

**Henri Hihnala**

# **MITTAUSTIEDON MONIPUOLISTAMINEN MODBUS-VÄYLÄLLÄ**

**Opinnäytetyö  
CENTRIA-AMMATTIKORKEAKOULU  
Automaatiotekniikan koulutusohjelma  
Marraskuu 2018**

## TIIVISTELMÄ OPINNÄYTETYÖSTÄ

<b>Centria-ammattikorkeakoulu</b>	<b>Aika</b> Marraskuu 2018	<b>Tekijä/tekijät</b> Henri Hihnala
<b>Koulutusohjelma</b> Automaatiotekniikka		
<b>Työn nimi</b> MITTAUSTIEDON MONIPUOLISTAMINEN MODBUS-VÄYLÄLLÄ		
<b>Työn ohjaaja</b> Hannu Ala-Pönttiö	<b>Sivumäärä</b> 32	
<b>Työelämäohjaaja</b> Petri Salo		
<p>Tämän opinnäytetyön tarkoituksena on esitellä Modbus-kommunikointiprotokolla ja mitä sen käytöllä voidaan saavuttaa. Modbus-kommunikointi pohjautuu master–slave-kommunikointiin. Kommunikointi tapahtuu Modbus-viesteillä, jotka ovat binäärisessä muodossa. Viestit ovat kooltaan muutamasta tavusta satoihin. Viesti on jaettu eri osiin, kuten osoite, funktiokoodi ja data. Masterin käskyt ovat usein datan kirjoittamista tai lukemista laitteilta. Master-laite lähettää ohjelmasta saamansa komennot slaveille. Slave toteuttaa masterin komentoja ja kommunikoi takaisin masterille.</p> <p>Opinnäytetyön toimeksiantajana toimi kokkolalainen Freeport Cobalt Oy. Freeport Cobalt on kemian alan tehdas, joka keskittyy koboltin jalostamiseen. Freeport Cobalt halusi korvata vanhat analogiset virtausmittarit uusilla Modbussia käyttävillä mittareilla. Vanhoilla mittareilla pystyttiin vain säätämään virtauksen arvoa ja mittaamaan sitä. Modbus-mittareita voitaisiin ohjata laajemmin ja mittaustietoja voitaisiin tuoda useita. Mittareita tulisi käyttää laboratorioissa, jossa voitaisiin saada suuri hyöty monipuolisemmasta mittaustiedosta.</p> <p>Opinnäytetyö alkaa teoriaosuudella. Teoriaosuudessa selitetään Modbus-kommunikoinnin perusteet ja esitellään yleisimmät Modbus-protokollat ja niiden ominaisuudet. Sen lisäksi esitellään myös standardit, jotka määrittelevät Modbus-laitteiden yhdistämisen toisiinsa. Lopuksi esitellään käytännön toteutuksessa käytetyt laitteet ja ohjelmistot.</p> <p>Käytännön osuudessa esitellään vaiheittain projektin eteneminen. Pienen skaalan testausvaiheesta edetään käytännön toteutukseen tehtaalla. Esitellään laitteiston konfigurointi, ohjelma ja sen teko ja graafisten käyttöpaneelien luonti. Tuodaan esille testaamisessa kohdattuja ongelmia ja miten ne ratkaistiin.</p> <p>Viimeistelty projekti oli toimiva kokonaisuus. Laboratorioissa työskentelevät kemistit saivat merkittävää hyötyä laajemmasta mittaustiedosta. Myös laajemmat mittarien ohjausmahdollisuudet helpottivat laboratoriotyöskentelyä.</p>		

<b>Asiasanat</b> ABB, master, mittaustieto, Modbus, ohjelmointi, sarjaliikenne, slave, väylä,
--

## ABSTRACT

<b>Centria University of Applied Sciences</b>	<b>Date</b> November 2018	<b>Author</b> Henri Hihnala
<b>Degree programme</b> Automation Technology		
<b>Name of thesis</b> DIVERSIFYING MEASUREMENT DATA WITH MODBUS		
<b>Instructor</b> Hannu Ala-Pönttiö		<b>Pages</b> 32
<b>Supervisor</b> Petri Salo		
<p>This thesis is about Modbus, which is a communication protocol used for connecting automation devices based on Master–slave communication. The aim of this thesis was to find out what are the benefits of replacing analog flow meters with Modbus flow meters, since old analogue flow meters could only measure the flow value of liquids and only allowed controlling of the liquid flow value. The new flow meters would allow better controlling of the device and allow measurement of additional variables. The implementation part of this thesis was done for Freeport Cobalt Oy, which is a chemical factory located in Kokkola, and it included device configuration, programming, graphical designing, and testing.</p> <p>The thesis starts with theory explaining how Modbus communication works and what are the main differences between different Modbus communication protocols. Modbus communication is done with Modbus messages, which are in binary form. Messages contain different parts like address, function code, and data. The Master device receives commands from the program and sends them to the slave devices. Usually the master wants to read or write data. The slave devices execute commands and respond to the master with information. This section also includes an explanation of the different physical connection interfaces and introduces the hardware used in the implementation.</p> <p>The following part of the thesis tells about the actual implementation, starting from the testing phase and leading up to the final execution of the project at the factory. The programming of the project and the graphical designing are also explained.</p> <p>The final application worked the way Freeport Cobalt wanted to and the additional measurement data proved to be beneficial for the laboratory staff. Better controlling of the flow meters compared to the old ones made working with the flow meters easier and more effective.</p>		
<b>Key words</b> ABB, master, measurement data, Modbus, programming, serial, slave		

## KÄSITTEIDEN MÄÄRITTELY

<b>ADU</b>	Application Data Unit
<b>ASCII</b>	American Standard Code for Information Interchange
<b>Asynkroninen tiedonsiirto</b>	Tiedonsiirto, joka ei tapahdu reaaliajassa.
<b>CBM</b>	Control Builder M
<b>CMT</b>	Control Module Type
<b>CRC</b>	Cyclic Redundancy check
<b>Id</b>	Identifier eli tunnistus
<b>LRC</b>	Longitude redundancy check
<b>MBAP</b>	Modbus Application Protocol header
<b>PDU</b>	Protocol Data Unit
<b>POU</b>	Program Organization Unit
<b>RTU</b>	Remote Terminal Unit
<b>SFC</b>	Sekvenssiohjelmoinnin esitysmuoto
<b>ST</b>	Structured text

**TIIVISTELMÄ**  
**ABSTRACT**  
**KÄSITTEIDEN MÄÄRITTELY**  
**SISÄLLYS**

<b>1 JOHDANTO .....</b>	<b>1</b>
<b>2 MODBUS .....</b>	<b>2</b>
2.1 Modbus-protokollat .....	8
2.1.1 Modbus RTU .....	8
2.1.2 Modbus ASCII.....	8
2.1.3 Modbus TCP.....	9
2.2 Fyysiset liitännät.....	10
2.2.1 RS232.....	11
2.2.2 RS485.....	12
2.2.3 RS422 & RJ45 .....	12
2.3 Laitteet ja ohjelmistot .....	13
2.3.1 Virtausmittari Bronkhorst Mini Cori-Flow .....	14
2.3.2 Moxa MGate MB3480.....	14
2.3.3 ABB 800xA-järjestelmä .....	15
2.3.4 ABB CI867 .....	16
2.3.5 ABB-Ohjelmistot .....	17
<b>3 KÄYTÄNNÖN TOTEUTUS.....</b>	<b>21</b>
3.1 Freeport Cobalt Oy .....	21
3.2 Ohjelmointi .....	22
3.2.1 Ohjelma .....	23
3.2.2 Käyttöpaneeli.....	26
3.3 Testaus.....	28
<b>4 YHTEENVETO JA PARANNUSEHDOTUKSIA .....</b>	<b>31</b>
<b>LÄHTEET .....</b>	<b>33</b>
<b>LIITTEET</b>	
<b>KUVAT</b>	
KUVA 1. OSI-malli sarjaliikennemuotoisesta Modbussista .....	3
KUVIO 2. Modbus-paketti sarjaliikenteessä .....	5
KUVA 3. Modbus TCP-arkkitehtuuri.....	10
KUVA 4. 800xA-automaatiosysteemin hierarkia. ....	18
KUVA 5. Esimerkki CMT:n muuttujien nimen muodostuksesta .....	23
KUVA 6. Modbus connect & read.....	24
KUVA 7. Modbus write-palikka & pulssigeneraattori .....	25
KUVA 8. Faceplate normaalissa ja extended-tilassa .....	27
KUVA 9. Puulaus-välilehti .....	27
KUVA 10. Laskurin-välilehti .....	28
<b>TAULUKOT</b>	
TAULUKKO 1. Modbus osoitteet.....	5

TAULUKKO 2. Yleisimmät Modbus funktiokoodit.....6

TAULUKKO 3. Exception-koodit.....7

## 1 JOHDANTO

Tämä opinnäytetyö kertoo Modbus-väylästä ja siitä, miten se monipuolistaa mittaustietoa verrattuna perinteiseen analogiamittaukseen. Opinnäytetyön käytännön osuus toteutettiin Freeport Cobalt Oy:lle, joka halusi vaihtaa laboratoriossaan käytetyt analogiset virtausmittarinsa Modbussia käyttäviin mittareihin. Työssä haluttiin selvittää, mitä tällä muutoksella voidaan saavuttaa. Mikäli saavutetut hyödyt olisivat merkittäviä, uskallettaisiin muut laboratoriossa käytetyt mittarit päivittää.

Opinnäytetyön teoriaosuudessa keskitytään Modbussiin. Tekstissä avataan sen toimintaperiaatteet ja selitetään, miten kommunikointi Modbus-laitteiden välillä tapahtuu. Eri Modbus protokollat ja liitännästandarit esitellään, jotta lukija saisi selvyuden niiden eroista. Käytännönsuudessa esitellään käytetyt laitteet ja ohjelmistot, jotta lukija saisi ymmärryksen niiden roolista itse toteutuksessa.

Käytännönsuudessa selitetään, miten itse toteutus eteni vaihe vaiheelta. Testausvaiheesta erillisessä järjestelmässä ja pienemmässä skaalassa edetään itse tehtaalla tapahtuvaan suoritukseen. Tehtaalla toteutus tapahtui suoraan tehtaan järjestelmään. Lisäksi kerrotaan vastaan tulleista ongelmista ja siitä, miten ne ratkaistiin. Yhteenvedossa tutkaillaan toteutusta kokonaisuutena ja sitä kuinka onnistunut se oli. Esitetään myös kehitysehdotuksia, joilla toteutusta oltaisiin voitu viedä eteenpäin.

Lähteinä käytettiin sekä kirjallisuutta että verkkolähteitä. Tärkeimpiä lähteitä olivat Modbus-organisaation laatimat ohjeet ja suositukset Modbussille. Kirjallisista lähteistä tärkein oli teos nimeltään ”Industrial process automation systems – Design and implementation”. Sen kirjoittajina toimivat Bodh Mehta ja Yaramala Reddy.

## 2 MODBUS

Modbus sai alkunsa vuonna 1979, kun se esiteltiin markkinoille. Kehittäjänä toimi Modicon-niminen yritys. Se luotiin alun perin Modiconin omien ohjelmoitavien logiikoiden kommunikointiprotokollaksi. Modbus päätettiin muuttaa avoimeksi protokollaksi ja sen käytöstä ei peritty rojalteja. Tämä mahdollisti sen, että muut laitavalmistajat pystyivät käyttämään sitä vapaasti omissa laitteissaan. Avoimuuden ja rojaltivapauden takia Modbus levisi nopeasti valmistajien keskuudessa ja siitä muodostui yksi käytetyimmistä kommunikointiprotokollista. Avoimuus helpotti Modbussia käyttävien sovellusten ja laitteiden luomista. Modbus on kehittynyt vuosien aikana eteenpäin sen alkuperäisestä versiosta. Tekniikan kehityksen tuomat vaatimukset ja mahdollisuudet ovat kehittäneet Modbus protokollaa eteenpäin vuosien kuluessa. (Mehta & Reddy 2015.)

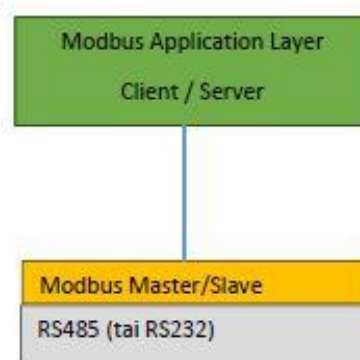
Modbus on sarjaliikennepohjainen kommunikointiprotokolla. Sen kantavana ideana on master–slave-kommunikointi. Järjestelmässä on yksi master ja yksi tai useita slaveja. Slavet erotetaan toisistaan väylällä Id:n avulla. Tämän vuoksi kahdella slavella ei voi olla samaa Id:tä. Yhdellä väylällä voi olla maksimissaan 247 slavea määritelmän mukaan, kun käytetään yhden tavun slave-Id:tä. Id:t väylällä ovat tällöin välillä 1–247. Tämä rajoitus voidaan kiertää käyttämällä kahden tavun Id:tä, jolloin slavejen määrä teoriassa kasvaa 65535:een asti. Kahden tavun Id:n käyttö edellyttää, että kaikki laitteet on konfiguroitu käyttämään kahden tavun Id:tä. Yhden tavun Id on vakio Modbus-organisaation määritelmien mukaan.

Modbussissa kommunikointi lähtee aina masterin aloitteesta. Se lähettää kyselyn väylälle, joka on osoitettu joko kaikille tai vain tietylle slavelle. Kun kysely on osoitettu kaikille väylällä oleville slaveille, on kyseessä broadcasting-kommunikointi. Siinä master lähettää viestin Id-osoitteeseen 0, jolla viesti lähetetään kaikille väylän slaveille. Broadcasting-kommunikoinnissa slavat eivät vastaa masterille missään vaiheessa. Broadcastingia käytetään silloin, kun halutaan kirjoittaa dataa kaikille väylän slaveille. Yleisempi master–slave-kommunikointitapa on nimeltään unicast. Siinä masterin kommunikointi kohdistuu aina tiettyyn slaveen. Unicast-kommunikoinnissa slave vastaa masterille. Slave toistaa ensin masterin lähettämän komennon, minkä jälkeen se toteuttaa masterin lähettämän komennon. Unicast on ehdottomasti yleisin Modbus-kommunikoinnin muoto. (Mehta & Reddy 2015; Modbus.org 2006c.)



Modbussia kokonaisuudessa voidaan kuvata hyvin käyttäen OSI-mallia (KUVA 1). OSI-mallin eli Open systems interconnection-mallin on luonut kansainvälinen standardointiorganisaation (ISO). Tällä mallilla voidaan kuvata kommunikointijärjestelmää kokonaisuudessa jakamalla se eri tasoihin. Tasot kuvaavat kommunikointia niin, että toiminnot kulkevat ylöspäin tasoilla. Alempi taso palvelee ylempää tasoa, joka taas palvelee sen yllä olevaa tasoa. Sarjaliikennemuotoinen Modbus käyttää tasoja application, data link ja physical. Application-taso on ylin taso, ja se käsittää modbus sovelluksien perustoiminnallisuuden. Sovellukset toimivat client–server-perusteella eli tutummin master–slave-periaatteella. Data link-taso on toiseksi alin taso, ja se kuvaa itse käytettyä protokollaa. Data link koostuu kahdesta osasta: master–slave-protokollasta ja datansiirtoon käytetystä protokollasta. Sarjaliikennettä käyttäessä datasiirto-protokolla on joko Modbus RTU tai ASCII. Physical layer on ensimmäinen eli alimmainen taso OSI-mallissa. Tällä tasolla kuvataan laitteiden yhdistämiseen käytettäviä liitäntästandardeja. Näitä ovat muun muassa RS485 ja RS232. (Mehta & Reddy 2015; Modbus.org 2006c.)

Taso	ISO/OSI Model	
7	Application	Modbus Application Protocol
6	Presentation	-
5	Session	-
4	Transport	-
3	Network	-
2	Data Link	Modbus Serial Line Protocol
1	Physical	RS485 (tai RS232)



KUVA 1. OSI-malli sarjaliikennemuotoisesta Modbussista (mukaillen Mehta & Reddy 2015, 329)

Modbus-väylällä laitteet eivät ole jatkuvasti samassa tilassa. Master- ja slave-laitteiden tilat ovat hyvin samankaltaisia, mutta niissä on joitain eroja. Käynnistyksessä sekä master- että slave-laitteet ovat idle-tilassa. Niille ei ole tässä vaiheessa vielä tullut pyyntöjä, joten ne ovat toimettomina. Kun master vastaanottaa ohjelman kautta käskyn, se lähettää sen eteenpäin slavelle tai slaveille. Jos kyseessä olisi broadcasting-kommunikointi, lähetettäisiin pyyntö kaikille slaveille. Broadcasting-tilassa master ei saa vastausta slaveilta. Master odottaa tällöin ennalta määritellyn ajan, että slavat kerkeävät prosessoida käskyn. Tämän jälkeen se palaa idle-tilaan. Unicastissa masterin tila muuttuu, ja se odottaa vastausta slavelta. Slave prosessoi saamansa käskyn, ja mikäli siinä ei ole mitään vikaa, se toteuttaa masterin käskyn. Tä-

män jälkeen slave vastaa masterin käskyyn ja palaa idle-tilaan. Master vastaanottaa vastauksen ja prosessoi sen. Mikäli vastaus on odotetunlainen, master palaa myös idle-tilaan. Silloin kun master ei ole idle-tilassa, se ei voi vastaanottaa uutta käskyä ohjelmalta. Mikäli jossain vaiheessa prosessia master tai slave havaitsee virheen, toiminta eroaa normaalista. Mikäli slave havaitsee virheen sille lähetetyssä käskyssä, se vastaa masterille virheilmoituksella. Mikäli master havaitsee virheen jossain vaiheessa, se prosessoi virhettä ja päättää mitä tehdään. Master havaitsee virheen, kun slave ei vastaa asetetun timeout-ajan sisällä tai kun slaven viesti on korruptoitunut. (Modbus.org 2006c.)

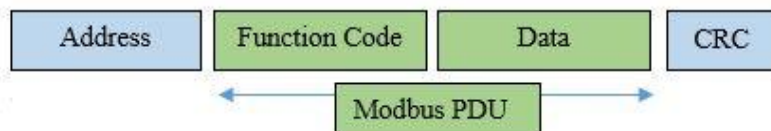
Modbussissa datan varastoiminen tapahtuu datarekistereihin. Modbus-laitteet sisältävät fyysisen muistin, joka on jaettu neljään osaan: Discrete output coils, discrete input contacts, analog input registers ja analog output holding registers. Fyysisessä muistissa nämä rekisteriosiot voivat olla eroteltuja muistin eri lohkoihin tai olla yhdessä lohkossa. Datan varastoinnin tapa on laitevalmistajan valittavissa, mutta data haun pitää toimia samalla tavalla muistin rakenteesta riippumatta. Jokaisessa rekisterinosiossa on 9999 rekisteriä arvojen säilyttämiseen. Rekisterit eivät ole kooltaan saman suuruisia. Discrete-rekisterit ovat kooltaan vain yhden bitin. Discrete-rekisterin data on niin sanottua on–off dataa, joten se voidaan esittää yhdellä bitillä. Analogipuolella rekisterit ovat kooltaan 16 bittiä eli kaksi tavua. Yksi tavu on siis kahdeksan bittiä. Analogi-rekisterit sisältävät esimerkiksi mittaustietoja, jotka vaativat huomattavasti enemmän tilaa. Analogipuolen rekistereissä data on esitetty yleensä niin, että painoarvoltaan merkittävämpi tavu on ensimmäisenä. Tätä tavujen lukutapaa kutsutaan myös nimellä big-endian. (Mehta & Reddy 2015; Modbus.org 2006c.)

Laitevalmistajat määrittelevät itse osoitteet, joihin eri datat varastoidaan. Kun dataa halutaan lukea tai kirjoittaa tietystä rekisteristä, kutsutaan sitä sen PDU-osoitteella. PDU-osoite on joko kokonaislukuna tai heksalukuna esitetty osoite, jossa data sijaitsee. Suurempikokoiset datat voivat olla jaettuna useaan rekisteriin, jolloin lukeminen pitää suorittaa niiltä rekistereiltä, joihin data on sijoitettu. Nämä datat sijaitsevat aina analogipuolella, jossa rekisterin koko oli suurempi. Useiden rekisterien kokoiset mittaustiedot saadaan yleensä esitettyä 32 bitin avulla eli kahdella rekisterillä. Usean rekisterin lukemisessa tulee ottaa huomioon, kumpi rekistereistä on painoarvoltaan merkittävämpi. Yleensä tässä noudatetaan samaa painoarvoa kuin normaalissa yhden rekisterin lukemisessa. Tällöin siis ensimmäinen rekisteri on painoarvoltaan merkittävämpi. (Mehta & Reddy 2015; Modbus.org 2006c.)

TAULUKKO 1. Modbus-osoitteet (mukaillen Mehta &amp; Reddy 2015, 330)

Rekisterinumero	Tyyppi	Rekisterin koko	Nimi
<b>1 – 9999</b>	Read/Write	1 bit	Discrete output coils
<b>10001 - 19999</b>	Read	1 bit	Discrete input contacts
<b>30001 - 39999</b>	Read	16 bit	Analog input registers
<b>40001 - 49999</b>	Read/Write	16 bit	Analog output holding registers

Modbus-kommunikaatio tapahtuu Modbus-paketteina. Paketin rakenne on esitelty kuvassa 2. Modbus-paketin ydin on Modbus PDU. Se koostuu kahdesta osasta, jotka ovat function code ja data. Function codella eli funktiokoodilla kerrotaan haluttu operaatio, yleisimmin kirjoitus tai luku. Funktiokoodi kertoo myös, mihin rekisterinosioon operaatio on kohdistettu. Funktiokoodi on ensimmäinen tavu PDU-viestistä. Data sisältää joko kirjoitettavan tai luettavan datan. Masterilta lähtiessä viesti sisältää myös PDU-osoitteen, johon funktio on kohdistettu. Slaven vastatessa se sisältää sen osoitteen datan, jonka master osoitti. Data on toinen tavu PDU-viestistä. Sarjaliikennemuotoisessa Modbus-kommunikoinnissa lisätään osia PDU-paketin ympärille. Nämä osat ovat osoite eli address ja CRC. Address osa tulee ennen Modbus PDU:ta. Siinä annetaan sen slaven osoite, johon halutaan olla yhteydessä. CRC tulee Modbus PDU:n jälkeen. Se on tarkoitettu pakettien tarkistukseen. ASCII käyttää omaa versiotaan CRC:stä, jota kutsutaan LRC:ksi. Kummatkin näistä ovat tarkistussummia, jotka laskevat addressin, funktiokoodin ja datan tavut yhteen. Tämä laskutoimitus tehdään, kun viesti lähtee masterilta slavelle. Slavella tämä lasku tehdään uudestaan. Jos slave ei saa samaa tarkistussummaa kuin master, viesti on korruptoitunut. Tällöin slave lähettää masterille virheilmoituksen. Modbus-viestejä ennen ja niiden jälkeen on vähintään 3,5 tavua tyhjää tilaa. Näillä tyhjillä väleillä väylä tunnistaa, missä viestien rajat menevät. Mikäli tyhjää tilaa on yli 1,5 tavua mutta alle 3,5 tavua, on viesti korruptoitunut ja se hylätään. (Mehta & Reddy 2015; Modbus.org 2006c.)



KUVA 2. Modbus-paketti sarjaliikenteessä (mukaillen Modbus.org 2006, 8)

Sarjamuotoisessa Modbus-kommunikoinnissa voidaan tarkastella myös viestien pariteettia. Tämä tarkoittaa sitä, että viestin tavujen pitää olla pariteetilta ennalta määritellyn kaltaisia. RTU:ssa käytössä olevat pariteetitilat ovat even eli parillinen, odd eli pariton ja no parity. Pariteetin avulla voidaan tarkistaa, että datan tavut eivät ole korruptoituneet matkalla laitteelta toiselle. Jokaisen lähetettävän tavun alussa on aloitusbitti. Tämän jälkeen seuraa kahdeksan bittiä, jotka sisältävät itse datan. Mikäli pariteettia käytetään, tulee datan jälkeen yksi pariteettibitti, jota seuraa lopetusbitti. Kun pariteetti ei ole käytössä, tarvitaan kaksi lopetusbittiä datan loppuun. Mikäli pariteetin tarkistus on käytössä, pitää kaikilla samassa järjestelmässä olevilla laitteilla olla sama pariteetin tarkastustila käytössä. Muuten laitteiden välinen kommunikointi ei onnistu datan ollessa bittimäärältään eroavaa, ja näin laitteet tulkitsevat datan virheelliseksi. Pariteetti toimii ASCII:ssa muuten samalla periaatteella, mutta kahdeksan databitin sijaan siinä on käytössä vain seitsemän. Pariteetti jätetään monesti pois käytöstä sekä RTU:ssa että ASCII:ssa, koska tarkistussumman avulla datan eheys voidaan varmentaa tarpeeksi luotettavasti. Tällä myös välteään mahdolliset virheet, mikäli laitteiden pariteetin tarkistus ei ole asetettu samaksi kaikissa laitteissa. Modbus-organisaatio suosittelee pariteetin käyttöä, mikäli mahdollista. (Modbus.org 2006c.)

Modbus-organisaatio on määritellyt tietyt funktiokoodit Modbus-laitteille, jotka on varattu tietyille funktioille. Nämä funktiokoodit ja niiden toiminnot on esitelty taulukossa 2. Valmistajat voivat kuitenkin lisätä myös omia funktiokoodejaan. Näillä funktiokoodeilla voidaan luoda laitekohtaisia funktioita, jotka toimivat valmistajan haluamalla tavalla. Käyttäjän itsemäärittelemät funktiokoodit sijoittuvat välille 65–72 ja 100–110. Näille alueille funktiokoodeja tehdessä tulee ottaa huomioon se, että nämä funktiokoodit toimivat vain laitteissa, joissa ne on määritetty. Mikäli näitä funktioita käytetään laitteissa, joissa niitä ei ole määritetty, vastaa laite virheilmoituksella. Funktiokoodit ovat kokonaislukuja väliltä 1–255. Tästä alueesta kuitenkin osa on varattu virheilmoituksia varten. Funktiokoodit voidaan lähettää myös heksalukuina, mutta kokonaislukuesitys on näistä huomattavasti yleisempi. (Mehta & Reddy 2015; Modbus.org 2006a.)

TAULUKKO 2. Yleisimmät Modbus funktiokoodit (mukaillen Mehta & Reddy 2015, 332)

Funktiokoodi	Funktio	Rekisteri
01	Read	Discrete output coils
05	Write Single	Discrete output coils
15	Write Multiple	Discrete output coils
02	Read	Discrete input contacts

(jatkuu)

TAULUKKO 2. (jatkuu).

<b>04</b>	Read	Analog input registers
<b>03</b>	Read	Analog output holding registers
<b>06</b>	Write Single	Analog output holding registers
<b>16</b>	Write Multiple	Analog output holding registers

Mikäli masterin lähettämä viesti ei välity ehjänä perille, slave vastaa masterille virheilmoituksella. Toinen mahdollinen virhe tapahtuu, jos slave ei tunne masterin lähettämää funktiokoodia. Virheilmoitus lähetetään myös silloin, jos kirjoitettavan tai luettavan datan osoite on väärä. Tämä tarkoittaa joko sitä, että osoitetta ei löydy muistista tai sitä, että yritetään lukea osaa datasta, joka on useassa rekisterissä. Näistä virheistä seurannutta virheilmoitusta kutsutaan exception-viestiksi. Slave vastaa masterin virheeliseen pyyntöön viestillä, joka koostuu masterin lähettämästä funktiokoodista ja exception-viestistä. Exception-viesti on lyhyt koodi, joka kertoo, mikä virhe on tapahtunut. Exception-koodeja on määritelty kahdeksan kappaletta. Ne on esitelty taulukossa 3. Jotkin laitevalmistajat ovat määritelleet omia exception-koodeja omien funktioidensa virheen määrittämiseen. (Modbus.org 2006a.)

TAULUKKO 3. Exception-koodit (mukaillen Modbus.org 2006a, 49)

<b>Koodi</b>	<b>Name</b>
<b>01</b>	Illegal function
<b>02</b>	Illegal data address
<b>03</b>	Illegal data value
<b>04</b>	Slave device failure
<b>05</b>	Acknowledge
<b>06</b>	Slave device busy
<b>08</b>	Memory Parity error
<b>0A</b>	Gateway path unavailable
<b>0B</b>	Gateway target device failed to respond

## 2.1 Modbus-protokollat

Vuosien varrella Modbussia on kehitetty eteenpäin automaatiotekniikan kehityksen mukana. Kehityksen myötä on luotu kokonaan uusia Modbus-protokollia, jotka ovat kehittäneet Modbussin perusidea eteenpäin. Tiedonsiirron kehitys sarjaliikenteestä verkkoliikenteeseen on vaikuttanut kenties kaikista eniten Modbussin kehitykseen. Ensimmäinen Modbus-protokolla, Modbus RTU, pohjautui sarjaliikennemuotoiseen kommunikointiin. RTU:n jatkokehityksenä syntyi ASCII, joka muutti datan muotoa helpommin ymmärrettäväksi operaattoreille. Kun teollisuus alkoi käyttämään sarjaliikenteen rinnalla verkkoliikennettä, syntyi Modbus TCP. TCP toi huomattavia parannuksia perinteiseen sarjaliikennettä käyttäviin protokolleihin. TCP toi ratkaisun perinteisten sarjaliikenneväylien heikkouksiin. Isompien datamäärien siirtäminen vaatii suurempia nopeuksia, joita verkkoliikenne pystyi tarjoamaan. Parannuksia saatiin myös datansiirron varmuuteen.

### 2.1.1 Modbus RTU

Modbus-väylän ensimmäinen versio luotiin aikanaan sarjaliikenneväylälle. Sarjaliikennemuotoinen datansiirto oli Modbussin julkistamisen aikaan ehdottomasti suosituin tapa siirtää dataa eri laitteiden välillä. Modbus RTU:ssa dataa siirretään binäärisessä muodossa kenttälaitteilta päätelaitteelle. Kaikkien laitteiden pitää toimia Modbus RTU-tilassa, jotta data voidaan käsitellä oikein. Data kulkee yhden tavun kokoisissa paketeissa. Näitä tavuja voi olla yhteensä 256, joka on maksimi koko Modbus RTU-viesteille. Yhdessä tavussa voidaan kuljettaa kaksi neljän bitin heksadesimaaliarvoa. Binäärisessä datansiirrossa data on pakattu tiiviisti, jolloin liikkuva datamäärä on hyvä suhteutettuna datansiirtonopeuteen. (Mehta & Reddy 2015; Modbus.org 2006c.)

### 2.1.2 Modbus ASCII

Modbus ASCII on toinen sarjaliikennettä käyttävistä Modbus-protokollista. Modbus ASCII syntyi RTU:n jatkokehityksenä. Toimintaperiaate on hyvin samanlainen kuin Modbus RTU:ssa, mutta data on binäärilukujen sijaan esitetty ASCII-kirjaimina. ASCII-kirjaimet ovat helpommin ymmärrettäviä operaattoreiden näkökulmasta kuin binääriluvut. Jotkin laitteet eivät pysty siirtämään haluttua dataa tarpeeksi nopeaa RTU-tilassa binäärisen data muodon vuoksi. Tällöin haluttua dataa ei saada siirrettyä kerralla, jolloin se jää epätäydelliseksi. Tällöin ASCII:n käyttäminen mahdollistaa tällaisten laitteiden

datansiirron onnistuneesti. Data siirretään ASCII-viesteinä osissa, jolloin RTU:n rajoitukset voidaan kiertää. Modbus ASCII:ta käyttävien laitteiden on pakko tukea myös RTU:ta, jonka päälle ASCII on luotu. ASCII:n heikkous verrattuna RTU:hun on se, että yhdellä tavunkokoisella viestillä ei voida viedä kuin kaksi ASCII-kirjainta. Dataa liikkuu huomattavasti vähemmän per viesti verrattuna RTU:hun samalla tiedonsiirtonopeudella. ASCII käyttää tarkistussummana CRC:n sijasta LRC:tä. Se toimii samalla periaatteella kuin CRC, mutta yhteenlaskettava data on eri muodossa. (Mehta & Reddy 2015; Modbus.org 2006c.)

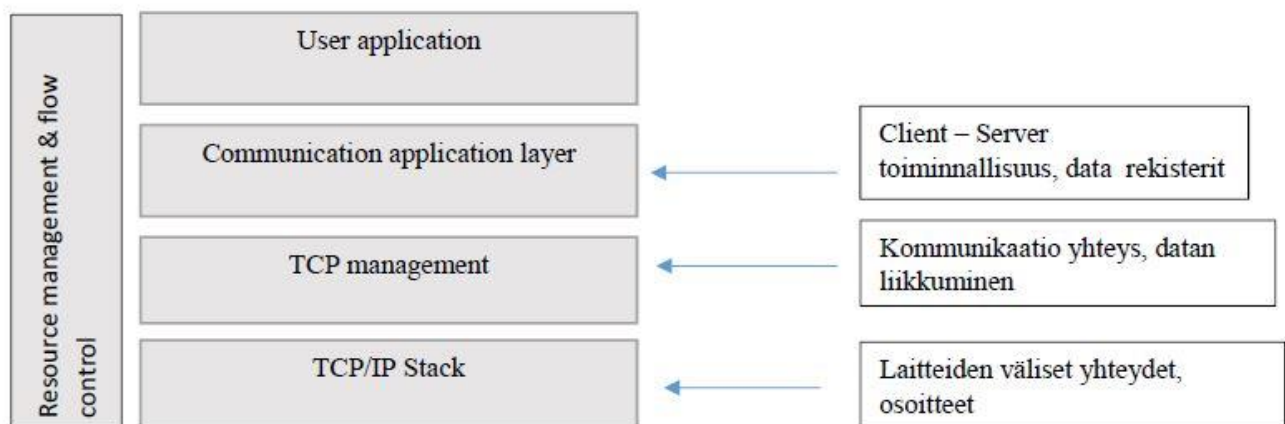
### 2.1.3 Modbus TCP

Erilaisten verkkoliikennettä käyttävien laitteiden yleistyttyä teollisuudessa havaittiin, että perinteistä sarjaliikennettä käyttävää Modbussia pitäisi päivittää. Tällöin luotiin Modbus TCP, jossa siirryttiin sarjaliikennemuotoisesta kommunikoinnista verkkoliikennepohjaiseen kommunikointiin. Modbus TCP:stä käytetään joskus myös nimitystä TCP/IP. Koska TCP:ssä verkon rakenne on samankaltainen normaalin lähiverkon kanssa, on siinäkin yleistynyt erilaisten verkkolaitteiden, kuten kytkimien tai protokollamuuntimien käyttö. Kytkimien avulla voidaan luoda järjestelmään eri aliverkkoja, joissa laitteet toimivat itsenäisesti riippumatta toisistaan. Tämä mahdollistaa sen, että TCP-väylään voidaan yhdistää huomattavasti enemmän laitteita kuin perinteisiin sarjaliikennettä käyttäviin Modbus-väyliin. Eri verkoissa oleviin laitteisiin saadaan yhteys helposti IP-osoitteiden avulla. Master lähettää sille kytkimelle viestin, jonka alla haluttu slave-laite on. Kytkin ohjaa viestin perusteella paketin oikeaan aliverkkoon, jossa haluttu laite sijaitsee. Aliverkossa kyseinen slave vastaanottaa kyselyn ja vastaa siihen. Protokollamuuntimen avulla taas voidaan yhdistää sarjaliikennettä käyttävät Modbus RTU- ja ASCII-laitteet Modbus TCP-verkkoon. Helpon liitettävyyden vuoksi myös sarjaliikenneprotokollaa käyttäviä laitteita valmistetaan yhä tänä päivänä. (Modbus.org 2006b.)

Modbus TCP-viestin ydinrakenne on samankaltainen verrattuna sarjaliikenneprotokollien viestiin. Perinteiseen PDU-viestiin lisätään Modbus TCP:n oma ADU. ADU tulee perinteisen PDU-viestin eteen ja sitä sanotaankin joskus MBAP-headeriksi. TCP:ssä ei käytetä sarjaliikenteessä käytettyä tarkistussummaa CRC:tä, koska TCP:ssä on sisäänrakennettuna datan tarkistusta useilla tasoilla. MBAP koostuu neljästä osasta, jotka ovat Transaction Identifier, Protocol Identifier, length ja Unit identifier. Transaction identifier on viestin yksilöllinen Id, jonka avulla se erotetaan muista viesteistä. Protocol identifier on yleensä aina 0, joka tarkoittaa Modbus-protokollaa. Length eli pituus kertoo seuraavien tavujen lukumäärän. Unit identifier on slaven osoite, johon master haluaa ottaa yhteyttä. TCP-kommunikointi käyttää

porttia 502, jota kutsutaan kuunteluportiksi. Se pitää olla avattuna ja varattuna vain TCP-kommunikointia varten. (Modbus.org 2006b.)

Modbus TCP:n toimintaa voidaan kuvata jakamalla se osiin kuvan 3 mukaan. Ylimmäisenä esitellään User application, joka on Modbus TCP:tä käyttävä sovellus. Sovellus on ladattuna prosessoriin, jossa sitä suoritetaan. Sovelluksen sisältä tulevat komennot operaatioista, joita halutaan suorittaa Modbus-laitteilla. Seuraava taso on nimeltään Communication application layer. Tämä taso kuvaa Modbussin perus client–server-toiminnallisuutta ja datan varastoimista eri rekistereihin. Seuraava taso on TCP management layer. Tämä taso kuvaa itse yhteyden muodostamista client–server-laitteiden välillä ja datan käsittelyä laitteilla. Alin taso on TCP/IP stack, joka kuvaa laitteiden välistä yhteyttä verkon tasolla. Näitä kaikkia tasoja tukee Resource management ja data flow control-taso. Se yhdistää kaikkien tasojen toiminnallisuudet toisiinsa mahdollistamalla datan liikkumisen tasojen välillä. Tämän tason avulla erotellaan myös datan liikkuminen eri suuntiin clientin ja serverin näkökulmasta. (Modbus.org 2006b.)



KUVA 3. Modbus TCP-arkkitehtuuri (mukaillen Modbus.org 2006b, 7)

## 2.2 Fyysiset liitännät

Modbus-väylä tukee useita eri liitännästandardeja. Nämä liitännät ovat standardein määritellyjä tapoja kytkeä Modbus-laitteita toisiinsa. Eri liitännät eroavat toisistaan ominaisuuksiltaan, ja käytettävä liitännä tapa on aina valittava tarpeiden mukaan. Modbus-organisaatio on määritellyt neljä kappaletta suositeltuja liitännästandardeja. Modbus-organisaatio suosittelee, että uusissa sovelluksissa käytettäisiin RS485-standardia kahden johdon konfiguraatiossa. Tämä johtuu siitä, että oletuskonfiguraatio Modbus-



laitteille on kolmen johdon rinnan kytkentä. Tässä kytkennässä on johdot D1, D0 ja maa. D1:tä kutsutaan myös 1-signaaliksi ja D0:aa 0-signaaliksi. Data liikkuu binäärimuodossa, joka esitetään ykkösinä ja nol-lina. Nollat ilmaistaan positiivisella jännitteellä ja ykköset negatiivisella. Tämä jännite saadaan linjojen D1 ja D0 välisestä jännitteen erotuksesta. Kaikki väylän laitteet on yhdistetty samaan kolmen johdon väylään. Fyysiset liitäntästandardit sijoittuvat OSI-mallissa ensimmäiselle tasolle eli physical layeriin. RS-kirjainyhdistelmä tulee sanoista ”recommended standard”. Muita liitäntästandardeja ja konfiguraatioita voidaan tarvittaessa käyttää, mutta suosituksien seuraaminen helpottaa eri laitevalmistajien välisen laitteiden yhteensopivuutta. Modbussin baud rate eli tiedonsiirtonopeus on oletusarvoisesti 19200 bittiä sekunnissa. Modbus-organisaatio vaatii, että tämä on asetettuna oletusarvoksi, joskin muitakin baud rateja voidaan halutessa käyttää. (Mehta & Reddy 2015; Modbus.org 2006c.)

### 2.2.1 RS232

RS232 on yksi käytetyimmistä sarjaliikenteen liitäntästandardeista. Electronic Industry Association (EIA) ja Telecommunications Industry Association (TIA) kehittivät RS232-standardin vuonna 1969. Se suunniteltiin alun perin modeemien ja päätelaitteiden väliseen binääriseen datansiirtoon. RS232:ssa binäärinen data siirretään yhdessä johtimessa, jossa data esitetään negatiivisena ja positiivisena jännitteenä. Tällä tavalla erotellaan johtimessa positiiviset ja negatiiviset bitit. Tätä yhden johtimen datansiirtoa kutsutaan epätasapainoiseksi datansiirroksi. Standardissa ei määritelty tiettyä liitintä, jota tulisi käyttää RS232:n kanssa. Yleisimmäksi liittimeksi muodostuivat D-liitimet, joista alkuun käytettiin DB25:stä ja myöhemmin DB9:ää. DB25 oli Modbussin julkistamisen aikaan yksi käytetyimmistä liittimistä sarjaliikennetekniikassa. DB25:stä siirryttiin DB9:ään, koska yhdeksän signaalin avulla voidaan saavuttaa asynkroninen tiedonsiirto. (Mehta & Reddy 2015; Modbus.org 2006c.)

RS232-kaapelin suositeltu maksimipituus on noin 25 metriä. Tämä pituus on teollisuuden tarpeisiin melko lyhyt, mikä estää sen käyttöä pitemmillä matkoilla. Teollisuudessa kenttälaitteiden ja ohjaavien laitteiden välit ovat usein satoja metrejä, mihin RS232 ei riitä. RS232:ta kannattaakin käyttää, kun halutaan kytkeä laitteita suoraan toisiinsa tai kun matkat laitteiden välillä ovat todella lyhyitä. RS232 kommunikoi full duplexina, eli se pystyy samaan aikaan sekä lähettämään että vastaanottamaan dataa. Tämä mahdollistaa nopean datansiirron laitteiden välillä, sillä data pystyy kulkemaan yhtä aikaa molempiin suuntiin. (Mehta & Reddy 2015; Modbus.org 2006c.)

### 2.2.2 RS485

RS232:n suurin heikkous on lyhyet datansiirtoetäisyydet, joihin väylä kykenee. RS485 luotiin, jotta voitaisiin viedä kenttälaitteita pitemmän matkan päähän ohjausyksiköstä ilman signaalihäviötä. RS485-standardi määriteltiin EIA:n toimesta, ja sitä kutsutaankin monesti EIA-485:ksi. RS485:tä pystytään käyttämään pitkillä etäisyyksillä, eikä sille ole määritetty tiettyä maksimipituutta. Kuitenkin 1200 metrin jälkeen signaalin vahvuus alkaa olemaan liian heikkoa. Linjan maksimipituus riippuu käytännössä käytetystä tiedonsiirtonopeudesta väylällä sekä väylällä olevien laitteiden määrästä. Mitä suurempi tiedonsiirtonopeus, sitä lyhyempi linjan maksimipituus käytännössä on. Väylällä olevat laitteet aiheuttavat vastusta väylälle, jolloin signaali heikkenee jokaista laitetta kohden. Signaalin vahvistamiseen on mahdollista käyttää vahvistinta, mutta sen käyttö on usein hankalaa ja kannattamatonta saatuun hyötyyn nähden. (Mehta & Reddy 2015.)

RS485:ssä binäärinen data kulkee kahta erillistä johdinta pitkin. Binäärinen data esitetään kahden johtimen välisenä jännite-erona, jolla ilmaistaan eri bittien tilat. Tätä datansiirtotapaa kutsutaan tasapainotetuksi datansiirroksi. Tämä signaalien erottelu on yksi päätekijä pitempien datansiirtomatkojen takana. Määritelmän mukainen kahden johdon RS485 väylä toimii half dublexina. Tämä tarkoittaa, että väylä ei voi yhtä aikaa sekä lähettää että vastaanottaa dataa. Vain yksi operaatio on sallittu kerrallaan. RS485-väylä voidaan myös konfiguroida käyttämään neljää johdinta, jolloin se toimii full dublexina. Neljän johdon konfiguraatio on harvinaisempi, ja huomattavasti yleisemmin käytössä on kahden johdon RS485-väyliä. RS485-väylään voidaan kytkeä teoriassa 256 laitetta, mutta laitteiden väylään aiheuttaman vastuksen vuoksi todellinen määrä on noin 32 laitetta per väylä. Nämä laitteet voivat olla sekä lähettäjiä että vastaanottajia. Ei ole rajaa, kuinka monta lähetintä tai kuinka monta vastaanotinta väylässä on. (Mehta & Reddy 2015.)

### 2.2.3 RS422 & RJ45

RS422 tai EIA-422 on todella samankaltainen liitännästandardi RS485:een verrattuna. Suurin ero verrattuna RS485:een tulee kuitenkin laitemäärästä ja siitä, minkälaisia laitteita väylään voidaan kytkeä. RS422 ei tue kuin yhtä lähettäjiä eli masteria kerrallaan. Vastaanottajia eli slaveja voi olla useita. Laitteita RS422:een voidaan kytkeä maksimissaan kymmenen. RS422 tukee sekä half- että full-dublexia. Kaapelipituudet ovat samaa luokkaa kuin RS485:ssä. Datansiirto tapahtuu RS485:n tapaan kahdella joh-

timella half duplexina. Pienemmän laitemäärän vuoksi RS422 on parhaimmillaan pienemmissä sovel-luksissa, joissa perinteisen RS232-väylän rajat tulisivat vastaan. RS485:een verrattuna, RS422 on jäänyt todella vähälle käytölle varsinkin teollisuudessa. RS422:n maksimilaitemäärä on yleensä suurin este sen käytölle. (Mehta & Reddy 2015.)

RJ45-liitäntä on tuttu monista verkkojen rakenteista. Normaalit lähiverkot ja tietoliikenneverkot ovat hyviä esimerkkejä RJ45:n käytöstä. RJ45:n käyttö Modbus-liitäntänä vakiintui Modbus TCP:n myötä. Standardin mukainen RJ45-liitäntä käsittää kahdeksan signaalia. Se on full duplexinen dataväylä, eli sillä pystytään lähettämään ja vastaanottamaan dataa yhtä aikaa. RJ45:ssä on neljä signaalia datansiirtoon. Sekä lähettämiseen että vastaanottamiseen on kaksi johdinta, joissa binäärinen data esitetään jän-nite-erona linjojen välillä. Näitä kahden johtimen pareja kutsutaan RX:ksi eli vastaanottopinniksi ja TX:ksi eli lähetyspinniksi. RX tulee sanasta receive ja TX sanasta transmit. RJ45-väylä ei ole yhtä al-tis signaalin heikkenemiselle kuin RS485-väylä. RJ45-väylässä voidaan myös saavuttaa suuremmat datansiirtonopeudet verrattuna perinteisempiin Modbus-linjoihin. RJ45 on myös hyvin häiriösuojattu, mikä parantaa datansiirron varmuutta. (Bronkhorst 2017; Mullins 2018.)

### 2.3 Laitteet ja ohjelmistot

Laitteiden hankkimisesta ja valinnasta vastasi Freeport Cobalt. Tilattujen laitteiden valmistajat olivat heille jo entuudestaan tuttuja, ja valmistajien laitteita oli jo käytössä eri puolilla tehdasta. Itse mittarin lisäksi tärkeimpiä laitteita olivat Modbus RTU / TCP-muunnin ja ABB:n järjestelmään liitetty Modbus TCP-kommunikointikortti. Näiden lisäksi tarvittiin johtoja, liittimiä ja muita asennustarvikkeita. Lait-teet oli tilattu kommunikointikorttia lukuun ottamatta ennen aloituspalaveria. Kommunikointikortin saapumisessa kesti noin puolitoista kuukautta, mikä hidasti projektin aloitusta. Kommunikointikortti oli linkki automaatiojärjestelmän ja mittarin välillä, joten ilman sitä ei voitu aloittaa testaamista. Vir-tausmittari oli valmistajan toimesta konfiguroitu Freeportin tarpeiden mukaiseksi. Laitteet ja käytetyt ohjelmistot esitellään seuraavissa osioissa tarkemmin.

### 2.3.1 Virtausmittari Bronkhorst Mini Cori-Flow

Bronkhorstin valmistama Mini Cori-Flow on nesteitä ja kaasuja varten kehitelty virtausmittari. Virtausmittarin mittaustekniikka perustuu coriolis-ilmiöön. Mittarin sisällä liikutetaan ohutta putkea edestakaisin. Nesteen tai kaasun virtaus aiheuttaa vastusta putken liikkeeseen, joka muuttaa putken edestakaisen liikkeen amplitudia. Amplitudin muutoksen avulla voidaan laskea muutokseen verrannollinen virtauksen arvo. Vain virtausmäärä vaikuttaa coriolis-ilmiön vahvuuteen, joten nesteen tai kaasun ominaisuudet ja niiden muutokset eivät vaikuta mittaukseen. Tämä mahdollistaa mittaukset eri lämpötiloissa ja tiheyksissä. Mittariin voidaan myös asentaa annostelupumppu. Tämän avulla mittarin läpi voidaan syöttää haluttu määrä nestettä. Mittari ohjaa pumppua ja seuraa läpi virrannutta nestemäärää. Kun haluttu nestemäärä on saavutettu, pumppu sulkee linjaston sisäisen venttiilin. (Bronkhorst 2015; Kallio, Mäkinen & Tantarimäki 2009, 189.)

Käytetty Cori-Flow-mittari oli rakennettu käyttämään Modbussia. Mittari ei eronnut normaalista Cori-Flow-mittarista ulkoisesti, mutta siinä oli yksi portti lisää Modbus-kaapelia varten. Kaapelina käytettiin viisijohtimista kaapelia. Näistä käytössä oli vain kolme, koska käytettiin RS485:tä kahden johdon konfiguraatiossa. Mittariin kytketty johdonpää käytti M12-liitintä viiden nastan konfiguraatiossa. Koska mittarissa käytettiin erillistä virtalähdettä, jätettiin M12-liittimen virransyöttöpinnit kytkemättä tuplajännitteen estämiseksi. Toinen pää johdosta oli kytkettynä Moxan muuntimeen. Datasignaalin laadun varmistamiseksi mittarin kanssa suositeltiin käytettäväksi terminointivastusta datalinjojen välillä. Tämän vastuksen arvo on  $120\Omega$ , joka on yksi yleisimmistä terminointivastuksen arvoista. Terminointi hoidettiin Moxan muuntimen sisäänrakennetun terminointivastuksen avulla. (Bronkhorst 2017.)

### 2.3.2 Moxa MGate MB3480

MGate MB3480 on Moxan valmistama sarjaväylä-Ethernet -muunnin. Sen avulla voidaan yhdistää eri Modbus-protokollia käyttäviä laitteita samaan järjestelmään. Muunnin muuntaa sarjaliikenneprotokollaa käyttävän Modbus RTU-datan Modbus TCP-dataksi. Päinvastainen muunnos tapahtuu, kun data liikkuu toiseen suuntaan. Fyysisen tason kytkennöissä muunnin mahdollistaa sarjaväylää käyttävien Modbus-laitteiden yhdistämisen Ethernetiä käyttäviin Modbus TCP-laitteisiin. Koska muunnin hoitaa Modbus-komentojen kääntämisen protokollien välillä, voidaan komennot lähettää suoraan Modbus TCP-viesteinä. Ohjelman näkökulmasta Modbus RTU-laitteet käyttäytyvät kuin ne olisivat Modbus

TCP-laitteita. Tämä helpottaa ohjelmointia, koska erillistä muunnosta protokollien välillä ei tarvitse tehdä. (Moxa Inc 2018.)

MB3480:ssa on neljä DB9-liittimellä varustettua sarjaväyläporttia. Se tukee RS232:ta, RS422:ta ja RS485:tä käyttäviä laitteita. RS485:tä käyttävillä laitteilla tuetaan sekä full duplex- että half duplex-datansiirtoa. Jokaisella sarjaliikenneportilla on mahdollista ottaa käyttöön terminointivastus, sekä hi-, että low pull-vastukset. Näitä vastuksia käytetään datalinjojen jännite-erojen ominaisuuksien muokkaukseen tarpeen mukaan. Tässä työssä käytettiin 120  $\Omega$ -terminointivastusta, koska mittarin manuaalissa suositeltiin sen käyttöä. Moxassa on yksi RJ45 Ethernet -portti, joka tukee sekä 10Mbps- että 100Mbps-nopeuksia. Tämä kytketään Modbus TCP-verkkoon. Muuntimen sarjaväylään voidaan kytkeä sekä RTU- että ASCII-protokollaa käyttäviä Modbus-laitteita. (Moxa Inc 2018.)

Moxan konfiguraatio voidaan tehdä joko Moxan valmistamalla MGate Manager-sovelluksella tai nettikonsolilla. Jotta voidaan käyttää nettikonsolia, pitää tietokoneen olla yhdistettynä samaan verkkoon muuntimen kanssa. Selaimella otetaan yhteys konfiguroitavan muuntimen IP-osoitteeseen. Moxan muuntimia on helppo konfiguroida ja konfiguraatiomahdollisuuksia on lukuisia. Yksi Moxan valmistamien muuntimien tärkeimmistä ominaisuuksista on mahdollisuus määritellä jokaiselle sarjaportille oma toimintatila. Näitä toimintatiloja ovat RTU Master, RTU Slave, ASCII Master ja ASCII Slave. Nämä tilat tarkoittavat, missä tilassa sarjaportteihin yhdistetyt laitteet toimivat suhteessa Ethernet -porttiin kytkettyyn laitteeseen. Tässä työssä mittarit olivat RTU Slaveja, joten jokainen portti oli konfiguroitu tähän tilaan. Muita tärkeitä konfiguroitavia parametreja ovat muuntimen IP-asetukset, sarjaporttien takana olevien slavejen Id:t ja slavejen timeout. Yhden sarjaportin taakse on mahdollista liittää useita slaveja, jolloin muunnin hoitaa niiden erottelun Moxan omalla sisäisellä Id:llä. Tässä työssä jokaiselle slavelle oli oma sarjaportti, ja Id:t menivät portin mukaan.

### 2.3.3 ABB 800xA-järjestelmä

800xA (extended automation) -järjestelmä on ABB:n kokonaisvaltainen automaatoratkaisu vaativiin teollisuuden tarpeisiin. 800xA-järjestelmän ytimessä ovat laitteet ja näitä varten luodut ohjelmistot. Laitteiston ytimessä on AC 800M-prosessorit ja niihin liitettävät laitteet. Näitä laitteita ovat muun muassa eri kommunikointikortit ja IO:t. Näiden laitteiden lisäksi valmistetaan sähköjakeluun keskittyneitä automaatiolaitteita, jotka voidaan myös liittää 800xA-järjestelmään. Perinteisten prosessoreiden ja kommunikointikorttien lisäksi ABB tarjoaa turva-automaatiolaitteita. Nämä kaikki voidaan liittää

ABB:n 800xA-järjestelmään, jolloin ohjaus ja datan keruu on keskitettynä yhteen paikkaan. 800xA-järjestelmän tarkoituksena onkin tarjota mahdollisimman monipuolinen automaatiojärjestelmä, joka kattaisi kaikki tehtaan automaatiotarpeet. 800xA-järjestelmän perustana on aspect objects-rakenne. Tehtaan laitteet tai isommat laitekokonaisuudet ovat aspect objecteja, ja niistä voidaan kerätä erilaista tietoa. Tämä tieto tuodaan eri aspecteihin, jotka keräävät tietynkaltaisen datan yhteen paikkaan. Näitä aspecteja ovat muun muassa grafiikat, ohjauslogiikat tai operointitiedot. Nämä tiedot ovat helposti järjestelmässä saatavilla ja jaoteltu selkeästi eri osa-alueisiin. (ABB 2018a.)

800xA-laitteisto käsittää muun muassa prosessorit, IO-kortit ja eri kommunikointikortit. Nämä laitteet on suunniteltu nykyaikaisia suuria tuotantolaitoksia varten, joissa laitteiden määrät voivat olla suuria. 800xA-järjestelmä on helposti skaalautuva. 800xA-järjestelmä panostaa laitteissaan helppoon huoltoon ja toimintavarmuuteen. Tämän mahdollistavat muun muassa laitteiden kahdennus ja itse laitteista löytyvät merkkivalot. Suurin osa laitteistosta koostuu kahdesta osasta: itse laitteesta ja sen pohjasta. Pohja sisältää laitteen ulkoiset portit, ja se kiinnitetään sähkökaapin kiskoon. Pohjaan kiinnitetään itse laite, joka sisältää suurimman osan elektroniikasta. Mikäli laite tai sen pohja menee rikki, ne voidaan korvata uusilla. Laitteet myös pystyvät ilmoittamaan mahdollisista ongelmista suoraan ohjelman kautta, jolloin mahdolliset laiterikot voidaan välttää ennakoimalla. Laitteiden hotswap eli rikkinäisten laitteiden korvaaminen ehjillä järjestelmän ollessa päällä on myös yksi järjestelmän eduista. Tämä vaatii kahdennuksen, jolloin toinen kahdennetuista laitteista hajoaa. Laiterikon sattuessa järjestelmä ei kaadu, vaan se vaihtaa toiminnan rikkoutuneen laitteen ehjälle parille. Hajonnut laite voidaan vaihtaa järjestelmän pyöriessä ja virtojen ollessa päällä turvallisesti. (ABB 2018a.)

### **2.3.4 ABB CI867**

CI867 on ABB:n valmistama kommunikointikortti Modbus TCP -protokollalle. Se liitetään ABB:n prosessoriin CEX-väylään. Se mahdollistaa järjestelmän kommunikoinnin Ethernet -porttien kautta Modbus-laitteiden kanssa. Siinä on kaksi RJ45 Ethernet -porttia, joista ensimmäinen portti tukee 100Mbps:n nopeutta ja full duplexia. Toinen portti on nopeudeltaan 10Mbps, ja se tukee vain half duplexia. Kortti tukee kahdennusta. Kahdennuksen ansiosta kommunikointijärjestelmään kytkettyjen Modbus-laitteiden kanssa ei katkea, mikäli toinen kommunikointikorteista hajoaa. (ABB 2018b.)

CI867 liitetään fyysisesti prosessoriin CEX-väylälle, minkä jälkeen se määritellään CBM-projektissa laitepuolelle. Kommunikointikorttia varten pitää projektiin lisätä kaksi kirjastoa: CI867TCPhwLib ja

ModbusTCPCommLib. Ensimmäinen mainittu kirjasto lisätään hardware-kirjastoon projektissa, ja se tuo projektiin CI867-kortin laitemääritelmät. Näitä määritelmiä ovat itse kortti, siihen kytkettävät TCP-slavet, gateway-protokollamuunnin ja serial-slavet. Laitteiden lisääminen projektiin vaatii parametrien määrittelyn. Näitä parametreja ovat esimerkiksi IP-osoitteet ja slaveihin liittyvät parametrit, kuten timeout. Toinen kirjasto on itse ohjelmaa varten, ja se sisältää Modbus kommunikointiin tarvittavat palikat. Näitä palikoita ovat muun muassa connect-, write-, ja read – palikat. Näillä palikoilla otetaan yhteys laitteeseen ja tehdään operaatioita, kuten kirjoitus ja luku.

### 2.3.5 ABB-Ohjelmistot

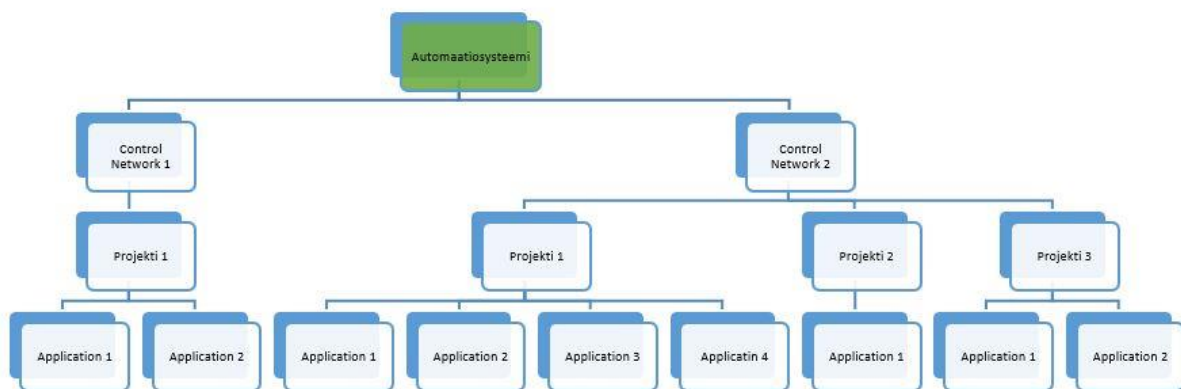
Työssä käytetyt ABB:n ohjelmistot keskittyivät 800xA-automaatiojärjestelmää varten luoduille ohjelmistoille. ABB tarjoaa muitakin ohjelmistoja, jotka on räätälöity esimerkiksi vanhemmille Advant-ohjausjärjestelmille. 800xA-ohjelmistoista tärkeimmät ovat ohjauslogiikan luontia varten suunniteltu Control Builder M. Tästä ohjelmasta on myös versio vanhemmille laitteille, nimeltään Compact Control Builder. Sitä käytetään myös pienemmissä kokonaisuuksissa, joissa ei tarvita kaikkia CBM-ominaisuuksia. Toinen tärkeimmistä ohjelmista oli Workplace, jota käytettiin eri rakenteiden selaamiseen ja niiden muokkaamiseen. Ohjelmoinnissa käytettiin Engineerin Workplacea, joka on ohjelmoijia varten luotu näkymä Workplacesta. Kaikkien grafiikkojen rakentamiseen käytettiin ABB:n Graphics Builderia. Seuraavissa alaluvuissa kerrotaan näistä ohjelmista hieman tarkemmin.

#### 2.3.5.1 Control Builder M

Control Builder M on ABB:n ohjausjärjestelmien ohjelmointityökalu. Ohjelmointi tapahtuu projekteihin, joissa yhdistyvät ohjelmat ja laitemääritykset. CBM pohjautuu pitkälti kirjastopohjaiseen ohjelmointiin. Kirjastot sisältävät valmiita ohjauslogiikoita eri objekteille, kuten esimerkiksi moottoreille. Tämä vähentää ohjelmoinnin tarvetta, koska samanlaisille objekteille ei tarvitse kerta toisensa jälkeen luoda uudestaan samankaltaisia ohjauslogiikoita. Jokainen CBM-asennus sisältää peruskirjastot, jotka sisältävät perusrakennuspalikat projektien luomiseen. Ne sisältävät myös IEC-standardien mukaisia funktioita, kuten ajastimia ja pulssigeneraattoreita. Käyttäjä voi luoda myös omia kirjastojaan, joita voidaan käyttää myöhemmin myös muissa projekteissa. CBM tukee IEC61131-standardin mukaisia ohjelmointikieliä, jotka ovat function block diagram, structured text, instruction list, ladder diagram ja

sequential function chart. ABB:n järjestelmä tukee näiden lisäksi function diagrammeja ja control module diagrammeja. Nämä kaksi ohjelmointikieltä pohjautuvat enemmän graafiseen ohjelmointiin. CBM mahdollistaa myös projektien eri osa-alueiden hallinnan. Näihin kuuluvat muun muassa turvallisuus, käyttäjien roolit ja oikeuksien hallinta. Yksi CBM:n tärkeimmistä ominaisuuksista on monen ohjelmoijan yhtäaikainen työskentely. Tämän mahdollistaa CBM:ään sisäänrakennettu resurssienvarausmekanismi, joka estää päällekkäisten muokkauksien tekemisen. Kun ohjelmoija haluaa muokata sovelusta, kirjastoa, tai laitemäärittäjiä, hänen pitää varata haluamansa resurssit. Resurssit voivat olla varattuna yhdelle käyttäjälle kerrallaan. Resurssien varaaminen tekee kahden käyttäjän päällekkäisen muokkaamisen mahdottomaksi. Tämä varmistuskeino mahdollistaa helpon ja huolettoman yhtäaikaisen ohjelmoinnin useille ohjelmoijille. (ABB 2016a, 2016b, 2018a.)

Yhden CBM-projektin alle voidaan luoda jopa 1024 applikaatiota, joista jokaisen sisällä voi olla 64 ohjelmaa tai 128 diagrammia. Diagrammiohjelmointi on tiiviimpi kooltaan verrattuna perinteisiin sovelluksiin, joten niitä voidaan luoda huomattavasti enemmän. Logiikan ohjelmointi CBM:ssä tapahtuu joko applikaatioon, ohjelmaan, diagrammiin, control moduleen tai function blockiin. Kaikkia näitä kutsutaan myös POU:ksi, joka on esitetty IEC61131-3-standardissa. POU:n sisällä voidaan määritellä koodin tarvitsemat parametrit ja muuttujat. Nämä kaikki ovat projektin alla. Yleensä ohjelmointi tapahtuu applikaation alle muilla ohjelmointitavoilla. Tehtaan automaatio voi olla jaettuna eri control networkkeihin. Nämä control networkit voivat olla jaettu esimerkiksi tehtaan tuotantolinjojen mukaan. Control networkien alla ovat kaikki siihen tehtaan osan ohjaukseen ja laitteisiin liittyvät projektit. Kaikki control networkit ovat automaatiojärjestelmän alla, mikä on koko hierarkiassa ylimmäisenä. Kuva 4 näyttää mallin tästä rakenteesta. (ABB 2016b.)



KUVA 4. 800xA-automaatiojärjestelmän hierarkia



### 2.3.5.2 Workplace

Workplace on ABB:n työkalu automaatiojärjestelmien esittelemiseen ja selaamiseen eri käyttäjäryhmille. Workplace-sovelluksessa on monia eri näkymiä, jotka on tarkoitettu eri käyttötarkoituksia varten. Nämä näkymät koostuvat aspekteista, joita eri laitteilla tai kokonaisuuksilla on. Kun Workplace otetaan käyttöön, siihen on määritelty vakiona kolme workplace-näkymää: Operator workplace sekä yhden että kahden näytön versiot plant explorer workplacesta. Kaikki workplace-näkymät voidaan konfiguroida useamman näytön näkymiksi. Tämä mahdollistaa helpon konfiguraation isoillekin valvoimille. Operator workplace on nimensä mukaisesti tarkoitettu operaattoreille. Operator workplace on tarkoitettu vain tehtaan laitteiden operointiin, eikä sitä kautta voida tehdä muokkauksia ohjelmiin tai projekteihin.

Plant explorer on tehty ohjelmoijia varten, ja sen kautta muokkaukset projekteihin ja rakenteisiin ovat mahdollisia. Plant explorerissa voidaan selata koko järjestelmää näyttäen eri rakenteita eli structureja. Näissä structureissa voidaan selata tehtaan eri aspekteja. Rakenteita ovat muun muassa control structure eli ohjaus ja library structure eli kirjastot. Yksinkertaistaen voidaan sanoa, että ohjelmoijat tekevät omassa workplace-näkymässään muokkauksia. Muokkauksia ovat esimerkiksi erilaiset määrittelyt, kuten hälytykset, datan keruu ja grafiikkojen teko. Sieltä muokkaukset otetaan käyttöön, jolloin ne tulevat näkyviin operaattoreille. (ABB 2013.)

Operator workplacen kautta operaattorit pääsevät ohjaamaan tehtaan laitteita ohjausnäkyistä. Nämä näkymät on tehty käyttäen Graphics builder-sovellusta, jolla luodaan käyttöpaneelit näille näkymille. Käyttöpaneeleita on eri skaaloissa: kokonaiselle tuotantolinjalle, muutamalle laitteelle tai yksittäisille laitteille. Käyttöpaneelit luodaan yleensä kaikille ohjattaville laitteille. Pelkille mittauksille tehdään yleensä vain graafiset objektit, jotka esittävät mittauksen näytöllä graafisesti. Kun näytetään isoja kokonaisuuksia käyttöpaneelissa, seurataan yleensä tuotannon yleiskuvaa tietyllä ajanhetkellä. Kun otetaan muutaman laitteen näkymä esille, voidaan alkaa seuraamisen lisäksi myös ohjaaman laitteita. Eri näkymien välillä voidaan helposti liikkua joko näkymiin luoduilla navigointi painikkeilla tai erikseen määritellyllä navigointisarakkeella. (ABB 2013.)

Operator workplace tuo operaattorin näkyviin listat hälytyksistä ja tapahtumista. Hälytyslistoilla on esillä varoitukset ja hälytykset. Varoitukset kertovat, kun esimerkiksi tankin täytössä ylitetään hälytysraja. Tällöin operaattori tajuaa seurata tämän tankin toimintaa ylivuodon varalta. Hälytykset kertovat laitevioista, yhteyksien menetyksistä ja muista mahdollisista ongelmista, joita järjestelmä havaitsee.

Tapahtumalistoilta nähdään, miten seurattujen laitteiden tilat ovat muuttuneet. Esimerkiksi jos venttiili avataan, tapahtumalistalla näkyy, että venttiilin tila on muuttunut. Tapahtumalistaan määritellään mitä tietoja sinne tuodaan. Operaattoreiden käytössä ovat myös erilaiset trendi näytöt, jotka prosessille on määritetty. Trendeillä voidaan seurata sekä yksittäisiä, että useita muuttujia kerrallaan. Näistä trendeistä muodostetaan käyriä, joilla prosessiarvojen kehitystä voidaan seurata pitkältä ajanjaksolta. (ABB 2013.)

### 2.3.5.3 Graphics Builder

Graphics Builder on ABB:n grafiikan suunnitteluun kehittämä ohjelma. Sillä pystytään luomaan kaikkea tehtaan toimintakuvista pienempiin graafisiin objekteihin. Graafisten käyttöliittymien rakentaminen on helppoa, sillä valmiita graafisia elementtejä on paljon. Yleisimmille laitteille on luotu jo valmiita graafisia elementtejä, ja uusien luominen on helppoa. Varsinkin kirjastoissa oleville CMT:lle ja function blockeille on jo valmiiksi määritettyjä grafiikkaobjekteja. Uusia graafisia elementtejä luodessa ne voidaan tallentaa, jolloin niitä voidaan käyttää uudelleen muissa kuvissa. Graphics Builderilla voidaan luoda myös koodipohjaista grafiikkaa. Esimerkki tästä on näytössä oleva merkkivalo, joka palaa, kun jokin ehto on täyttynyt. Kun ehto on epätosi, merkkivalo ei ole päällä. (ABB 2018c.)

Graphics Builderilla näyttöjen määrittäminen eri laitteille on helppoa. Koska grafiikka on vektoripohjaista, on se helppo skaalata eri resoluutiota käyttäville näytöille. Graphics Builderilla on myös mahdollista kääntää muiden laitevalmistajien grafiikoita ABB:n järjestelmälle. Tämä grafiikkojen tuonti vaatii vain vähän määrittelyä tuonnin jälkeen, mikä nopeuttaa grafiikkojen luomista verrattuna niiden uudelleen luomiseen. Graphics Builder mahdollistaa myös grafiikan esikatselun muokatessa. Mikäli muokattavan grafiikan ohjelma on pyörimässä prosessorilla, voidaan muokatessa laittaa grafiikka live-tilaan. Tällöin voidaan esikatsella grafiikkaa oikeassa prosessiympäristössä. CBM-projektien keräämän tiedon tuominen näytöille on helppoa, sillä se voidaan tehdä suoraan Graphics Builderin-projektin selaustoiminnolla. Ohjelmapuolella muuttujien pitää olla grafiikalle näkyviä, jotta ne pystytään tuomaan grafiikkaan. (ABB 2018c.)

### 3 KÄYTÄNNÖN TOTEUTUS

Freeport Cobalt Oy halusi korvata laboratoriossaan käyttämänsä vanhat virtausmittarit. Näitä vanhempia mittareita ohjattiin analogisesti, ja ainoa mittaustulos mitä mittarilta saatiin, oli virtauksen arvo. Tämä arvo lähetettiin järjestelmään milliampeeriviestinä ja se piti skaalata vielä ohjelman puolella oikeaan skaalaan. Virtausmittaria piti kalibroida usein, ja tämä kalibrointi jouduttiin suorittamaan manuaalisesti. Modbussilla varustetut mittarit mahdollistaisivat useiden eri mittausten suorittamisen, ja mittarin laajemman ohjaamisen ohjelman kautta. Mittarin läpi ajetusta nesteestä haluttiin saada enemmän tietoa selville, jotta voitaisiin tehdä tarkempia laskelmia aineen ominaisuuksista. Selville haluttavia tietoja olivat esimerkiksi tiheys ja lämpötila.

Käytännön toteutus aloitettiin palaverilla, jossa käytiin läpi käytettävät laitteet ja mitä niillä haluttiin saavuttaa. Käytiin läpi jo olemassa olevat laitteet ja missä ne sijaitsivat. Käytiin läpi, miten uudet laitteet muuttaisivat nykyisen järjestelmän kokoonpanoa. Tutustuttiin laboratorioon, jonne itse mittarit asennettaisiin. Kun kaikki laitteet olivat saapuneet, aloitettiin itse projekti. Aluksi tutustuttiin käytettäviin laitteisiin, ja testailtiin näitä erillisessä järjestelmässä. Laitteiden ja ohjelman testaamisen jälkeen projekti eteni siihen pisteeseen, että siirryttiin tehtaalle luomaan lopullista toteutusta. Tehtaalla tapahtunut käytännön toteutus kesti kuusi täyttä työpäivää. Tehtaalla tapahtunut toteutus sisälsi ohjelman tekemisen, grafiikkojen luonnin, testaamisen ja käyttöohjeiden laatimisen.

#### 3.1 Freeport Cobalt Oy

Freeport Cobalt Oy on Kokkolan teollisuusalueella sijaitseva koboltin tuotantoon keskittynyt kemian alan tehdas. Freeport Cobalt Oy on amerikkalaisen Freeport-McMoran-konsernin omistama yritys. Freeportilla on tehtaan lisäksi myyntiyksiköitä eri mantereilla. Freeport Cobaltin tuotannon pääraaka-aineena käytetään koboltia, jota jalostetaan Kokkolassa eri käyttötarkoituksia varten. Raaka koboltti saadaan kaivoksilta ympäri maailmaa, josta se kuljetetaan Freeportille. Koboltia myös saadaan kiertämällä esimerkiksi akkuja tai muun teollisuuden kobolttipitoisia tuotannon sivutuotteita. Raaka-aineet ja valmiit tuotteet liikkuvat suurimmaksi osaksi meriteitse, Kokkolan teollisuusalueen sataman kautta. Freeport Cobalt on yksi maailman johtavista koboltin jalostajista. Jalostamisen lisäksi Freeport

Cobalt panostaa koboltin ja siihen liittyvien kemikaalien tutkimiseen sekä koboltti pohjaisten tuotteiden kehitykseen. Koboltista pyritään luomaan uusia yhdisteitä, joita voitaisiin käyttää laajemmin teollisuudessa. (Freeport Cobalt.)

Koboltin jalostus hoidetaan Freeport Cobaltilla raaka-aineesta lopputuotteeksi asti. Koboltti erotetaan malmista tai muusta kierrätysmateriaalista liuottamalla se kemikaaliseksi avulla erikseen muusta materiaalista. Liuottamisen jälkeen liuos puhdistetaan ylimääräisestä materiaalista, minkä jälkeen se siirtyy uuttoon. Siinä erotetaan koboltti kemiallisesti muusta nesteestä. Sen jälkeen puhdasta kobolttia aletaan jalostamaan eri käyttötarkoituksia varten. Tuotanto on jaettu osittain eri tehdasrakennuksiin, jotka ovat pitkälle automatisoituja. Tehdas työllistää noin 400 kemian, tekniikan ja näitä tukevien alojen ammattilaista. (Freeport Cobalt.)

Lopulliset tuotteet ovat kobolttipulvereita ja koboltti pohjaisia kemikaaleja. Näitä tuotteita käytetään muussa teollisuudessa raaka-aineina. Yksi tärkeimmistä käyttökohteista on akkuteollisuus, jossa käytettävät kobolttijalosteet ovat tärkeässä osassa. Koboltti on myös tärkeässä osassa muissa energiateollisuuden sovelluksissa. Kobolttia käytetään lisäksi muun muassa väriaineena ja metalliteollisuudessa sidosaineena. Tuotteet toimitetaan käyttökohteen mukaan pakattuna ympäri maailmaa teollisuuden tarpeisiin. (Freeport Cobalt.)

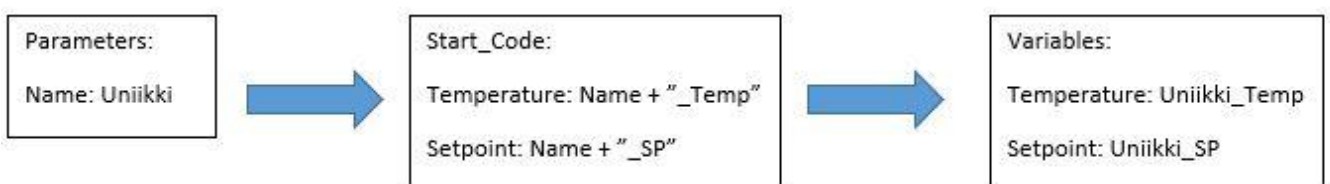
### 3.2 Ohjelmointi

Ohjelmointi tehtiin lähes kokonaan Control Builder M -ohjelmistolla. Käytännön toteutuksessa käyttöpaneeli- ja historiankeruuominaisuudet tehtiin Graphics builderilla ja Workplacella. Ohjelmasta luotiin testiversio erillisessä järjestelmässä ennen tehtaalla tapahtunutta ohjelmointia. Tähän testiohjelmaan luotiin samankaltaiset toiminnot, joita tehtaalla tulisi käyttää. Käyttöpaneeleita ei tähän testiversioon luotu, koska niiden luominen oli entuudestaan tuttua. Laitemääritykset tehtiin lähes samalla tavalla kuin tehtaalla. Skaala testiversiossa oli pienempi kuin käytännön toteutuksessa, ja siinä luettiin ja kirjoitettiin vain muutamia muuttujia. Kun oltiin saatu ohjelma toimimaan halutulla tavalla, oltiin valmiita siirtymään tehtaalle suorittamaan käytännön toteutusta. Tehtaalla ohjelmointi tapahtui atk- ja valvomohuoneessa. Ohjelmointi tehtiin suoraan tehtaan järjestelmään, jossa myös testaus tapahtui.

### 3.2.1 Ohjelma

Ohjelma luotiin Control module typenä. Type eli tyyppi tarkoittaa sitä, että tätä control modulea voidaan käyttää uudelleen samanlaisten mittareiden ohjaukseen. Kun CMT tuodaan applikaatioon, ei siihen tarvitse määritellä kuin parametreit. Parametrit ovat CMT:n ulkopuolelle näkyviä muuttujia, kun taas CMT:n sisällä käytettävät muuttujat ovat variaabeleita. CMT:n parametrit määrittelevät laitteelle uniikit ominaisuudet, kuten esimerkiksi nimen tai sijainnin laitteistossa. Uusilla mittareilla pitäisi vain määritellä uniikit parametrit, jolloin saavutetaan sama toiminallisuus ilman lisäohjelmointia. Modbus-lukemiseen ja -kirjoittamiseen liittynyt koodi tehtiin kokonaan ST:nä. Freeport oli valinnut sen standardiohjelmointikieleksi sovelluksissaan, joten sitä käytettiin. Suurin osa ohjelmasta tehtiin puhtaasti ST-koodina, mutta myös SFC:tä käytettiin eri toimintojen luomiseksi mittareille. CMT luotiin kirjaston sisään, joka mahdollisti sen käytön myös kokonaan muissa projekteissa.

Ohjelma koostui viidestä osasta. Ensimmäinen osa oli ST:nä tehty ohjelmaosio nimeltä "start\_code". Tämän ohjelmaosion funktiona oli käyttäjän CMT:n sisälle antamien parametrien perusteella muodostaa luodulle CMT:lle uniikit parametrit CMT:n sisällä oleviin variaabeleihin. Tämä tapahtui niin, että käyttäjä antaa laitteelle uniikin nimen CMT:n parametrit-osiossa. Annettu nimi yhdistetään eri muuttujille tehtyihin nimen päätteisiin. Kutsuttaessa tietyn nimisen CMT:n lämpötilamuuttujaa sitä kutsutaan kuvan 5 mukaisesti nimellä "nimi\_Temp". Kuvassa 5 on esitelty yksinkertaisuudessaan, miten nimen muodostaminen muuttujille tapahtuu parametreissa annetun nimen perusteella.



KUVA 5. Esimerkki CMT:n muuttujien nimen muodostuksesta

Toinen osio oli nimeltään "code", joka oli myöskin tehty ST:nä. Tässä osiossa esiteltiin kaikki Modbus-funktiot. Modbus-funktiot ovat funktiopalikoita, joita kutsutaan koodissa. Ennen tätä ne pitää ilmoittaa function block-välilehdellä, jossa annetaan palikan nimi ja tyyppi. Koodissa niitä kutsutaan niille annetuilla nimillä. Kaiken alkuna oli MBTCPCConnect-palikka, jonka funktio oli muodostaa yh-

teys haluttuun laitteeseen. MBTCPConnect-palikan sisälle annetaan parametrit, joilla se yhdistää haluttuun laitteeseen hardware-osiossa. MBTCPConnect enableoidaan enable-muuttujan avulla, joka on boolean muuttuja. Se on oletusarvoisesti päällä, joten järjestelmä yhdistää automaattisesti Modbus-laitteisiin käynnistyksessä. MBTCPConnectin jälkeen oli määritelty timer, jonka tehtävä oli odottaa ennalta määritelty aika yhteyden varmistamisesta tulevaa signaalia. Tällä estettiin se, että järjestelmä ei heti käynnistyksessä asettaisi MBTCPConnect-palikkaa virhetilaan. Virhe loisi tarpeettoman hälytyksen, joka ei ole suotavaa.

Tämän jälkeen oli MBTCPReadCyc-palikoiden vuoro. Nämä lukupalikat lukevat määritellyn syklin välein dataa yhdistetyltä laitteelta halutusta osoitteesta. Syklin pituudeksi valittiin toteutuksessa viisi sekuntia. Jokaiselle palikalle annettiin osoite, josta data luetaan, ja variaabeli, johon luettu data tuodaan. Näillä palikoilla oltaisiin myös voitu lukea useita peräkkäisiä rekistereitä, jos data olisi ollut sijoiteltuna näin. Tämän mittarin tapauksessa luettavat rekisterit olivat pitkin poikin muistia, jolloin ei pystytty lukemaan useita muuttujia samalla kertaa. MBTCPReadCyc-palikoiden yhteyteen oli luotu lukupalikoiden tilasta kertova muuttuja Statusbool. Jos jokin lukupalikka aiheuttaisi virheen, aktivoituisi statusbool-muuttuja. Tämä helpottaa virheen etsintää. Lukemisen jälkeen koodissa olivat MBTCPWrite-palikat. Näillä voitiin kirjoittaa haluttuun osoitteeseen dataa. Kirjoitettava data annettiin CMT:n sisäisessä variaabelissa. Kirjoituspalikat tarvitsivat kirjoituspyyntösignaalin kirjoituksen suorittamiseen. Kirjoituspyynnöt aktivoitiin pulssigeneraattorilla, jota ohjasi boolean muuttuja. Puls-sigeneraattorin aktivoituminen antoi MBTCPWrite-palikalle kirjoituspulssin kolmen sekunnin välein. Edellä mainitut funktiopalikat on esitelty kuvissa 6 ja 7.

```
(* Connection *)

MBTCPConnect( En_C := Enable,
              Channel := Channel,
              Partner := Partner,
              Id := Id );

TONConnectValidDelay( In := MBTCPConnect.En_C and MBTCPConnect.Valid,
                     PT := ConnectionValidDelayTime,
                     Q => ConnectionValid );

(* Read Data *)

MBTCPRead1( Id := Id,
            Enable := Enable,
            CycleTime := Time5s,
            StartAddr := MAddressRead1,
            Error => Error1,
            Status => Status1,
            Rd[1] := ReadData1 );

StatusBool1 := Status1 <> 1 OR NOT Enable;
```

KUVA 6. Modbus connect & read

```

(* Write Data *)
PulsGenWriteReq( Enable := Write_req,
                  PeriodTime := PeriodTimeReq,
                  PulseTime := PulseTimeReq );

MBTCPWrite1( Id := Id,
              Req := PulsGenWriteReq.Out AND (s_InitReset_64.X OR s_InitReset_0.X) ,
              StartAddr := MBAddressWrite1,
              Sd[1] := WriteData1 );

```

#### KUVA 7. Modbus write-palikka & pulssigeneraattori

Kolmas osio oli nimeltään nollaus, ja se oli ensimmäinen kolmesta sekvenssistä. Kaikki sekvenssit toteutettiin käyttäen SFC-ohjelmointikieltä. Nollauksessa mittari kalibroitiin linjastossa olevalle aineelle vallitsevissa olosuhteissa. Nollauksen toteutus ohjelmallisesti oli järkevintä toteuttaa sekvenssinä, sillä sen etenemistä olisi helppo seurata tarvittaessa. Nollaussekvenssi käynnistetään lähettämällä initreset muuttujalle arvo 64. Tämä sallii suojattujen muuttujien muokkaamisen. Muut nollaukseen tarvittavat muuttujat olivat control mode ja calibration mode. Control mode kertoo laitteen ohjauksen tilan, ja calibration mode on nimensä mukaisesti kalibrointitila. Calibration mode on suojattu muuttuja, jonka muuttaminen vaati initresetin kirjoittamisen. Seuraavaksi asetettiin control mode 9:ksi, minkä jälkeen calibration mode vaihdettiin ensin 0:ksi ja sitten 9:ksi. Tämä sekvenssissä tapahtunut operaatio käynnistää nollauksen. Nollauksen päätyttyä control mode vaihtuu takaisin normaaliin tilaan. Tämän jälkeen initresetillä pitää lukita suojattujen muuttujien muokkaaminen lähettämällä sille arvo 0.

Neljäs osio oli puulaus. Puulauksella avataan venttiili täysin auki linjaston huuhtelua varten. Tämä toiminnallisuus luotiin myös sekvenssinä, jotta sen seuraaminen olisi helppoa. Sekvenssi odottaa, että operaattori aktivoi käyttöpaneelista puulauksen. Start-painikkeen painamisen jälkeen puulaussekvenssi lähtee käyntiin. Puulauksessa vaihdetaan vain control moden tilaa. Sekvenssissä kirjoitetaan control modeen arvo 8, joka käynnistää puulaustilan. Puulaustila on niin kauan päällä, kunnes operaattori pysäyttää puulauksen stop-painikkeella. Kun stop-painike on aktivoitu, control mode muutetaan takaisin normaaliin tilaan. Kun tila on palannut takaisin normaaliksi, on sekvenssi palannut takaisin alkuun ja käynnistyspainike vapautuu. (Bronkhorst 2018.)

Viimeinen osio oli nimeltään batch. Batch eli nestemäärän laskuri toteutettiin myös sekvenssinä. Laskuritila käynnistetään käynnistyspainikkeella. Sekvenssin käynnistyksessä laskurille kirjoitetaan halutun nestemäärän arvo. Tämä arvo asetetaan käyttöpaneelista, johon operaattori antaa sen ennen lasku-

rin käynnistystä. Seuraavaksi laskurille annetaan käsky muuttaa virtauksen asetusarvon halutun ainemäärän saavutettua. Sen jälkeen lähetetään laskurille uusi asetusarvo, jonka laskuri ainemäärän saavutettua asettaa. Asetusarvo asetettiin sekvenssissä nollassi, jolloin venttiili sulkeutuisi ainemäärän saavutettuaan. Laskuri asetetaan seuraavassa sekvenssin askeleessa laskuritilaan. Sen jälkeen sekvenssi odottaa siihen asti, kunnes haluttu nestemäärä on saavutettu. Sen jälkeen määritetty asetusarvo asetetaan, mittari laitetaan automaattitilaan ja laskuri nollataan. (Bronkhorst 2018.)

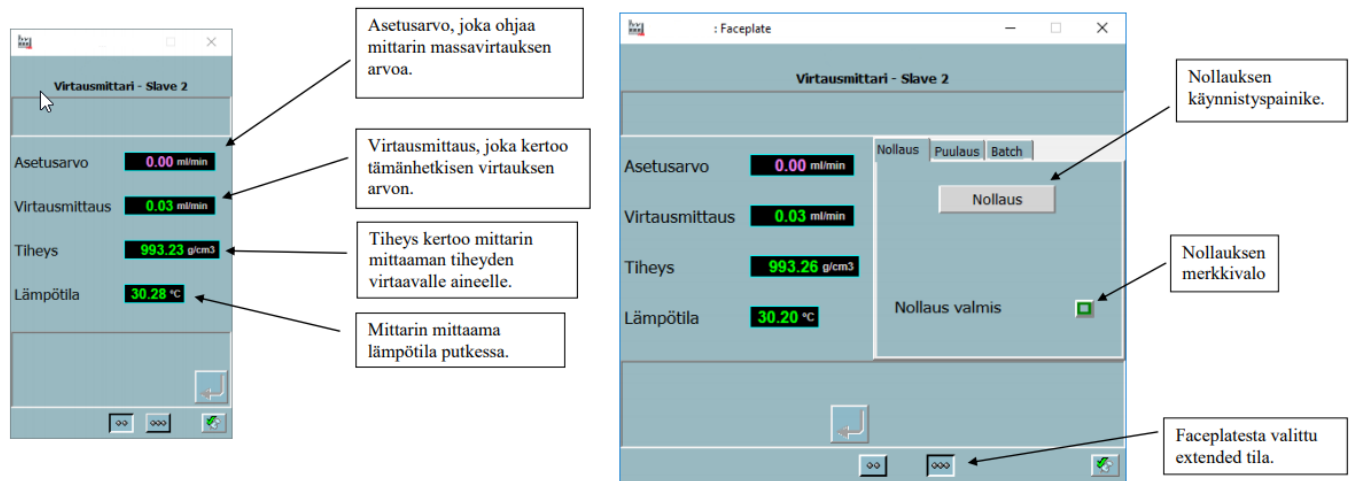
### 3.2.2 Käyttöpaneeli

Valmiin ohjelman pohjalta lähdettiin suunnittelemaan käyttöpaneelia operaattoreille. Käyttöpaneelit tehtiin ABB:n Graphics builder-ohjelmalla. Käyttöpaneelin tulisi tuoda ohjelman keräämät tiedot selkeästi esille, ja mahdollistaa mittarin ja sen toimintojen ohjaamisen paneelistä. Operaattorille esitettäisiin vain oleelliset mittaustiedot, ja operoinnin kannalta tarpeettomat tiedot jätettäisiin ohjelman sisäisiksi. Paneelistä haluttiin pystyä käyttämään virtaussäädön lisäksi mittarin nollausta, linjaston puulausta, ja nestemäärän laskuria. Itse laboratorion ohjausnäkymään luotiin graafinen objekti, joka näytti vain virtauksen arvon. Kun tätä objektiä painettiin, avautui itse mittarin operointipaneeli.

Käyttöpaneeliin luotiin kaksi erikokoista näkymää. Oletuksena aukeava paneeli on niin sanottu normaali paneeli. Siinä tuodaan esille seuraavat tiedot: virtauksen asetusarvo, virtausmittaus, tiheys ja lämpötila. Käyttöpaneeli voidaan suurentaa niin sanottuun laajennettuun tilaan, jolloin paneeliin tulee näkyviin muutkin ominaisuudet. Nämä ominaisuudet ovat nollaus, puulaus ja batch. Kuvassa 8 on esitelty operointipaneelit sekä normaalissa että laajennetussa tilassa.

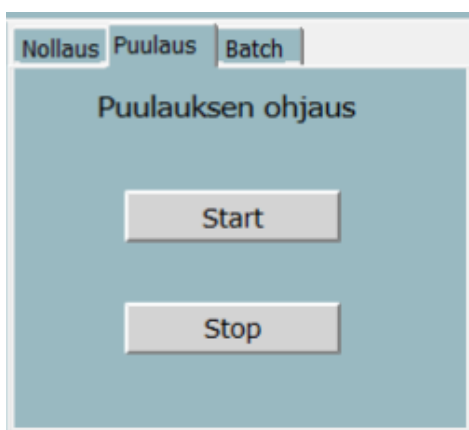
Nollaus-välilehdellä on mittarin nollaukseen liittyvät ohjaukset. Nollauksen käynnistyspainikkeella käynnistetään itse nollaus. Painike painuu pohjaan. Ennen nollausta pitää mittarin olla lämmennyt noin puolen tunnin ajan ja olla täytettynä prosessissa käytettävällä aineella. Virtaus pitää olla pysäytettynä, jotta kalibrointi onnistuu hyvällä tarkkuudella. Tämän jälkeen voidaan käynnistää nollaus, jonka aikana mittari kalibroi itsensä. Nollaus kestää noin 30s–180s riippuen nesteen tai kaasun ominaisuuksista. Kun mittari on suorittanut nollauksen, se palaa takaisin normaaliin tilaan. Välilehdellä oleva merkkivalo muuttuu vihreäksi sen merkiksi, että nollaus on suoritettu loppuun. Myös nollauspainike palaa takaisin ylös.





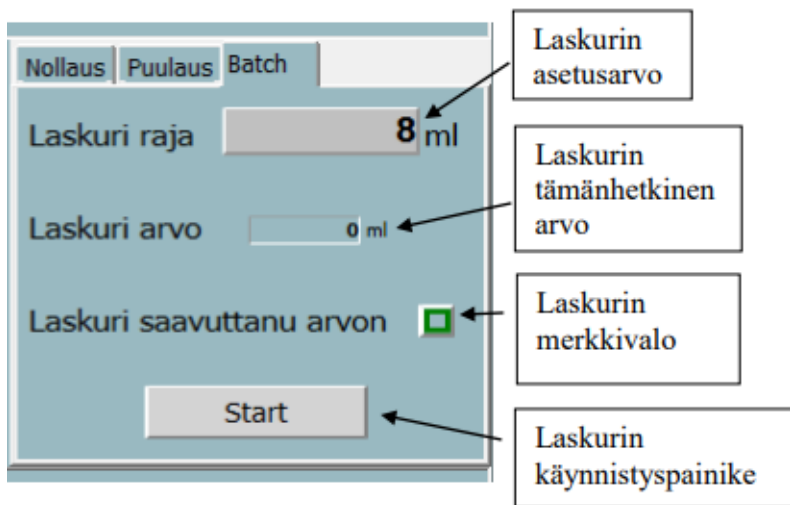
KUVA 8. Faceplate normaalissa ja extended-tilassa

Puulaus-välilehti on tarkoitettu puulaus-ominaisuuden käyttöä varten. Tämä on esitelty kuvassa 9. Puulausta käytetään ennen kalibrointia ja aineen vaihdon yhteydessä. Sitä käytetään myös puhdistettaessa linjastoa edellisen aineen jäämistä. Puulauksessa venttiili avataan täysin auki, ja virtauksen arvoa nostetaan mittarin asetettuun maksimiarvoon. Puulaus-välilehdellä on kaksi painiketta: käynnistys- ja pysäytyspainike. Käynnistyspainike aktivoi puulaus-ominaisuuden. Puulaus on niin kauan aktiivisena, kunnes puulaus-ominaisuus otetaan pois päältä pysäytyspainikkeella. Käynnistyspainike jää pohjaan, kun se on aktivoitu. Se on pohjassa niin kauan, kunnes pysäytyspainiketta painetaan ja tila on palautunut normaaliksi. Virtaus pysähtyy, ja mittari palaa takaisin normaaliin tilaan.



KUVA 9. Puulaus-välilehti

Batch-välilehti (KUVA 10) on tarkoitettu virtausmäärän laskuria varten. Tällä ominaisuudella voidaan ajaa mittarin läpi haluttu ainemäärä millilitroina. Ainemäärä annetaan laskurin rajakenttään millilitroina, minkä jälkeen voidaan käynnistää laskuri. Laskurin käynnistytyn jälkeen annetaan mittarin asetusarvoksi haluttu virtauksen arvo. Mittari nostaa virtauksen tähän arvoon ja alkaa laskea läpi virrannutta nestemäärää. Kun haluttu nestemäärä on saavutettu, mittari sulkee venttiilin. Tämän jälkeen mittari nollaa laskurin ja vaihtaa mittarin manuaalitilasta automaattitilaan. Tämän jälkeen mittarin ohjaustila vaihdetaan laskuritilasta takaisin normaaliin tilaan.



KUVA 10. Laskuri-välilehti

### 3.3 Testaus

Kun ohjelma ja grafiikat oli saatu valmiiksi, aloitettiin niiden testaus järjestelmässä. Testaus aloitettiin lataamalla applikaatio laboratorion prosessorille. Lataus saatiin suoritettua onnistuneesti ensimmäisellä kerralla. Ensimmäinen testaus aloitettiin heti latauksen jälkeen. Ohjelman perusominaisuudet, kuten yhdistäminen, kirjoitus ja luku, toimivat ensimmäisellä latauksella. Mittareihin saatiin yhteys, ja niiltä saatiin luettua tietoa. Tiedot välittyivät käyttöpaneeliin, jolloin myös operaattorit näkisivät nämä tiedot. Ensimmäinen käytännön testi oli suorittaa nollaus jollekin mittareista. Tämä suoritettiin operaattoreille tehdyn käyttöpaneelin kautta. Nollaus onnistui, joskin tässä vaiheessa grafiikkaan ei ollut vielä merkkivaloa nollauksen valmistumisen seuranta varten. Nollauksen etenemistä pystyttiin kuitenkin seuraa-

maan mittarin omista merkkivaloista, jolloin voitiin todeta ominaisuuden toimivuus. Datan kirjoittaminen mittarille onnistui myös, mutta siinä havaittiin ongelma. Ensimmäisessä versiossa kirjoituspyyntö annettiin yksittäisenä pulssina. Yhdellä pulssilla kirjoitus ei aina mennyt perille, joten sitä piti muuttaa.

Seuraavana päivänä tehtiin pieniä parannuksia ennen seuraavaa testaamista. Nollausominaisuuteen luotiin merkkivalo, joka kertoo, kun nollaus on suoritettu. Nollauksen valmistumista jouduttaisiin muuten seuraamaan mittarista itsestään, mikä aiheuttaisi turhaa kävelyä pitkin laboratoriota. Kirjoituspyynnön ongelma korjattiin pulssigeneraattorin avulla. Kirjoituspyyntö aktivoi pulssigeneraattorin, joka ohjasi MBTCPWrite-palikkaa. Kun kirjoituspyyntö aktivoitiin, pulssigeneraattori alkoi antamaan kirjoituspyynnölle pulssia kolmen sekunnin välein. Kun kirjoitus on onnistuneesti suoritettu, kirjoituspyyntö deaktivoituu ja pulssigeneraattori menee pois päältä. Tällä keinolla pystyttiin varmistamaan kirjoittamisen onnistuminen tilanteessa kuin tilanteessa.

Syöttö- ja puulausominaisuuksien testaamista varten mittareille syötettiin vettä, jonka avulla voitiin turvallisesti testata aineiden pumppaamista mittarin läpi. Ensimmäisenä testattiin puulausta, jonka avulla saatiin linjasto täyteen vettä nopeimmin. Puulausominaisuus toimi juuri suunnitellusti, joskin virtauksen aikaan saamisessa kesti jonkin aikaa. Tämä johtuu siitä, että pumppu on hidas imemään ainetta linjastoon, jolloin virtauksen aikaansaamisessa kestää aikaa. Kun linjastossa on jo valmiina ainetta, on virtauksen aikaansaaminen huomattavasti nopeampaa. Puulaus käynnistyi painikkeesta, ja pysähtyi stop-painikkeesta, kuten se oli tarkoitettukin. Puulauksen jälkeen testattiin nestemäärän laskuria. Laskuri ei lähtenyt toimimaan aluksi oikein, sillä mittari ei toiminut annetuilla laskurin parametreilla. Mittarilla oli kahdet mahdolliset parametrit laskurin käyttöön, ja näistä oltiin valittu väärät. Alun perin laskurin ainemäärää ja laskettua arvoa käsiteltiin liukulukuina, mutta se ei onnistunut. Nämä parametrit muutettiin liukuluvuista kokonaisluvuiksi, jolloin saatiin ohjelma toimimaan tarkoitettulla tavalla. Nestemääriä pitäisi antaa kokonaislukuina, mikä ei haitannut. Laskuriominaisuuden korjaamiseen kului loput toisesta testauspäivästä.

Kolmantena testauspäivänä tutkailtiin pääosin laskuriominaisuutta. Mittari vastaanotti ainemäärän rajan ja näytti käynnissä ollessaan lasketun ainemäärän arvon. Laskuri toimi muuten suunnitellusti, mutta laskurin nollauksen jälkeen virtaus palautui asetusarvoon. Tämä johtui siitä, että laskurin nollaus ei asettanut virtauksen asetusarvoa pois kuin väliaikaisesti. Heti kun laskurin nollaus oli suoritettu, virtauksen asetusarvo pomppasi takaisin siihen arvoon, johon se oli alussa asetettu. Tämä pystyttiin kor-

jaamaan asettamalla mittari käsikäytöltä automaatille laskurin nollausta ennen. Tällöin asetusarvo pysyi nollassa halutun ainemäärän jälkeen. Mittari pitää laskurin nollauksen jälkeen asettaa käsitilaan ja antaa sille asetusarvo, mikäli halutaan alkaa syöttää nestettä läpi tietyllä asetusarvolla. Jos halutaan syöttää toinen erä ainetta mittarista läpi, pitää laskuriominaisuus käynnistää uudelleen.

Kolmannen testauspäivän lopulla todettiin ohjelman toimivan kaikilta osin niin kuin oli suunniteltu. Työelämänohjaajan kanssa totesimme, että käytännön toteutus on saavuttanut sille asetetut tavoitteet. Pieniä viilauksia tehtiin vielä grafiikkaan. Batch-välilehden lasketun nestemäärän kenttä oli alun perin hieman liian pieni. Jotta operaattorin olisi helpompi nähdä se, sitä isonnettiin. Ohjelmapuolen valmistuttua tehtiin vielä operaattoreita varten opas ohjelman käyttöön. Oppaassa kerrottiin, miten ohjelmaa ohjattiin käyttöpaneelin kautta. Itse ohjelman koodiin ei pureuduttu lähes ollenkaan, koska operaattorit eivät siitä hyötyisi. Eri ominaisuuksien toimintaperiaatteet selitettiin yksinkertaisesti, jotta operaattori olisi selvillä, mistä hänelle esitetty tieto olisi peräisin. Manuaalin loppuosaan tehtiin ohjelmoijille muutama huomio siitä, mitä tulisi ottaa huomioon uusia mittareita käyttöönottaessa.

#### 4 YHTEENVETO JA PARANNUSEHDOTUKSIA

Alun testaamisen ja laitteisiin tutustumisen jälkeen käytännön toteutukselle oli hyvät pohjat. Joitain osa-alueita, joita testiohjelmassa ei voitu testata tai käyttää, jouduttiin vielä tutkimaan tehtaalla. Näiden asioiden tutkiminen hoitui kuitenkin sujuvasti ohjelman teon yhteydessä, eikä se hidastanut käytännön toteutusta paljoa. Koska tehtaalla tapahtuneessa toteutuksessa ei ollut tiukkaa aikataulua, pystyttiin toteutukseen keskittymään. Tämä karsi mahdollisesta kiireestä johtuvia virheitä merkittävästi. Oppimisen kannalta tämä oli myös tärkeää, sillä pystyttiin rauhassa tutustumaan itselle vieraisiin asioihin. Kun ohjelmaa testattiin ensimmäistä kertaa, se toimi pääpiirteissään mainiosti. Pieniä korjauksia jouduttiin tekemään, jotta voitaisiin varmistaa ohjelman toiminta halutulla varmuudella. Nämä olivat kuitenkin pieniä muokkauksia, jotka eivät kuluttaneet loppujen lopuksi mahdolltomasti aikaa. Kokonaisuudessa voidaan sanoa, että projekti oli onnistunut. Freeport Cobaltilla oltiin erittäin tyytyväisiä ohjelmaan ja käyttöpaneeliin. Mittareista saatu tieto oli arvokasta laboratoriossa työskenteleville kemisteille. Freeport Cobalt aikoo korvata myös muut vanhat virtausmittarinsa Modbus-mittareilla. Toteutus kokonaisuudessaan on osoittautunut toimivaksi ja on askel eteenpäin vanhasta analogi ohjatusta järjestelmästä.

Henkilökohtaisesti koen tämän olleen hyvä osoitus omista taidoistani tällä hetkellä. Koska käytännön työkokemukseni on vielä vähäistä, ei toteutus varmasti ole kokeneemman insinöörin silmin täydellinen. Toisaalta taas jokainen toteutus on tekijänsä näköinen. Kukaan ei ole täydellinen, ja tällä alalla ei ole yleensä yhtä oikeaa tapaa toteuttaa asioita. Epävarmuus oli välillä esillä vähäisen kokemuksen vuoksi. En antanut sen haitata tekemistäni ja käytin sitä hyödykseni työn mittaan. Nuori insinööri osaa ajatella helpommin laatikon ulkopuolelta, johon kokeneet insinöörit eivät välttämättä taivu tapojen vakiinnuttua. Nuorena insinöörinä osaa epäillä asioita eikä pidä mitään itsestänselvyytenä. Toteutuksen aikana olin välillä hieman hukassa, mutta löysin kuitenkin aina keinon edetä eteenpäin työssäni. Tein työn lähes kokonaan itsenäisesti, mutta kysyin apua, jos jäin jumiin johonkin. Monet koulussa opitut asiat konkretisoituivat projektin myötä, joka oli minulle merkittävä oppimisen paikka.

Valmis ohjelma on toimiva kokonaisuus, mutta myös parannuksia siihen voidaan tehdä. Valmiissa ohjelmassa pitää erikseen määritellä mittauksien hälytysrajat. Nämä luultavasti vakiintuvat tiettyihin arvoihin, kun ollaan käytetty mittareita jonkin aikaa. Nämä vakiintuvat hälytysarvot voitaisiin kirjoittaa ohjelmaan jo valmiiksi, jolloin niitä ei erikseen tarvitsisi määritellä joka kerta, kun uudet mittarit otetaan käyttöön. Jos jotkin hälytysrajat eläisivät aineiden ominaisuuksien vuoksi, voitaisiin ne asettaa

määriteltäviksi. Muita säädettäviä rajoja olivat muun muassa virtauksen asetusarvon rajat, jotka pitää säätää ohjelman kautta. Jonkin käyttöajan jälkeen nämäkin rajat olisivat varmasti tiedossa, jolloin nekin voitaisiin kirjoittaa ohjelman sisällä jo valmiiksi. Näiden rajojen asettelu nopeuttaisi mittareiden käyttöönottoa myöhemmässä vaiheessa.

Mittausarvojen päivittyminen käyttöpaneeliin oli välillä hieman hidasta. Varsinkin virtausmittauksen arvon päivittyminen jumiutui välillä pitemmäksi aikaa. Tähän vaikuttaa suurimmaksi osaksi käytetty liitäntä, joka on RS485:n kahden johdon konfiguraatiossa. Tämä ei mahdollista yhtäaikaista datan vastaanottamista ja lähettämistä, mikä hidastaa datan liikkumista. Mikäli useita operaatioita osuu samalle ajanhetkelle, osa jää toteutumatta. Mikäli datansiirtoa haluttaisiin nopeuttaa, on kolme vaihtoehtoa: joko hienosäätää kirjoitus- ja lukutoimintojen ajoitusta ohjelmassa, vaihtaa kaapelointi full duplexia tukevaan tai kasvattaa tiedonsiirtonopeutta. Hienosäädön avulla voitaisiin estää useiden operaatioiden suorittaminen väylällä yhtäaikaaisesti, mutta kirjoittamisen aikana mittaustiedot eivät päivittyisi siltikään. Tähän ongelmaan tehtiin pieni korjaus muokkaamalla muuntimen konfiguraatiota. Muuntimessa laitettiin päälle first in first out-ominaisuus (FIFO). Tämän ominaisuuden avulla muuntimelle tulevat paketit käsitellään siinä järjestyksessä, kun ne muuntimelle saapuvat. Tämä toi hieman parannusta mittaustietojen päivittymiseen, mutta se ei ratkaissut ongelmaa täysin. Toinen merkittävä tekijä on käytetty verkon nopeus, joka oli 10Mbps. Tämä osaltaan ahdistaa datan liikkumisen nopeutta, mutta tässä tapauksessa tämä on toissijainen aiheuttaja. Mikäli kaapeli olisi eri, ei nopeuden kasvattamista tarvitsisi harkita. Nopeuden kasvattaminen ei kiertäisi käytetyn väylän rajoituksia. Nopeampi tiedonsiirtonopeus voisi estää todella lähekkäin tapahtuvien operaatioiden päällekkäin suorittamista.

Mittarin toimintoja voitaisiin jatkokehittää tulevaisuudessa. Kun opinnäytetyössä tehty ohjelma oli ollut käytössä tehtaalla jonkin aikaa, lisättiin siihen Freeportin toimesta moolisuhdesäätö. Tällöin kolmen mittarin syöttämän ainemäärän suhdetta pystyttiin säätämään, jotta saataisiin aikaan halutunkaltaista seosta. Moolisuhde vaikuttaa seoksen kemiallisiin ominaisuuksiin, ja pienetkin muutokset voivat muuttaa kemiallisia ominaisuuksia merkittävästi. Jokainen mittari syöttää ainetta läpi, ja tätä seurataan ohjelmassa. Valitun moolisuhteen mukaan ohjelma seuraa syötettyä ainemääriä ja vertaa, ovatko ne oikeassa suhteessa haluttuun moolisuhteeseen. Ohjelma säätää virtausta pois ja isommaksi tarvittavien ainemäärien mukaan.

## LÄHTEET

- ABB. 2018a. ABB Ability™ System 800xA. Saatavissa: <http://search-ext.abb.com/library/Download.aspx?DocumentID=3BSE047351&LanguageCode=en&DocumentPartId=&Action=Launch>. Viitattu 12.11.2018.
- ABB. 2016a. AC 800M Getting Started. Saatavissa: <https://search-ext.abb.com/library/Download.aspx?DocumentID=3BSE041880-600&LanguageCode=en&DocumentPartId=&Action=Launch>. Viitattu 6.11.2018.
- ABB. 2016b. AC 800M Configuration. Saatavissa: <https://search-ext.abb.com/library/Download.aspx?DocumentID=3BSE035980-600&LanguageCode=en&DocumentPartId=&Action=Launch>. Viitattu 1.11.2018.
- ABB. 2018b. CI867 System 800xA hardware selector. Saatavissa: <http://www.800xahardwareselector.com/pdf/download/ci867-modbus-tcp>. Viitattu 26.10.2018.
- ABB. 2018c. Graphics Builder. Saatavissa: <https://new.abb.com/control-systems/system-800xa/800xa-dcs/engineering/graphics-builder>. Viitattu 8.11.2018.
- ABB. 2013. Operator Workplace Configuration. Saatavissa: [https://library.e.abb.com/public/16baa156a30aabcb1257b40001f6724/3BSE030322-510\\_D\\_en\\_System\\_800xA\\_Operations\\_5.1\\_Operator\\_Workplace\\_Configuration.pdf](https://library.e.abb.com/public/16baa156a30aabcb1257b40001f6724/3BSE030322-510_D_en_System_800xA_Operations_5.1_Operator_Workplace_Configuration.pdf). Viitattu 6.11.2018.
- Bronkhorst. 2015. Mini CORI-FLOW. Saatavissa: [https://www.bronkhorst.co.uk/files/downloads/manuals\\_english/917050manual\\_mini\\_coriflow.pdf](https://www.bronkhorst.co.uk/files/downloads/manuals_english/917050manual_mini_coriflow.pdf). Viitattu: 1.11.2018.
- Bronkhorst. 2017. Modbus slave interface for digital Mass Flow/ Pressure instruments. Saatavissa: [https://www.bronkhorst.co.uk/files/downloads/manuals\\_english/917035manual\\_modbus\\_slave\\_interface.pdf](https://www.bronkhorst.co.uk/files/downloads/manuals_english/917035manual_modbus_slave_interface.pdf). Viitattu 8.11.2018.
- Bronkhorst. 2018. Operational instructions for digital Multibus Mass Flow / Pressure instruments. Saatavissa: [https://www.bronkhorst.co.uk/files/downloads/manuals\\_english/917023\\_operation\\_instructions\\_digital\\_instruments.pdf](https://www.bronkhorst.co.uk/files/downloads/manuals_english/917023_operation_instructions_digital_instruments.pdf). Viitattu 20.11.2018.
- Freeport Cobalt. Kemiä pelaavat! Connecting the world with cobalt. Esite.
- Kallio, R., Mäkinen, M., Tantarimäki, R. 2009. Prosessiteollisuuden sähkö- ja automaatioasennukset. Keuruu: Otava.
- Mehta, B., Reddy, Y. 2015. Industrial Process Automation Systems. Design and Implementation. Oxford: Elsevier Inc.
- Modbus.org. 2006a. Modbus Application Protocol Specification. Saatavissa: [http://www.modbus.org/docs/Modbus\\_Application\\_Protocol\\_V1\\_1b.pdf](http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf). Viitattu 3.11.2018.
- Modbus.org. 2006b. Modbus Messaging on TCP/IP Implementation Guide. Saatavissa: [http://www.modbus.org/docs/Modbus\\_Messaging\\_Implementation\\_Guide\\_V1\\_0b.pdf](http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf). Viitattu 3.11.2018.

Modbus.org. 2006c. Modbus over Serial Line. Saatavissa: [http://www.modbus.org/docs/Modbus\\_over\\_serial\\_line\\_V1\\_02.pdf](http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf). Viitattu 26.10.2018.

Moxa Inc. 2018. MGate MB3000 Modbus Gateway User's Manual. Saatavissa: [https://www.moxa.com/doc/man/MGate\\_MB3000\\_Series\\_UM\\_e10.2.pdf](https://www.moxa.com/doc/man/MGate_MB3000_Series_UM_e10.2.pdf). Viitattu 26.10.2018.

Mullins, M. 2018. History of the RJ45: A Case of Mistaken Identity. Saatavissa: <https://www.flukenet-works.com/blog/cabling-chronicles/history-rj45-case-mistaken-identity>. Viitattu 14.11.2018.