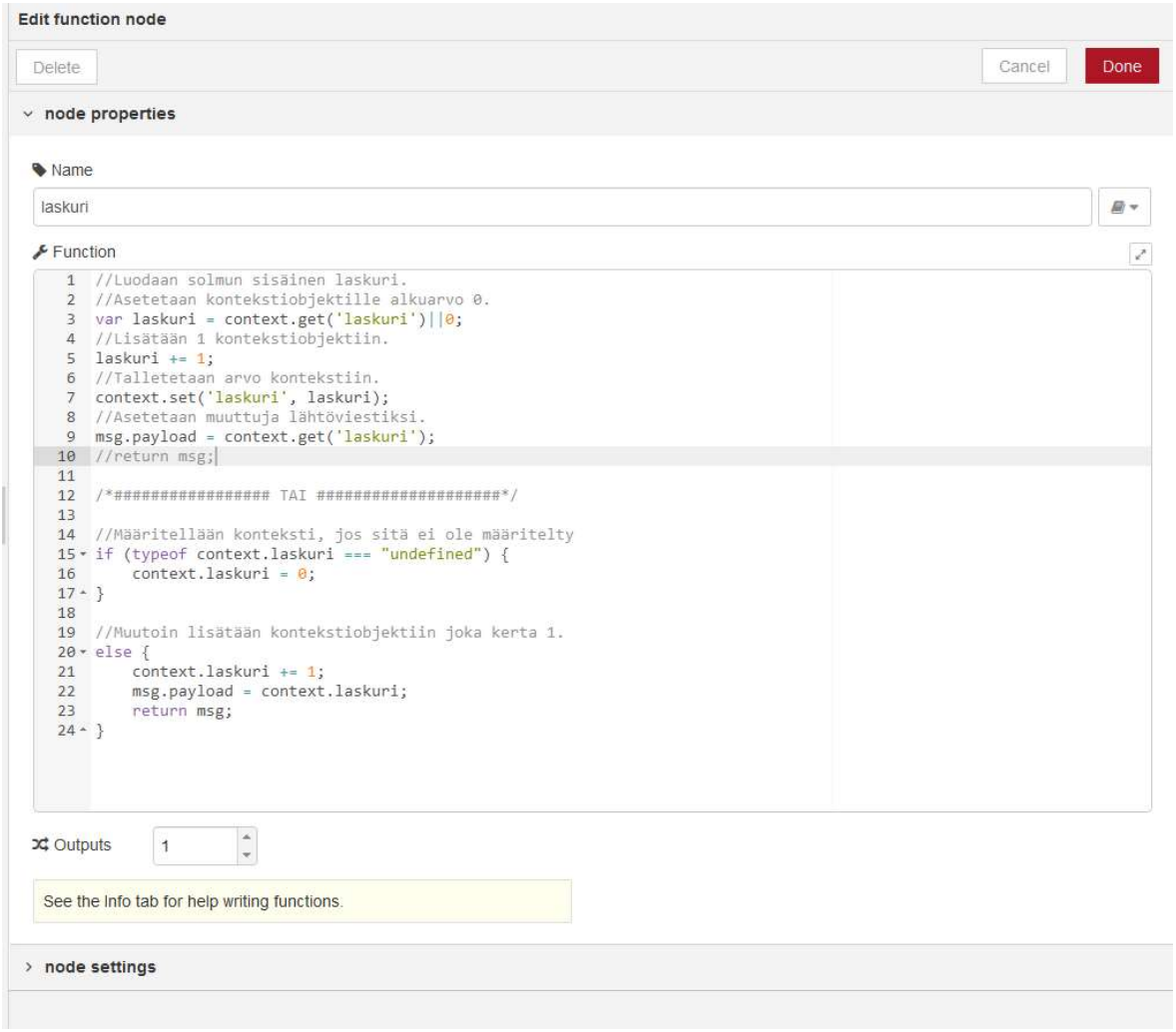


Kuva 11. Yksinkertainen esimerkki toimintosolmun sisältämästä JavaScript -ehtolauseesta

Toimintosolmun avulla voidaan korvata monimutkainen solmuista koostuva rakenne yhdellä solmulla. Yksinkertaisissa ohjelmissa riittää se, että valitsee oikeat solmut ohjelmaan ja yhdistää ne keskenään siten, että ohjelma toimii. Monimutkaisemmissa ohjelmissa on mielestäni helpompi käyttää toimintosolmuja, joiden sisälle on ohjelmoitu käsin kyseisen solmun toimintaa, kuin yhdistellä ja käyttää suuria määriä valmiita solmuja. Esimerkiksi toimintosolmun avulla voitaisiin luoda ohjelmakierto, joka kommunikoi etälukijan kanssa OSDP-protokollan avulla, koska yleistä kirjastoa OSDP:lle ei ole saatavilla.

8.3.5 Kontekstiobjektit

Ohjelmakierron aikana voidaan pysyviä tietoja tallettaa kolmeen erilaiseen kontekstiobjektiin. Kontekstiobjektit palautuvat alkutilaan Node-RED:n uudelleen käynnistyessä. Solmujen sisäisiä pysyviä muuttujia talletetaan solmun kontekstiobjektiin, jota voidaan käyttää ainoastaan kyseisessä solmussa.

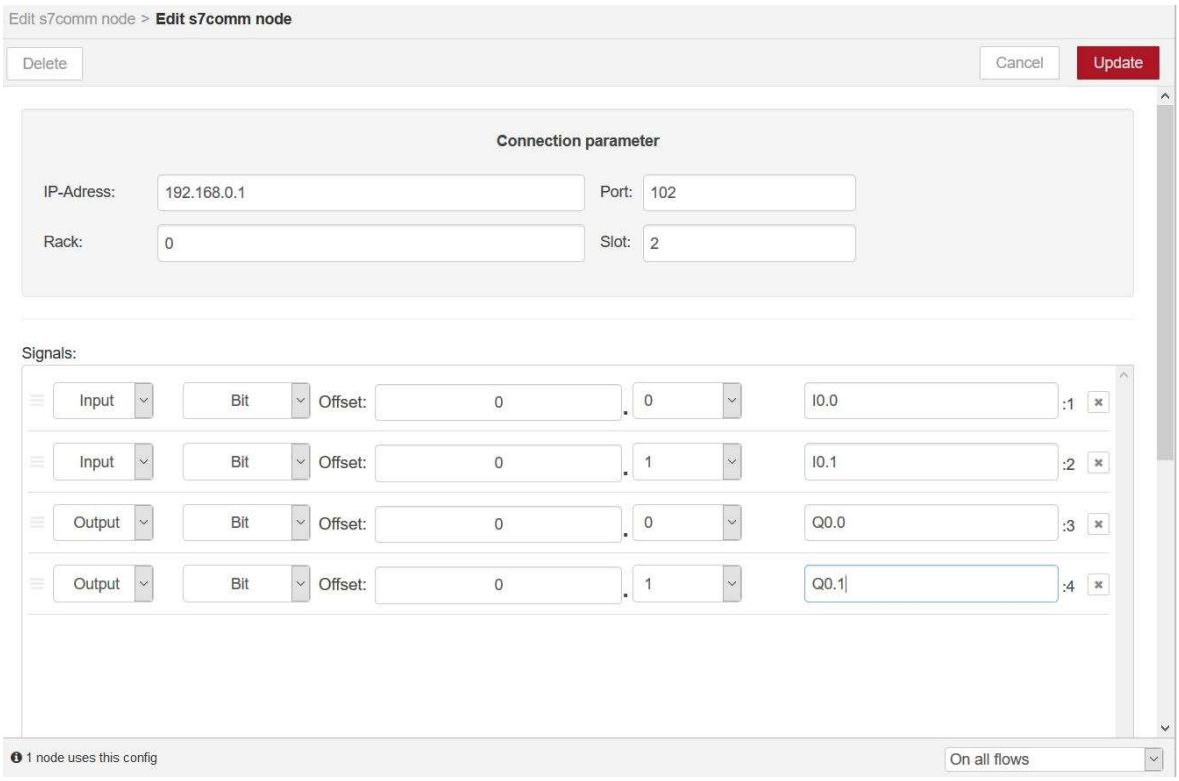


Kuva 12. Solmun sisäisen kontekstiobjektin määrittäminen toimintosolmussa

Solmujen sisäiset kontekstiobjektit ovat sidottu vain kyseiseen solmuun. Eri solmujen sisällä voidaan käyttää samannimisiä solmun sisäisiä kontekstiobjekteja ilman päällekkäisyyksiä. Virtauksen kontekstiobjektia voidaan käyttää virtauksen sisällä. Tällöin sitä voidaan käyttää kaikissa virtauksen sisältävissä solmuissa. Globaalia kontekstiobjektia nimensä mukaisesti voidaan käyttää missä tahansa ohjelmaa. (JS Foundation 2017e.) Virtauksen pysyviksi muuttujiksi voidaan määrittää arvoja, joiden käyttö ei rajoitu yhteen solmuun. Esimerkiksi virtauksen ensimmäisessä solmussa määritetään kontekstiobjektin arvoksi yksi tai nolla ja virtauksen toisessa solmussa määritetään tuleva toiminto ensimmäisen solmun kontekstiobjektin perusteella. Kuvassa 12 on esimerkkinä yhden toimintosolmun sisäisestä yksinkertaisesta laskurista. Tämä laskuri on pysyvä muuttuja, mutta on voimassa vain yhdessä solmussa. Globaaliksi muuttujaksi voidaan määrittää arvoja, joiden käytettävyys ei saa rajoittua pelkästään yhden virtauksen sisään. Tiettyjä varauksia on oltava käytettäessä globaaleja muuttujia, koska tällöin voi syntyä päällekkäisyyksiä samanlailla nimettyjen kontekstiobjektien kanssa. Globaalien kontekstiobjektien hallinnointi voi muuttua työlääksi, jos ohjelma koko kasvaa.

8.3.6 S7comm-solmut

Node-red-contrib-s7comm-solmu on suunniteltu kommunikoimaan Siemens SIMATIC S7-tuoteperheiden logiikoiden kanssa. Solmu lukee ja kirjoittaa logiikan osoitteita tietyillä S7-tietotyypeillä. S7comm -solmun voi asentaa “Manage palette” -toiminnon avulla tai suoraan komentorivillä. (JS Foundation 2017d.) Tutkitaan S7comm-solmun ominaisuuksia luomalla TIA Portal -työkalun avulla yksinkertainen testiohjelma S7-1513 CPU:n sisälle. “Optimized block access” -valinta täytyy ottaa pois käytöstä. S7-1500 -tuoteperheen ohjaimessa täytyy ottaa käyttöön “Enable GET/PUT Access” -toiminto. Tämä valinta sallii muiden sovellusten pääsyn ohjaimeen. (Github Inc 2018.)

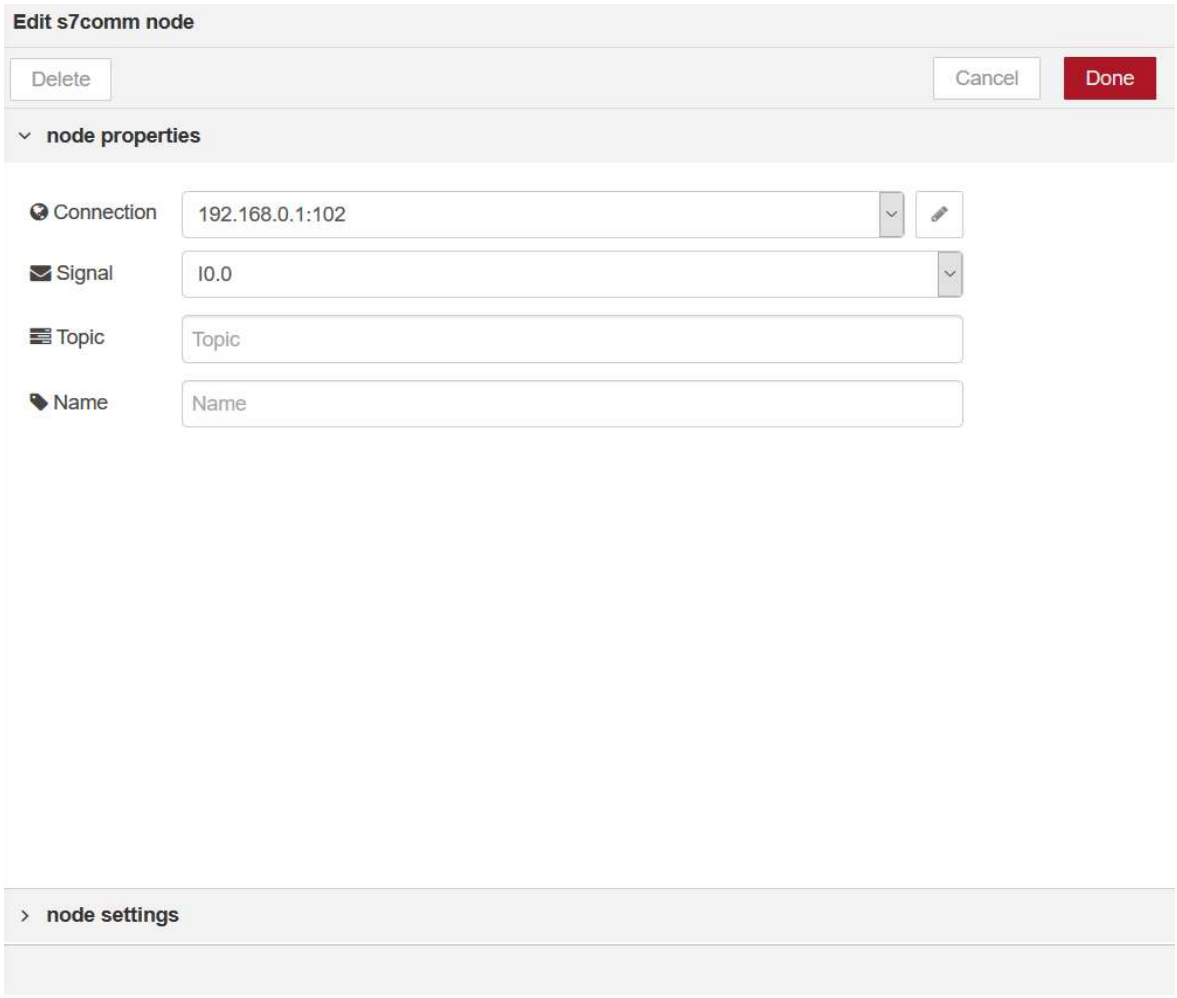


Kuva 13. S7comm solmun yhteysparametrit

S7comm solmun yhteysparametreihin (kuva 13) määritetään luvun 7.2.1 mukaiset yhteyden muodostamiseen tarvittavat arvot. IP-osoitteen ja porttinumeron määrittäminen liittyy S7comm-protokollan ensimmäiseen vaiheeseen. IP-osoitteeksi asetetaan logiikan IP-osoite ja porttinumeroksi oletuksena 102. Yhteyden muodostamisen toisessa vaiheessa määritetään yhteysparametreihin logiikan telineen ja korttipaikan numero. Kuvassa 13 telineen numeroksi on määritetty nolla ja korttipaikaksi numero kaksi. S7comm-solmun yhteysparametreihin määritetään myös käytettävät signaalit. Kuvassa 13 on signaaleiksi määritetty kaksi tuloa ja kaksi lähtöä. Signaalin asetuksissa valitaan tiedon laji, tietotyyppi, osoite ja vapaavalintainen nimi.

Solmun asetuksista (kuva 14) otetaan käyttöön määritetty yhteys ja siihen liittyvä signaali. Yhden S7comm-solmun avulla voidaan käyttää vain yhtä yhteysparametrin signaalia. Käytettäessä kuvan 13 kaikkia signaaleja yhtä aikaa samassa ohjelmakierrossa tarvitaan neljä solmua. Siemensin ohjelmoitaville logiikoille on olemassa myös S7-niminen Node-RED-kirjasto, joka perustuu samaan protokollaan kuin S7comm-kirjasto. S7-kirjastossa voidaan kaikkia määritettyjä logiikan muuttujia lukea samaan viestiin, joka voi tiivistää S7comm-protokollaan perustuvien solmujen määrää.

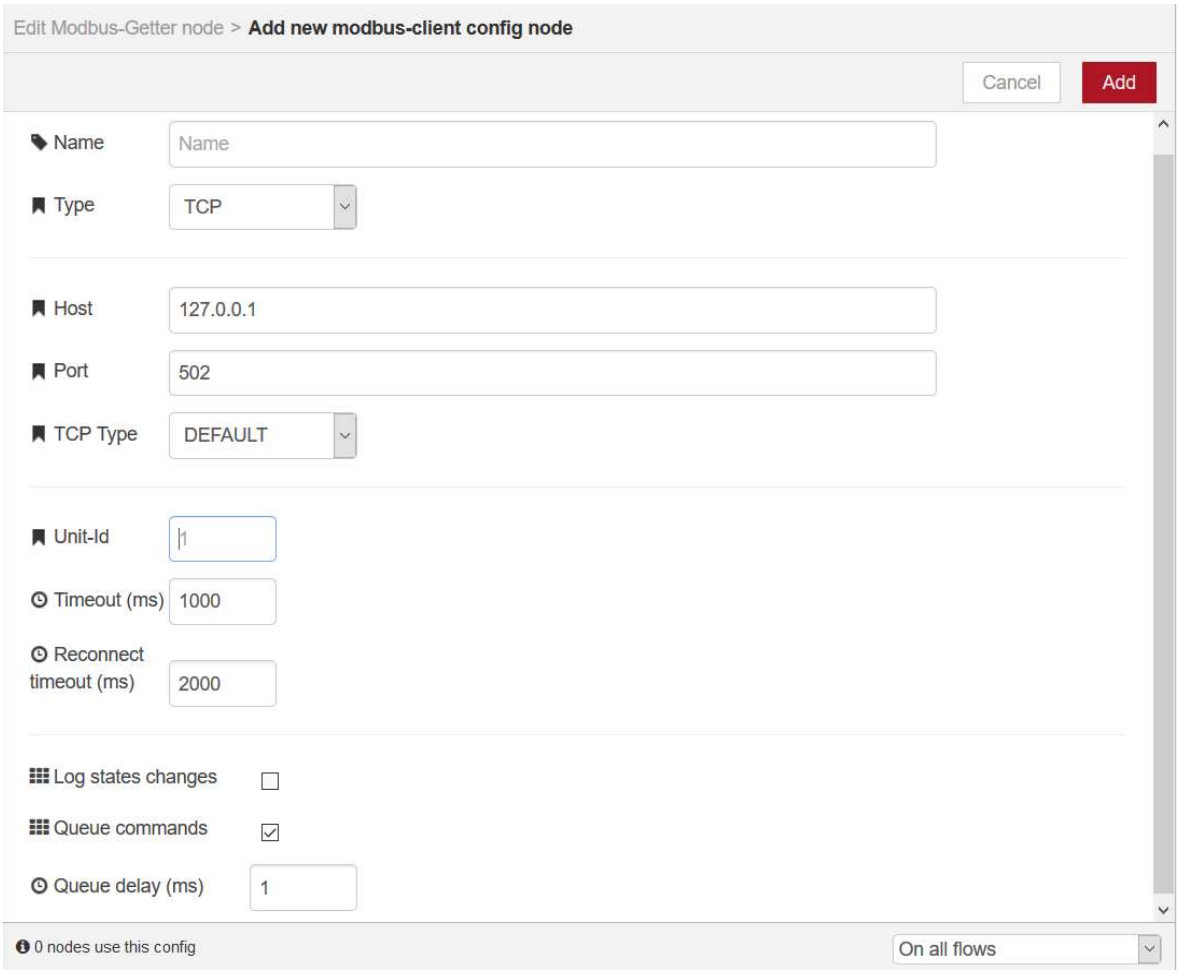
Testasin S7comm-kirjastoa yksinkertaisen logiikkaohjelman avulla. TIA Portal -työkalulla loin S7-1513 -suorittimeen ohjelman, joka asetti 32-bittiselle muistipaikalle tietyn luvun riippuen siitä, mikä muistipaikka on päällä. Loin ohjelmaan myös ”Watch table” -taulun, jota pystyi tarkkailemaan logiikkasuorittimen pieneltä näytöltä. Node-RED-ohjelmassa ohjasin logiikan kaksiarvoisia muistipaikkoja, jolloin 32-bittinen rekisteri sai tietyn arvon. Yksinkertainen testaus toimi hyvin ja Node-RED-ohjelman hakemat logiikan arvot vastasivat logiikan ”Watch table” -taulun arvoja.



Kuva 14. S7comm-solmun asetukset

8.3.7 Modbus-solmut

Vuoden 2016 huhtikuussa moduulin node-red-contrib-modbus tekijäksi on nimetty Klaus Landsdorf. Moduulin avulla voidaan luoda yhteys käyttäen Modbus/TCP tai Modbus RTU -protokollia. Moduulin käyttöä opastetaan lähinnä videoilla ja esimerkkiohjelmalla, joten kirjoitan moduulin toiminnasta omien havaintojeni perusteella.

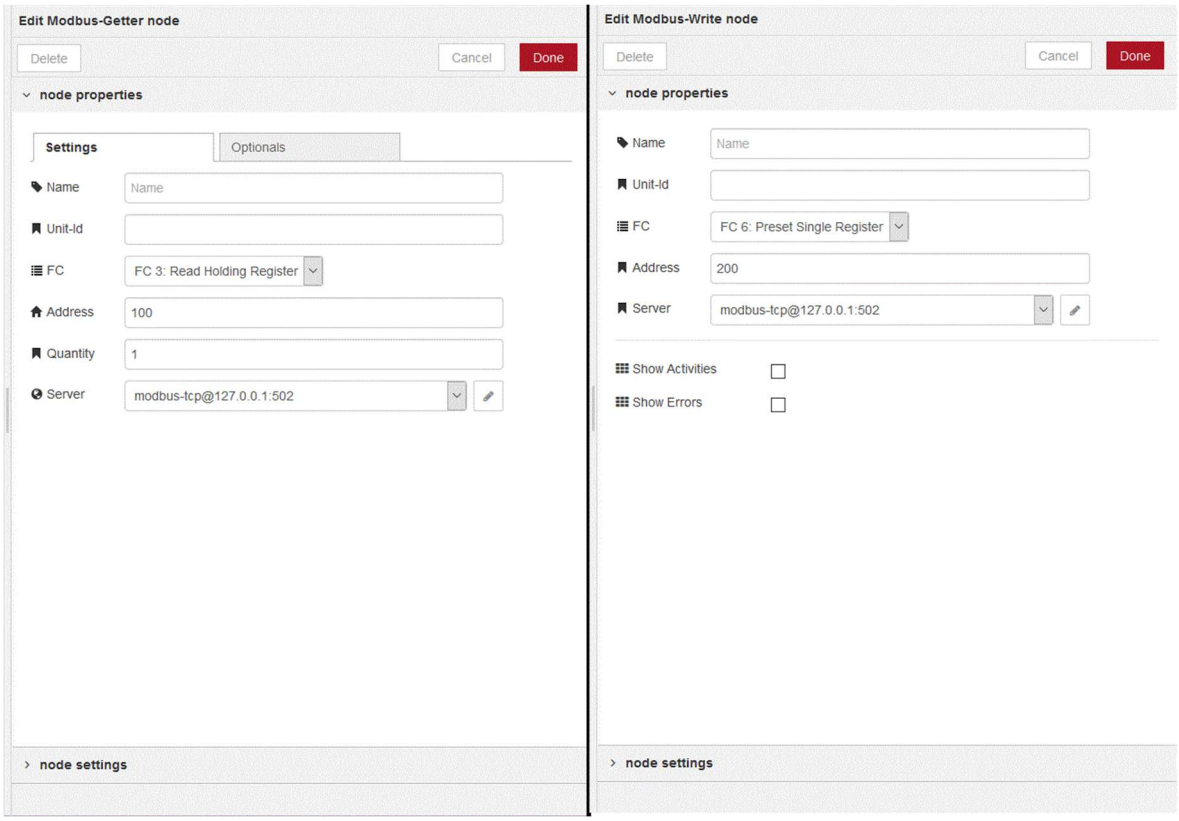


Kuva 15. Modbus-solmun yhteysparametrit

Solmun yhteysparametreihin (kuva 15) määritetään luvussa 7.2.2 ilmoitettuja muuttujia. Solmun yhteysparametrit liittyvät sovelluksen tietoyksikköön. Yhteysparametreihin määritetään monia asioita, mutta tärkeimmät yhteyden kannalta ovat tyyppi, laitteen IP-osoite ja porttinumero Modbus/TCP-protokollaa käytettäessä, tietotyyppi ja yksikön tunniste. Modbus-solmujen asetuksissa (kuva 16) määritetään luvussa 7.2.2 käsitelty protokollan tietoyksikkö. Asetuksissa valitaan toimintokoodi ja rekisterin osoite. Jos kyseessä on rekisterin lukeminen, määritetään myös rekisterien määrä. Opinnäytetyön kannalta tärkeimmät modbus -solmut ovat Modbus-Getter- ja Modbus-Write-solmut. Modbus-Getter-solmu on tarkoitettu lukemistoiminnoille ja Modbus-Write-solmua käytetään kirjoitustoiminnoissa.

Voidaan modbus-kirjaston avulla rajata toimintaa siten, että ohjelmaan luotaisiin erillinen osio modbus-muistialueiden lukemiseen ja oma alue modbus-laitteen ohjaamiseen, jolloin kirjoitetaan tietyille muistialueille haluttuja komentoja.

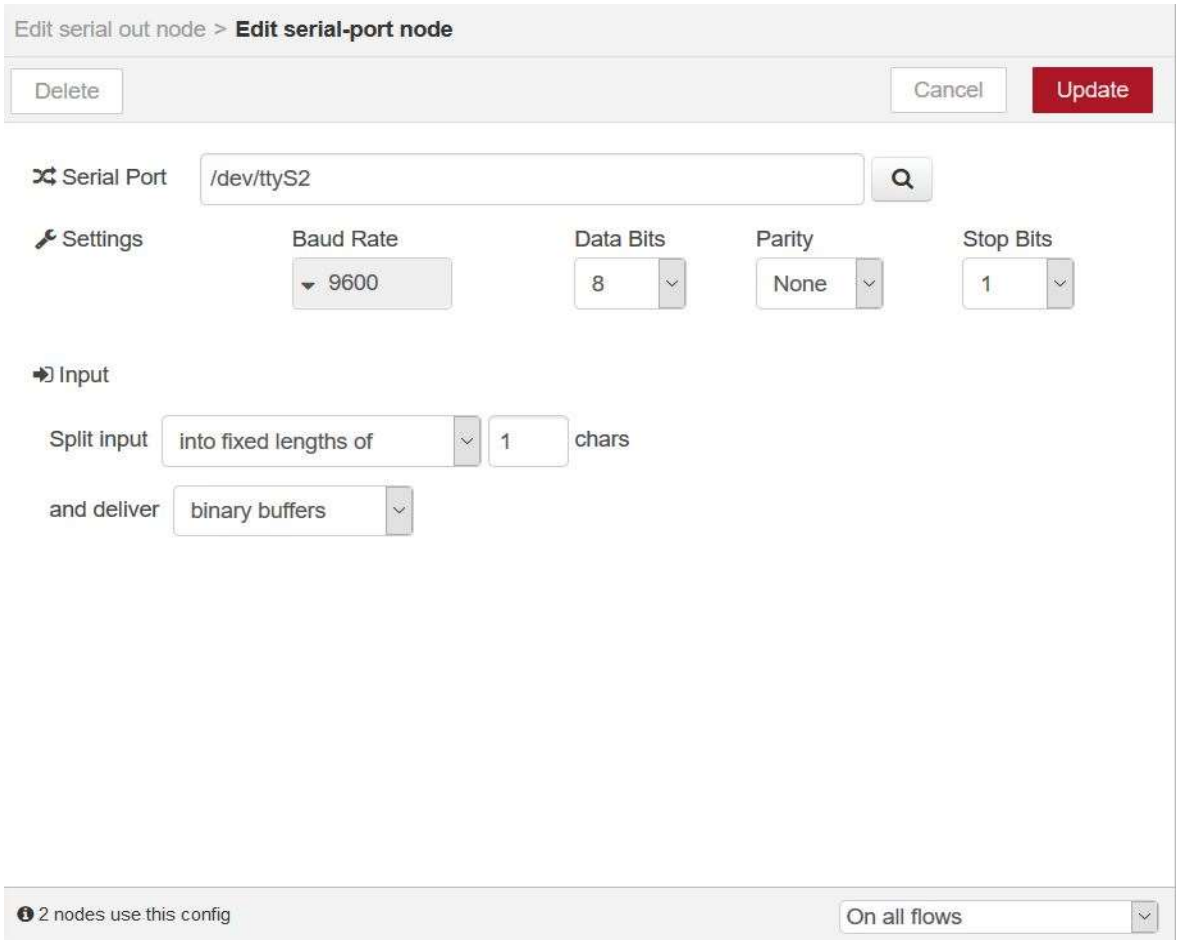
Testasin modbus-kirjastoa WA-805-vaa'an avulla. Tein Node-RED-ohjelman, joka luki joitain vaa'an rekistereitä ja kirjoitti komentoja vaa'alle. Rekistereistä saatiin luettua oikeanlaista tietoa ja vaa'an ohjaus onnistui. Testissä oli käytössä myös Modbus poll -ohjelma, jolla tarkistin testin onnistumisen. Modbus poll on yleinen ohjelma, jolla voidaan suorittaa modbus-toimintoja laitteille.



Kuva 16. Modbus-Getter- ja Modbus-Write-solmujen asetukset

8.3.8 Serialport-solmut

Serialport-solmut on suunniteltu yhteydenpitoon paikallisten sarjaporttien kanssa. Tähän sisältyy kaksi eri solmua, toinen vastaanottamaan ja toinen lähettämään viestejä sarjaporttiin. Sarjaportin yhteysasetuksiin (kuva 17) määritetään yhteyteen liittyvät asiat, kuten siirtonopeus, lähetettävät databitit, pariteetti- ja stop-bitit. Lisäksi määritellään sarjaporttiin lähetettävän ja sarjaportista vastaanotetun viestin rakenne. Esimerkiksi kuvan 17 mukaisissa asetuksissa sarjaportilta saapuva viesti vastaanotetaan yksi merkki kerrallaan ja sarjaporttiin lähetettävä viesti lähetetään binäärisenä puskurina.



Kuva 17. Serialport-solmun yhteysasetukset

Kirjastoa testattiin lähettämällä erilaisia sanomajonoja lukijalle. Viestin saapuessa etälukijalle, se käsittelee sen ja lähettää vastauksen. Yksinkertainen Node-RED-ohjelma lähetti etälukijalle viestejä ja se vastasi takaisin ja toteutti viestin toiminnon.

8.4 Testausvaiheen lopputulos

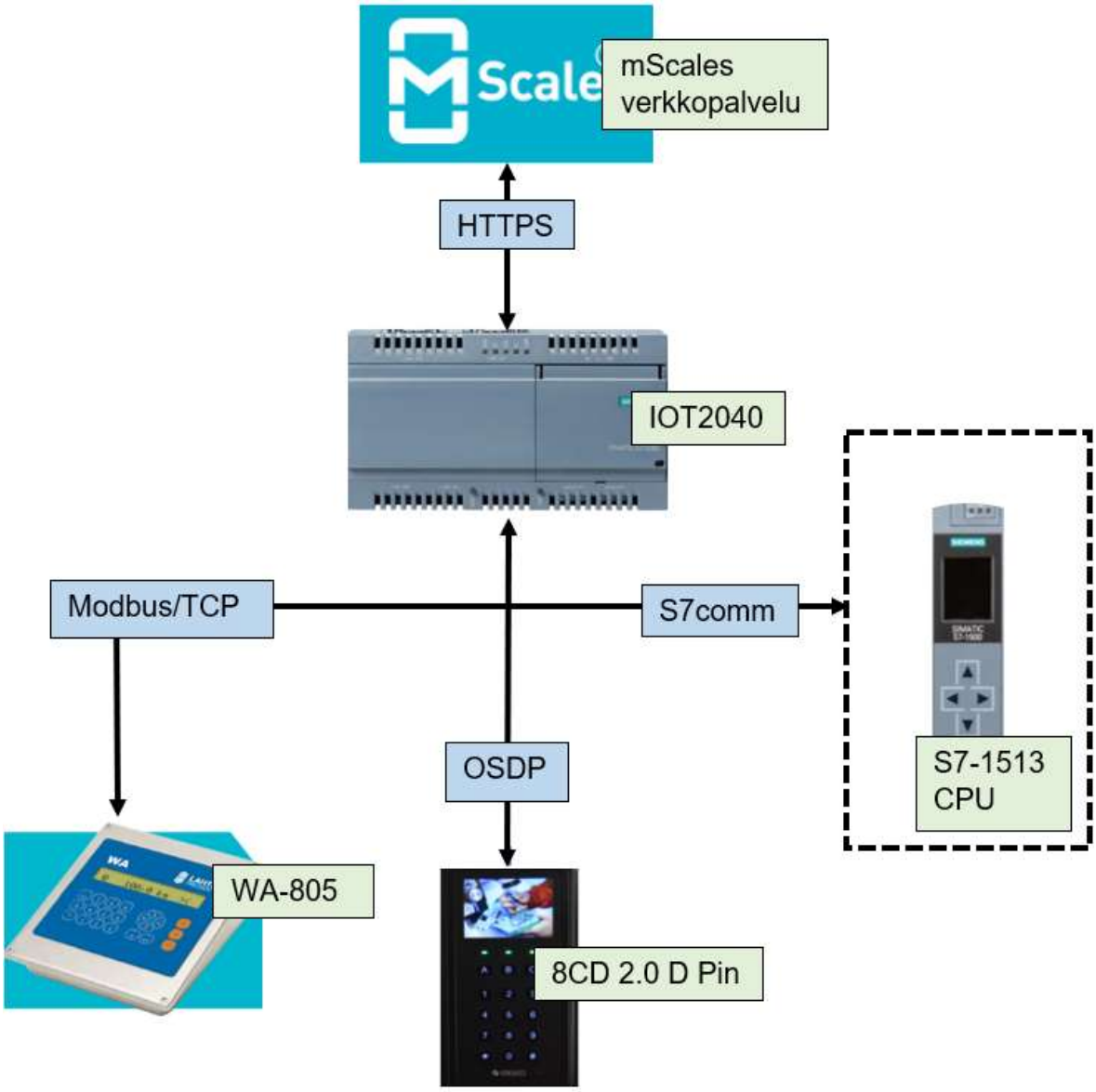
Testausvaiheessa ei havaittu ongelmia laitteiden yhdistämisessä gateway-laitteeseen. Teknisiä haasteita voi ilmetä käytettäessä S7comm-kirjastoa sellaisen logiikan kanssa,

joka sisältää suuren määrän luettavaa ja kirjoitettavaa tietoa. Node-RED-ympäristöön ladattavat kirjastot vaikuttivat luotettavilta ja kaikki niistä oli luotettavien tahojen luomia. Testausvaiheessa tärkein tavoite oli todeta, että käytettävillä komponenteilla ja työkaluilla voidaan edetä toteutusvaiheeseen. Tämä tarkoittaa sitä, että näiden laitteiden ja työkalujen ominaisuudet riittävät kokonaisen sovelluksen tekoon. Testausvaiheessa saatiin Node-RED-ohjelmointityökalun ja IOT2040-yhdyskäytävän avulla luotua yhteys Lahti Precision Oy:n vaakapäätteeseen, Idescon etälukijaan, Siemensin ohjelmoitavaan logiikkaan ja mScales-verkkopalveluun. Node-RED on mielestäni kattava työkalu punnituksen pilvipalvelun laajennuksen tekemiseen. Sillä on suhteellisen helppo tehdä testiohjelmia ja sen toiminnallisuudet ovat riittävät.

9 TOTEUTUSVAIHE

9.1 Alkutilanne

Tämän opinnäytetyön toteutusvaiheessa tehtiin soveltuvuusselvitys eli Proof of Concept. Opinnäytetyössä mallinnettiin oikeaa tilannetta soveltuvuusselvityksessä, jolloin voitiin alustavasti jo todeta tietyn sovelluksen toimivuus. Mallinnettava tilanne oli siltavaaka. Kenttälaitteina käytetään WA-805-vaakapäätettä ja Idescon näytöllistä RFID-lukijaa. Tarkoituksena oli mallintaa tilannetta, jossa kuljettaja saapuu siltavaa'alle ja tekee onnistuneen kaupallisen punnituksen etälukijalla. Vaaka suorittaa punnituksen etälukijan käskystä ja lähettää punnituksen tiedot etälukijan näyttöön, kun punnitus on tehty.



Kuva 18. Proof of Consept -mallin rakenne

Kuvassa 18 on mallinnettu PoC-mallin fyysistä rakennetta. Huomioitavaa on se, että PoC-mallissa kommunikoidaan mScales-verkkopalvelun kehitysympäristössä, joka toimii paikallisesti. Verkkopalvelusta puhuttaessa tarkoitetaan kehitysympäristöä. Ohjelmoitava logiikka on asetettu katkoviivan sisään, koska sitä ei tulla käyttämään mallinnettavassa tilanteessa. Ohjelmoitavan logiikan käyttäminen voi kuitenkin olla oleellinen asia tulevaisuudessa, jonka takia yhteydenpitoa siihen testattiin testausvaiheessa. Proof of Consept -mallin avulla on myös tarkoituksena lisätä ymmärrystä tällaisten sovellusten kehittämisessä.

9.2 WA-805

WA-805-vaaka kuuluu WA-800-sarjaan. Sarjan vaakapäätteet ja annostusohjaimet on tarkoitettu vaativiin teollisen punnituksen sovelluksiin. Niitä voidaan käyttää itsenäisinä vaakapäätteinä tai osana laajempaa järjestelmää. WA-800-sarjan laitteet asennetaan seinään tai pöydälle. WA-805 on täyttävä annostusohjain, jota käytetään automaattisissa annostussovelluksissa. Liittyminen pääjärjestelmään on helppoa ja yhteydenpitoon on käytössä Modbus/RTU-, Modbus/TCP-, Profibus-, Profinet- ja DeviceNet-protokolla. WA-800 tuoteperhe on hyväksytty kaupalliseen punnitukseen. (Lahti Precision Oy 2016b.)

PoC-mallissa on käytössä Lahti Precision Oy:n WA-805-vaakapääte. Lahti Precision Oy:llä on aikaisemmin ollut kehitystyötä WA-903+-vaa'an parissa, minkä tuloksena vaa'an ohjelmistossa on protokolla mScales-verkkopalvelua varten. Tällainen vaaka ei tarvitse gateway-laitetta saadakseen yhteyden verkkopalveluun, ainoastaan internet-yhteyden. WA-903+-vaa'alla voidaan suorittaa erilaisia toimintoja, joita käytetään ajoneuvon punnituksessa. Tavoitteena on suunnitella ja luoda samankaltainen ohjelma gateway-laitteeseen. Ohjelman avulla voitaisiin yhdistää WA-805-vaaka verkkopalveluun. Ohjelmassa tulisi olla mahdollisuus tehdä komentoja vaa'alla sekä suorittaa kaupallinen punnitus. Luodaan ohjelma Node-RED-työkalun avulla ja yhdistetään vaaka Modbus/TCP-protokollaa käyttäen.

9.2.1 Toimintaperiaate

Vaakaohjelma lähettää tietyn väliajoin tilatietoa verkkopalveluun. Verkkopalveluun lähetetään tilatietoa HTTP-protokollan avulla, johon on valmis kirjasto Node-RED-ympäristössä. Verkkopalvelu vastaa tilatietoon suoritettavalla toiminnolla. Vaakaohjelmaan määritetään tietyt toiminnot, jotka suoritetaan tietyn komennon perusteella. Tällaisia komentoja ovat:

- ei komentoa
- punnitse

- kuittaa
- liikennevalo päälle
- nollaa vaaka.

Komento on osa tilatietoon vastatusta viestistä, joka sisältää myös komennon tapahtumaan liittyviä asioita. Vaakaohjelma suorittaa komentoa vastaavan toiminnon ja palauttaa komennon tilatiedon verkkopalveluun. Verkkopalvelu vastaa komennon tilatietoon uudella komennolla. Node-RED-ympäristöön tulevan vaakaohjelman toimintaperiaate ja rajapinnat pidetään samanlaisena valmiiseen WA-903+-vaakaan nähden.

”Ei komentoa” -komento vaatii vaakaohjelmalta uuden tilatiedon. Tilatietoa varten kerätään Modbus/TCP-protokollaa hyödyntäen vaa’an modbus-rekisteristä ja vaakaohjelmasta tiettyjä arvoja, joista rakennetaan verkkopalveluun lähetettävä viesti. Viestiin sisällytetään esimerkiksi vaa’an painolukemaa ja nollaustietoa. Painolukemaa päivitetään käyttäjälle verkkosivuilla ja nollaustietoa käytetään sallimaan nollaustoiminto käyttäjälle.

”Nollaa”, ”kuittaa”, ”punnitse” ja ”liikennevalo päälle” -komentoja vastaavat toiminnot ovat Modbus-Write-solmun käyttämistä. Näiden toimintojen jälkeen verkkopalveluun lähetetään viesti komennon onnistumisesta. Komennoilla on tietyt tapahtumat:

- ”Nollaa” -komennolla vaaka nollaantuu, jos se on sallitun nollausrajan sisällä.
- ”Kuittaa” -komento kuittaa tapahtuneen virheen vaakapäätteeltä.
- ”Punnitse” -komento käynnistää vaa’an kaupallisen punnitustapahtuman. Tällöin vaaka punnitsee vaa’alla olevan painon ja lähettää punnitustapahtuman tiedot verkkopalveluun.
- ”Liikennevalo päälle” -komennolla pakotetaan vihreä liikennevalo päälle tietyksi ajaksi. Tämä komento lähetetään verkkopalvelulta vaakaohjelmaan yleensä punnituksen jälkeen, jolloin käyttäjä voi ajaa pois vaa’alta.

9.2.2 Toimintojen erottelu

Ensimmäisessä vaakaohjelman versiossa käytettiin globaaleja kontekstiobjekteja erottamaan eri toiminnot toisistaan. Objektien tarkoituksena oli määrittää ohjelman sisällä seuraavaksi suoritettava toiminto. Globaalien kontekstiobjektien käyttäminen osoittautui epäedulliseksi ideaksi, koska niiden hallinnointi tulisi työlääksi, kun eri ohjelmien määrä kasvaisi. Koska globaaleja kontekstiobjekteja voidaan käyttää joka puolella ohjelmaa, voi samanlailla nimettyjä muuttujia esiintyi vahingossa muissa ohjelmissa, jolloin virheen

syntyminen on erittäin todennäköistä. Parempi vaihtoehto olisi käyttää virtauksen sisäisiä kontekstiobjekteja, jolloin ne voidaan rajata yhteen virtaukseen.

Vaakaohjelman ollessa rakenteeltaan sellainen, että siinä kulkevat viestit yleensä aiheutuvat tapahtumamuutoksesta, on hyödyllisintä käyttää msg.topic-muuttujaa eri toimintojen tallettamiseen. Tämä muuttuja on osa viestiä, joka kulkee virtauksessa viestin mukana. Toimintojen erittely suoritetaan niin, että msg.topic-muuttuja saa tietyn arvon, joka perustuu verkkopalvelun vastaukseen. Tämän arvon perusteella ohjelmassa aloitetaan sitä vastaava toiminto. Viestissä on msg.topic-muuttuja mukana muuttumattomana koko viestiketjun ajan. Tällöin vältytään kontekstiobjektien käyttämiseltä ja toiminnot ovat riippuvaisia jokaisesta viestiketjusta.

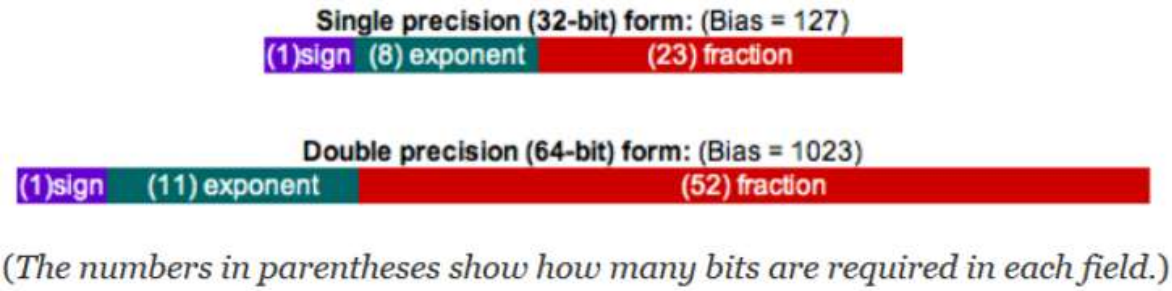
9.2.3 Tietotyypin muunnokset

Määrätyssä ohjelman kohdassa vaaditaan joissain tapauksissa tietyn tyyppistä lauseketta. Tämä lauseke voisi olla myös suure, joka sijoitetaan ohjelmakierron aikana sellaisen muuttujan arvoksi, joka vaatii suureelta tiettyjä rakenteellisia ominaisuuksia. Suure ei aina ole oikean tyyppinen, jolloin suoritetaan tietotyypin muunnos. Esimerkiksi vaaditun suureen täytyy olla kokonaisluku, mutta alkuperäinen arvo onkin liukuluku. (Korpela & Larmela 1992, 90.)

Vaa'alta luetaan Modbus/TCP-protokollalla eri osoitteissa olevia rekistereitä, joissa säilytetään kahden tai neljän tavun pituisia arvoja. Rekisterin arvot ovat kokonais- tai liukulukuja. Rekistereistä haetaan muun muassa vaa'an paino-, tila-, punnitus- ja virhetietoja. Tietyt rekisteristä luetut arvot ovat sellaisessa muodossa, joita ei voida suoraan lukea. Tällöin täytyy tehdä tietotyypin muutos, jolloin käännetään rekisterin arvo luettavaan muotoon. Vaakaohjelmaan tuleville arvoille yleisimpiä tehtäviä muutoksia olivat liukulukustandardin IEEE 754 mukainen ja kokonaisluvun muunnos binäärimuotoon.

9.2.4 IEEE 754 -muunnos

IEEE 754 on standardi, jota käytetään muuttamaan IEEE 754 lukuja liukuluvuiksi ja päinvastoin. Liukulukujen esittäminen tietokoneelle on haasteellinen asia, jonka takia on luotu tieteellinen tapa tehdä liukulukujen muunnos.



Kuva 19. IEEE 754-standardin bittirakenne (The Oxford Math Center 2018)

Kuvassa 19 on esiteltynä kahden erilaisen IEEE 754 -muunnoksen bittirakennetta. Binääriluku jaotellaan kolmeen erilaiseen ryhmään; merkki- (sign), eksponentti- (exponent) ja murtolukubitteihin (fraction). Liukuluku muodostuu kaavalla 1:

$$(-1)^{\text{merkkibitti}}\,(1+\text{murtoluku})\times 2^{\text{eksponentti}-\text{poikkema}}$$

Merkkibitti on rakenteen ensimmäinen bitti ja se määrittää liukuluvun etumerkin. Saadessaan arvon nolla, liukuluku on positiivinen ja arvolla yksi liukuluku on negatiivinen. Eksponenttibitin tarvitsee merkitä positiivisia ja negatiivisia eksponentteja, joten alkuperäiseen eksponenttiin lisätään poikkeama (bias), joka 32-bitin rakenteessa saa arvon 127. IEEE 754 kantalukuna on kaksi, jolloin eksponentti saadaan, kun kerrotaan liukulukua niin kauan, että se saa arvon, joka on yhden ja kahden välillä. Siten tiedetään, millä kahden eksponentilla luku muutettiin sellaiseksi, että se kuuluu lukujen yksi ja kaksi välille. Murtolukubitit kuvailevat liukuluvun murtolukuja binäärimuodossa. (The Oxford Math Center 2018.) Opinnäytetyössä muunnetaan lukuja IEEE 754 -muodosta liukuluvuiksi, joten lasketaan esimerkkilasku tämän mukaisesti. Muunnettava luku on heksadesimaalina 0x46445F33, joka binääriluvuksi käännettynä on 01000110010001000101111100110011. Seuraavaksi jaotellaan binääriluku IEEE 754 -muodon kolmeen ryhmään;

- merkkibitti: 0
- eksponenttibitit: 10001100
- murtolukubitit: 10001000101111100110011.

Merkkibitti saa arvon nolla, joten liukuluvun etumerkki on positiivinen. Muunnetaan laskimella eksponenttibitit kokonaisluvuksi, jolloin saadaan vastaukseksi 140. Murtolukubittien mukaan lasketaan yhteen jokaisen bitin kohdalla saatu arvo. Arvo lasketaan kaavalla 2:

$$\text{tila} \times 2^{-1 \times \text{järjestysnumero}},$$

jossa tila saa arvon nolla tai yksi riippuen bitin tilasta sekä järjestysnumero on sama kuin bitin sijainti jonossa. Ensimmäisenä jonossa oleva bitti saa järjestysnumeroksi yksi ja toisena oleva saa luvun kaksi. Esimerkiksi jonon kolmas bitti on tilassa yksi, jolloin kaavan 2 mukaan sen arvo on:

$$1 \times 2^{-1 \times 3} = 0.125$$

Murtoluvun arvo saadaan, kun lasketaan yhteen jokainen murtolukubitin arvo. Murtoluvun arvo voidaan laskea esimerkiksi taulukossa, mutta arvon laskemista varten voidaan muodostaa summaoperaattorin avulla kaava 3:

$$\sum_{\text{i}=1}^{s} \text{b} * 2^{((-1)*\text{i})},$$

jossa b on kyseisen bitin tila, s on murtolukubittien määrä ja i on bitin järjestysnumero. Tällä hetkellä on laskettu merkkibitti, joka osoitti luvun olevan positiivinen. Eksponenttibittien perusteella eksponentti on 140 ja murtolukubittien perusteella murtoluvun arvoksi saadaan noin 0.53415525. Sijoitetaan arvot liukuluvun kaavaan 1:

$$(-1)^0 \, (1 + 0.53415525) \times 2^{140-127} = \, 12567{,}79 \, \approx \, 12567{,}80$$

Binääriluku 10001100100010001011111100110011 on IEEE 754 -standardin avulla muunnettuna 12567,80. Kuvassa 20 on toimintosolmun sisäinen ohjelma, jonka avulla voidaan IEEE 754 -lukuja muuttaa liukuluvuiksi. Ohjelma muuntaa IEEE 754 -luvut oikein ainoastaan silloin, kun luvut saapuvat käänteisessä järjestyksessä solmuun. Solmu lähettää lähtöporttiin muunnetun liukuluvun.

Function

```
1  msg.topic = 'grossWeight1';
2  var raw = new ArrayBuffer(4);
3  var intView = new Int16Array(raw);
4  var floatView = new Float32Array(raw);
5  intView[0] = msg.payload[1];
6  intView[1] = msg.payload[0];
7  msg.payload = floatView[0];
8  return msg;
```

Kuva 20. IEEE 754 -muodon muuttaminen liukuluvuksi ohjelmallisesti

9.2.5 Kokonaisluvusta binäärimuotoon

Tässä kappaleessa käsitellään kymmenjärjestelmän lukujen muuntaminen binäärijärjestelmään. Käsin tehtäviä muunnoksia on monia erilaisia ja esimerkissä lasketaan kymmenjärjestelmän luku binäärijärjestelmään jakaen luvulla kaksi. Tarkoituksena on ilmaista jakojäännös kokonaislukuna ja lisätä siihen tarvittaessa yksi tai nolla. Luku yksi lisätään jakojäännökseen, jos se sisältää desimaalin 0,5. Jakojäännöksen mennessä tasan, lisätään siihen nolla. Luvun sisältäessä desimaalin 0,5 pyöristetään luku alaspäin seuraavaan kymmenykseen ja jatketaan luvun jakamista. Lasku lopetetaan, kun viimeisen jakojäännöksen tulos on alle yksi. Jakojäännöksistä saadaan käänteisesti lukemalla kokonaislukua vastaava binääriluku. (Jyväskylän yliopisto 2008.)

541	270	135	67	33	16	8	4	2	1
270,5 | 270	135 | 135	67,5 | 67	33,5 | 33	16,5 | 16	8 | 8	4 | 4	2 | 2	1 | 1	0,5 | 0
+	+	+	+	+	+	+	+	+	+
1	0	1	1	1	0	0	0	0	1

Kuva 21. Kokonaisluvun käsin laskettu binäärimuoto

Kuvassa 21 on excel-taulukkoon laskettu muunnos kokonaisluvusta binäärilukuun. Käsin laskettaessa kirjotetaan binääriluku päinvastaisessa järjestyksessä jakojäännökseen nähden. Luku 541 on binäärijärjestelmässä 1000011101. Node-RED-ohjelman toimintasolmussa muunnos tehdään Javascript-kielellä, joka tarjoaa suoran komennon muunnokselle (kuva 22). Kuvan 22 mukainen komento muuttaa luvun binäärimuotoon, mutta arvo on merkkijonona.

```
//Talletetaan kokonaisluku
var kokonaisluku = msg.payload;
//Muunnon binääriksi
var binaari = kokonaisluku.toString(2);
msg.payload = binaari;
return msg;
```

Kuva 22. Muutos kokonaisluvusta binääriluvuksi ohjelmallisesti

9.3 8 CD 2.0 D Pin -etälukija

Idescon uudessa 8 CD 2.0 D Pin -RFID-lukijassa on pin-kooditunnistus ja näyttö. Näppäimistö sisältää numeronäppäimet ja vapaasti ohjelmoitavat funktionäppäimet. Näyttö lisää vuorovaikutteisuutta kulunvalvontajärjestelmän ja käyttäjän välille. Näytön kautta voidaan välittää tietoa kulunvalvonnan tapahtumista käyttäjälle. Laite sopii ulkokäyttöön, koska sen suojausluokka on IP67 ja käyttölämpötila-alue -30 – 55 celsiusastetta. Yhteydenpitoon lukija käyttää OSDP tai OSDP v2 protokollaa ja sarjaliikenneväylänä RS-485. (Idesco Oy 2017b.)

Etälukijaa on tarkoitus käyttää punnitustapahtuman käynnistämisessä. Etälukijan avulla vaa'an käyttäjä syöttää tunnisteen, jonka avulla liitetään käyttäjä tehtävään punnitukseen. Tunniste voisi olla numeroyhdistelmä tai RFID-tunniste. Tunnistetta vastaan palautetaan punnitukseen ja käyttäjään liittyvät tiedot ja käyttäjän toimesta aloitetaan punnitus. Onnistuneen punnitustapahtuman jälkeen näytetään etälukijan näytöllä punnituksen tietoja käyttäjälle, kunnes käyttäjä hyväksyy kuitin ja lähtee vaa'alta pois.

9.3.1 Toimintaperiaate

Etälukijan toimintaa suunniteltaessa tultiin johtopäätökseen, että lukijan näytöllä on monia erilaisia tapahtumiin liittyviä näyttöpohjia. Lukijassa olisi kielen valinta -pohja, jonka jälkeen lukijan näytölle ilmestyisi käyttäjän tunnistus -pohja. Lukijan toiminta perustuu tiettyyn tapahtumakiertoon liittyviin näyttöpohjiin, joita näytetään käyttäjälle. Tiettyyn tapahtumaan liittyy myös rajaukset siitä, mitä käyttäjä voi tietyssä pohjassa tehdä. Etälukija toimisi niin sanottuna tilakoneena, jolla olisi tapahtumaan liittyviä tiloja. Jokaisessa tilassa on

määritelmät käyttäjälle sallituista toiminnoista ja näytölle tulostettavasta näyttöpohjasta. Käynnissä olevaan tilaan liittyisi aina aloitustoiminnot, joilla rakennetaan näyttöpohja lukijan näytölle. Joihinkin tiloihin liittyy tiettyjä toimintoja, joita käyttäjä voi tehdä, kuten esimerkiksi nappien painaminen tai RFID-tunnisteen näyttäminen. Lukija ja IOT2040 välillä vaihdetaan tilaviestejä, jotka muutetaan komentoviesteiksi, jos lukijalla tehdään toimenpiteitä.

9.3.2 Tilakone

Automaation sovelluksissa tilakone on tyypillinen ratkaisu. Tilakoneessa on useita eri tiloja ja tilasiirtymiä. Esimerkiksi robotilla voisi olla kolme erilaista tilaa: kappaleeseen tarttuminen, kappaleen siirtäminen ja kappaleen jättäminen loppupisteeseen. (Metropolia, 2018) Etälukijan toiminta päätettiin suunnitella niin, että se toimisi kuin tilakone. Jokainen erilainen näyttö on oma tilansa, jolla on tilan tyypin perusteella erilaisia ominaisuuksia. Tällöin voidaan helposti erottaa tiettyyn tilaan liittyvät ominaisuudet ja niiden siirtämisen ohjelmakiertoon sekä tilasiirtymät. Tietyn tilan ominaisuudet määrittävät tilasiirtymän aloittamisen. Tilakone-tyyppinen ratkaisu etälukijan ohjelmoinnissa tuo lisäarvoa järjestelmällisyyden takia. Tilakoneessa on tietyt määritelmät, jotka toteutuvat tilasiirtymän jälkeen. Tilakone-tyyppinen ratkaisu tuo järjestelmällisyyttä ohjelman rakenteeseen.

Käytännössä etälukijalle määritellään tietynlaiset tilat sovelluksesta riippuen. Punnitustapahtuman käyttöliittymänä etälukijalla on tila, jossa tunnistetaan vaa'an käyttäjä. Punnitustapahtuman jälkeen käyttäjälle näytettäisiin punnitustapahtuman kuittia, joka on oma tilansa. Punnitustapahtumaan liittyisi myös useita erilaisia tiloja, joita ovat esimerkiksi kielenvalinta, käyttäjätietojen näyttäminen ja kehotusviestit.

9.3.3 CRC-algoritmi

CRC (cyclic redundancy check) on virheen havaitsemismenetelmä. Tarkastettava sanoma tulkitaan polynomin kertoimiksi ja kyseinen polynomi jaetaan tarkastuspolynomilla. Tarkastuksessa saatava jakojäännös liitetään alkuperäiseen sanomaan ja tallennetaan tai lähetetään eteenpäin. Tarkastettaessa suoritetaan samanlainen jakolasku koodisanan ja jakojäännöksen muodostamalle kokonaisuudelle. Sanoma on virheetön, jos jako menee tasan, muutoin sanoma on virheellinen. CRC-algoritmia käytetään muun muassa siirtojärjestelmien bittivirhesuhteen laskentaan eli siirron laadun valvontaan. Algoritmi on myös mukana useissa virheenkorjaavissa siirtoprotokollissa. Virheen havaitseminen vastaanottopäässä aiheuttaa uudelleenlähetyspyynnön ja uudelleenlähetyksen. (Haltsonen & Levomäki & Rautanen 2013, 186-187.)

Opinnäytetyössä CRC-algoritmia käytetään etälukijan ja yhdyskäytävän välisessä yhteydenpidossa. Etälukijalle lähetettävän viestin kaksi viimeistä merkkiä ovat CRC-algoritmilla laskettu tarkastussumma. Myös lukijalta saapuva viesti tarkastetaan samalla algoritmilla. Seuraavaksi ratkaistaan tarkastuksessa saatava jakojäännös esimerkkilaskuna. Tarkastuspolynomi on:

$P(x) = x^3 + x + 1,$

missä jokainen polynomin ilmoitettu muuttuja on tarkastuspolynomissa oleva “1” -bitti. Tarkastuspolynomin perusteella tarkastuslukumme on binäärimuodossa 1101. Lausekkeessa ei ilmoitettu x^2 -muuttujaa, jolloin sen tilalle tulee 0 eli tarkastusluvussa kolmannen bitin kohdalle. Tarkistettava polynomi on:

$M(x) = x^3 + 1,$

jolloin se on binäärimuodossa 1001. Tarkistettavan polynomin perään täytyy myös lisätä n määrä nollia, joka saadaan tarkastuspolynomin suurimman potenssin mukaan. Tässä tapauksessa

n = 3,

koska tarkastuspolynomissa suurin potenssi on kolme. Tarkastuspolynomin pituus myös saadaan selville kaavalla 4:

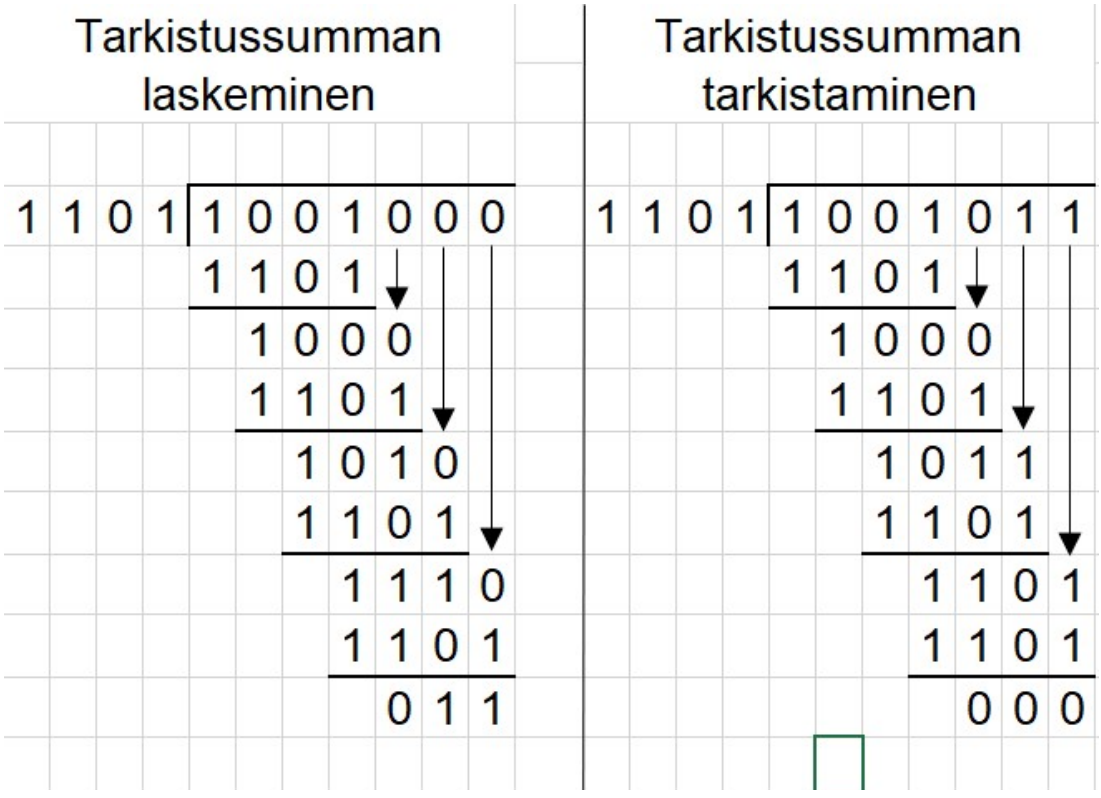
$l = n + 1,$

missä l on tarkastuspolynomin merkitsevien bittien lukumäärä ja n suurin potenssi tarkastusluvussa. Lisäksi CRC-laskussa on nolla-arvo, joka on yleensä 0000. CRC-menetelmän perusperiaate on aina sama, mutta siinä käytettävät luvut vaihtuvat. Yksinkertaisuudessaan polynomi jaetaan tarkistuspolynomilla. Jäljelle jäävän bittijonon ensimmäisen bitin perusteella tehdään jakolasku. Ollessaan 1 se jaetaan tarkistuspolynomilla. Kun ensimmäinen bitti on 0, niin jono jaetaan nolla-arvolla. Jako tehdään XOR-portin avulla (kuva 23). Jonon ensimmäistä bittiä ei jaeta. Lopussa lisätään seuraava jonon bitti jakojonon perään. Kun lisättävät bitit loppuvat, niin tarkistussumma on laskettu.

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Kuva 23. XOR -portin totuustaulu (DYclassroom, 2018)

Esimerkki laskussa lasketaan alussa annetun tarkistuspolynomin avulla tarkistettavalle polynomille tarkistussumma, joka on binäärimuodossa 011 (kuva 24). Tarkistussumma on jakolaskun jakojäännös. Tarkistussumma siten lisätään polynomin perään lisättyjen nollien tilalle, jolloin tehdään tarkistussumman tarkistaminen. Sanoman ollessa virheetön, jako-jäännökseksi saadaan nolla, kuten kuvassa 24.



Kuva 24. Tarkastussumman ratkaisu ja tarkistus CRC-menetelmällä

9.4 Konseptin todennus

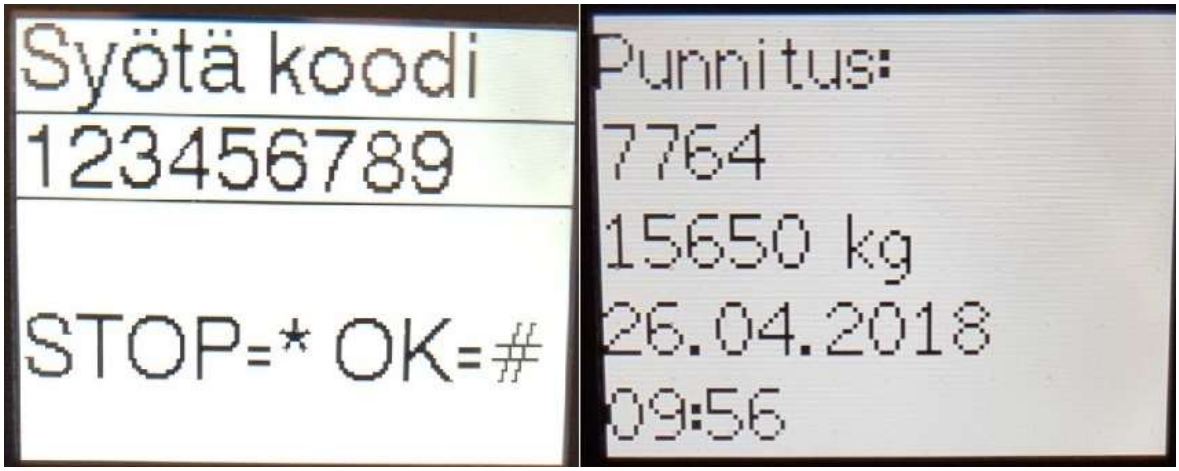
Konseptin todennus varten luotiin käyttäjätarina, jonka perusteella suunniteltiin Node-RED-ohjelma. Käyttäjätarina kuvailee samankaltaista sovellusta, joka voitaisiin toteuttaa oikeasti. Tavoitteena on todentaa, että eri kenttälaitteisiin saadaan yhteys ja niiden toimintaa voidaan ohjata Node-RED-ohjelmasta. Huomioon otettavia asioita ovat muun muassa etälukijan ja vaa'an yhteistoiminta, laitteiden tietojen hallinta yhdyskäytävässä ja verkkopalvelun rajapinnat.

9.4.1 Todennuksen käyttäjätarina

Konseptia simuloidaan oikean tilanteen mallinnuksella, johon kuuluu esimerkkitoiminnan käyttäjätarina:

- Saadaan toimeksianto lisätä verkkopalvelun asiakkaaksi Oy Asiakas Ab. Asiakkaalla on tilaus, jonka kuljetusliikkeenä on Kuljetusliike Oy.
- Luodaan punnitusnumero kuljetusliikkeen kuljettajalle, joka voi käyttää punnitusnumeroa punnituksessa.
- Kuljettaja saapuu vaa'alle, kun liikennevalo on vihreänä. Etälukijan näyttö on tähän asti ollut näytönsäästö -tilassa, mutta ajoneuvon paino laukaisee etälukijan päälle ja liikennevalon punaiseksi.
- Etälukijan ensimmäisessä näytössä näytetään kuljettajalle kielivaihtoehtoja.
- Kielen valinnan jälkeen kuljettaja syöttää punnituskoodinsa, jonka perusteella kuljettaja tunnistetaan (Kuva 25).
- Kuljettajalle näytetään tietoja punnitustapahtumasta, jolloin hän hyväksyy tiedot ja laukaisee punnitustapahtuman.
- Punnituksen onnistuessa näytetään kuljettajalle punnituksen tietoja näytöllä (Kuva 25).

- Tämän jälkeen liikennevalo syttyy vihreäksi ja kuljettajaa pyydetään poistumaan vaa'alta. Punnitustapahtuma on suoritettu onnistuneesti.



Kuva 25. Vasemmalla käyttäjä syöttää koodia lukijalle ja oikealla punnitus on tehty ja siitä on saatu punnituskuitti

9.4.2 Konseptin toteutuminen

Luvun 9.3.1 mukainen käyttäjätarina toteutui suurimmalta osaltaan opinnäytetyössä. Tietokantaa ei ollut vielä käytössä, joten käyttäjätiedot olivat raakakoodattuina ohjelmassa. Ilman tietokantaa voitiin kuitenkin varmistua siitä, että konsepti toimii ja se on kehitettävissä tulevaisuudessa. Node-RED-työkalun avulla luotiin toimiva Modbus/TCP-yhteys vaa'an ja gateway-laitteen välille. Gateway kommunikoi myös etälukijan kanssa ja yhdisti laitteet siten, että vaakaa voitiin ohjata etälukijalta. Gateway-laitteella ollessa yhteys kenttälaitteisiin, piti se yhteyttä myös verkkopalveluun, mihin se lähetti muun muassa vaa'an tilatietoa. Laitteiden tietoja vastaanotetaan ja hallitaan onnistuneesti yhdyskäytävässä.

9.4.3 Tulevaisuus

Konseptin todennus onnistui hyvin ja kehitettäviä asioita riittää tulevaisuuteen. Gateway-laitteella voidaan yhdistää kenttälaitteita verkkopalveluun ja uskon tällaisen menetelmän olevan hyödyllinen asia tulevaisuudessa teollisen internet -käsityksen yleistyessä. Gateway-laitteella voisi myös testata erilaisia ohjelmointiympäristöjä ja todeta parhain tällaiseen IoT-kehitykseen.

10 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli tehdä laajennus Lahti Precision Oy:n punnituksen pilvipalveluun ja löytää siihen tarkoitukseen sopiva laite. Laajennus käsittää teollisia komponentteja, joita liitetään osaksi pilvipalvelun toimintaa. Tavoitteena oli toteuttaa Proof of Concept -malli laajennuksesta ja simuloida toimintaa mallinnetulla tilanteella. Opinnäytetyössä perehdyttiin teolliseen internetiin ja sen sovelluksen konseptin todentamisen eri vaiheisiin. Työssä käsiteltiin erilaisia teollisen internetin tiedonsiirron ja laitteiden yhteyskäytäntöjä. Laitteiden yhdistäminen pilvipalveluun tehtiin Node-RED-työkalun avulla. Laitteiden yhdistäminen palveluun onnistui ja laajennuksen konsepti todettiin toimivaksi, mikä oli opinnäytetyön tavoite.

Työn onnistumisen myötä löydettiin sopiva laite, jota voidaan tulevaisuudessa käyttää teollisen internetin sovelluksen kehittämiseen. IOT2040-yhdyskäytävällä saatiin luotua valmius yhdistää kolme erilaista laitetta mScales-pilvipalveluun. Yhdistämisen valmiudella tarkoitan sitä, että laitteet saadaan yhdistettyä yhdyskäytävään. Tämä tarkoittaa sitä, että laitteiden välittäessä tietoa yhdyskäytävään, voidaan tieto lähettää eteenpäin pilvipalveluun erilaisten rajapintojen avulla. Tulevaisuudessa IOT2040-laitteeseen voidaan liittää uusia laitteita ja testata myös muita ohjelmointiympäristöjä. Node-RED-työkalu sopi ihan hyvin opinnäytetyön toteuttamiseen, koska sen käyttö oli helppoa ja ominaisuuksiltaan laaja. Silti olisi hyvä testata erilaisia ohjelmointiympäristöjä ja -kieliä sovelluksen tekemiseen, koska silloin voidaan vertaamalla valita parhain tapa kehittää sovellusta.

Opinnäytetyö oli mielestäni laaja ja tarpeeksi haastava. Alussa käytin aikaa protokollien ja ohjelmistoteknisten asioiden opettelussa. Kokonaisen sovelluksen kehitys jäi hieman kesken, mutta opinnäytetyön tavoite konseptin todentamisesta onnistui. Opinnäytetyötä tehdessä opin uusia asioita tiedonsiirrosta, protokollista, Node-RED-työkalusta ja Lahti Precision Oy:n vaakapäätteistä. Kehitystyö jatkuu opinnäytetyön aiheen parissa.

LÄHTEET

Painetut lähteet

Airila, M. 1999. MEKATRONIIKKA. Luku 6. Helsinki: Hakapaino Oy.

Collin, J & Saarelainen, A. 2016. Teollinen internet. Helsinki, Talentum.

Haltsonen, S & Levomäki, J & Rautanen, E.T. 2013. Digitaalitekniikka. Porvoo. Bookwell Oy.

Korpela, J & Larmela, T. 1992. C-ohjelmointikieli. Espoo. OtaDATA ry.

Elektroniset lähteet

A Medium Corporation 2018. How To Use Node-RED [viitattu 24.5.2018]. Saatavissa: https://medium.com/node-red/how-to-use-node-red-af8afa12ddcc

Barry & Associates Inc 2018. Service-Oriented Architecture (SOA) Definition [viitattu 6.2.2018]. Saatavissa: https://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html

DYclassroom 2018. Logic gateXOR [viitattu 27.10.2018]. Saatavissa: https://www.dyclass-room.com/logic-gate/xor-and-xnor-logic-gate

FläktWoods 2016. Funktiokoodi [viitattu 25.10.2018]. Saatavissa: http://resour-ces.flaktwoods.com/Perfion/File.aspx?id=5ae5d3e3-2af6-46c6-9244-f7d3e1304f54

Github Inc 2018. nodeS7 [viitattu: 20.5.2018]. Saatavissa: https://git-hub.com/plcpeople/nodeS7/blob/master/README.md

Idesco Oy 2017a. What is RFID technology? -diaesitys [viitattu 15.3.2018]. Saatavissa: https://prezi.com/4itj8pp3rdsr/what-is-rfid-technology/?utm_campaign=share&utm_me-dium=copy

Idesco Oy 2017b. 8 CD 2.0 D Pin [viitattu 25.3.2018]. Saatavissa: https://idesco.fi/pro-duct/rfid-reader-display/

Internetix. 2018. OSI-malli [viitattu 6.5.2018]. Saatavissa: http://oppimateriaalit.interne-tix.fi/fi/avoimet/6tekniikkatalous/lahiverkko/standardointi

JS Foundation 2017a. About [viitattu 14.3.2018]. Saatavissa: https://nodered.org/about/

JS Foundation 2017b. Browser-based flow editing [viitattu 14.3.2018]. Saatavissa: https://nodered.org/

JS Foundation 2017c. Creating your first node [viitattu 14.3.2018]. Saatavissa: https://nodered.org/docs/creating-nodes/first-node

JS Foundation 2017d. node-red-contrib-s7comm [viitattu: 20.5.2018]. Saatavissa: https://flows.nodered.org/node/node-red-contrib-s7comm

JS Foundation 2017e. Writing Functions [viitattu 14.3.2018]. Saatavissa: https://node-red.org/docs/writing-functions

Jyväskylän yliopisto 2008. II.3. Lukujärjestelmistä [viitattu 24.5.2018]. Saatavissa: http://www.math.jyu.fi/matpo/kirja/dam/index-13.html

Kauppalehti 2016. Lahti Precision Oy [viitattu 21.1.2018]. Saatavissa: https://www.kauppalehti.fi/yritykset/yritys/lahti+precision+oy/08984550

Kuva 3. TechTarget. 2018. A gateway's position in the IoT ecosystem [Viitattu 16.3.2018]. Saatavissa: https://internetofthingsagenda.techtarget.com/definition/gateway

Lahti Precision Oy 2016a. Historia [viitattu 21.1.2018]. Saatavissa: https://www.lahtiprecision.com/fi/yritys/historia

Lahti Precision Oy 2016b. Lahti on punnitustalo [viitattu 25.3.2018]. Saatavissa: https://www.lahtiprecision.com/fi/tuotteet/vaakapaatteet-ja-punnituskomponentit/55/vaakapaatteet

Lahti Precision Oy 2016c. mScales punnituspalvelu [viitattu: 6.2.2018]. Saatavissa: https://www.lahtiprecision.com/fi/tuotteet/automaatio-ja-tiedonkeruujarjestelmat/50/mscales-punnituspalvelu

Lahti Precision Oy 2016d. Punnittua kokemusta [viitattu 21.1.2018]. Saatavissa: https://www.lahtiprecision.com/

Lahti Precision Oy 2016e. Yritys [viitattu 21.1.2018]. Saatavissa: https://www.lahtiprecision.com/fi/yritys

Lahti Precision Oy 2017. mScales -verkkopalvelu [viitattu 21.1.2018]. Saatavissa: https://www.mscales.com/#kenelle

Metropolia 2018. Tilakone [viitattu 20.5.2018]. Saatavissa: https://wiki.metropolia.fi/display/koneautomaatio/Tilakone

Modbus Organization 2018a. MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3 Määrittelytiedosto [viitattu 19.5.2018]. Saatavissa: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

Modbus Organization 2018b. Modbus FAQ: About the Protocol [viitattu 13.5.2018]. Saatavissa: http://www.modbus.org/faq.php

Modbus Organization 2018c. MODBUS PROTOCOL [viitattu 13.5.2018]. Saatavissa: http://www.modbus.org/specs.php

Mozilla 2018a. An overview of HTTP [viitattu 20.5.2018]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Mozilla. 2018b. HTTP request methods [viitattu 20.5.2018]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

OPC Foundation 2018. What is OPC? [viitattu 6.2.2018]. Saatavissa: https://opcfoundation.org/about/what-is-opc/

Real Time Automation 2018. Control IEC 61131-3 [viitattu 7.2.2018]. Saatavissa: https://www.rtaautomation.com/technologies/control-iec-61131-3/

Security Industry Association 2018. Open Supervised Device Protocol (OSDP) [viitattu 6.5.2018]. Saatavissa: https://www.securityindustry.org/industry-standards/open-supervised-device-protocol/

Siemens AG 2018a. SIMATIC OPC UA S7-1500 [viitattu 18.2.2018]. Saatavissa: http://w3.siemens.com/mcms/automation-software/en/tia-portal-software/step7-tia-portal/simatic-step7-options/opc-ua-s7-1500/pages/default.aspx

Siemens AG 2018b. The intelligent gateway for industrial IoT solutions [viitattu 4.3.2018]. Saatavissa: http://w3.siemens.com/mcms/pc-based-automation/en/industrial-iot/pages/default.aspx

Siemens AG 2018c. Tuotekuvaus ja hyödyt [viitattu 25.3.2018]. Saatavissa: http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic/s7_1500.php

TechTarget 2018. gateway [viitattu 16.3.2018]. Saatavissa: https://internetofthingsagenda.techtarget.com/definition/gateway

Tieto Oyj 2018. Teollinen internet – mikä se on? [viitattu 24.5.2018]. Saatavissa: https://www.tieto.fi/nakemyksia-ja-visioita/teollinen-internet-mika-se-on

The Oxford Math Center 2018. IEEE 754 Format [viitattu 24.5.2018]. Saatavissa: http://www.oxfordmathcenter.com/drupal7/node/43