Jouni Juntunen

Käyttöliittymä automaatiolaitteiden parametrisointiin

Opinnäytetyö

Sähkö- ja automaatiotekniikka

2022

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Jouni Juntunen
Työn nimi	Käyttöliittymä automaatiolaitteiden parametrisointiin
Toimeksiantaja	Jemtek Oy
Vuosi	2022
Sivut	25 sivua
Työn ohjaaja(t)	Teemu Manninen

TIIVISTELMÄ

Työn tavoitteena oli suunnitella ja toteuttaa ohjelmistoalusta automaatiolaitteiden yksinkertaiseksi käyttöliittymäksi. Käyttöliittymä on käyttäjän konfiguroitavissa.

Opinnäytetyön alussa selvitettiin yleisiä vaatimuksia Modbus-protokollan pohjalta, jonka perusteella suunniteltiin yksinkertainen skaalautuva käyttöliittymä.

Toteutus muodostui ohjelmakoodin kirjoittamisesta C#-kielellä sekä valmiin ohjelman testauksesta. Testaus muodostui automaattisista yksikkötesteistä sekä manuaalitesteistä. Manuaalitesteissä käytettiin useampaa erilaista laitetta.

Työssä tuotetulla sovelluksella käyttäjä pystyy muodostamaan yksilöllisen käyttöliittymän Modbus-laitteeseen. Alustan skaalautuvuus selviää vasta lisättäessä ohjelmaan uusia erityyppisiä protokollia.

Asiasanat: Käyttöliittymä, Modbus, C#

South-Eastern Finland
University of Applied Sciences

Degree	Barchelos of Engineering
Author (authors)	Jouni Juntunen
Thesis title	Bachelor’s / Master’s thesis title in English
Commissioned by	Jemtek Oy
Time	April 2022
Pages	25 pages
Supervisor	Teemu Manninen

ABSTACT

The objective of the thesis was to design and implement a software platform for a simplified user interface for automation devices. With the application the user can form an individual user interface to the Modbus device.

The program was written in C#. Testing was done with manually and automated unit tests. The manual tests were done using several different kinds of devices.

Keywords: User Interface, Modbus, C#

SISÄLLYS

Lyhenteet ja käsitteet

ADU	Application Data Unit
IP	Internet Protocol
PDU	Protocol Data Unit
RS485	Sarjaliikenneväylä
RTU	Remote Terminal Unit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
XML	eXtensible Markup Language
CSV	Comma Separated Values

1 JOHDANTO

Tämä työ on suunniteltu työnantajalleni Jemtek Oylle. Jemtek Oy on sulautettua ohjelmistosuunnittelua tarjoava yritys. Yritys on toiminut vuodesta 2018 Helsingissä.

Työn tavoitteena on suunnitella ja toteuttaa PC:ssä toimiva ohjelmistoalusta erilaisten teollisuus- ja rakennusautomaatiolaitteiden numeeriseksi käyttöliittymäksi. Ohjelman kohderyhmän muodostaa tekninen henkilöstö, jonka tarvitsee tarkastella asetuksia ja tehdä pieniä muutoksia tuotekehityksen, konfiguroinnin tai esimerkiksi kunnonvalvonnan yhteydessä. Vai vaikka testata kehittämänsä uuden protokollan toimintaa.

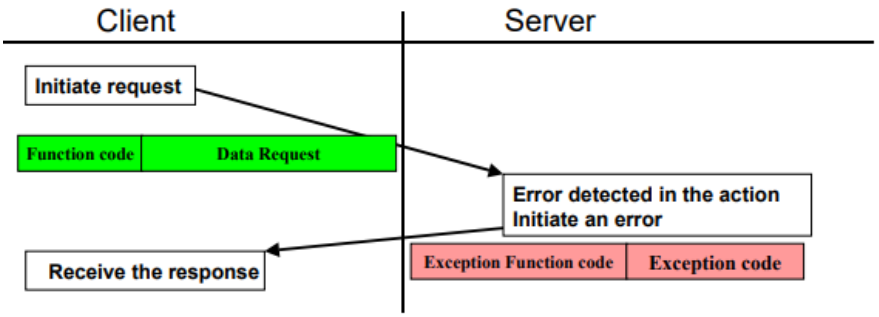
Työn alkuun perehdytään Modbus-protokollaan. Modbus valittiin ensimmäiseksi ohjelmaan toteutettavaksi yksinkertaisuuden vuoksi, mutta joka kuitenkin samalla tarjoaa useita erilaisia variaatioita niin esitysmuodon kuin laitteistorajapinnan osalta. Tämän jälkeen käsitellään hieman vaatimuksia ja käyttöliittymää. Lopuksi luodaan valmiilla ohjelmalla esimerkkiprojekti sekä pohditaan tuloksia sekä jatkokehityshankkeita.

2 MODBUS

Modbus on Modiconin vuonna 1979 julkaisema tietoliikenneprotokolla, joka oli alun perin tarkoitettu käytettäväksi heidän valmistamiensa ohjelmoitavien logiikkojen kanssa. Protokolla oli menestys ja siitä onkin muodostunut jonkinlainen ”de facto” -standardi rakennus- ja teollisuusautomaatiossa. Protokolla on yksinkertainen, riittävän luotettava yleisimmissä käyttötapauksissa sekä lisenssimaksuton. Myös dokumentaation on vapaasti saatavilla. Nykyisin se on The Modbus Organizationin hallinnoima avoin standardi. /1./

Protokollan toiminta on periaatteeltaan varsin yksinkertainen. Verkkoon kuuluvat laitteet kytketään samaan väylään, jossa jokaisella laitteella on yksilöllinen osoite. Yksi laitteista toimii isäntänä (master tai client), joka kyselee muilta laitteilta (server tai slave) vuoron perään tarvitsemiaan tietoja. Laitteen vastattua

voidaan jatkaa kyselyä seuraavalta laitteelta. Muut laitteet eivät voi aloittaa kommunikaatiota tai viestiä keskenään. /2./

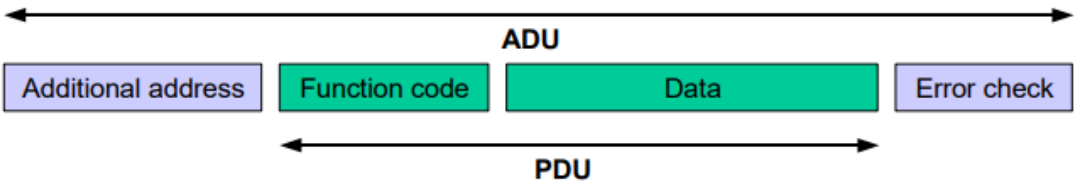


Kuva 1. Modbus transaction /2/

Modbus-protokollasta on suuri määrä erilaisia variaatioita, ja se on toteutettu useille eri väylille. Yleisimmät ovat Modbus RTU tai Modbus TCP. Modbus TCP käyttää IP-verkkoa ja RTU-laitteet kytketään RS485 väylään.

2.1 Sanoman rakenne

Modbus-sanoma (ADU) muodostuu kolmesta osasta: osoitteesta, protokolla-datasta (PDU) sekä mahdollisesta virheentarkistuksesta. Osoite ja virheentarkistus ovat väyläriippuvaisia.



Kuva 2. Modbus sanoman yleinen rakenne /2./

PDU muodostuu operaatiokoodista (Function code) ja yhdestä tai useammasta 16-bittisestä arvosta. Operaatiokoodi määrittelee, millaista tietoa viesti sisältää. Operaatiokoodia seuraa yleensä osoite, joka kertoo, mihin rekisteriin operaatio kohdistuu. Kentän koko on 16 bittiä, joten osoiteavaruus kattaa rekisterit 1…65535. /2./

Request

Function code	1 Byte	**0x02**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Inputs	2 Bytes	1 to 2000 (0x7D0)

Response

Function code	1 Byte	**0x02**
Byte count	1 Byte	**N***
Input Status	**N*** x 1 Byte	

Kuva 3. PDU:n rakenne /2/

Dokumentaatiossa osoitteet kuvataan usein 5 tai 6 merkin avulla. Alun perin osoiteavaruus oli jaettu neljään ryhmään seuraavasti. /2./

Taulukko 1. 5-numeroinen osoite

Objekti	Tyyppi	Koko	Osoitteet
Kela	Luku/kirjoitus	1 bitti	00001-09999
Diskreetti tulo	Vain luku	1 bitti	10001-19999
Tulorekisteri	Vain luku	16 bittiä	30001-39999
Pitorekisteri	Luku/kirjoitus	16 bittiä	40001-49999

Alkuperäisen määrittelyn lisäksi on muodostunut 6-numeroinen de facto -esitysmuoto, jolloin koko osoiteavaruus on käytettävissä.

Taulukko 2. 6-numeroinen osoite

Objekti	Tyyppi	Koko	Osoitteet
Kela	Luku/kirjoitus	1 bitti	000000 065535
Diskreetti tulo	Vain luku	1 bitti	100000-165535
Tulorekisteri	Vain luku	16 bittiä	300000-365535
Pitorekisteri	Luku/kirjoitus	16 bittiä	400000-465535

Kummassakaan esitysmuodossa osoitteen ensimmäistä merkkiä ei käytetä väylällä osoitteen osana, vaan se ilmaisee käytettävän operaatiokoodin.

Lisäksi on määritelty, että väylällä ensimmäinen osoite on nolla. Tämä saattaa aiheuttaa joskus sekaannusta dokumentaation ja käytännön kanssa. On epäselvää, pitäisikö väylällä käyttää samaa osoitetta kuin dokumentaatiossa vai vähentää siitä yksi. Usein siihen ei ole muuta konstia kuin lukea joku tunnettu rekisteri ja verrata sisältöä laitteen manuaaliin.

2.2 Modbus RTU

Modbus RTU on binäärinen esitysmuoto sarjamuotoista liikennettä varten. Liitäntä on asynkroninen, ja siinä siirretään tietoa yksi merkki kerrallaan. Merkin pituus voi vaihdella tarpeen mukaan, mutta tyypillisesti se koostuu kahdeksasta databitistä, yhdestä start- ja stop-bitistä sekä mahdollisesta pariteettibitistä.

Modbus-sanomat muodostuvat tavuista ja protokollassa sanomat erotetaan toisistaan ajan perusteella. Uuden sanomat katsotaan alkavan, kun väylä on ollut lepotilassa yli 3,5 kertaa yhden merkin lähettämiseen käytetty aika. Merkkien välillä ei saa olla 1,5 merkin ylittävää taukoa. Varsinkin suurella baudinopeudella merkkien väliseen ajan mittaaminen saattaa olla haasteellista PC ympäristössä. /2./

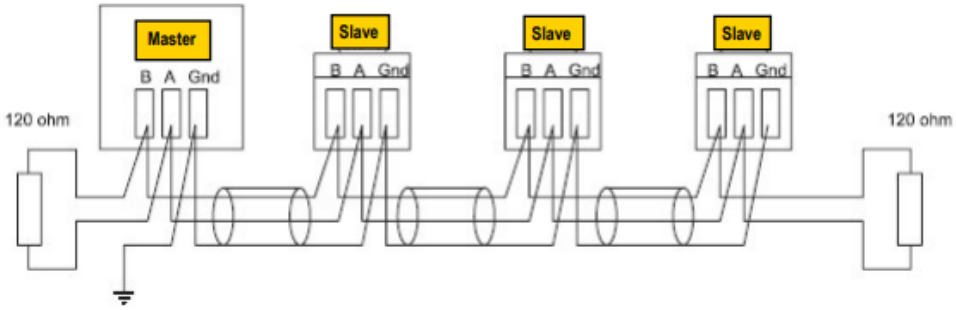
Tarkasta mittauksesta olisi hyötyä esimerkiksi sellaisen tilanteen havaitsemiseen, kun vastapää lähettää oikeansisältöisen viestin, mutta merkkien välille muodostuu liian pitkä tauko. Tällöin vastaanotettava viesti olisi hylättävä virheellisenä.

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

Kuva 4. Modbus RTU -viestin sisältö

RTU-viesti koostuu kohdelaitteen osoitteesta, operaatiokoodista, datasisällöstä sekä tarkistussummasta. Fyysinen siirtoväylä voi olla esimerkiksi kahden laitteen välinen RS232 tai useamman laitteen RS485-väylä.

RS485 on balansoitu sarjaliikenneväylä, jossa standardi määrittelee väylän sähköiset ominaisuudet.



Kuva 5. Tyypillinen RS485-verkko, väylän päissä balansointivastukset /3/

2.3 Modbus TCP/IP ja UDP

Modbus TCP on kehittynein teknologia. Siinä informaatio siirtyy paketeissa laitteiden välillä. Tiedonsiirto on pakettitasolla luonteeltaan yhteydetön, eli verkosta ei varata kiinteää yhteyttä laitteiden välille, vaan verkon laitteet välittävät paketit osoitekentän perusteella käytettävän verkon yli IP-protokollan avulla. IP-protokollan päällä voidaan käyttää muita kuljetuskerroksen protokollia, kuten TCP tai UDP. /4./

TCP muodostaa loogisen luotettavan yhteyden kahden laitteen välille. Se huolehtii kuittauksista ja uudelleenlähetyksistä sekä varmistaa, että paketit saapuvat oikeassa järjestyksessä. TCP on luonteeltaan yhteydellinen protokolla.

UDP on taas yhteydetön protokolla. Siinä IP-kerros huolehtii pakettien välittämisestä verkossa, mutta protokolla ei takaa viestin perille välittämisestä. Se jää sovelluksen vastuulle.

Automaatiotekniikassa UDP-protokollaa käytetään usein prosessidatan välittämiseen. Käyttökohteesta riippuen viestien määrä voi olla hyvin suuri ja vasteaika halutaan pitää mahdollisimman pienenä tai ainakin vakiona.

Aiemmin mainittiin, että väylässä voi olla vain yksi isäntä. IP-pohjaiset protokollat tekevät tässä yleensä poikkeuksen. Toteutuksesta riippuen laite saattaa pystyä palvelemaan useita samanaikaisia yhteyksiä.

3 YLEISTÄ OHJELMASTA

Tässä luvussa käsitellään ohjelman käyttöympäristöä ja vaatimuksia.

3.1 Käyttöympäristö

Ohjelma on toteutettu PC:lle pääasiassa C#-kielellä Visual Studiolla. C# on Microsoftin .NET-alustalle kehitetty vahvasti tyypitetty oliopohjainen ohjelmointikieli.

.Net Framework koostuu kahdesta osasta: CRL:stä (Common Language Runtime) ja luokkakirjastoista (Framework Class Library, FCL), jotka tarjoavat ohjelmointikielen tarvitsemat palvelut.

Alimmaisena on CLR, joka hoitaa laitteiston ja käyttöjärjestelmän välisen tiedonvälityksen. Luokkakirjasto (FCL) tarjoaa rajapinnat kaikkiin CLR:n tarjoamiin palveluihin. Lähdekoodi käännetään Visual studiolla tavukoodiksi, jonka ajonaikainen ympäristö kääntää kohdeympäristön konekielelle. Käännös tapahtuu käyttäjältä näkymättömissä. /5./

3.2 Vaatimukset

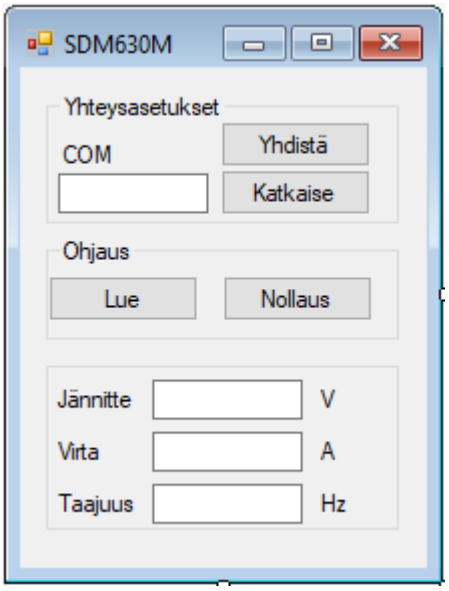
Tässä työssä on tarkoituksena muodostaa ohjelma, jolla voidaan lukea ja kirjoittaa laitteen asetuksia käyttäen jotain tunnettua rajapintaa, kuten esimerkiksi Modbus-protokollaa.

Verkossa voi olla erilaisia laitteita, joten näkymä pitäisi pystyä määrittelemään jokaiselle laitteelle yksilöllisesti. Jos samassa verkossa on useita laitteita, ohjelma voi kommunikoida usean erityyppisen laitteen kanssa samanaikaisesti sekä esittää numeerisia arvoja myös graafisesti.

Muodostettu konfiguraatio täytyy pystyä tallentamaan myöhempää käyttöä varten.

3.3 Käyttöliittymä

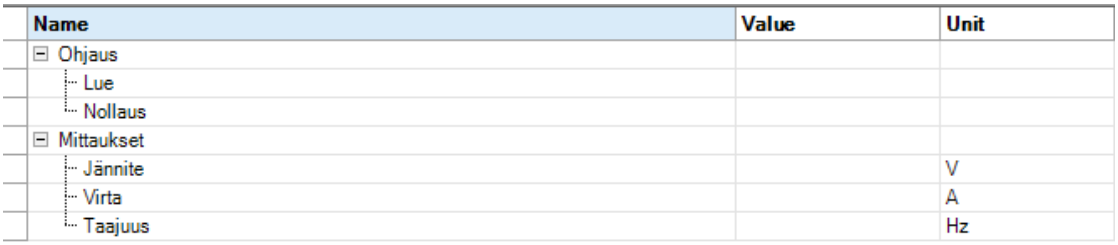
Tyypillisesti käyttöliittymät ovat räätälöity käyttäjän tai laitteiston tarpeisiin ja sisältää erilaisia tietoruutuja, nappuloita, valikoita ja grafiikkaa.



Kuva 6. Yksinkertainen käyttöliittymä

Käytännössä yllä mainitun tapainen käyttöliittymä olisi tässä työssä mahdollinen, mutta ei välttämättä tarkoituksen mukainen. Työssä tutkittiin vaihtoehtoisia menetelmiä helposti skaalautuvan käyttöliittymän muodostamiseen.

Perinteisen painikenäkymän sijasta DataGrid osoittautui käyttökelpoiseksi. DataGrid on muokattava ohjelmistokomponentti taulukkomuotoisten tietojen esittämiseen.
Komponentti sisältää tyypillisesti editorin, joka mahdollistaa automaattisen rivien lisäämisen ja poistamisen ajonaikaisesti. Solujen, sarakkeiden tai rivien muokkaaminen on varsin vapaata, ja soluihin voidaan lisätä tarvittaessa myös painikkeita. Tietolähteenä voidaan käyttää monen tyyppisiä tiedostoja. Käyttäjä voi antaa asetukselle nimen sekä yksikö. Työssä käytetty komponentti tukee myös puumaista esitystapaa, jolloin käyttäjä voi muodostaa asetuksista loogisia kokonaisuuksia tarpeensa mukaan.



Kuva 7. Hierarkkinen esitys

Kuvassa alussa mainittu ”yksinkertainen käyttöliittymä” esitettynä uudella tavalla.

3.4 Tallennus

Käyttäjän luoma tietorakenne tallennetaan uudelleenkäyttöä varten. Ohjelmoinnissa käytetty NET-komponenttikirjasto tarjoaa valmiin rajapinnan objektien serialisointiin joko binääri- tai XML- muodossa. Serialisoinnissa objektin kentät muutetaan tallennuskelpoiseksi merkkijonoksi. Deserialisoinnissa merkkijonon palautetaan takaisin objektiksi. /6./
Käytännössä kuitenkin luokkien rakenne muuttuu ohjelman elinkaaren aikana, joka saattaisi aiheuttaa vähintään yhteensopivuusongelmia eri ohjelmistoversioiden välillä.

Automaattisen serialisoinnin sijaan päädyttiin määrittelemään yksinkertainen ja ehkä myös hieman geneerisempi XML-pohjainen tallennusformaatti. Extensible Markup Language (XML) on yleisesti käytetty joustava kuvauskieli tietorakenteiden esittämiseen tekstimuodossa. /7./

Tekstimuotoisena tiedostot ovat helposti siirrettävissä laitteesta toiseen, ja niiden sisältöä voidaan tarkastella ja muokata lähes millä tahansa editorilla, mikä binääriformaattia käytettäessä olisi hyvin hankalaa.

3.5 Visualisointi

Visualisoinnilla laitteen tapahtumia voidaan tarkastella graafisesti ajan funktiona. Käytännössä tämä tehdään lukemalla laitteen asetuksia peräkkäin tietyllä nopeudella ja merkitsemällä tulokset taulukkoon. Lähtökohtaisesti mikä tahansa laitteesta luettava numeerisesti esittävä asetus tulisi olla esitettävissä graafisesti.

Kerätty data voidaan tallentaa nauhoituksen lopuksi aikaleimoineen tiedostoon CSV-formaatissa myöhempää tarkastelua varten. CSV on tiedostomuoto, jolla voidaan tallentaa yksinkertaista taulukkomuotoista dataa tiedostoon. /8./

Formaatti on toteutukseltaan tekstitiedosto, jossa taulukkorakenteen kentät on erotettu toisistaan pilkuilla ja rivinvaihdoilla. Tässä tapauksessa formaatti ei ole erityisen tehokas suorituskyvyn tai levynkäytön suhteen, mutta avoimena formaattina kerättyä dataa voi tarkastella helposti muilla ohjelmilla.

4 ASETUKSET

Aluksi kartoitettiin konfiguroitavia asioita ja jaettiin niitä toiminnallisiin ryhmiin. Ryhmittelyn jälkeen aseteltavat asiat jakaantuvat neljään hierarkkiseen pääryhmään - protokollavalintaan, laitteistoasetuksiin, parametrin lukuun liittyviin asioihin sekä parametrin esitystapaan. Asetukset riippuvat osittain toisistaan.

4.1 Protokolla

Protokollavalinta määrittelee käytettävän protokollan tai yleisemmin kommunikointitavan, jonka perusteella voidaan esittää laitetason asetukset. Valinnan yhteydessä laitteelle annetaan myös sen osoite.

4.2 Laitteen asetukset

Laiteasetukset sisältävät laitetta ja yhteyttä koskevia asetuksia. Tällaisia ovat esimerkiksi.

- Laitteen yksilöivä osoite
- Laitteen nimi
- Vastauksen odotusaika
- Viestien välinen aika
- Tavujärjestys.

Asetusten ryhmittely ei ole tieteellisen tarkka. Esimerkiksi tavujärjestys voisi olla myös parametrikohtainen, mutta koska sen oletetaan olevan kaikille parametreille samanlainen, on se yksinkertaisuuden vuoksi määritelty laitteen asetuksissa. Tavujärjestystä voi vaihtaa tarvittaessa myös laskentakaavassa laskemalla tavut yhteen yksitellen.

Osoite on laitteen yksilöivä tunniste, minkä perusteella se löytyy verkosta tai väylältä. Laitteen nimi on käyttäjän vapaasti valittavissa.

4.3 Parametri

Tässä yhteydessä parametrilla tarkoitetaan yhtä riviä parametrinäkymässä. Parametri voi muodostua yhdestä tai useammasta rekisteristä tai niiden osasta, mikäli niiden arvot ovat luettavissa yhdellä komennolla:

- Rekisterin numero
- Operaatiokoodi lukemiseen
- Operaatio kirjoitukseen
- Rekistereiden määrät

Parametriasetukset ovat lähes kokonaan protokollariippuvaisia. Esimerkiksi Ethernet/IP:n vastaava asetus voisi muodostua luokasta, instanssista sekä attribuutista. /12./

4.4 Esitystapa

Esitystapa sijoittuu OSI-mallin sovelluskerrokselle ja määrittelee nimensä mukaisesti, kuinka parametrin arvo esitetään käyttäjälle. Sisältö voi käytännössä olla lähes mitä tahansa, tekstiä tai numeerisia arvoja erilaisissa muodoissa, eikä sen sisältöä ole yleensä sidottu käytettävään protokollaan.

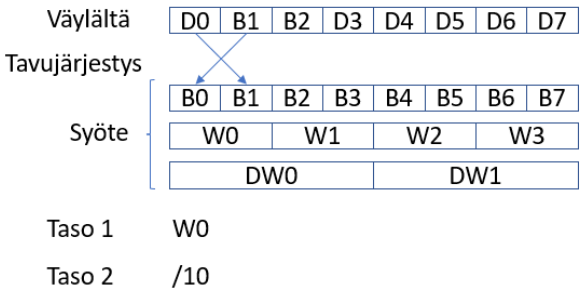
Parametrinäkymä muodostuu kolmesta tekijästä.

- Nimestä
- Arvosta
- Tyypistä.

Tässä mallissa on oletettu, että käyttäjä konfiguroi parametrin nimen ja tyypin ja arvo luetaan laitteesta. Ohjelma on toteutettu niin, että esitystapakerros saa laitteelta luetun parametrin protokollariippumattomana merkkijonona, josta se parsitaan käyttäjälle sopivaan muotoon.

Tulostuksen muotoitu tehdään kolmitasoisessa prosessissa seuraavasti.

Ensimmäisellä tasolla voidaan vaihtaa tavujärjestystä kahden tai neljän tavun lohkoissa.



Kuva 8. Tulostuksen muotoilu

Seuraavat kaksi tasoa ovat laskukaavoja, joiden lopputuloksena on käyttäjälle näytettävä arvo. Ensimmäisen kaavan syötteenä toimii väylältä merkkijono tavumuunnoksen jälkeen, johon voidaan viitata tavuittain (Bx), sanoittain (Wx) tai kaksoissanoin (Dx). Laskentakaavan parina on asetus, joka kertoo lopputuloksen tyypin. Ensimmäisen kaavan tulos toimii myös syötteenä toiselle kaavalle.

Oletetaan esimerkiksi, että luettaessa laitteesta kolme peräkkäistä 16-bittistä arvoa, joista ensimmäinen sisältää virran [A], toinen on tyhjä ja viimeinen jännitteen [V]. Käyttäjälle voidaan esittää teho [VA] kertomalla ensimmäinen ja viimeinen arvo keskenään, jolloin ensimmäinen laskukaava olisi ”W0*W2”. Esimerkissä keskimmäinen arvo jätetään käyttämättä.

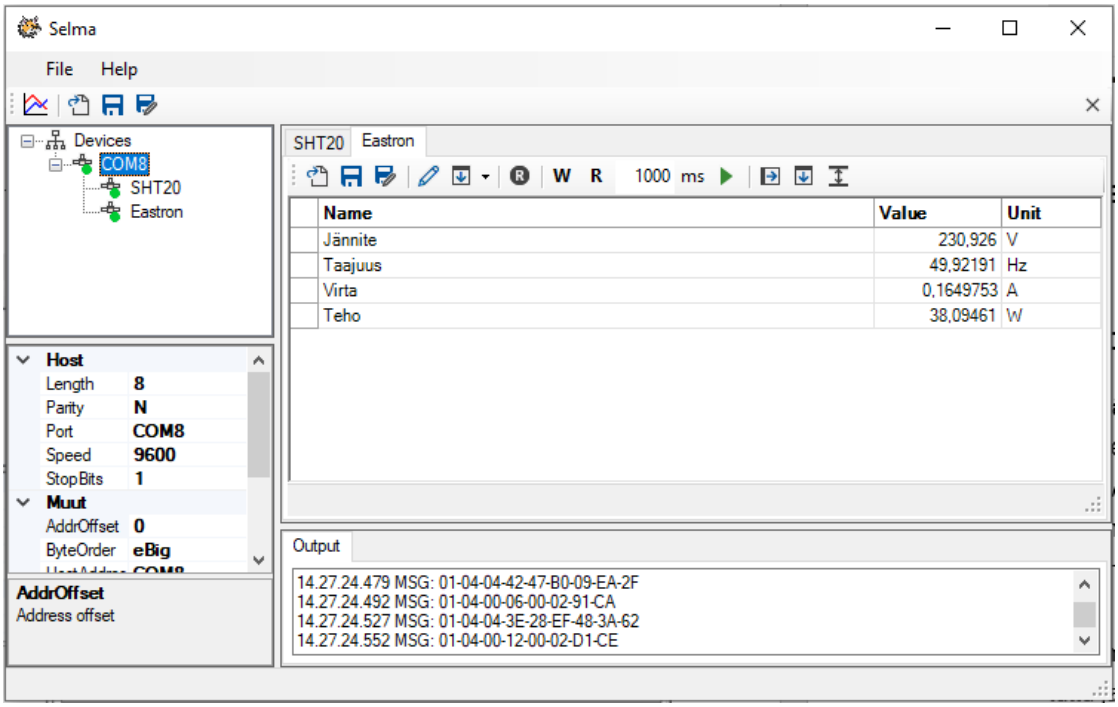
Arvojen kirjoittaminen tehdään omilla kaavoilla päinvastaisessa järjestyksessä, mutta tämä suunta on selvästi rajoitetumpi kuin luku. Esimerkiksi edellä mainitussa esimerkissä käyttäjän muuttaessa tehoa, virtaa ja jännitettä ei voida kirjottaa takaisin laitteeseen, koska niitä ei voida johtaa tehosta takaperin. Parametri on näin ollen vain luettavissa.

Lähtökohtaisesti esitystapakerros on riippumaton muista asetuksista.

5 KÄYTTÖLIITTYMÄ

Tässä kappaleessa esitellään valmis käyttöliittymä ja luodaan sen avulla esimerkkiprojekti.

Varsinainen käyttöliittymä on jaettu neljään erilliseen paneeliin. Laitteet sijaitsevat vasemmalla ja parametrinäkymä sen vieressä oikealla. Vasemmalla alhaalla on tietoruutu, jonka sisältö vaihtuu käyttäjän valinnan mukaisesti. Asetusten konfigurointi tapahtuu tässä ruudussa

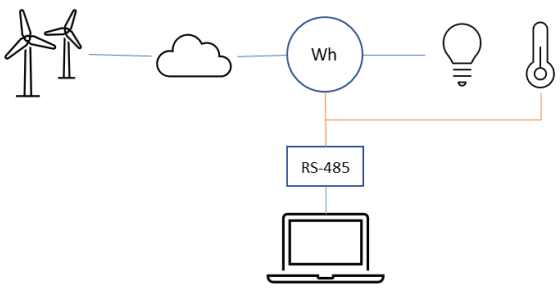


Kuva 9. Ohjelman ulkoasu

Alimmaisena on infokenttä. Tässä kentässä esitetään käyttäjälle tietoja ohjelman toiminnasta.

5.1 Käyttö

Esimerkissä luodaan käyttöliittymä kulutus- ja lämpömittarille. Kytkennässä mitataan hehkulampun energian kulutusta sekä lämpötilaa. Molemmat laitteet on kytketty samaan väylään.



Kuva 10. Mittauskytkentä

Energiamittari Eastron SDM630 (M) on DIN kiskoon asennettava 3-vaiheinen kulutusmittari, jota voidaan käyttää myös yksivaiheisena. Mittarissa on integroitu näyttö, ja se mittaa muun muassa seuraavia sähköverkon ominaisuuksia.

- Taajuutta
- Jännitettä
- Virtaa
- Tehoa.

Tehomittaus tehdään molempiin suuntiin, kulutettua ja verkkoon syötettyä energiaa. /3./

Lampun tuottamaa lämpöä seurataan SHT20-anturilla, joka mittaa myös suhteellista ilmankosteutta. /9./

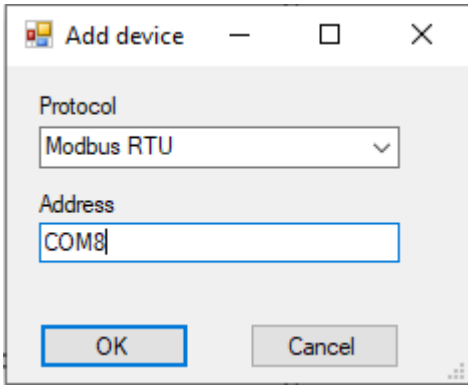
Konfiguroinnin avuksi tarvitaan laitteen manuaali. Manuaalista selviää laitteen kommunikaatioprotokollat sekä niiden asetukset. Tässä tapauksessa se on Modbus RTU, joka käyttää RS485-väylää. Laitteiden kytkemiseksi tietokoneeseen tarvitaan lisäksi USB-RS485-muunnin.

Manuaalien mukaan laitteiden oletusasetukset ovat identtiset. Kommunikaatioasetukset sopivat sellaisenaan, mutta koska laitteet kytketään samaan väylään, toisen Modbus Node ID täytyy vaihtaa toiseksi.

Muutosten jälkeen asetukset ovat seuraavat

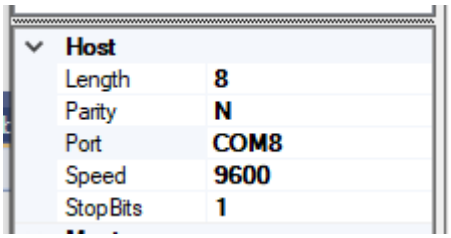
- Kommunikaatioasetukset 9600 8,N,1
- Eastron Node ID 1
- SHT 20 Node ID 2.

Ohjelman käyttö aloitetaan protokollavalinnalla. Valitaan listasta Modbus RTU, ja osoitteeksi USB-RS485 muuntimen käyttämä COM-portti.



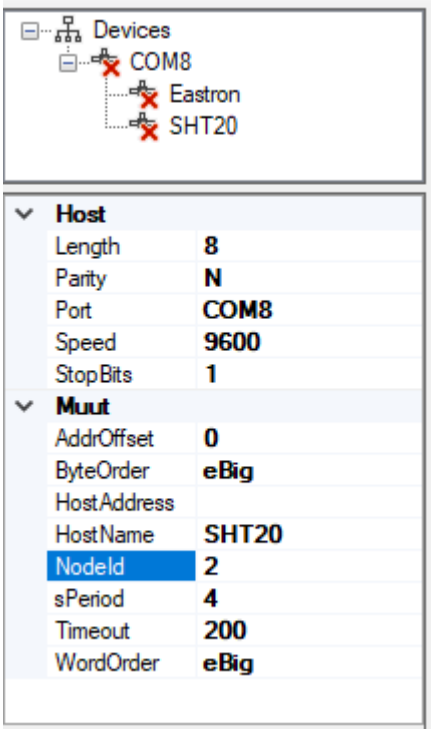
Kuva 11. Protokollavalinta

Seuraavaksi täytetään COM portin tietoliikenneasetukset.



Kuva 12. Kommunikaatioasetukset

Tämän jälkeen laitepuuhun lisätään Modbus RTU laitteet energia- ja lämpömittarille sekä täytetään molempien laiteosoitteet. Puussa näkyvät nimet ovat vapaasti valittavissa. Nimen edessä näkyvä punainen ruksi tarkoittaa, ettei laitteeseen ole muodostettu yhteyttä. Kun yhteys on aktiivinen, symboli vaihtuu vihreäksi ympyräksi.



Kuva 13. Kommunikaatioasetukset

Seuraavaksi muodostetaan varsinainen käyttöliittymä. Esimerkissä haluamme tarkastella jännitettä, taajuutta, virtaa sekä hetkellistä tehoa. Konfigurointia varten selvitämme laitteen käyttöohjeesta seuraavat asetukset. /3; 9./

- Rekisterin numero
- Luettavien rekistereiden määrä
- Operaatiokoodi
- Sisällön esitysmuoto.

Taulukko 3. Energiamittarin asetukset

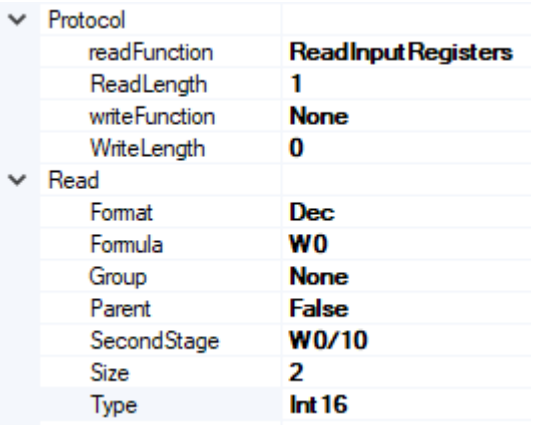
Nimi	Rekisteri	Operaatiokoodi	Koko	Tyyppi
Jännite	30001	Read Input register	2	Float
Taajuus	30071	Read Input register	2	Float
Virta	30007	Read Input register	2	Float
Teho	30019	Read Input register	2	Float

Kaikkien asetusten arvot ovat neljän tavun liukulukuja IEE 754 -standardin määrittelemässä formaatissa. Ohjelmassa tämä on float32-tyyppinen arvo. Asetuksen koko on neljä tavua, jolloin laitteesta luetaan kaksi Modbus-rekisteriä.

Taulukko 4. Lämpömittarin rekisterit

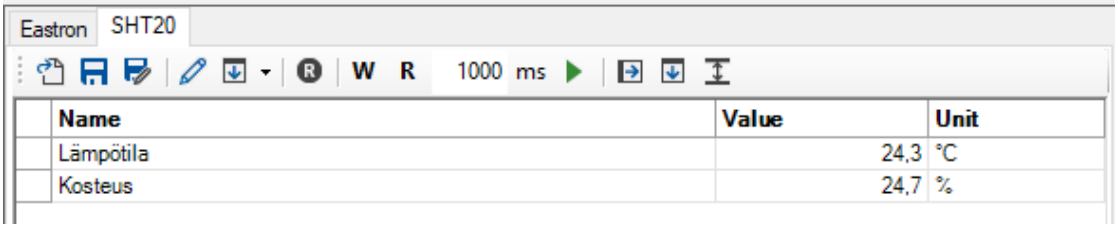
Nimi	Rekisteri	Operaatiokoodi	Koko	Tyyppi
Lämpötila	1	Read Input register	1	Int16 *10
Kosteus	2	Read Input register	1	Int16 *10

Lämpötila ja kosteus ovat 16-bittisiä etumerkillisiä kiinteän pilkun kokonaislukuja. Ohjelmassa ne voidaan käsitellä niin, että ensimmäisessä kaavassa arvo määritellään int16-tyyppiseksi ja toisessa jaetaan kymmenellä.

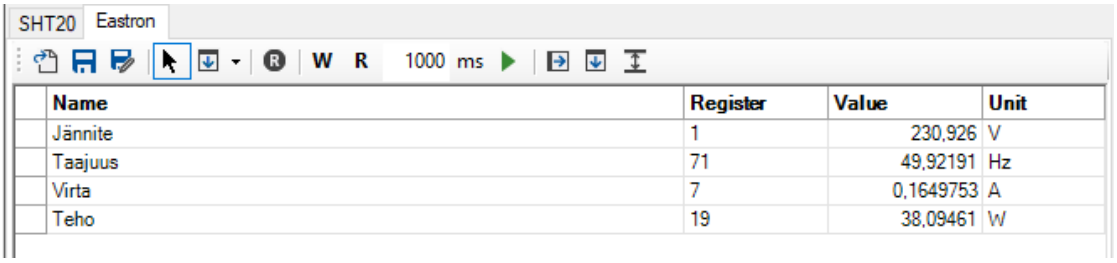


Kuva 14. Asetukset lämpötilan lukemiseen

Muodostetut rekisterikartat näyttävät valmiina seuraavilta. Rekisterien numerot ovat näkyvissä vain editointitilassa.

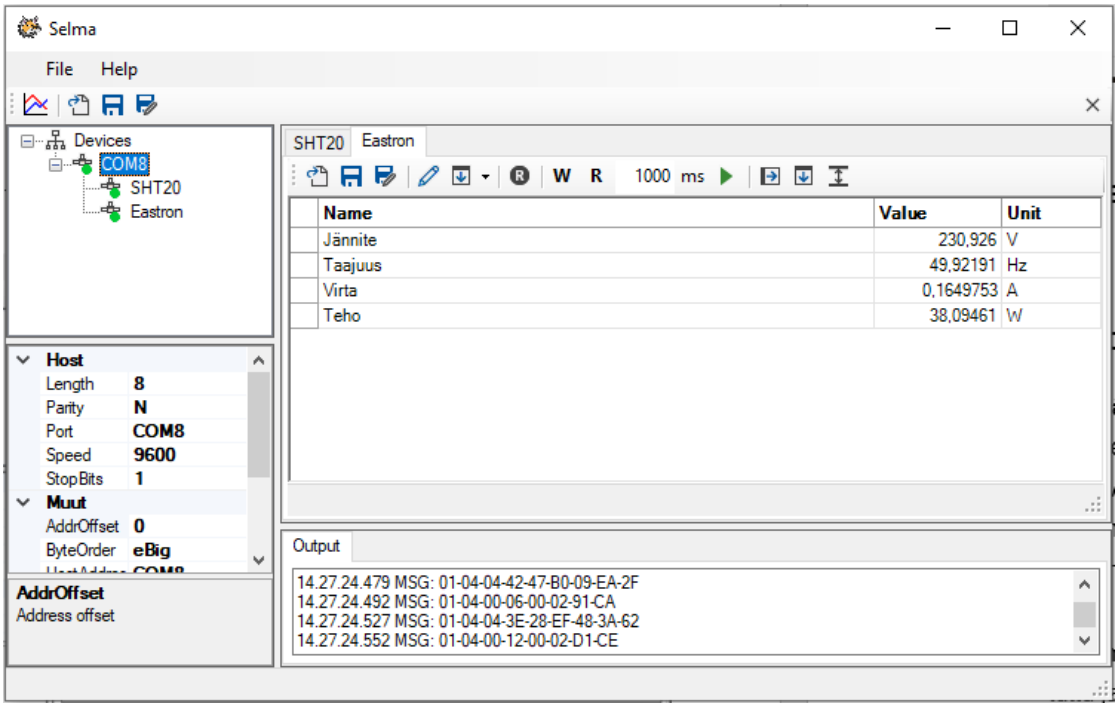


Kuva 15. Lämpömittari



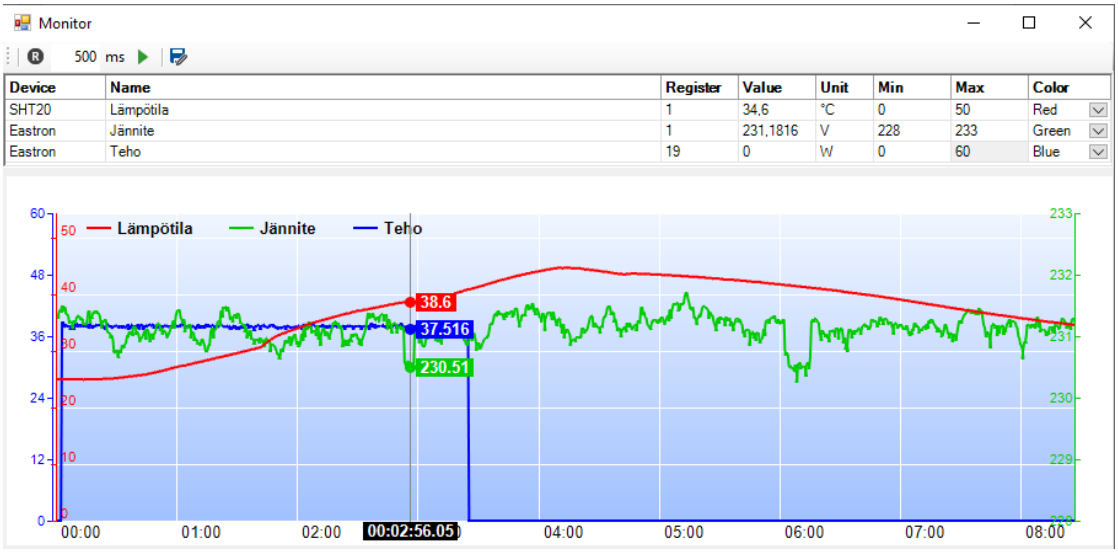
Kuva 16. Energiamittari

Valmis käyttöliittymä näyttää tältä.



Kuva 17. Valmis käyttöliittymä

Parametrien arvoja voidaan tarkastella myös visuaalisesti. Alla nauhoitus esimerkkikytkennästä. Kuorman kytkentä lämmittää ympäröivää ilmaa, joka on nähtävissä myös mittarilla pienellä viiveellä.



Kuva 18. Mittaustulokset

6 TESTAUS

Testaamisen tavoitteena on havaita ohjelmistossa ilmenevät häiriöt, jotta viat voidaan löytää ja korjata. Testaus voidaan suorittaa usealla eri tavalla, automaattisesti tai manuaalisesti sekä testattavan kokonaisuuden mukaan jaoteltuna esimerkiksi yksikkö- ja integraatiotestaukseen.

Yksikkötestissä testataan yksittäisen ohjelmistokomponentin toimintaa. Testattava objekti voi olla yksittäinen funktio, luokka tai mikä tahansa eristettävissä oleva komponentti. Komponentti testataan itsenäisesti ilman muita järjestelmän osia. Yksikkötestit ovat lähtökohtaisesti lähes aina automaattisia ja osa regressiotestausta. Regressiotestauksessa varmistetaan, että osa toimii myös jatkossa ohjelman elinkaaren aikana.

Kattava yksikkötestauskirjasto mahdollistaa koodin myös hyvän ylläpidon. Tämä auttaa varmistamaan, että uusien ominaisuuksien lisääminen tai koodin refaktorointi ei muuta komponentin toimintaa. Refaktorointi tarkoittaa prosessia, jossa ohjelman lähdekoodia muutetaan siten, että sen toiminnallisuus säilyy, mutta sisäinen rakenne muuttuu. /11./

```
public void TestCase_SHT20Example()
{
    Converter c = new Converter();
    Display prop = new Display();
    prop.Formula = "W0";
    prop.SecondStage = "W0/10";
    prop.Type = Display.eType.Int16;
    byte[] reg = new byte[2];

    // Negatiivinen arvo
    reg[0] = 0xFF;
    reg[1] = 0x33;
    string result = c.Parse(prop, reg);
    Assert.AreEqual(result, "-20,5");

    // Positiivinen arvo
    reg[0] = 0x01;
    reg[1] = 0x31;
    result = c.Parse(prop, reg);
    Assert.AreEqual(result, "30,5");
}
```

Kuva 19. Yksikkötesti

Esimerkissä yksikkötesti, jolla varmistetaan, että lämpömittarin käyttöoppaassa käytetyt esimerkkiarvot esitetään oikealla tavalla. Integraatiotestauksessa varmistetaan komponenttien toiminta yhdessä suurempana ja mahdollisesti lopullisena kokonaisuutena. Ohjelman toimintaa on testattu manuaalisesti kehityksen ohella useammalla erilaisella laitteella.

7 LOPPUPÄÄTELMÄ

Opinnäytetyön tehtävänä oli suunnitella ja toteuttaa käyttöliittymä automaatiolaitteiden konfigurointiin. Lähtökohdat olivat varsin vapaat. Ainoina varsinaisena vaatimuksena olivat, että käyttäjä pystyy luomaan käyttöliittymän ilman ohjelmointitarvetta sekä pystyy kommunikoimaan useamman laitteen kanssa samanaikaisesti.

Ohjelma toteutettiin Windowsille C#-ohjelmointikielellä. C# oli entuudestaan tuttu, ja se muistuttaa läheisesti C/C++ kieltä, jota käytän työssäni päivittäin.

Windows oli sen sijaan uusi tuttavuus. Vei jonkin verran aikaa miettiä, kuinka käyttöliittymä toteutetaan ja kuinka se saadaan kommunikoimaan useiden laitteiden kanssa niin, että mahdolliset viiveet eivät jumita varsinaista käyttöliittymää.

Opinnäytetyö antoi nähdäkseni aika kattavan leikkauksen PC-ohjelmointiin sekä erilaisten tekniikoiden käytöstä, kuten säikeistä, sanomajonoista, oheislaitteista ja UI-ohjelmoinnista.

Työ laajeni lopulta aiottua suuremmaksi, ja ajan säästämiseksi päätettiin hankkia kaksi komponenttia valmiina - grid control- ja chart-kirjastot. Toisaalta myös komponenttien integrointi oman sovelluksen osaksi vei oman aikansa.

Ohjelmaa testattiin jatkuvasti kehityksen edetessä manuaalisesti ja joillekin komponenteille tehtiin myös automaattitestejä. Tällöin mm. koodin optimointi on turvallisempaa, kun mahdolliset virheet jää heti kiinni.

LÄHTEET

(1) Modbus Organization. About modbus organization. WWW-dokumentti. Saatavissa: https://modbus.org/ [viitattu 16.4.2022].
(2) Modbus Organization. Modbus Application Protocol V1.1b3. PDF-dokumentti. Saatavissa: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf [viitattu 16.4.2022].
(3) Eastron SDM630-Modbus V2 User Manual V 1.6B.
(4) Modbus messaging on TCP/IP Implementation guide V1.0b, 2006. PDF-dokumentti. Saatavissa: https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf [viitattu 16.4.2022].
(5) Tom Archer, Inside c#, IT Press 2001, ISBN 951-826-455-5, 2001
(6) Riku Saikkonen, 2012. Ohjelmoinnin peruskurssin laaja oppimäärä. Luento 5, Serialisointi. PDF-dokumentti. Saatavissa: https://wiki.aalto.fi/download/attachments/63548818/luento5.pdf [viitattu 16.4.2022].
(7) XML Tutorial. WWW-dokumentti. Saatavissa: https://www.w3schools.com/xml/ [viitattu 16.4.2022].
(8) CSV file format. WWW-dokumentti. Saatavissa: https://docs.fileformat.com/spreadsheet/csv/ [viitattu 16.4.2022].
(9) Temperature and humidity transmitter SHT20, Sensor Modbus RS485. User Manual.
(10) Haikala, I &Märijärvi, J. 2004 Ohjelmistotuotanto. Talentum.
(11) Refactoring. WWW-dokumentti. Saatavissa: https://refactoring.com/ [viitattu 16.4.2022].
(12) ODVA, Inc. The Common Industrial Protocol (CIP) and the Family of CIP Networks. PDF-dokumentti. Saatavissa: https://www.odva.org/wp-content/uploads/2020/06/PUB00123R1_Common-Industrial_Protocol_and_Family_of_CIP_Networks.pdf [viitattu 16.4.2022]