**TEKNIIKKA JA LIIKENNE**

**Kone- ja tuotantotekniikka**

**Koneautomaatio**

**INSINÖÖRITYÖ**

# ETHERNET-POHJAISEN MODBUS-KORTIN HYÖDYNTÄMINEN QT CREATORILLA

**Työn tekijä: Tuomo Rinta-Paavola**
**Työn ohjaaja: Jari Savolainen**

**Työ hyväksytty: ___. ___. 2010**

**Jari Savolainen**
**lehtori**

**ALKULAUSE**

Tämä insinöörityö tehtiin Metropolia Ammattikorkeakoulun koneautomaation yksikölle. Haluan kiittää työn ohjaajaa lehtori Jari Savolaista sekä projekti-insinööri Markus Aunolaa, joka auttoi ohjelman luomisessa.

Helsingissä 31.5.2010

Tuomo Rinta-Paavola

**TIIVISTELMÄ**

**Työn tekijä:** Tuomo Rinta-Paavola

**Työn nimi:** Ethernet-pohjaisen Modbus-kortin hyödyntäminen Qt Creatorilla

**Päivämäärä:** 31.5.2010 **Sivumäärä:** 30 s. + 2 liitettä

**Koulutusohjelma:** Kone- ja tuotantotekniikka

**Suuntautumisvaihtoehto:** Koneautomaatio

**Työn ohjaaja:** Jari Savolainen

Tämä insinöörityö tehtiin Metropolia Ammattikorkeakoulun koneautomaation laboratorioon. Työn tarkoituksena oli selvittää Modbus-kortin käyttöominaisuuksia Qt Creatorissa, joka on avoimen lähdekoodin ohjelmointiympäristö. Aikaisemmin ei ole tehty juurikaan ilmaisia, graafisen käyttöliittymän sisältäviä, avoimen lähdekoodin sovelluksia Ethernet-pohjaiselle Modbus-väylälle eli Modbus/TCP:lle.

Aluksi perehdyttiin Qt Creatorin käyttöön ja selvitettiin, miten Modbus/TCP:n funktiokomennot toimivat Qt:ssä. Seuraavaksi tehtiin testiohjelma, jonka avulla selvitettiin Modbus/TCP-moduulin käyttömahdollisuuksia. Ohjelmasta kerrotaan sen perustoiminnot, joilla ohjataan Modbus/TCP-moduulin tuloja ja lähtöjä. Lisäksi kerrotaan miten Modbus-kirjastot asennetaan ja saadaan toimimaan Qt Creatorissa. Olisikin suositeltavaa, että työn lukijalla olisi perusteet C++-ohjelmoinnista.

Työn tuloksena rakennettin ohjelma, jolla pystyi lukemaan ja ohjaamaan Modbus/TCP-moduulin tuloja ja lähtöjä. Muutettiin moduulin tulojen ja lähtöjen tilaa. Piirrettiin graafinen kuvaaja Modbus-kortin analogitulon jännitteen perusteella. Selvitettiin myös Qwt-lisäosan hyödyntämistä ohjelmassa. Sen avulla piirrettiin analogiatulon jännitesignaalin arvosta graafinen kuvaaja ja tallennettiin tämä mittausdata tietokantaan. Pienellä jatkokehityksellä tehty sovellus pystyttäisiin muuttamaan yksinkertaiseksi mittausohjelmaksi.

Qt-tekniikoilla ja luokkakirjastoilla pystyy melko helposti tekemään graafisen käyttöliittymän sovelluksia. Avoimen lähdekoodin ohjelmien suosio tulee tulevaisuudessa todennäköisesti vain kasvamaan, mikä oli myös yksi syy käyttää Qt Creatoria tässä työssä.

**Avainsanat:** Qt Creator, Modbus/TCP, Qwt, Ethernet

Metropolia

**ABSTRACT**

**Name:** Tuomo Rinta-Paavola

**Title:** The Use of Ethernet based Modbus Module in Qt Creator

**Date:** May 31, 2010 **Number of pages:** 30 + 2

**Department:** Mechanical Engineering

**Study Programme:** Machine Automation

**Supervisor:** Jari Savolainen

This Bachelor's thesis was carried out at the machine automation laboratory of Metropolia University of Applied Sciences. The objective of this thesis was to examine the features of the Ethernet-based Modbus card and how to use it in Qt Creator, which is an open source programming environment. Previously there have not been many open source and graphical user interface applications for Modbus/TCP Protocol.

Firstly, the use of programming tools and Modbus/TCP card were examined and the principles of how to use Modbus function commands in Qt Creator were studied. Secondly, a test program which uses abilities of Modbus/TCP module was made. This study describes the basic functions of controlling Modbus/TCP module inputs and outputs. In addition, it is described how to install and link Modbus fuction libraries with Qt Creator. It is advisable that the reader of this study would be familiar with basics of C++ programming.

As a result, the built program is able to read input values and modify output values of the Modbus/TCP module. With Qwt addon it was possible to draw a graph of the input voltage signal with Qt Creator. The value of voltage signal was also stored in the database so that minor further development the program could be used as a simple measurement application.

Qt programming and its class libraries are quite good tools to create graphical user interface applications. In the future the popularity of open source applications is likely to grow even more, which was also one of the reasons to use Qt Creator in this thesis.

**Keywords:** Modbus/TCP, Qt Creator, Qwt, Ethernet

**SISÄLLYS**

**KÄYTETYT LYHENTEET**

| | |
|---|---|
| ASCII | *American Standard Code for Information Interchange* |
| CAN | *Controller Area Network* |
| CSMA/*CD* | *Carrier Sense Multiple Access with Collision Detection*; siirtotien monikäyttötekniikka |
| Ethernet/IP | *Ethernet/Industrial Protocol*; kenttäväyläratkaisu |
| Firmware | Laitteen sisäinen ohjelmisto |
| GUI | Graphical user interface |
| I/O | *Input/Output*; tulo/lähtö |
| IP | *Internet Protocol* |
| Kb/s | *Kilobits per second* |
| Mb/s | *Megabits per second* |
| RTU | *Remote Terminal Unit* |
| SQL | Tietokantakieli |
| SQLite | SQL-tietokanta |
| TCP | *Transmission Control Protocol*; kuljetuskerroksen tiedonsiirtoprotokolla |
| UDP | *User Datagram Protocol*; yhteydetön tiedonsiirtoprotokolla |

# 1 JOHDANTO

Tämä insinöörityö on tehty Metropolia Ammattikorkeakoulun koneautomaation laboratorioon. Työn aiheena oli selvittää Ethernet-pohjaisen TCP/IP-moduulin käyttömahdollisuuksia avoimen lähdekoodin ohjelmointiympäristössä, Qt Creatorissa. Aikaisemmin ei ole tehty graafisen käyttöliittymän Modbus/TCP-sovelluksia Qt Creatorilla. Tietoliikenneväylänä käytettiin siis Ethernet-pohjaista Modbus/TCP-väylää ja Telemerkki-nimisen yrityksen EFDC-moduulia.

Tavoitteena oli selvittää, miten Telemerkin EFDC-moduuli saadaan toimimaan tietokoneen kanssa ja miten sen Modbus-ominaisuuksia hyödynnettäisiin avoimen lähdekoodin ohjelmontiympäristössä. Työssä selostetaan miten Modbus-kirjastot asennetaan ja otetaan käyttöön Qt Creatorissa sekä millä funktioilla ohjataan kortin digitaalisia ja analogisia tuloja ja lähtöjä. Lisäksi selvitettiin Modbus-protokollan toimintaperiaatetta ja miten Qt Creatorin Qwt-lisäosaa pystyttäisiin hyödyntämään työssä.

Tällä hetkellä teollisuus-Ethernet-väyliä ja Modbus/TCP-väylää hyödyntäviä avoimen lähdekoodin sovelluksia ilmestyy nopeaa tahtia. Esimerkiksi tätä työtä tehtäessä ilmestyi toinenkin Qt Creatoria hyödyntävä Modbus-sovellus, QModBus. Se on kuitenkin pelkästään sarjaporttiin perustuva versio, eikä ole Ethernet-yhteensopiva.

# 2 MODBUS

Tässä luvussa käsitellään Modbus-tekniikan perusteita ja sen TCP/IP-protokollaa.

## 2.1 Taustaa

Modbus on alun perin Modicon-nimisen yrityksen, nykyään Schneider Electricin, vuonna 1979 julkaisema sarjaliikenneprotokolla, joka suunniteltiin alun perin käytettäväksi Modiconin ohjelmoitavien logiikkojen kanssa. Vuosien myötä siitä on tullut kuitenkin teollisuuden johtava standardi. Pääasialliset syyt tähän ovat sen avoimuus, lisenssimaksuttomuus sekä riippumattomuus laitevalmistajasta. Modbusin avulla useat samaan verkkoon kytketyt laitteet pystyvät kommunikoimaan keskenään. Siksi se soveltuukin hyvin valvontaan ja automaatiolaitteiden ohjaukseen.

Modbus-protokollasta on saatavilla sekä sarjaportti- että Ethernet-versiot. Nykyään sitä käytetään ennen kaikkea elektroniikkalaitteiden välisessä kommunikoinnissa, mutta myös esimerkiksi rakennuskohteissa ja pitkänmatkan tiedonsiirrossa. [6.]

## 2.2 Modbusin toimintaperiaate

Modbus on master/slave -protokolla, joka toimii half duplex periaatteella. Se siis lähettää ja vastaanottaa viestejä vuorotellen. Tietoliikenne toimii kiertokyselynä: isäntä lähettää pyynnön ja orjat vastaavat.

Modbus-tiedonsiirron täytyy olla jatkuvaa. Se rajoittaa etäyhteyksien tyypin sellaisiin, jotka voivat puskuroida dataa katkosten välttämiseksi. Modbus-protokolla määrittää kehykset tiedonsiirto ja -valvontafunktioille, mutta se ei kuitenkaan määrittele verkkojen välistä liikennöintitapaa. Modbus voi toimia useissa eri verkkokerroksissa. Esimerkiksi TCP-protokolla toimii kaikissa TCP/IP:tä tukevissa verkkokerroksissa, myös Ethernetissä.

Modbusista on saatavilla useita eri versiota: Modbus-TCP, -RTU (Remote Terminal Unit), -ASCII ja -Serialport. Niissä kaikissa on samat tietomallit ja

toimintokutsut. Tässä työssä kerrotaan pääasiassa vain Modbus TCP-protokollasta ja sen toiminnasta. [1; 6.]

## 2.3 Modbus TCP/IP-protokolla

Modbus/TCP on Schneider Electricin kehittämä, TCP/IP-pohjainen muunnos Modbus RTU-protokollasta. Sitä käytetään tiedonsiirtoprotokollana Ethernet TCP/IP -verkoissa. Protokolla välittää tietonsa TCP:n avulla. TCP-kehyksen sisällä on Modbus-kehys, joka puolestaan sisältää lähetettävän tiedon.

Protokolla käyttää binäärikoodattua dataa ja TCP/IP:n virheentunnistusmenetelmää tiedonsiirtovirheiden havaitsemiseen. Sen tiedonvälitys perustuu neljään viestityyppiin: pyyntö-, vahvistus-, osoitus- ja vastausviestiin. Orjalaite lähettää aluksi pyyntöviestin, isäntä vastaanottaa ja käsittelee sen osoitusviestinä. Seuraavaksi isäntä palauttaa orjalaitteelle vastausviestin, jonka orja käsittelee vastausviestinä.

Jos TCP/IP-yhteys aikakatkeaa tai järjestelmään tulee jostain syystä virhe, isäntä sulkee sekä avaa yhteyden uudelleen ja yrittää lähettää viestin uudelleen.

Modbus TCP/IP -väylän kautta kommunikoivan moduulin suurin etu on se, että se voidaan liittää tavanomaiseen Ethernet-verkkoon. Ethernet-verkon hyödyistä on kerrottu lisää luvussa 3. Sen suurin heikkous on puolestaan hidas jaksonaika, 5 - 10 millisekuntia, joka estää protokollan käytön reaaliaikaisissa sovelluksissa.

Protokollan tekniset tiedot ovat avoimia, mikä on yksi syy siihen, miksi Modbus/TCP on houkutellut toistasataa valmistajaa tukemaan sitä. Se onkin tällä hetkellä yksi käytetyimmistä teollisuuden Ethernet-järjestelmissä. Modbus/TCP toimii tavallisilla Ethernet-komponenteilla, joten esimerkiksi jo olemassa olevia kaapelointeja voidaan käyttää hyväksi. [13; 1, s. 122.]

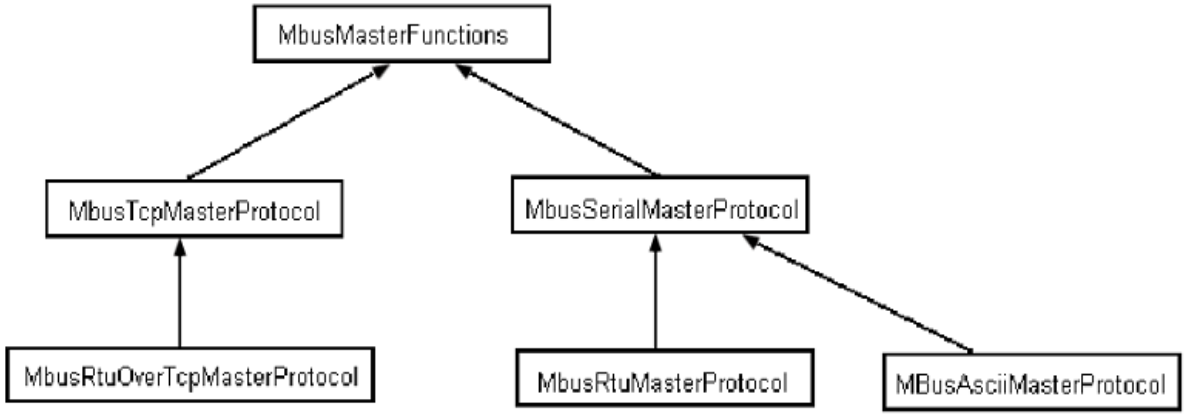
### *2.3.1 Orjalaitteen tunnistus*

Orjalaitteen pätevä osoite on väliltä 1 - 247. Jokainen Modbus-väylään liitettävä laite saa yksilöllisen osoitteen. Osoitetta nolla käytetään tiedon lähettämiseen isännältä kaikille verkkoon kytketyille orjalaitteille. Paluuviestissä orja kertoo myös dataviestin lisäksi osoitteensa, jotta isäntä tunnistaa mikä orja on kyseessä.

Jokainen väylään kytketty laite voi lähettää Modbus-komennon. Yleensä kuitenkin vain yksi isäntä toimii lähettäjänä. Modbus-komento sisältää vain sen laitteen osoitteen, jolle käsky on tarkoitettu. Vain tämä laite suorittaa komennon, vaikka kaikki laitteet voivatkin sen vastaanottaa. Kaikilla Modbus-komennoilla on myös tarkiste, jolla varmistetaan käskyn virheetön kulku. [6; 12, s. 19.]

### *2.3.2 Modbus-kirjaston protokollaluokat*

Tässä työssä käytetty Fieldtalk Modbus Master C++-kirjasto sisältää viisi erilaista luokkaa, yhden jokaiselle protokollalle, sekä pääluokan, joka pätee kaikille protokollatyypeille. Kuvasta 1 selviää, että *MbusMasterFunctions* on pääluokka, josta kaikki muut protokollatyypit periytyvät.




*Kuva 1. Modbus-kirjaston rakenne [1, s. 5]*

MbusMasterFunctions-luokka sisältää kaikki protokollien määrittelemättömät funktiot, esimerkiksi Modbusin määrittelemät data- ja ohjausfunktiot. [1.]

## 2.4 Rekisterimalli ja datataulut

Modbusin datafunktiot perustuvat rekisterimalliin. Se puolestaan perustuu taulusarjoihin, joilla on kullakin omat tunnuspiirteensä. Rekisteri on pienin osoitekokonaisuus Modbusissa. Taulukko 1:ssä on näytetty FieldTalk Modbus Master C++-kirjaston neljä datataulua.

*Taulukko 1. Datataulut [1, s. 119]*

| Taulu | Termi | Rekisteriosoite | Ominaisuudet |
|---|---|---|---|
| Discrete outputs | Coils | 0:00000 | 16-bittinen sana, sovelluksella muutettava tieto, luku/kirjoitus |
| Discrete inputs | Inputs | 1:00000 | yksi bitti, tulojen sekä lähtöjen tilatiedot, vain luku |
| Input registers | Input registers | 3:00000 | 16-bittinen sana, tulojen sekä lähtöjen tilatiedot, vain luku |
| Output registers | Holding registers | 4:00000 | yksi bitti, sovelluksella muutettava tieto, luku/kirjoitus |

Laitteen sovellusmuistiin sijoitetaan käsiteltävät tiedot, joista tehdään myös linkitys tietoviittauksilla laitteen fyysisiin osoitteesiin. Jokaiselle tiedolle on määritelty osoite väliltä 0 - 65535. Modbus-protokollan tietomallin rakentuu neljästä osasta, jotka sisältävät useita elementtejä. Käytettävän tietomallin toiminta perustuu neljään tauluun, joita ovat erilliset lähdöt (Discrete outputs), erilliset tulot (Discrete inputs), tulorekisteri (Input registers) ja pitorekisteritaulut (Output registers).

Yhteen tauluun voidaan määrittää 65536 erillistä tietoalkiota. Funktiokoodi määrittelee käsiteltävien tietoalkioden määrän luku- ja kirjoitustehtävissä. Modbus-protokollassa ei ole myöskään tulojen tai lähtöjen tietoalkioden välille määritelty mitään rajoja. Tauluja pystytään käsittelemään joko erillisinä tai yhtenä isona lohkona. [1; 10, s. 6 - 8.]

## 2.5 Rekisterien ja diskreettien numerointi

Modiconin PLC-rekisterit ja diskreetit järjestetään muistityypin mukaisesti. Esimerkiksi rekisteripaikassa 3:00001 olisi ensimmäinen viittaus Input-rekisteriiin. Sopiva rekisteritaulu valitaan käyttämällä ohjelmassa vastaavaa funktiokutsua, kuten taulukosta 2 selviää.

*Taulukko 2. Rekisteritaulukko [1, s. 120]*

| **Funktiokutsu** | **Rekisteriosoite** |
|---|---|
| readCoils(),<br>writeCoil(),<br>forceMultipleCoils() | 0:00000 |
| readInputDiscretes | 1:00000 |
| readInputRegisters() | 3:00000 |
| writeMultipleRegisters(),<br>writeSingleRegister(),<br>maskWriteRegister(),<br>readWriteRegisters() | 4:00000 |

Jos siis ohjelmassa halutaan esimerkiksi lukea digitaalisia tuloja, käytetään *readCoils*-funktiota, joka sijaitsee rekisteripaikassa 0:00000. Toisin kuin perinteisesti, Modbusin rekisterien numerointi ei ala 0:sta vaan 1:stä. Tämä tarkoittaa, että ensimmäinen pätevä rekisteriosoite on esimerkiksi 3:00001, eikä 3:00000. Modbusin diskreettien numerointi alkaa myös 1:stä, joka on merkitsevin bitti 16-bittisessä sanassa. Tämäkin eroaa selvästi perinteisestä ohjelmointitavasta, jossa ensimmäinen viittaus on nimetty nollaksi ja vähiten merkitsevä bitti on ensimmäinen bitti. [1.]

# 3 TEOLLISUUS-ETHERNET-VÄYLÄT

Modbus/TCP:n lisäksi on olemassa useita muitakin Ethernet-pohjaisia protokollia. Tässä luvussa on tehty pienimuotoista vertailua Ethernet-protokollien ja CAN-väylän välillä.

## 3.1 Ethernet-protokollat

Ethernet-pohjaisissa järjestelmissä on käytössä kymmeniä erilaisia protokollia. Tämä johtuu pääasiassa siitä, että eri valmistajat eivät ole päässeet yhteisymmärrykseen siitä, mitä protokollalta halutaan, ja lisäksi sillä on useita käyttökohteita, joten standardisointi ei ole helppoa.

Ethernet-järjestelmien yleisimmät protokollat ovat

- EtherCAT
- Ethernet Powerlink
- Ethernet/IP
- Modbus/TCP
- ProfiNet [9].

Eri protokollien ominaisuuksissa on suuria eroja, ja niiden soveltuvuus eri käyttökohteisiin vaihtelee suuresti. Jotkin protokollat soveltuvat esimerkiksi paremmin reaaliaikaisten sovellusten ohjaamiseen, toiset taas ovat helpommin laajennettavissa tai omaavat suuremman nopeuden.

Ethernet-pohjaisten järjestelmien edut verrattuna muihin väyliin, esimerkiksi CANiin ovat tiedonsiirron nopeus, olemassa olevien komponenttien käytettävyys ja erittäin pitkät tiedonsiirtomatkat.

Ethernet-pohjaisten järjestelmien heikkouksia ovat

- hitaus reaaliaikaisissa sovelluksissa
- TCP/IP-järjestelmän hallittavuus
- suurilla nopeuksilla alhainen tiedonsiirron tehokkuus [9].

Ainakin teoriassa Ethernet TCP/IP-pohjaiset järjestelmät voisivat olla satoja kertoja nopeampia kuin CAN-pohjaiset järjestelmät. Käytettäessä suuria nopeuksia kasvaa kuitenkin myös lähetettävän kehyksen koko, mikä tarkoit-

taa, että samalla myös tiedonsiirron tehokkuus kärsii. Käytännössä siis suurten tiedonsiirtonopeuksien tuomat edut menevät hukkaan.

Ethernet TCP/IP-pohjaiset eivät sovellu myöskään yhtä hyvin reaaliaikaisten sovellusten ohjaamiseen kuin esimerkiksi CAN tai CANopen, joskin viime vuosina on kehitetty reaaliaikaisuutta paremmin tukevia protokollia. Aikaisemmin Ethernet ja CAN eivät ole juurikaan kilpailleet samoilla markkina-alueilla, mutta uusien, reaaliaikaisten Ethernet-järjestelmien myötä tilanne on kuitenkin muuttumassa. [9.]

## 3.2 CAN- ja Ethernet-väylien tiedonsiirtotavat

### *3.2.1 Ethernet-väylän tiedonsiirtotapa*

Ethernet käyttää kaistanvarausmenetelmänään CSMA/CD-tekniikkaa (Carrier Sense Multiple Access With Collision Detection). Jos väylä on viesteistä vapaa eli mikään laite ei lähetä viestejä, on kaikilla laitteilla oikeus aloittaa lähettäminen. Jos taas useampi laite lähettää samanaikaisesti viestejä, tapahtuu viestien törmäys ja lähetys katkeaa. Viestien lähettämistä uudelleen yritetään satunnaisen ajan jälkeen. [4.]

### *3.2.2 CAN-väylä ja sen tiedonsiirtotapa*

CAN (Controller Area Network) on alun perin kehitetty autoteollisuuden tarpeisiin. Tämän takia se on luotettava ja sietää erinomaisesti häiriöitä. CAN-viestien rakenne ja lähetystapa eroaa merkittävästi Ethernet-pohjaisista väylistä: jokainen CAN-viesti sisältää tarkisteen, jolla varmistetaan sen perillemeno. CAN-viestit eivät sisällä lähettäjän tai vastaanottajan osoitetta vaan ne sisältävät ID-kentän, jonka perusteella tunnistetaan eri laitteille kulkevat viestit. Tästä syystä viestien koko on pystytty minimoimaan, jolloin saadaan aikaan suurempi tiedonsiirtokapasiteetti.

Usealle laitteelle kulkevaa viestiä ei tarvitse lähettää moneen kertaan, sillä väylällä kulkevat viestit ovat kaikkien laitteiden luettavissa. CAN-viestit priorisoidaan eli viestien törmäystilanteissa pienimmän ID-luvun omaava laite lähettää viestinsä ensimmäisenä ja muut vasta, kun väylä on jälleen vapaa. [7.]

# 4 EFDC-MODUULI

## 4.1 EFDC-moduulin tekniset tiedot

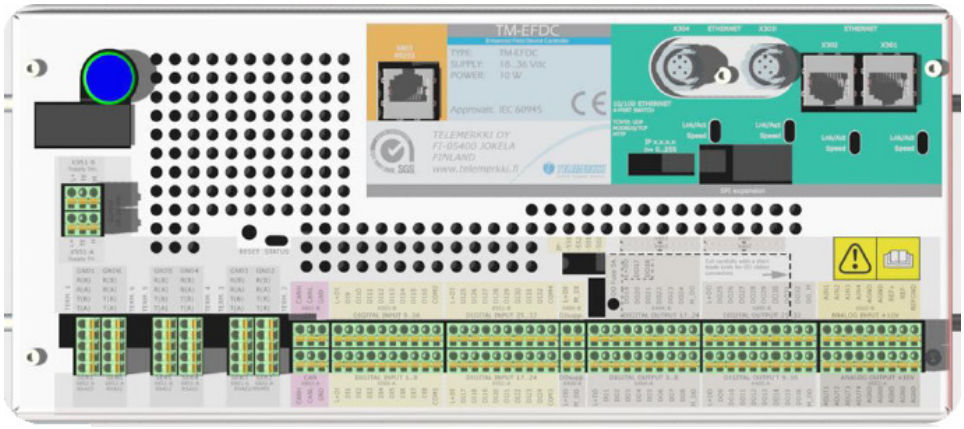
Tässä työssä ohjattiin Telemerkki-nimisen yrityksen valmistaman EFDC-moduulin tuloja ja lähtöjä. Telemerkin EFDC-moduuli on Ethernet-pohjainen I/O-yksikkö, jota voidaan käyttää esimerkiksi valvonta- ja ohjaustehtävissä. Alun perin se on suunniteltu käytettäväksi laivateollisuudessa, esimerkiksi laivojen komentosilloilla, mutta se soveltuu käytettäväksi myös teollisuus-automaatiossa.

EFDC-moduuli toimii 18-36 VDC:n jännitteellä. Siinä on kaksi erillistä syöttö-jännitteen sisäänmenoa sekä automaattinen syötönvaihto. Laite sisältää 32 digitaalista tuloa, 32 digitaalista lähtöä sekä 4 analogista tuloa ja 4 analogista lähtöä. Moduuli sisältää myös referenssijännitelähteen, johon voidaan kytkeä joko potentiometri tai anturi. Taulukossa 3 on esitelty moduulin tärkeimmät liitännät.

*Taulukko 3. Tekniset tiedot [2]*

| | |
|---|---|
| Digitaaliset tulot ja lähdöt | 32 sisäänmenoa ja 32 ulostuloa |
| Sarjamuotoisten tietojen käsittely | 6 kpl RS-422-portteja. Yhdessä portissa lisäksi RS-232-mahdollisuus |
| Analogiset tulot ja lähdöt | 4 analogista sisäänmenoa ja 4 ulostuloa |
| Ethernet-liityntä 10/100 | 5-porttinen Ethernet-kytkin (4 ulkoista) |
| CAN-Bus-liityntä | 1 CAN-liitäntä. Galvaaninen erotus |

Laitteen digitaalisia ja analogisia I/O:ita voidaan ohjata joko Modbus- tai TCP-protokollan avulla. Moduuli sisältää oman sisäänrakennetun Modbus/TCP-palvelimen, jolla voidaan sekä lukea digitaalisia ja analogisia tuloja että muokata vastaavia lähtöjä. Muunnosalue analogisille tuloille ja lähdöille on ±10 volttia ja sen muunnostarkkuus on 16 bittiä. TM-EFDC-moduulissa on sisäänrakennettu 5-porttinen Ethernet-kytkin, jonka avulla voidaan linkittää usempi EFDC-moduuli yhteen ilman erillistä Ethernet-kytkintä. Kuvassa 2 on EFDC-moduuli kuvattuna ylhäältäpäin. [2, s. 1.]




*Kuva 2. Telemerkin EFDC-moduuli*

## 4.2 Web-konsoli

EFDC-moduuli sisältää myös selainpohjaisen konfigurointiohjelman, jonka avulla voidaan lukea tulojen ja lähtöjen tila. Kun laite on kytketty verkkokaapelilla tietokoneeseen, päästään Internet-selaimen avulla muokkaamaan laitteen asetuksia. Internet-selaimen osoitekenttään kirjoitetaan EFDC-palvelimelle asetettu IP-osoite, jolloin päästään sisäänkirjautumisen jälkeen konsolin aloitusvalikkoon. Useimmat konsolin toiminnoista ovat muokattavissa myös tässä työssä Qt Creatorilla rakennetun ohjelman kautta, josta lisää luvussa 7. Web-konsolin käyttöliittymä on salasanasuojattu. Käyttäjänimi ja salasana ovat kovakoodattuja, eikä niitä voida vaihtaa ilman firmwaren päivitystä. Kuvassa 3 on internet-selaimessa näkyvä konsolin aloitusvalikko. [2.]


*Kuva 3. Web-konsolin aloitusvalikko*

### 4.3 EFDC-moduulin liittäminen tietokoneeseen

Telemerkin EFDC-moduuli liitetään 24 V:n virtalähteeseen ja yhdistetään ristikaapelilla Ethernet-liitännästä tietokoneeseen. Tietokone toimii isäntänä ja EFDC orjana. TCP-palvelintilassa EFDC-moduuli odottaa passiivisesti, kunnes isäntä ottaa siihen yhteyden. Kun tietokone on luonut yhteyden, voidaan tiedonsiirto aloittaa. TCP-tila tukee jopa neljää samanaikaista yhteyttä orjalaitteelle, jolloin useat isännät voivat kerätä dataa samalta orjalaitteelta. [2, s. 28.]

Tietokoneen IP-osoite on vaihdettava, jotta moduuliin saataisiin luotua yhteys. Osoitteen voi vaihtaa joko verkkoasetusten tai komentorivin kautta. Yleensä vaihtaminen onnistuu helpoiten komentorivin kautta. Tässä työssä on selostettu IP-osoitteen vaihtaminen vain Linux-ympäristössä.

Taulukosta 4 nähdään kaikki sallitut IP- ja Netmask-osoitteet isännälle, eli tässä tapauksessa tietokoneelle.

*Taulukko 4. Sallitut IP-arvot [2 s. 28]*

| **Sallittu isäntä** | **IP / Netmask** |
|---|---|
| Mikä tahansa | 0.0.0.0/255.255.255.255 (kaikille yhteyksille) |
| 192.168.137.100 | 192.168.137.100/255.255.255.255 |
| 192.168.137.1 - 192.168.137.254 | 192.168.137.0/255.255.255.0 |
| 192.168.0.1 - 192.168.255.254 | 192.168.0.0/255.255.0.0 |
| 192.168.137.1 - 192.168.137.126 | 192.168.137.0/255.255.255.128 |
| 192.168.137.129 - 192.168.137.254 | 192.168.137.128/255.255.255.128 |

Esimerkki IP-osoitteen vaihtamisesta. Kirjoitetaan komentoriville:

```
sudo ifconfig eth0 192.168.137.11 netmask 255.255.255.0
```

eth0 on Ethernet-kortin nimi, jolle annetaan uusi IP-osoite. Korttien numerointi alkaa nollasta ja toisen kortin nimi olisi vastaavasti eth1. Sallittu IP-osoite riippuu Netmaskin arvosta. Tässä tapauksessa IP-osoite voi olla väliltä 192.168.137.1 – 192.168.137.254. Netmask on TCP/IP-protokollan asetuksissa oleva kohta, joka jakaa IP-osoitteen verkko- ja koneosoitteeksi. Yleinen arvo on 255.255.255.0, joka jättää viimeisen kahdeksan bitin joukon koneosoitetta varten.

IP-osoite on nyt vaihdettu, minkä jälkeen internet-selaimen osoiteriville kirjoitetaan:

```
192.168.137.1
```

Selain kysyy käyttäjänimeä ja salasanaa. Oletusarvot ovat:

```
Username: admin
Password: admin
```

Onnistuneen kirjautumisen jälkeen näkyy web-konsolin aloitusvalikko ja yhteys on luotu. Tietoa voidaan nyt siirtää orjalta isännälle ja toisinpäin.

# 5 OHJELMOINTITYÖKALUT

Tässä työssä käytettiin Modbus-funktioiden hyödyntämiseen Qt Creatoria, jolla tehtiin Modbus-väylää hyödyntävä sovellus. Myös kaikki myöhemmin näkyvät koodiesimerkit on tehty Qt:lla. Qwt-lisäosaa käytettiin puolestaan graafisen kuvaajan piirtämisessä.

## 5.1 Qt Creator

Qt Creator on alustariippumaton ohjelmistojen ja graafisten käyttöliittymien kehitysympäristö. Se on vapaa, avoimen lähdekoodin ohjelmisto. Pohjimmiltaan Qt on C++-luokkakirjasto, jossa syntyvät tiedostot ovat samoja kuin normaalissakin C++-ympäristössä. On kuitenkin muutama poikkeus, kuten projektitiedosto ja xml-resurssitiedostot. Qt on alun perin norjalaisen Trolltechin kehittämä ohjelmisto. Nokia osti Trolltechin vuonna 2008 ja nykyisin yritys toimii osana Nokiaa Qt Development Frameworks -nimellä.

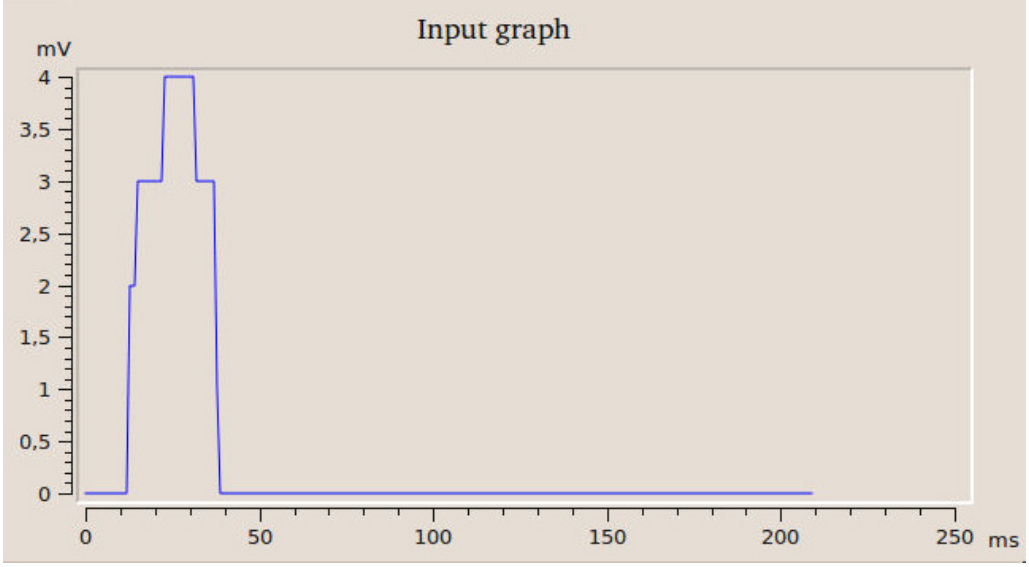
Qt on julkaistu seuraaville käyttöjärjestelmille:

- Linux/X11 – X Window System (Unix / Linux)
- Mac OS X – Apple Mac OS X
- Windows – Microsoft Windows
- Embedded Linux – sulautetut Linux-pohjaiset käyttöjärjestelmät
- Windows CE – Microsoft Windows CE [3; 11.]

## 5.2 Qwt

Qwt on ilmainen avoimen lähdekoodin laajennus Qt Creatorin GUI (Graphical User Interface) -kirjastoon. Qwt:n kirjasto sisältää komponentteja esimerkiksi 2D-grafiikan piirtämiseen sekä useita erilaisia säätimiä, kuten liukupalkkeja. Qwt-laajennuksen avulla voidaan Qt Creatorilla tehdä sovelluksia, jotka piirtävät esimerkiksi mittaustuloksista reaaliaikaista grafiikkaa. Sitä voidaankin käyttää esimerkiksi teknisissä ja tieteellisissä sovelluksissa. Tällä hetkellä sille ei ole saatavilla muita käyttödokumentteja kuin funktioluettelot, mikä tekee siitä melko hankalan käyttää. Se on kuitenkin erittäin monipuolinen ja monikäyttöinen lisäosa Qt:hen, mutta sen käyttö vaatii vahvaa koke-

musta ohjelmoinnista ja Qt:sta. Kuvassa 4 on Qwt:llä tehty graafinen kuvaaja. [8.]



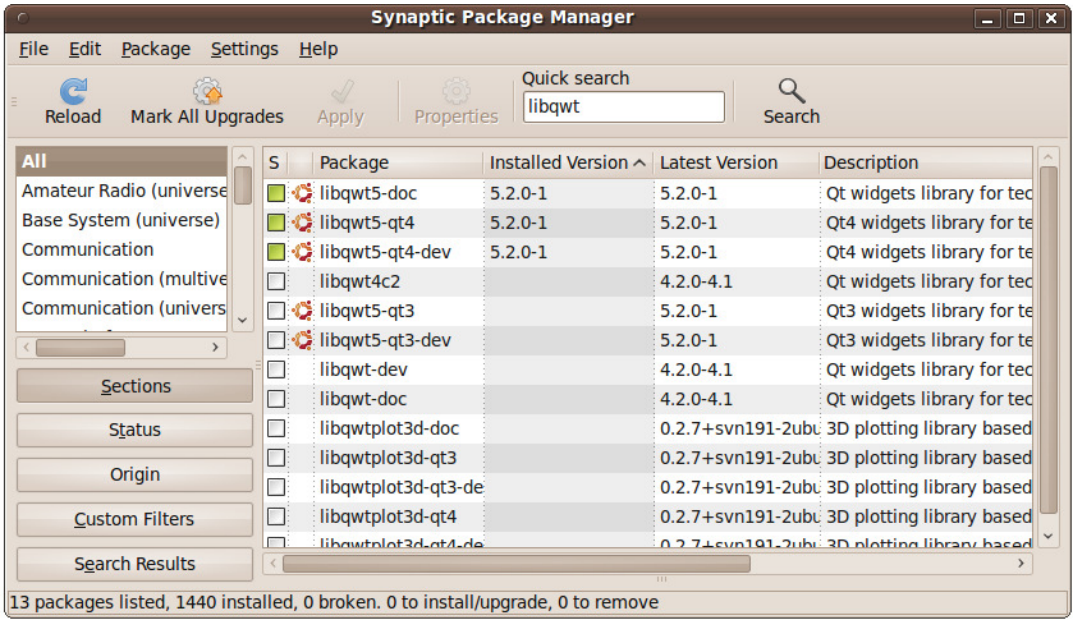

*Kuva 4. Graafinen kuvaaja tehtynä Qwt:llä*

### 5.3 Qwt:n asennus ja linkitys

Toimiakseen Qwt vaati Qt Creatorin asennettuna tietokoneen kovalevylle. Qwt toimii sekä Linux- että Windows-ympäristöissä. Seuraavassa on selostettu asentaminen vain Ubuntuun, joka on Linux-käyttöjärjestelmä. Asennusohjeet muille käyttöjärjestelmille ovat saatavissa lähdeluettelosta löytyvästä internet-osoitteesta. Ubuntussa Qwt:n asennus onnistuu helpoiten Synapticin paketinhallintaohjelmalla, jonka kautta sen pystyy asentamaan muutamalla klikkauksella:

Kirjoitetaan paketinhallintaohjelman hakukenttään *libqwt.* Löytyvistä qwt -tiedostoista valitaan kaksi tiedostoa (kuva 5):

```
libqwt5-qt4-dev        (development)
libqwt5-qt4            (runtime)
```




*Kuva 5. Qwt:n asennus Synaptic -paketinhallintaohjelmalla*

Asennetaan valitut tiedostot, minkä jälkeen Qwt-laajennus on liitetty osaksi Qt:ta ja Qwt:n lisäominaisuudet ilmestyvät Qt Creatorin työkaluvalikkoon. Qwt pitää vielä linkittää Qt Creatorilla tehtävän ohjelman projektitiedostoon, jotta sen ominaisuudet toimisivat.

### *Qwt:n linkitys Qt Creatoriin*

Qwt pitää linkittää Qt Creatorissa tehtävän ohjelman projektitiedostoon. Linkitys on erittäin tärkeää ja se on tehtävä huolellisesti. Tiedostopolkujen oikeellisuus on tarkastettava huolella, mikäli lisäosa ei jostain syystä toimi tai ilmenee ongelmia.

Projektitiedostoon lisätään rivit:

```
INCLUDEPATH += /(asennushakemisto)/qwt-qt4
LIBS += -L/usr/lib -lqwt-qt4
```

Qwt on nyt linkitetty osaksi Qt Creatoria, joka voi nyt käyttää kaikkia sen ominaisuuksia.

# 6 MODBUS-KIRJASTOJEN ASENNUS

Ennen kuin Modbus-kirjastojen funktiota voidaan käyttää Qt Creatorissa, täytyy ne asentaa käyttöjärjestelmään ja linkittää ohjelma niihin. Tästä lisää seuraavaksi.

## 6.1 Asentaminen ja lähdekoodin käyttö

Kirjastofunktiot saatiin FieldTalk Modbus Master C++ Librarysta. Kirjasto voidaan asentaa useisiin käyttöjärjestelmiin kuten esimerkiksi Linux- ja Windows-ympäristöihin sekä muutamiin vain teollisuuskäyttöön tarkoitettuihin käyttöjärjestelmiin. Seuraavassa on selostettu asentaminen Linux- ja Windows-käyttöjärjestelmiin.

### *6.1.1 Asennus Linuxiin*

Tiedostot *FT-MBMP-LX-ALL.2.4.1.tar.gz* ja *FT-MBSV-LX-ALL.2.4.0.tar.gz* puretaan. Purkamisen jälkeen tiedostopuu näyttää seuraavalta:

```
omaprojekti
|
+-- Fieldtalk_Master
   |
   +-- doc
   +-- include
   +-- samples
   +-- src
```

Käännetään kirjasto lähdekoodista:

```
# cd (asennushakemisto)/src/
# ./make
```

Make-komennolla tunnistetaan käyttöjärjestelmä, suoritetaan kääntäjä ja tehdään linkitys. Nyt tiedostorakenne näyttää seuraavalta:

```
omaprojekti
|
+-- fieldtalk
   |
   +-- doc
   +-- src
   +-- include
   +-- samples
   +-+ lib
     |
     +-- linux
```

Kirjasto on valmis käytettäväksi. [1, s. 123.]

### *6.1.2 Asennus Windowsiin*

Asennus Windows-ympäristöön on samantyylinen kuin Linuxilla. Puretaan tiedostot *FT-MBMP-WIN-ALL.2.4.1.zip*. ja *FT-MBSV-WIN-ALL.2.4.0.zip*, minkä jälkeen käännetään kirjasto lähdekoodista. Tiedostopuu on samanlainen kuin Linux-ympäristössä. Kääntäminen voidaan suorittaa esimerkiksi Microsoft C++:n, Visual Studion tai Borland C++:n avulla.

Käytettäessä Microsoft C++:aa ja nmakea kirjoitetaan komentoriville:

```
# cd (asennushakemisto)\fieldtalk\src
# nmake
```

Kääntämisen jälkeen tiedostopuu näyttää seuraavalta:

```
omaprojekti
|
+-- fieldtalk
    |
    +-- doc
    +-- src
    +-- include
    +-+ lib
      |
      +-- win32_xx   (nimi riippuu kääntäjästä)
```

Kirjasto on valmis käytettäväksi. [1, s. 124.]

## 6.2 Ohjelman linkitys kirjastoon

Asennuksen jälkeen ohjelma ja Modbus-kirjastot täytyy vielä linkittää toisiinsa sekä lisätä Qt Cretorilla tehtävän ohjelman projektitiedostoon kirjaston asennuspolku.

### *6.2.1 Linkitys Linuxiin*

Asennuksen jälkeen ohjelma linkitetään Modbus-kirjastoon. Kirjaston hakemisto lisätään kääntäjän polkuun. Kirjoitetaan komentoriville:

```
# c++ -I fieldtalk/include -c ohjelma.cpp
```

Seuraavaksi kirjaston nimi (*limbusmaster.a*) lisätään ohjelmaan, joka halutaan linkittää kirjastoon.

Esimerkiksi:

```
c++ -o ohjelma ohjelma.o fieldtalk/lib/linux/libmbusmaster.a
```

### *6.2.2 Linkitys Windowsiin*

Kirjaston hakemisto lisätään kääntäjän polkuun. Esimerkki käytettäessä Visual C++:aa:

```
cl -Ifieldtalk/include -c ohjelma.cpp
```

Kirjaston nimi lisätään ohjelmaan, joka linkitetään kirjastoon. Koska käytetään Modbus/TCP-protokollaa, täytyy lisätä myös Winsock2-kirjasto:

```
cl -Fe ohjelma ohjelma.obj field-
talk/lib/win/win32/release/libmbusmaster.lib Ws2_32.lib
```

### *6.2.3 Linkitys Qt Creatoriin*

Qt Creatorin projektitiedostoon lisättävät rivit:

```
INCLUDEPATH += /(asennushakemisto)/fieldtalk_2.4.1/include/
LIBS += -L/(asennushakemisto)/fieldtalk_master/lib/linux -
lmbusmaster
```

Ohjelma on nyt linkitetty Modbus-kirjastoon ja Qt Creator voi käyttää Modbus-käskyjä sekä toimia masterina. [1, s. 123.]

# 7 OHJELMAN TOIMINTAPERIAATE

## 7.1 Modbus ja Qt Creator

Linkityksen jälkeen selvitetään, miten Qt Creatorin käskyt toimivat ymmärrettävästi Telemerkin EFDC-moduulissa. Modbus-kirjaston tiedosto *FT-MBMP-LX-ALL.2.4.1.tar.gz* puretaan, ja se sisältää dokumentin nimeltä *limbus-master.pdf*, jossa on Modbus-standardien mukaan tehtyjä esimerkki-funktioita.

Jotta TCP-protokollan funktiokutsut toimisivat Qt Creatorissa, täytyy Qt:llä luotavan ohjelman otsikkotiedostoon lisätä:

```
#include <MbusTcpMasterProtocol.hpp>
```

ja tehdä siitä vielä olio:

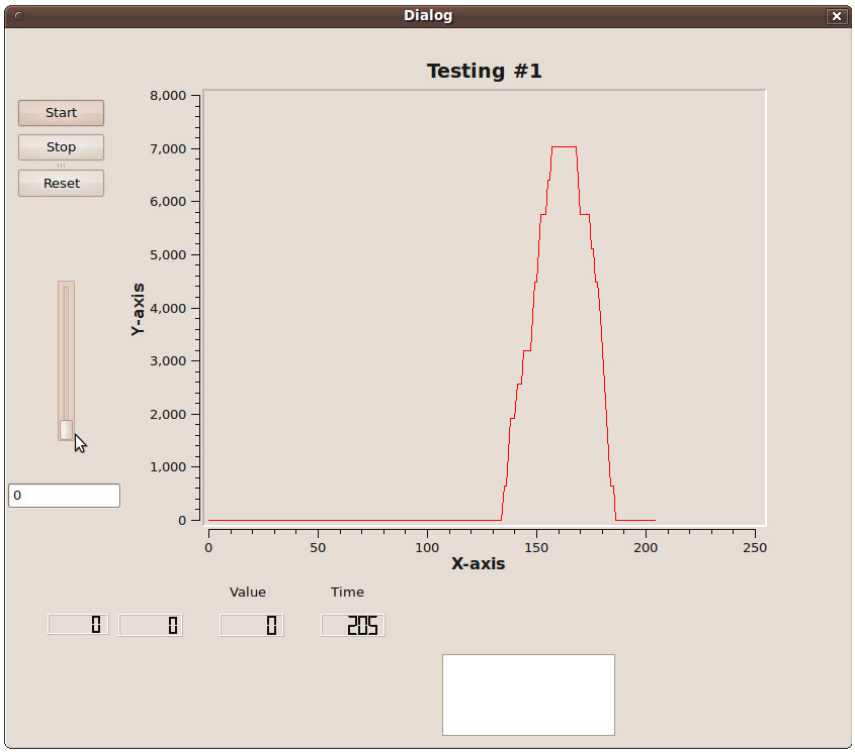
```
MbusTcpMasterProtocol mbusProtocol;
```

funktion kutsuesimerkki:

```
mbusProtocol.writeCoil(…);
```

## 7.2 Ohjelman kehittäminen

Rakennettiin Qt Creatorilla ohjelma, jonka avulla pystyttiin ohjamaan Telemerkin Modbus-kortin tuloja ja lähtöjä. Aluksi tehtiin muutamia testiversiota, joilla perehdyttiin Modbus-funktioiden käyttöön ja graafisen kuvaajan piirtämiseen ilman väyläliitäntää. Koska Qwt:n käytöstä ei ollut saatavilla käytännössä yhtään dokumenttia, kului graafisen kuvaajan piirtokoodin tekemiseen yllättävän paljon aikaa. Kuvassa on 6 testisovellus, jolla kokeiltiin aluksi pelkästään graafisen kuvaajan piirtoa.

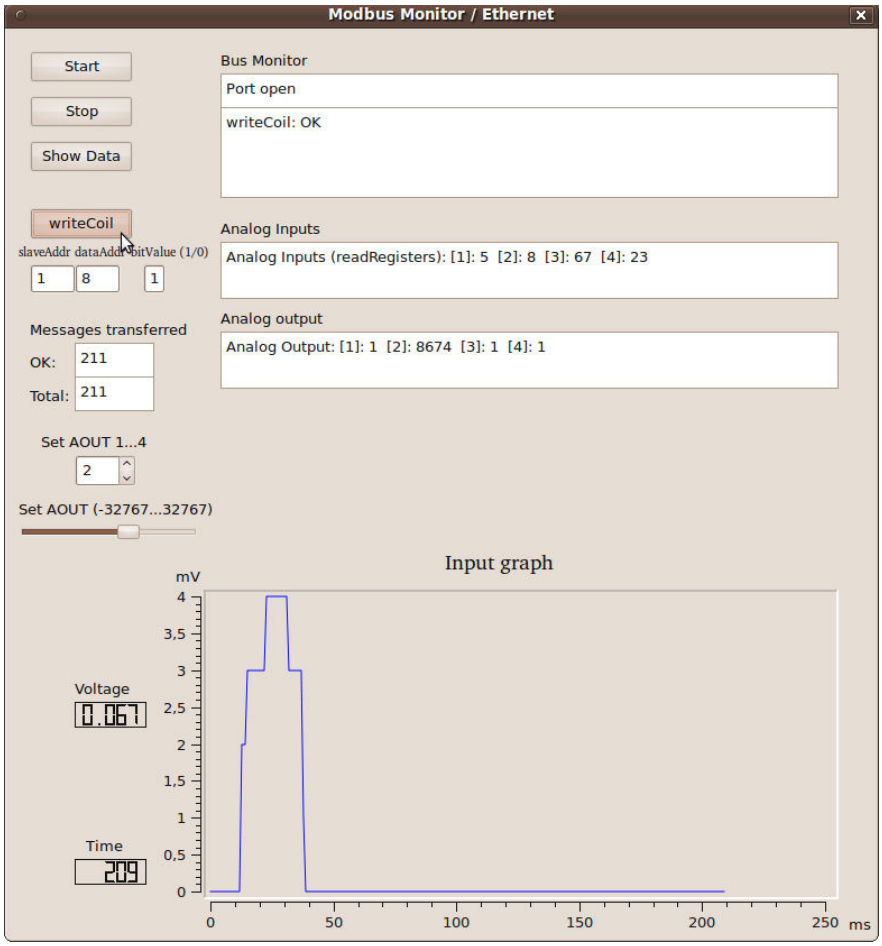



*Kuva 6. Ensimmäisiä ohjelmaversioita: graafisen kuvaajan piirtoa Qwt:llä*

Kun kuvaaja saatiin onnistuneesti piirrettyä, lisättiin ohjelma tukemaan Modbus-väylää ja Telemerkin EFDC-moduulia. Ennen kuin Modbus-komentoja pystyttiin käyttämään, täytyi kuitenkin muodostaa yhteys isännän ja orjan välille.

Onnistuneen yhteydenluonnin jälkeen pystyttiin käyttämään Modbus-funktioita ja lisäämään ohjelmaan uusia ominaisuuksia. Yksi tärkeimmistä oli analogisen tulon lukeminen, jonka pohjalta graafinen kuvaaja piirrettäisiin. Ohjelmaan tehtiin myös muita ominaisuuksia, kuten digitaalisten lähtöjen muokkaaminen ja analogialähdön jännitteen muuttaminen. Digitaalilähtöjen muokkaus onnistui writeCoil-komennolla, jolla pystyi muuttamaan yksittäisten bittien tilaa.

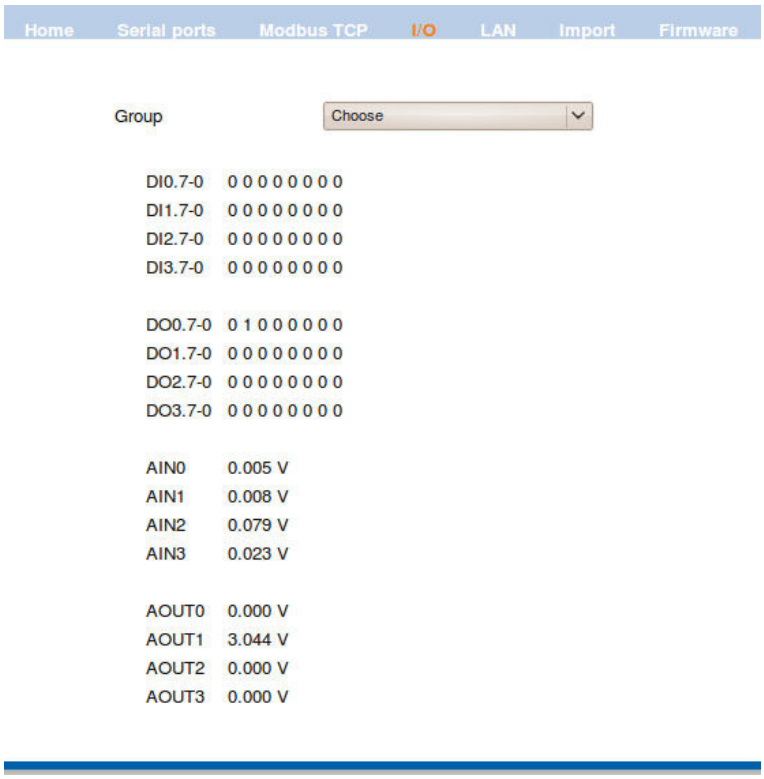
Tuloksena oli yksinkertainen Modbus-väylää hyödyntävä sovellus. Jos EFDC-moduuliin liitettäisiin esimerkiksi paine- tai voima-anturi, tulisi ohjelmasta periaatteessa yksinkertainen mittaussovellus, joka osaisi tallentaa datan tietokantaan. Qwt:n avulla voidaan piirtää graafista kuvaajaa analogiatulon tai -lähdön arvoista. Kuvassa 8 on valmis Modbus-väylää käyttävä sovellus.




*Kuva 8. Valmiin ohjelman käyttöliittymä*

Qt Creatorilla tehdyn ohjelman avulla pystyttiin siis sekä lukemaan moduulin tuloja että muuttamaan lähtöjen tilaa. Tehdyt muutokset pystyttiin vielä varmistamaan web-konsolin kautta. Qt:llä tehty ohjelma pystyi myös suoraan näyttämään analogiatulon jännitteen millivoltteina kokonaislukujen sijaan.

Analogiatulon jännitesignaalin ohjaus onnistui kytkemällä moduulin referenssijännitelähteeseen potentiometri. Potentiometriä käytettiin säätövastuksena, jolloin saatiin liukukosketinta säätämällä kytkentäpisteiden välille resistanssi 0:n ja 10 V:n väliltä. Qwt-lisäosaa hyödyntävä testiohjelma pystyi piirtämään ”mittausdatasta” graafisen kuvaajan. Mittaustulokset tallennettiin yksinkertaiseen tietokantaan. Kuvassa 9 EFDC:n tulot ja lähdöt nähtynä web-konsolin kautta.




*Kuva 9. EFDC- moduulin tulot ja lähdöt web-konsolin käyttöliittymässä*

### 7.3 Käytetyt funktiot

Modbus-protokollassa isäntä käskee orjaa toteuttamaan tietyn tehtävän funktiokoodin kautta. Funktiokoodi on yhden tavun kokoinen kokonaisluku väliltä 1-255. Suurin osa funktiokoodista on ohjelmoitu toimimaan useassa eri ympäristöissä ja laitteessa. [12, s. 19.]

Funktiokoodit käskevät esimerkiksi orjalaitetta, eli tässä tapauksessa EFDC-moduulia, muuttamaan digitaalisen lähdön eli tietyn bitin tilaa tai lukemaan vaikkapa analogiatuloa. Orja lukee halutun muistipaikan, osoitteen tai tulon ja palauttaa sen arvon isännälle.

Aluksi pitää avata Modbus-yhteys, jonka jälkeen pystytään vasta käyttämään Modbus-funktioita.

Taulukossa 5 on esitelty openProtocol-funktion parametrit ja käyttö.

*Taulukko 5. openProcol [1, s. 92]*

| | |
|---|---|
| **Funktion nimi** | openProtocol |
| **Käyttö** | int openProtocol (const TCHAR _const hostName) |
| **Parametrit** | **hostName** IP-osoite tai isännän nimi |
| **Palauttaa** | FTALK_SUCCESS onnistumisesta, muuten virheilmoitus |

Funktio openProtocol luo yhteyden isännän ja orjalaitteen välille. Yhteyden muodostuksen jälkeen kaikki data- ja valvontafunktiot ovat käytettävissä. Jos yhteyttä ei kuitenkaan pystytä muodostamaan 1000 millisekunnin aikana, se aikakatkaistaan. Kyselyssä määritellään isännän IP-osoite tai nimi. [1, s. 92.]

Koodiesimerkki 1:ssä näytetään yhteyden muodostus Qt Creatorissa:

*Koodiesimerkki 1.*

```
void openModbus()
{
int result;
#if defined (_UNICODE)
strcpy(hostName, "192.168.137.1");
#endif
result = mbusProtocol.openProtocol(hostName);
if (result != FTALK_SUCCESS)
            {...
            }
            else
            {…
            }
}
```

Kutsutaan oliolla *mbusProtocol MbusTcpMasterProtocol*-luokan openProtocol-funkiota. Otetaan yhteys isäntälaitteeseen, ja mikäli yhteyttä ei pystytä muodostamaan antaa result-muuttuja virheilmoituksen.

Vastaavasti *closeProtocol*-funktio sulkee TCP/IP-yhteyden orjalaitteeseen ja vapauttaa käytössä olleet resurssit.TCP/IP-yhteys on suositeltavaa sulkea käytön jälkeen, sillä se varaa melko paljon tietokoneen resursseja.

Taulukossa 6 on esitelty readInputRegisters-funktion parametrit ja käyttö.

*Taulukko 6. InpuRegisters [1, s. 17]*

| | |
|---|---|
| **Funktion nimi** | readInputRegisters |
| **Käyttö** | int readInputRegisters (int slaveAddr, int startRef, short regArr[ ], int refCnt) |
| **Parametrit** | • ***slaveAddr*** Orjalaitteen Modbus-osoite tai tunniste (Alue: 1 - 255)<br>• ***startRef*** Rekisteriosoite (Alue: 1 - 0x10000)<br>• ***regArr*** Datapuskuri; täytetään luettavalla datalla<br>• ***refCnt*** Luettavien rekisterien määrä (Alue: 1 - 125) |
| **Palauttaa** | FTALK_SUCCESS onnistumisesta, muuten virheilmoitus |

Funktio lukee valitun analogiatulon arvon. Se lukee Input-rekisterien sisällön rekisteripaikasta 3:00000. Kyselyssä määritellään orjalaitteen osoite, mistä rekisteriosoitteesta lukeminen aloitetaan, datapuskurin koko sekä luettavien rekisterien määrä. [12, s. 28.]

Koodiesimerkki 2 näyttää, miten analogiatuloja luetaan Qt Creatorissa:

*Koodiesimerkki 2.*

```
void CModbus::readAnalogInputRegisters(short* regArr,)
int result;
result= mbusProtocol.readInputRegisters(1,1,regArr,4);
if (result == FTALK_SUCCESS)
…
}
```

Funktion parametrien arvot ovat tässä tapauksessa seuraavat: Orjalaitteen osoite (slaveAddr) on 1, startRef 1, datapuskurin kooksi on määritetty 16 (regArr) ja luettavien rekisterien määrä on 4. Result-muuttujalla tehdään virheentarkistus. WriteMultipleRegisters [1, s. 18]

Taulukossa 7 on esitelty WriteMultipleRegisters-funktion parametrit ja käyttö.

*Taulukko 7. WriteMultipleRegisters [1, s. 18]*

| | |
|---|---|
| **Funktion nimi** | writeMultipleRegisters |
| **Käyttö** | int writeMultipleRegisters (int slaveAddr, int startRef, const short regArr[],int refCnt) |
| **Parametrit** | • ***slaveAddr*** Orjalaitteen Modbus-osoite tai tunniste (Alue: 0 - 255)<br>• ***startRef*** Rekisterisoite (Alue: 1 - 0x10000)<br>• ***regArr*** Puskuri lähetettävälle datalle<br>• ***refCnt*** Kirjoitettavien rekisterien määrä (Alue: 1 - 100) |
| **Palauttaa** | FTALK_SUCCESS onnistumisesta, muuten virheilmoitus |

Funktio muokkaa valitun analogialähdön arvoa eli se muuttaa Output-rekisterien arvoja. Kyselyssä määritellään orjalaitteen osoite, mistä rekisteriosoitteesta lukeminen aloitetaan, datapuskurin koko sekä kirjoitettavien rekisterin määrä.

Lähdön arvo pakotetaan annettuun tilaan, riippumatta siitä mikä sen tila on ollut ennen käskyn vastaanottamista. Vastausviesti on sama kuin isännän lähettämä asetuskäsky. [12, s. 34.]

Koodiesimerkki 3 näyttää, miten analogialähtöjä muokataan Qt Creatorissa:

*Koodiesimerkki 3.*

```
void CModbus::writeAnalogOutputs( int* pwm_par1, int* out1)
{
result = mbusProto-
col.writeMultipleRegisters(1,513,dataArr,sizeof(dataArr) /
sizeof(short));
}
```

Taulukossa 8 on esitelty writeCoil-funktion käyttö ja parametrit.

*Taulukko 8. writeCoil [1, s. 16]*

| | |
|---|---|
| **Funktion nimi** | writeCoil |
| **Käyttö** | int writeCoil (int slaveAddr, int bitAddr, int bitVal) |
| **Parametrit** | • ***slaveAddr*** Orjalaitteen Modbus-osoite tai tunniste (Alue: 0 - 255)<br>• **bitAddr** Bitin osoite (Alue: 1 - 0x10000)<br>• **bitVal** Bitin arvo (tosi/epätosi) |
| **Palauttaa** | FTALK_SUCCESS onnistumisesta, muuten virheilmoitus |

Funktio muuttaa digitaalisten lähtöjen tilaa. Se muuttaa kerrallaan vain yhden bitin tilan joko ON- tai OFF-tilaan. Kyselyssä määritellään orjalaitteen osoite, kirjoitettavan bitin osoite sekä bitin arvo, joka on joko 1 tai 0.

Koodiesimerkki 4 näyttää miten, yksittäisten bittien tilaa muutetaan:

*Koodiesimerkki 4.*

```
void CModbus::writeCoil(, int* coil_par1, int* coil_par2, int*
coil_par3)
{
int result;
result = mbusProto-
col.writeCoil(*coil_par1,*coil_par2,*coil_par3);
```

Arvot muuttujille *coil_par1 (slaveAddr), coil_par2 (bitAddr)* ja *coil_par3 (bitVal)* haetaan ohjelman käyttöliittymän tekstikenttään syötetyistä arvoista.

## 7.4 Mittaustulosten tallennus tietokantaan

Samalla kuin analogiatulon jännitesignaalin perusteella piirretään graafista kuvaajaa, tallennetaan arvo myös yksinkertaiseen tietokantaan. Aina 50 millisekunnin välein päivitetään graafista kuvaajaa ja tehdään uusi lisäys tietokantaan.

### *7.4.1 SQLite*

SQLite on relaatiotietokanta, jossa vain yhden tiedoston sisältävää tietokantaa operoidaan suoraan käytettävästä sovelluksesta. SQLite-tietokanta linkitetään sitä tarvitsevaan ohjelmaan, jolloin ei esimerkiksi tarvita erillisiä hallintaohjelmia tai tietokantapalvelimia. Sqlite-kirjastoa voidaan vapaasti muokata ja levittää eteenpäin, eikä sen linkittämiseen muiden sovellusten kanssa ole mitään estettä. Sitä käytetään esimerkiksi sulautettujen järjestelmien, kuten kännyköiden, tietokantana. [5]

### *7.4.2 SQLiten käyttö ja linkitys*

Testiohjelmalla suoritetut, analogiatulosta saadut jännitearvot tallennettiin SqLitellä tehtyyn tietokantaan.

SQLiten linkitys ohjelmaan:

Projektitiedostoon lisätään seuraavat rivit:

```
#include <QtSql>
QT += sql
```

Kuvassa 10 analogiatulon jännitesignaalin arvo taulukossa.

Mittaustulokset

| | Name | Voltage | Time | Date | Description |
|---|---|---|---|---|---|
| 1 | Mittaus nro 1 | 0.069 | 1.0 | 2010-04-29T09:47:58 | Comment 1 |
| 2 | Mittaus nro 2 | 0.069 | 2.0 | 2010-04-29T09:47:58 | Comment 2 |
| 3 | Mittaus nro 3 | 0.067 | 3.0 | 2010-04-29T09:47:58 | Comment 3 |
| 4 | Mittaus nro 4 | 0.069 | 4.0 | 2010-04-29T09:47:58 | Comment 4 |
| 5 | Mittaus nro 5 | 0.07 | 5.0 | 2010-04-29T09:47:58 | Comment 5 |
| 6 | Mittaus nro 6 | 0.067 | 6.0 | 2010-04-29T09:47:58 | Comment 6 |
| 7 | Mittaus nro 7 | 0.066 | 7.0 | 2010-04-29T09:47:59 | Comment 7 |
| 8 | Mittaus nro 8 | 0.07 | 8.0 | 2010-04-29T09:47:59 | Comment 8 |
| 9 | Mittaus nro 9 | 0.068 | 9.0 | 2010-04-29T09:47:59 | Comment 9 |
| 10 | Mittaus nro 10 | 0.067 | 10.0 | 2010-04-29T09:47:59 | Comment 10 |
| 11 | Mittaus nro 11 | 0.067 | 11.0 | 2010-04-29T09:47:59 | Comment 11 |
| 12 | Mittaus nro 12 | 0.066 | 12.0 | 2010-04-29T09:47:59 | Comment 12 |
| 13 | Mittaus nro 13 | 0.067 | 13.0 | 2010-04-29T09:47:59 | Comment 13 |
| 14 | Mittaus nro 14 | 0.067 | 14.0 | 2010-04-29T09:47:59 | Comment 14 |
| 15 | Mittaus nro 15 | 0.07 | 15.0 | 2010-04-29T09:47:59 | Comment 15 |
| 16 | Mittaus nro 16 | 0.069 | 16.0 | 2010-04-29T09:47:59 | Comment 16 |
| 17 | Mittaus nro 17 | 0.069 | 17.0 | 2010-04-29T09:48:00 | Comment 17 |
| 18 | Mittaus nro 18 | 0.074 | 18.0 | 2010-04-29T09:48:00 | Comment 18 |

*Kuva 10. Mittausdata näytettynä Qt Creatorilla*

# 8 YHTEENVETO

Insinöörityössä oli tarkoituksena selvittää, miten Qt Creatoria pystyttäisiin hyödyntämään Modbus/TCP-moduulin ohjaamisessa. Tehtiin graafisen käyttöliittymän omaava ohjelma, jolla oli mahdollista ohjata Modbus/TCP-moduulin tuloja ja lähtöjä. Aikaisemmin Qt Creatorilla ei ole tehty vastaavia Modbus/TCP-väylää hyödyntäviä sovelluksia. Lisäksi opinnäytetyössä kerrottiin Modbus-väylästä ja selostettiin, miten Modbus-funktiokirjastot otetaan käyttöön sekä miten Modbus/TCP-moduuli liitetään tietokoneeseen.

Ohjelman rakentaminen aloitettiin tutustumalla Telemerkin EFDC-moduulin teknisiin tietoihin ja ominaisuuksiin. Seuraavaksi perehdyttiin Modbus-funktiokirjastoihin, jotka sisälsivät tarvittavat funktiokomennot moduulin ohjaamiseen. Toimivan ohjelman rakentamisessa oli monia ongelmia, varsinkin funktiokirjastojen linkityksessä ja Qwt:llä tehdyssä graafisessa kuvaajassa, joka on lisäksi melko vaikeakäyttöinen. Lopulta ongelmat kuitenkin ratkaistiin.

Kaikki tässä työssä käytetyt ohjelmointityökalut lukuun ottamatta Modbus-funktiokirjastoja ovat ilmaisia ja perustuivat avoimeen lähdekoodiin. Lisenssimaksuttomien, avoimen lähdekoodin sovellusten määrä tulee varmasti kasvamaan tulevaisuudessa entisestään. Qt Creatorilla on suhteellisen vaivatonta tehdä graafisen käyttöliittymän sovelluksia, jos omaa hieman aikaisempaa ohjelmointikokemusta.

Modbus/TCP-väylän suurimmat vahvuudet verrattuna sen kilpailijoihin ovat sen avoimuus ja maksuttomuus. Sen avulla pystytään yksinkertaistamaan toimintoja, jolloin järjestelmästä saadaan helpommin hallittava ja tehokkaampi. Onkin todennäköistä, että tulevaisuudessa teollisuudessa käytettyjen avointen Ethernet-standardien määrä tulee kasvamaan entisestään.

Insinöörityön tulos täytti asetetut vaatimukset. Tehdyllä ohjelmalla pystyttiin ohjaamaan Modbus/TCP-moduulia ja sen analogisia tuloja ja lähtöjä voitiin onnistuneesti lukea ja muuttaa. Ohjelma on hyvä pohja, mikäli Modbus/TCP-protokollaa päätetään hyödyntää myöhemmissä projekteissa.

**VIITELUETTELO**


[1] Focus Software Engineering. FieldTalk™ Modbus® Master C++ Library [verkkodokumentti]. 20.10.2006 [viitattu 4.4.2010]. Saatavissa: http://www.focus-sw.com/

[2] Miettinen, Otto, Telemerkki Oy.TM-EFDC Users Manual [verkkodokumentti]. 9.12.2009 [viitattu 4.4.2010]. Saatavissa: http://www.telemerkki.fi/

[3] Blanchette, Jasmin – Summerfield, Mark. *C++ GUI Programming with Qt 4*. Westford: Massachusetts. 2008.

[4] Ethernet [verkkodokumentti]. 2010 [viitattu 10.5.2010]. Saatavissa: http://fi.wikipedia.org/wiki/Ethernet

[5] SQLite [verkkodokumentti, viitattu 16.4.2010]. Saatavissa: http://www.sqlite.org/

[6] Modbus Organization, Inc. Modbus [verkkodokumentti, viitattu 18.4.2010]. Saatavissa: http://www.modbus.org/faq.php

[7] CAN in Automation [verkkodokumentti, viitattu 10.5.2010]. Saatavissa: http://www.can-cia.de/

[8] Qwt - Qt Widgets for Technical Applications [verkkodokumentti, viitattu 14.4.2010]. Saatavissa: http://qwt.sourceforge.net/

[9] Industrial Ethernet [verkkodokumentti]. 2010 [viitattu 8.5.2010]. Saatavissa: http://en.wikipedia.org/wiki/Industrial_Ethernet

[10] Modbus-IDA. Modbus Application Protocol Specification V1.1b [verkkojulkaisu]. 2006 [viitattu 5.5.2010]. Saatavissa: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

[11] Nokia. Qt Creator [verkkodokumentti]. 2010 [viitattu 14.4.2010]. Saatavissa: http://qt.nokia.com/

[12] Modicon Inc. Modicon Modbus Protocol Reference Guide [verkkojulkaisu]. 1996 [viitattu 5.5.2010]. Saatavissa: http://www.modbustools.com/PI_MBUS_300.pdf

[13] Modbus-IDA. Modbus Messaging on TCP/IP Implemantation Guide V1.0b [verkkojulkaisu]. 2006 [viitattu 20.4.2010]. Saatavissa: http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

Ohjelman koodi

Ohjelman projektitiedostossa tehdään linkitys, projektitiedosto:

```
TARGET = qt_modbus
TEMPLATE = app

## Huom. hakemistopolut
INCLUDEPATH += /home/koneauto/qt_modbus/fieldtalk_master/include/
INCLUDEPATH += /usr/include/qwt-qt4
LIBS += -L/home/koneauto/qt_modbus/fieldtalk_master/lib/linux \
    -lmbusmaster

LIBS += -L/usr/lib \
    -lqwt-qt4
SOURCES += main.cpp \
    dialog.cpp \
    modbus.cpp \
    datatable.cpp \
    datatableconnection.cpp
HEADERS += dialog.h \
    modbus.h \
    datatable.h \
    datatableconnection.h
FORMS += dialog.ui
QT += sql
```

Parametrit tietokantataululle, datatable.h:

```
#ifndef SCOOTERWINDOW_H
#define SCOOTERWINDOW_H
#include <QWidget>

class QSqlTableModel;
class QTableView;

enum {
    Data_Id = 0,
    Data_Name = 1,
    Data_Force = 2,
    Data_Delta = 3,
    Data_Time = 4,
    Data_Description = 5
};

class CDataTable : public QWidget
{
    Q_OBJECT
public:
    CDataTable();
    void DrawTable();
private:
    QSqlTableModel *model;
    QTableView *view;
};
#endif
```

Tietokannan kehykset, datatable.cpp:

```
#include <QtGui>
#include <QtSql>
#include "datatable.h"

CDataTable::CDataTable()
{
}
void CDataTable::DrawTable()
{
// Luodaan malli sql-taululle
    model = new QSqlTableModel(this);
// Lisätään malliin taulu
    model->setTable("tbl_data");
    // Annetaan sarakkeille otsikot:
    model->setHeaderData(Data_Name, Qt::Horizontal, tr("Name"));
    model->setHeaderData(Data_Force, Qt::Horizontal, tr("Voltage"));
    model->setHeaderData(Data_Delta, Qt::Horizontal, tr("Time"));
    model->setHeaderData(Data_Time, Qt::Horizontal, tr("Date"));
    model->setHeaderData(Data_Description, Qt::Horizontal,
tr("Description"));

//Otetaan malli käyttöön
    model->select();
        // Näytetään taulu
    view = new QTableView;
    view->setModel(model);
    view->setSelectionMode(QAbstractItemView::SingleSelection);
    view->setSelectionBehavior(QAbstractItemView::SelectRows);
    view->setColumnHidden(Data_Id, true);
    view->resizeColumnsToContents();
    view->setEditTriggers(QAbstractItemView::NoEditTriggers);

    QHeaderView *header = view->horizontalHeader();
    header->setStretchLastSection(true);
// Luodaan layout
    QHBoxLayout *mainLayout = new QHBoxLayout;
// Layout päänäkymään
    mainLayout->addWidget(view);
    setLayout(mainLayout);
// Annetaan ikkunalle nimi
    setWindowTitle(tr("Mittaustulokset"));
}
```

Datatableconnection.h:

```
#ifndef DATATABLECONNECTION_H
#define DATATABLECONNECTION_H
#endif // DATATABLECONNECTION_H
#include <QString>
#include <QObject>

class CDataTableConnection
{
public:
```

```
    CDataTableConnection();
    void insertRow(QString, double, double, QString);
    void fillDemoData(int*, int*);
    void createTable(bool);
    bool createConnection();
};
```

Tiedon tallennus tietokantaan, datatableconnection.cpp:

```
#include <QtGui>
#include <QtSql>
#include "datatableconnection.h"

CDataTableConnection::CDataTableConnection()
{
}

bool CDataTableConnection::createConnection()
{
// Luodaan SQLite-tietokanta
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
// Nimetään tietokanta
db.setDatabaseName("testdb3.dat");
// Avataan tietokanta ja testataan onnistuuko yhteydenluonti
   if (!db.open())
 {
// Jos tietokanta ei aukea, naytetään virheilmoitus
QMessageBox::warning(0, QObject::tr("Database Error"),
                            db.lastError().text());
        printf("DEBUG: connection NOT created\n");
        return false;
    }
        printf("DEBUG: connection created\n");
    return true;
}

void CDataTableConnection::createTable(bool delete_if_exists)
{
    QSqlQuery q;
    if (delete_if_exists)
    {
        q.prepare("DROP TABLE tbl_data");
        q.exec();
    }
    q.clear();

    // Luodaan sql-käskyllä tietokantataulun sarakkeet
    q.prepare("CREATE TABLE tbl_data ("
                        "id INTEGER PRIMARY KEY AUTOINCREMENT, "
                        "name VARCHAR(40) NOT NULL, "
                        "force DOUBLE NOT NULL, "
                        "time DOUBLE NOT NULL, "
                        "date DATETIME, "
                        "description VARCHAR(80) NOT NULL)");
     q.exec();
}
void CDataTableConnection::insertRow(
```

```
                QString      name,
                double       force,
                double       time,
                QString      descr = "No comments"
                )
{
                 QDateTime now = QDateTime::currentDateTime();
                 QSqlQuery q;
                 q.prepare("INSERT INTO tbl_data (name, force,
time,date,description) VALUES (?,?,?,?,?)");
                 q.addBindValue(name);
                 q.addBindValue(force);
                 q.addBindValue(time);
                 q.addBindValue(now);
                 q.addBindValue(descr);
                 q.exec();
                 q.clear();
}
```

Modbus/TCP-protokollan parametrit, modbus.h:

```
#ifndef MODBUS_H
#define MODBUS_H
#include <QString>
#include <QObject>
#include <MbusTcpMasterProtocol.hpp>


class CModbus
{
public:
    CModbus();
    void openModbus(QString*);
    void closeModbus();
    void readAnalogInputRegisters(short*, QString*);
    void writeCoil(QString*, int*, int*, int*);
    void writeAnalogOutputs(QString*, int*, int*);
    void MessageCounter(long*, long*)
    #if defined (_UNICODE)
    TCHAR hostName[50];
    #else
    char hostName[50];
    #endif
    MbusTcpMasterProtocol mbusProtocol;
    private:
        QString Comment;
        QString Auxv;
        int result;
        short dataArr[10];
        int dataArr2[10];
        short regArr[16];
        int coil_par1;
        int coil_par2;
        int coil_par3;
};
#endif // MODBUS_H
```

Modbus-funktiokäskyjen toteutus, modbus.cpp:

```
#include "modbus.h"
#include <cstdio>
#include <cstring>
#include <MbusTcpMasterProtocol.hpp

CModbus::CModbus()
{
void CModbus::openModbus(QString *Comment)
{
   int result;
    #if defined (_UNICODE)
    strcpy(hostName, L"10.0.0.11");
    #else
    strcpy(hostName, "192.168.137.1");
    #endif

// Avataan Modbus-protokolla
   result = mbusProtocol.openProtocol(hostName);
if (result != FTALK_SUCCESS)
   {
       Comment->sprintf("Port closed: %s!\n", getBusProtocolError-
Text(result));
   }
   else
   {
       Comment->sprintf("Port open");
   }
}

// Suljetaan yhteys
void CModbus::closeModbus()
{
       mbusProtocol.closeProtocol();
}

void CModbus::MessageCounter(long* SuccessCounter, long* TotalCounter)
{
// Palauttaa onnistuneen viestin siirron
        *SuccessCounter = mbusProtocol.getSuccessCounter();
// Palauttaa ajetun viestin siirron
        *TotalCounter = mbusProtocol.getTotalCounter();
}

// Luetaan analogiset tulot. Muutujat: data and kommentti
void CModbus::readAnalogInputRegisters(short* regArr, QString* Comment)
{
        int result;
// Luetaan tulorekisterien sisältö
        result= mbusProtocol.readInputRegisters(1,1,regArr,4);

// Näytetään analogiatulojen arvot, jos ei virheitä
        if (result == FTALK_SUCCESS)
          {
// Näytetään neljän analogiatulon arvot
Comment->sprintf("Analog Inputs (readRegisters): ",regArr[0]);
            for(int i = 0; i < 4; i++)
            {
                 Auxv.sprintf("[%d]: %hd  ",1+i,regArr[i]);
```

```
                Comment->operator +=(Auxv);
            }
        }
// Virheilmoitus
        if (result != FTALK_SUCCESS)
        {
            Comment->sprintf("readRegisters Error: %s! \n",
getBusProtocolErrorText(result));
        }
}

// Muutetaan kerrallaan yhden bitin tilaa (tosi / epätosi)

void CModbus::writeCoil(QString* Comment, int* coil_par1, int* coil_par2,
int* coil_par3)
{
        int result;
        result = mbusProto-
col.writeCoil(*coil_par1,*coil_par2,*coil_par3);
// Virheilmoitus
        if (result != FTALK_SUCCESS)
        {
            Comment->sprintf("writeCoil error: %s!\n", getBusProtocolEr-
rorText(result));
        }
        else Comment->sprintf("writeCoil: OK");
}

// Muutetaan analogisen lähdön arvoa
void CModbus::writeAnalogOutputs(QString* Comment, int* pwm_par1, int*
out1)
{
        short dataArr[4];
        int out;
        out=*out1-1;
        int i, result;
        for( i=0; i<4; i++ )
        {
                dataArr[i] = out;
        }
        dataArr[out]=*pwm_par1;

 //kirjoitetaan sliderilta saatu arvo valitun AOUT:n kohdalle
result = mbusProtcol.writeMultipleRegisters(1,513,dataArr,sizeof(dataArr)
/ sizeof(short));
        Comment->sprintf("Analog Output: ",dataArr[0]);
 // Naytetan neljan analogialahdon arvot
            for(int i = 0; i < 4; i++)
            {
                Auxv.sprintf("[%d]: %hd  ",1+i,dataArr[i]);
                Comment->operator +=(Auxv);
            }
        if (result != FTALK_SUCCESS) // Virheilmoitus
        {
            Comment->sprintf("writeMultipleRegisters error: %s!\n", get-
BusProtocolErrorText(result));
        }
}
```

Käyttöliittymän parametrit, dialog.h:

```
#ifndef DIALOG_H
#define DIALOG_H
#include <QtGui/QDialog>
#include <MbusTcpMasterProtocol.hpp>
#include <QTimer>
#include <qwt_plot_marker.h>
#include <qwt_legend.h>
#include <qwt_scale_draw.h>
#include <qwt_math.h>
#include <qwt_plot_item.h>
#include <qwt_plot_curve.h>
#include <qwt_text.h>
#include "modbus.h"
#include "datatable.h"
#include "datatableconnection.h

namespace Ui
{
    class Dialog;
}
class Dialog : public QDialog
{   Q_OBJECT
    QTimer *timer;
    CModbus modbus;

public:
    Dialog(QWidget *parent = 0);
    ~Dialog();
    CDataTableConnection dtablec;
    CDataTable dt;

private:
    Ui::Dialog *ui;
    QwtPlotCurve *Curve;
    double Tulo, time;
    double x[10000], y[10000];
    void UpdatePlot();
    short dataArr[10];
    int dataArr2[10];
    short Table1[10];
    short regArr[16];
    long Success, Total;
    QString Comment;
    QString Message;
    int flag1, flag2;

private slots:
    void OpenPort();
    void ClosePort();
    void Read_Input();
    void WriteBit_Output();
    void WritePWM(int);
    void UpdateAll();
    void MessageCounter();
    void ShowData();
};


#endif // DIALOG_H
```

Graafisen käyttölittymän toteutus, dialog.cpp:

```
#include "dialog.h"
#include "ui_dialog.h"
#include <MbusTcpMasterProtocol.hpp>
#include <QMessageBox>
#include <qwt_plot_marker.h>
#include <qwt_legend.h>
#include <qwt_scale_draw.h>
#include <qwt_math.h>
#include <qwt_plot_item.h>
#include <QtGui>
#include <QtSql>

Dialog::Dialog(QWidget *parent)
    : QDialog(parent), ui(new Ui::Dialog)
{
    //Qwt Plot
    time=0;
    Curve = new QwtPlotCurve();
// Asetetaan piirtoväriksi sininen
    Curve->setPen(QPen(Qt::blue));

// Antialiasointi, tasoitetaan piirrettävän käyrän kulmikkuutta
    Curve->setRenderHint(QwtPlotItem::RenderAntialiased);
      ui->setupUi(this);
    timer = new QTimer(this)
//Annetaan ikkunalle nimi
    setWindowTitle(tr("Modbus Monitor"));

// Yhdistetään signaalit ja slotit
    connect(timer, SIGNAL(timeout()), this, SLOT(UpdateAll()));
    connect(ui->B_Open,SIGNAL(clicked()),this,SLOT(OpenPort()));
    connect(ui->B_Close,SIGNAL(clicked()),this,SLOT(ClosePort()));
    connect(ui-
>B_WriteCoil,SIGNAL(clicked()),this,SLOT(WriteBit_Output()));
    connect(ui-
>S_WritePWM,SIGNAL(valueChanged(int)),this,SLOT(WritePWM(int)));
    connect(ui->B_Database,SIGNAL(clicked()),this,SLOT(ShowData()));

//Lippumuuttujat, joilla ohjataan Start, Stop ja Show Data -nappien
//toimintaa. Estetään päällekkäisyydet
    flag1=2;
    flag2=0;
}
Dialog::~Dialog()
{
    delete ui;
}

// Suoritetaan funktio kun painetaan Start-nappia
void Dialog::OpenPort()
{
    flag1=1;
    flag2=1;
// Avataan yhteys. Kommenttina saadaan tieto onnistuiko avaus
   modbus.openModbus(&Comment);
//  Kirjoitetaan tekstikenttään
```

```
   ui->StateMonitor->setPlainText(Comment);
// Käynnistetään ajastin
   timer->start(100);
// Estetään turhien virheilmoitusten näyttäminen
   if(flag1==1 && flag2==1)
   {
       ui->StateMonitor->setPlainText("Port open");
   }
}

void Dialog::ClosePort()
{
    flag1=0;
    timer->stop();          // Pysäytetään ajastin
   modbus.closeModbus();    // Suljetaan yhteys
   ui->StateMonitor->setPlainText("Port closed");
}
void Dialog::ShowData()
{
//Mittaus käynnistetty ja lopetettu. Naytetään mittausdata uudessa
ikkunassa
    if(flag1==0 && flag2==1)
{
   ui->StateMonitor->setPlainText("Port closed");
   dt.DrawTable();
   dt.resize(800,600);
   dt.show();
}
// Mittausta ei ole aloitettu
    else if (flag1==2)
    {
        ui->StateMonitor->setPlainText("Start measurement first");
    }
// Mittaus päällä
     else if (flag1==1)
    {
    ui->StateMonitor->setPlainText("Error: Measurement in progress! Stop
measurement first");
}
// Stop-nappia painettu, mutta mittausta ei aloitettu
     else if (flag1==0)          {
          ui->StateMonitor->setPlainText("Start measurement first");
       }
}

// Luetaan analogiset tulot
// Suoritetaan Paivita-funktio aina kun timer saa uuden arvon
void Dialog::UpdateAll()
{
    Read_Input();
    MessageCounter();
}

// Näytetään kulkeneiden modbus-viestien lukumäärä
void Dialog::MessageCounter()
{
    modbus.MessageCounter(&Success, &Total);
    Comment.sprintf("%li",Success);
// Näytetään onnistuneiden modbus-viestien määrä
    ui->MessageOK->setPlainText(Comment);
    Comment.sprintf("%li",Total);
```

```
// Näytetään kokonaisviestien määrä
    ui->MessageTotal->setPlainText(Comment);
}

void Dialog::Read_Input()
{
// Luetaan analogiset tulot. Muuttujat: data ja kommentti
    modbus.readAnalogInputRegisters(Table1, &Message);
// Näytetään data
    ui->Analog_Inp->setPlainText(Message);
    UpdatePlot(); //function to draw a curve
}



// Kirjoitetaan yksi bitti (0/1)
void Dialog::WriteBit_Output()
{
        int writecoil1, writecoil2, writecoil3;
        QString Asetus;
        bool ok;
// Haetaan käyttöliittymästä syötetyt arvot
        Asetus=ui->writecoil_slave->text();
        writecoil1=Asetus.toInt(&ok,10);
        Asetus=ui->writecoil_addr->text();
        writecoil2=Asetus.toInt(&ok,10);
        Asetus=ui->writecoil_bit->text();
        writecoil3=Asetus.toInt(&ok,10);
        modbus.writeCoil(&Message, &writecoil1, &writecoil2,
&writecoil3); //writeCoil. Muuttujat: slave addr, coil addr, bit value
        ui->ModbusMonitor->append(Message);
}


//Päivitetään kuvaaja
void Dialog::UpdatePlot()
{
    QString Asetus;
    time++;
// Analoginen tulosignaali (volttia)
    Tulo=Table1[2]/1000.0;

   //Plottaus
    y[(int) time]=(int)Tulo; //input
    x[(int) time]=(int)time; //time
// set curve data
    Curve->setData(x, y, time);
// Näytetään graafinen kuvaaja
    Curve->attach(ui->Qwt_Plot);
// Päivitetään plotti
    ui->Qwt_Plot->replot();
// Näytetään tulosignaali
    ui->Lcd_Input->display(Tulo);
// Näytetään aika
    ui->Lcd_Time->display(time);

// Tallennetaan mittausdata tietokantaan
    dtablec.insertRow("Mittaus nro "+ QString::number(time), Tulo,time,
"Comment " + QString::number(time));
}
```

```
// Säädetään analogista lähtöä
void Dialog::WritePWM(int Out)
 {
    int aout1;
    QString Asetus;
    bool ok;
// Haetaan käyttöliittymästä syötetty arvo (Output 1-4)
    Asetus=ui->set_aout->text();
    aout1=Asetus.toInt(&ok,10);
    modbus.writeAnalogOutputs(&Comment, &Out, &aout1);
    ui->Analog_out->setPlainText(Comment);
}
```

Pääohjelma, main.cpp:

```
#include <QtGui/QApplication>
#include "dialog.h"
#include <QtGui>
#include <QtSql>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    CDataTableConnection dtablec;     // luodaan olio

// luodaan yhteys tietokantaan
    if (!dtablec.createConnection())
         return 1;
// tuhotaan vanha tietokanta
    dtablec.createTable(1);

    Dialog w;  // tehdaan olio Dialog-luokasta
    w.show();  // ajetaan ohjelma

  return a.exec();
}
```

EFDC-moduulin järjestelmäkaavio