VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Adis Tosumovic

RELEEN ASETUSTYÖKALU QT:LLA

Tekniikka ja liikenne
2012

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Adis Tosumovic
Opinnäytetyön nimi	Releen asetustyökalu Qt:lla
Vuosi	2012
Kieli	suomi
Sivumäärä	30
Ohjaaja	Martti Mustonen

Tässä työssä käydään läpi, kuinka Qt:lla on tehty asetustyökalu releelle ja miten se toimii. Asetustyökalun tarkoituksena on ohjata relettä etäisesti verkon kautta.

Työssä on käytetty Qt Creator versio 2.4.1:ä ja Qt versio 4.8.1:ä.

Työn aikana on selvinnyt, että Qt:n etuina on erityisesti sen alustariippumattomuus, hyvä dokumentaatio sekä Qt:n omat luokat. Alustariippumattomuus oli tärkeä tekijä, koska releen käyttöliittymän tarkoitus oli olla aika samanlainen kuin asetustyökalun käyttöliittymä. Releessä käyttöjärjestelmänä toimii Linux, kun taas asetustyökalua tullaan käyttämään enimmäkseen Windowsilla. Qt:n avulla pystyi käyttämään samoja lähdekoodeja releen käyttöliittymää varten ja asetustyökalua varten. Qt on myös open source-ohjelmisto, joten Qt:n omia luokkia pystyy muokkaamaan.

Avainsanat rele, ohjelmointi, käyttöliittymä, Qt, C++

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information technology

ABSTRACT

Author	Adis Tosumovic
Title	Relay Setting Tool by means of Qt
Year	2012
Language	Finnish
Pages	30
Name of Supervisor	Martti Mustonen

In this work we will go through, how Qt is used to create a setting tool for a relay and how the setting tool works. The purpose of the setting tool is to control the relay remotely through the network.

Qt Creator version 2.4.1 and Qt version 4.8.1 have been used in this project.

During this work it has become clear, that the advantages of Qt are its cross-platform support, great documentation, and its own classes. Cross-platform support was an important factor, because the UI of the relay was supposed to be quite identical compared to setting tool's UI. Relay is running on Linux OS, while setting tool will be mainly used on Windows OS. By using Qt it was possible to use the same source codes for the relay UI and the setting tool. Qt is also open source software, so it is possible to edit Qt's own classes.

Keywords relay, programming, user interface, Qt, C++

SISÄLLYS

MERKINNÄT JA LYHENTEET

X11	ikkunointijärjestelmä
C++	ohjelmointikieili
Qt Creator	alustariippumaton ohjelmiston kehitysympäristö
Refaktorointi	automatisoitu tapa muuttaa lähdekoodia
Firmware	laiteohjelmisto
Strukti	datarakenne
FTP	File Transfer Protocol, tiedostonsiirtomenetelmä
XML	Extensible Markup Language, kuvauskieli

1 JOHDANTO

Tässä työssä on tehty releelle asetustyökalu käyttäen Qt:a. Asetustyökalun tarkoituksena on ohjata relettä etäisesti verkon kautta. Asetustyökalu sisältää myös logiikkaeditorin, jolla voi suunnitella logiikkaa käyttäen erilaisia logiikkaportteja. Projekti on toteutettu Qt:lla. Projektissa on käytetty Qt:n versiota 4.8.1 ja Qt Creatorin versiota 2.4.1.

Työ sai alkunsa, kun yritys nimeltään Arcteq etsi pari opiskelijaa tekemään yhtä projektia. Projektin tarkoituksena oli tehdä releelle käyttöliittymä ja asetustyökalu. Aluksi tämä aiottiin toteuttaa Javalla. Joku Vaasan ammattikorkeakoulun opettajista kutenkin ehdotti, että projekti toteutettaisiin Qt:lla. Tähän sitten päädyttiin.

Arcteq on sähköalan yritys, joka sijaitsee Palosaarella Vaasassa. Yrityksessä on noin 10 työntekijää. Yritys on perustettu vuonna 2010. Yrityksen tärkeimpiä tuotteita ovat suojareleet.

Tämä projekti oli tärkeä yritykselle, koska se tarvitsi asetustyökalua releelle. Yrityksessä ei myöskään ole käytetty Qt:a aikaisemmin.

2 QT

2.1 Mikä on Qt?

Qt on alustariippumaton ohjelmistokehys. Qt:lla kehitetään alustariippumattomia ohjelmia sekä graafisia käyttöliittymiä. Qt on C++ pohjainen, joten ohjelmointi tapahtuu C++ kielellä. Qt:lla voi samaa lähdekoodia käyttäen kääntää ohjelmiston monelle eri alustalle. Yleisiä ohjelmia, joiden käyttöliittymä on toteutettu käyttäen Qt:n kirjastoja, ovat mm. VLC player, Google Earth, KDE sekä Skype (Qt:n omat sivut; Schumacher 2011, Qt and the Future of KDE).

Qt on tarjolla kahdella eri lisenssillä. On GNU Lesser General Public License (LGPL) sekä kaupallinen lisenssi.

2.2 Qt:n historiaa

Qt sai alkunsa vuonna 1991 kun Eirik Chambe-Eng ja Haavard Nord alkoivat kehittää sitä. Vuonna 1994 he perustivat oman yrityksen nimeltä Quasar Technologies. Tämä nimi vaihtui myöhemmin Troll Techiksi, jonka jälkeen nimi vaihtui Trolltechiksi. Vuonna 1995 Metis-niminen yhtiö antoi Trolltechille sopimuksen Qt-ohjelmistojen kehitystä varten. Samana vuonna he julkaisivat ensimmäisen julkisen version Qt:sta. Qt:lla on alusta asti ollut 2 eri lisenssiä. Avoimen lähdekoodin lisenssi avoimen lähdekoodin ohjelmistoja varten sekä kaupallinen lisenssi kaupallisia ohjelmistoja varten.

Kun Matthias Ettrich päätti vuonna 1997 käyttää Qt:a KDE:n kehittämiseen, auttoi se Qt:a tulemaan teollisuusstandardiksi Linux-maailmassa. Qt 2.0 julkaistiin kesällä 1999. Se toi mukanaan 40 uutta luokkaa ja monia uusia arkkitehtuurisia muutoksia. Se toi mukanaan myös uuden avoimen lähdekoodin lisenssin, Q Public lisenssin (QPL). Samana vuonna Qt voitti LinuxWorld-palkinnon parhaana kirjastona/työkaluna.

Vuonna 2000 Trolltech julkaisi Qt/Embeddedin. Se oli suunniteltu sulautettuja laitteita varten, joissa pyöri Linux. Se toi mukanaan oman kevyen ikkunointijärjestelmän korvaamaan X11:n.

Vuoden 2000 loppuun mennessä Trolltech oli perustanut Trolltech Inc.:n Amerikkaan sekä julkaissut ensimmäisen version Qtopiasta. Qtopia on ohjelmistoalusta Linux-pohjaisia mobiililaitteita varten, joka tunnetaan nykyään nimellä Qt Extended. Qt/Embedded voitti LinuxWorldin ”paras sulautettu linux ratkaisu” (best embedded linux solution) palkinnon vuosina 2001 ja 2002.

Vuonna 2001 Trolltech julkaisi Qt 3.0:n. Se toi mukanaan 42 uutta luokkaa. Tämän jälkeen Qt oli saatavilla Windowsille, Unixille, Liuxille, sulautetulle Linuxille sekä Mac OS X:lle. Vuonna 2002 Qt 3.0 voitti Software Development Timessin ”Jolt tuottavuuden palkinnon” (Jolt Productivity Award).

Qt julkaisi 4.0 version vuonna 2005 (Binner 2005, Trolltech Releases Qt 4.0). Se toi mukanaan 5 uutta teknologiaa (Tulip, Interview, Arthur, Scribe sekä MainWindow). Vuonna 2008 Qt siirtyi Nokian omistukseen. Versio 4.8.0 julkaistiin 2011 joulukuussa ja versio 4.8.2 julkaistiin 2012 toukokuussa (Tanilkan 2011, Qt 4.8.0 Released). Vuoden 2012 elokuussa Digia osti Qt:n Nokialta.

(Blanchette, Summerfield, C++ GUI Programming with Qt 4; Thelin 2010 Cross Platform Qt; Aarhus, Getting started with Qt; Pentopia, What is Trolltech?)

2.3 Qt:n hyvät ominaisuudet

Qt:n parhaimpia ominaisuuksia ovat ehdottomasti sen alustariippumattomuus, sen omat luokat sekä Qt Creator (**Kuva 1.**). Qt Creatorissa on mm. graafinen editori, jolla voi suunnitella käyttöliittymän graafisen puolen sekä tekstieditori, jolla tehdään lähdekoodit.

Qt on avoimen lähdekoodin ohjelmisto, joten käyttäjä pääsee Qt:n omiin luokkiin käsiksi ja pystyy muokkaamaan näitä oman tahdon mukaan. Qt:n pystyy myös rakentamaan suoraan lähdekoodista, joten Qt:n kirjastoja voi myös muokata. Tämä on hyvä keino säästää muistia, jos on tekemässä ohjelmistoa sulautetulle laitteelle.

Qt käyttää signaaleiksi ja sloteiksi kutsuttua menetelmää. Tämä toimii siten, että tehdään jokin signaali ja kytketään se kiinni johonkin slottiin. Slotti on aivan normaali funktio, joka voidaan suorittaa aivan normaalisti kutsumalla sitä tai lähettämällä signaalin, joka on yhdistetty kyseiseen slottiin. Tämä on erittäin tärkeä käytäntö, kun tekee eri säikeitä omaan ohjelmaan. Myös nappien painallukset tunnistetaan tällä menetelmällä. Kun nappia painaa, se lähettää signaalin. Jos haluaa jotain tapahtuvan, tämä signaali pitää yhdistää johonkin slottiin.

Qt:n mukana tulee myös erinomainen dokumentaatio. Tämä on yhdistetty Qt Creatoriin niin, että dokumentaation pystyy avaamaan, esim. jostakin luokasta klikkaamalla luokan nimeä ja painamalla näppäimistön F1-näppäintä. Sama toimii myös luokan funktioihin. Sama dokumentaatio löytyy myös Internetistä. Ohjeita saa myös eri foorumeilta. Hyviä foorumeita ovat mm. qtforum.org, qt-project.org/forums, sekä qtcentre.org/forum/.

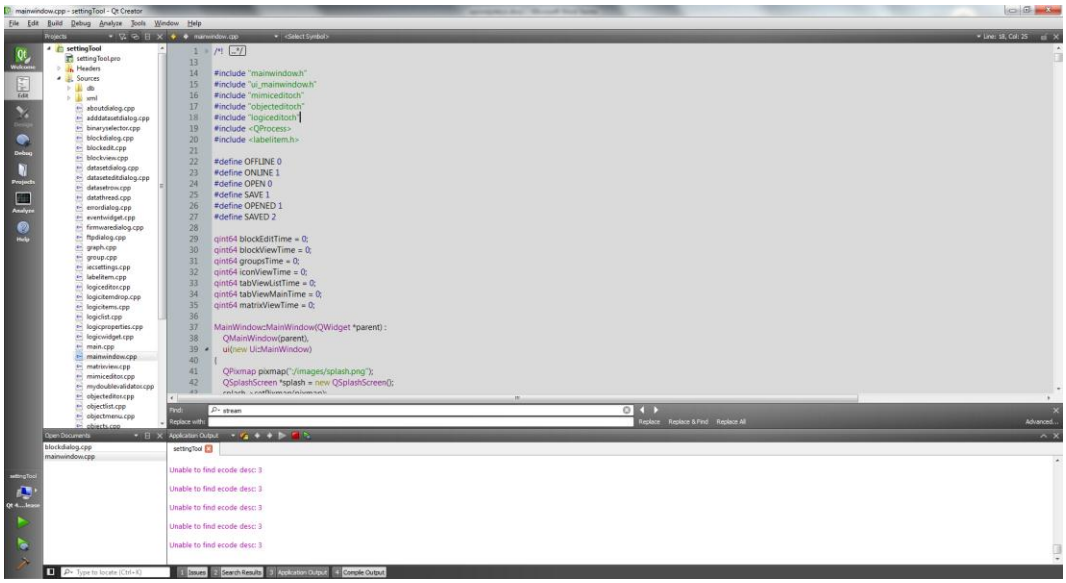
2.4 Qt:n parantamisen mahdollisuudet

Qt:n päivitystyökalu, SDKMaintenanceTool.exe, saattaa kaatua välillä.

Jos haluaa linkittää ohjelman käyttämät kirjastot staattisesti, käyttäjän on pakko kääntää Qt suoraan lähdekoodeista ja tämä voi olla haastavaa Windowsilla.

Refaktorointi voi olla vaarallista Qt:ssa. Jos refaktoroimalla vahingossa uudelleen nimeää jotain Qt:n omaa luokkaa, ei ole mitään varoitusta, joka varoittaisi käyttäjää tästä. Jos käyttäjä menee vahingossa tekemään näin, on käyttäjän asennettava ne Qt:n kirjastot uusiksi, missä refaktorointi tapahtui.

Qt SDK:n ei myöskään päivitetä Qt Creatoria kovin nopeasti. Jos haluaa uusimman version Qt Creatorista, joutuu sen usein lataamaan erikseen ilman Qt SDK:a.



Kuva 1: *Qt Creator.*

3 YRITYS

3.1 Yleisesti yrityksestä

Yritys, jossa projektia on tehty, on nimeltään Arcteq. Arcteq on sähköalan yritys ja se on perustettu vuonna 2010. Yrityksen tärkeimpiä tuotteita ovat suojareleet. Yrityksen kohteena ovat kansainväliset markkinat. Arcteqilla on noin 10 työntekijää ja yritys sijaitsee Palosaaressa, Vaasassa.

3.2 Yrityksen historiaa

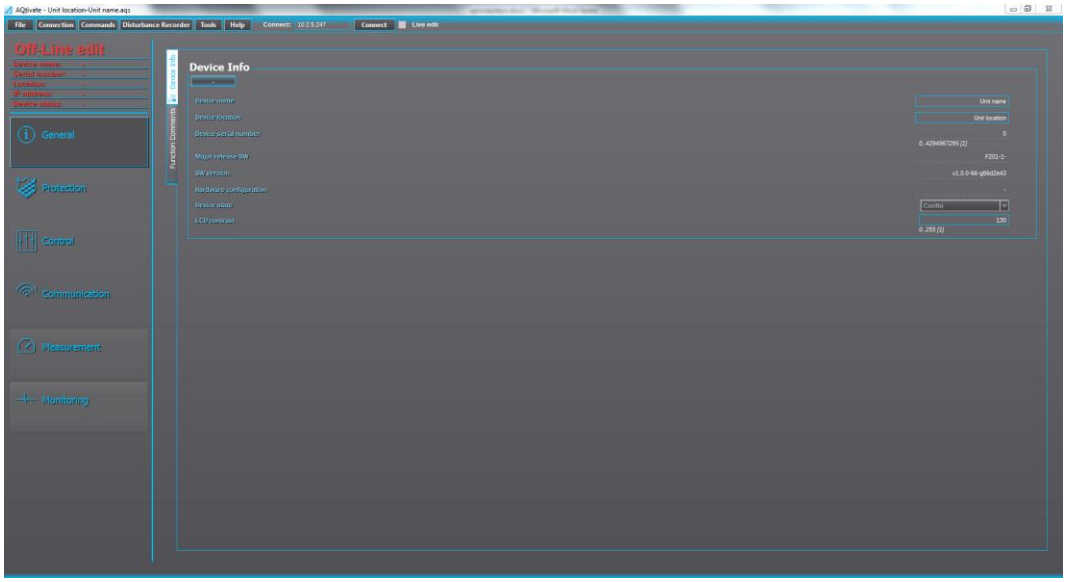
Yritys perustettiin 1. kesäkuuta 2010. Yrityksen perustajina oli ryhmä kokeneita suojareleitä valmistavia ammattilaisia.

4 ASETUSTYÖKALU

4.1 Asetustyökalusta yleisesti

Releen asetustyökalu (**Kuva 2.**) on ohjelma, jota käyttämällä voidaan ohjata relettä. Asetustyökalua voidaan käyttää joko offline-tilassa tai online-tilassa. Offline-tilassa muutokset tallennetaan tiedostoihin, jotka voidaan myöhemmin lähettää releelle verkon kautta. Online-tilassa voidaan joko tehdä muutokset vain tiedostoihin ja lähettää ne sitten releelle, tai voidaan tehdä muutoksia suoraan releeseen verkon kautta.

Asetustyökalu sisältää myös logiikkaeditorin, jolla voidaan suunnitella logiikkaa ja mimiikkaeditorin, jolla voidaan suunnitella releen mimiikkanäkymä. Myös releen firmware voidaan päivittää asetustyökalulla. Asetustyökallussa on myös matriisieditori, jolla voidaan yhdistää signaaleja toisiinsa. Asetustyökalu sisältää myös IEC 61850-editorin, jolla voidaan säätää releen kommunikaatio asetuksia.



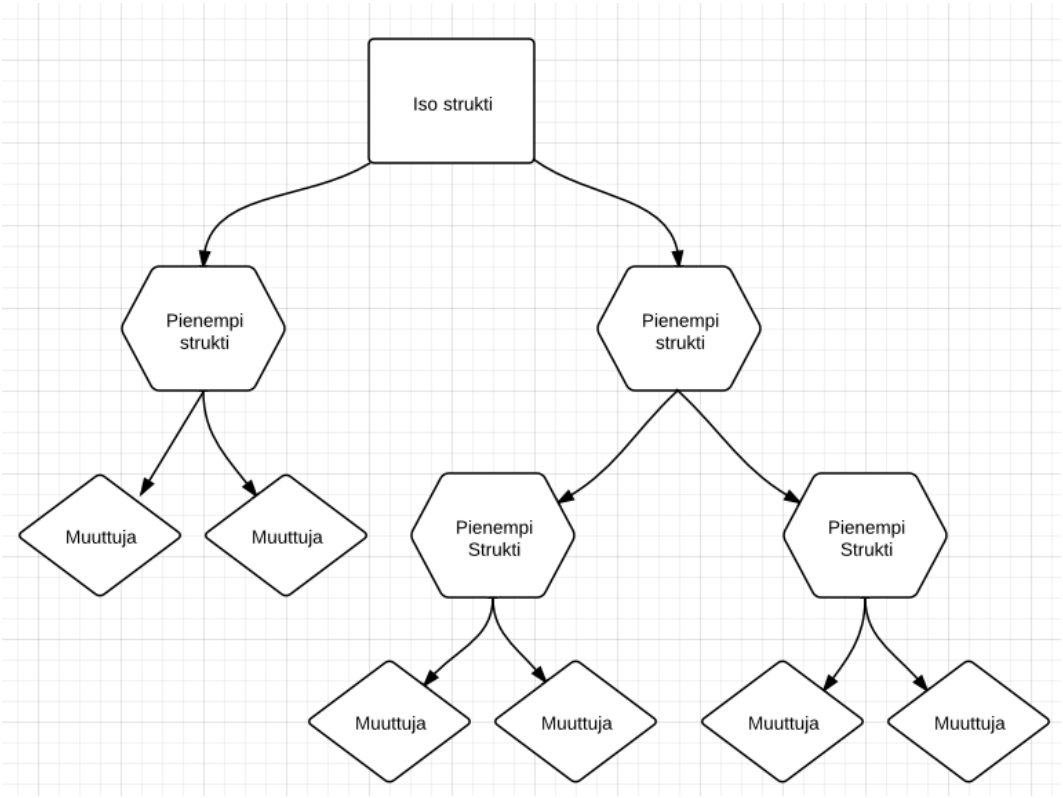
Kuva 2: *Asetustyökalu.*

4.2 Asetystyökalun toiminta

4.2.1 XML-luku

Kun avataan asetustyökalu, ensimmäinen asia joka tehdään, on .aqs-tiedoston avaaminen. Tämä on zippi-paketti, joka sisältää muita tiedostoja. Yksi näistä tiedostoista on XML-tiedosto, joka sisältää releen menurakenteen sekä muuta tärkeää dataa. Tämä XML-tiedosto jaetaan osiin ja luetaan muistiin. Sitten mennään menurakenteen osia muistista läpi ja laitetaan tiedot isoihin strukteihin, joilla on muita pienempiä strukteja (**Kuva 3.**). Jokainen osa menusta laitetaan omaan isoon struktiin.

Tässä vaiheessa on käytetty Qt:n QFile–luokkaa tiedoston avaamiseen sekä QDomDocument–luokkaa XML–tiedoston lukemiseen. Zip–tiedoston purkamiseen käytetään 7zip–ohjelmaa.



Kuva 3: *Esimerkki strukteista.*

4.2.2 Objektien luonti

Kun yksi osa menurakenteesta on menty läpi ja laitettu struktiin, aletaan tämän struktin avulla luomaan käyttöliittymää. Struktit sisältävät tietoa objektien tyypeistä ja niiden attribuuteista. Näiden avulla tiedetään millainen objekti luodaan minnekkin ja millaista sisältöä se tulee sisältämään. Kun yksi osa käyttöliittymästä on valmis, mennään seuraava osa menurakenteesta läpi ja luodaan se osa käyttöliittymästä. Tätä toistetaan kunnes koko menurakenne on menty läpi.

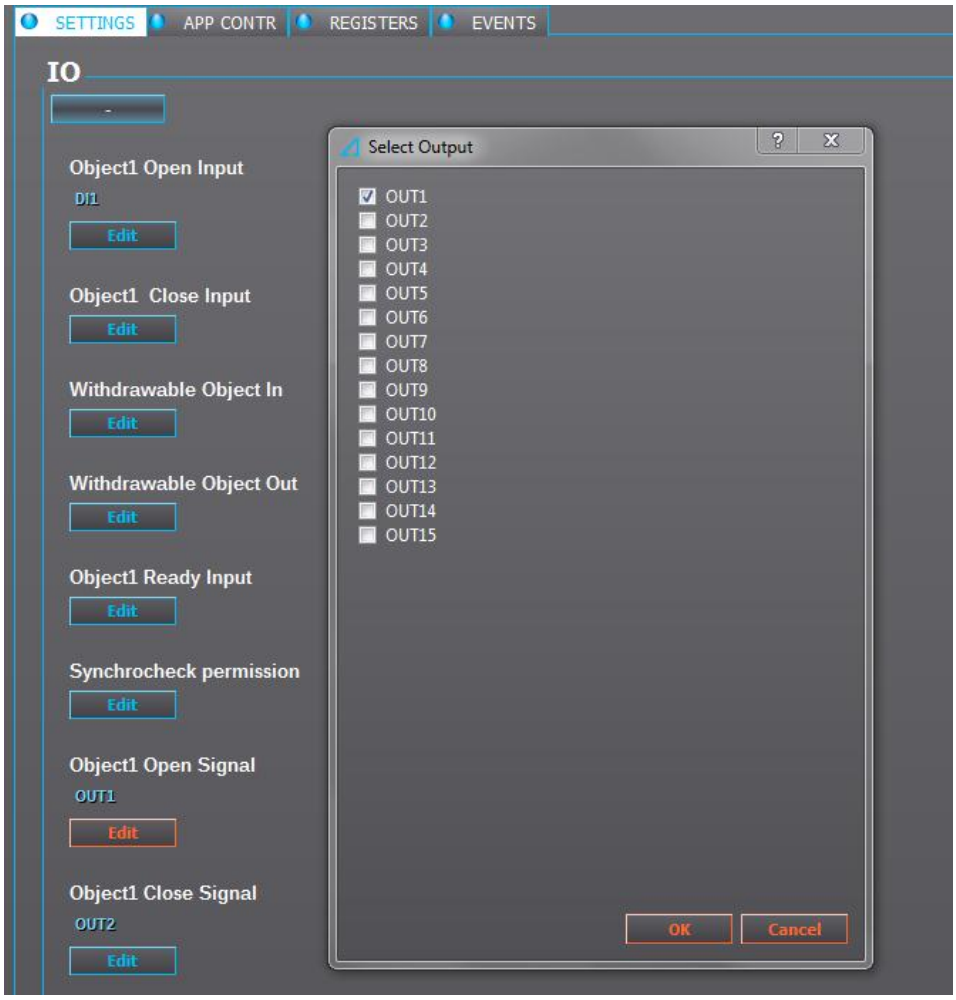
Alussa asetustyökalu on melko tyhjä ohjelma. Vasta XML-tiedoston lukemisen jälkeen aletaan luomaan käyttöliittymää kunnolla. Kaikkia osia ei kuitenkaan luoda heti. Pienemmät objektit jätetään luomatta, koska tämä hidastaa käyttöliittymän luomista paljon. Nämä pienemmät osat luodaan vasta silloin, kun ne halutaan laittaa näkyviin. Tähän olisi ollut muitakin vaihtoehtoja, joita on jopa kokeiltu. Yksi vaihtoehto on lukea näkyvä osa menurakenteesta ja luoda se osa käyttöliittymästä. Sitten kun painetaan jotakin nappia, joka avaa seuraavan näkymän, luetaan se kohta menurakenteesta ja luodaan vasta sitten se näkymä. Tämä tosin hidasti aika paljon asetustyökalun käyttöliittymän toimintaa.

4.2.3 Inputit ja outputit

Inputit ja outputit ovat signaaleja, joita voidaan yhdistää. Päänäkymässä näkyy lista yhdistetyistä signaaleista sekä edit-nappi. Kun edit-nappia painaa, tulee lista mahdollisista inputeista tai outputeista näkyviin (**Kuva 4.**), joita voi sitten yhdistää tai katkaista. Input- tai outputobjektin näyttämisen aikana luetaan tekstitiedostosta mitkä signaalit on yhdistetty ja laitetaan kyseiset signaalit ”connect”-tilaan. Tämä tarkoittaa sitä, että laitetaan rasti ruutuun jokaisen yhdistetyn signaalin kohdalle.

Kun tekee jotain muutoksia listassa ja painaa OK-nappia, tehdyt muutokset tallentuvat tekstitiedostoon. Tämän jälkeen kyseisen input tai output objektin värit muuttuvat oranssiksi. Jotta nämä muutokset tulisivat voimaan releessä, pitää käyttäjän ensiksi lähettää tämä tiedosto releelle. Tämä onnistuu valitsemalla asetustyökalun yläpalkista ”Commands” ja sen alta ”Write Configurations”. Tiedoston lähetyksen jälkeen asetustyökalu lähettää komennon releelle, joka käskee sitä ottamaan uudet asetukset käyttöön, jonka jälkeen kaikki oranssiksi muuttuneet input ja output objektien värit menevät takaisin sinisiksi.

Valintaruudut ovat QTreeWidgetItem-olioita ja ne ovat listattuna QTreeWidget-olion sisään. Ikkuna, jossa on lista signaaleista, on tehty käyttämällä Qt:n QDialog–luokkaa.



Kuva 4: *Objekti 1 ja sen signaalit. Select output listassa näkyy sen open-signaalin yhdistäminen mahdolliseen outputtiin.*

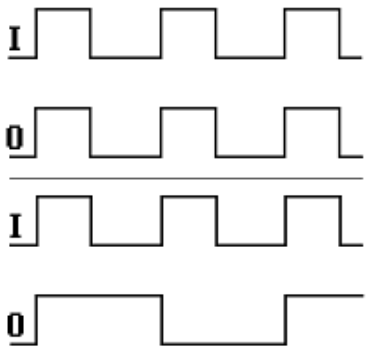
4.2.4 Matriisieditori

Matriisieditori on editori, jossa yhdistetään signaaleja outputteihin. Tässä on 2 eri tapaa yhdistää signaali outputtiin. Signaalin voi yhdistää outputtiin connectilla, joka tarkoittaa sitä, että kun signaalin tila muuttuu, output menee samaan tilaan, tai sitten latchilla, joka tarkoittaa sitä, että silloin kun signaali menee päälle, outputin tila muuttuu (**Kuva 5.**).

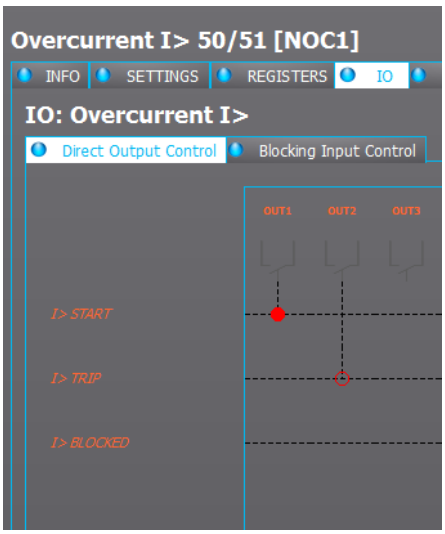
Matriisieditorissa pystyy yhdistämään yhden signaalin moneen outputtiin sekä monta signaalia yhteen outputtiin. Kun käyttäjä tekee jonkun muutoksen, se kirjoitetaan tekstitiedostoon. Muutos ei tule voimaan releeseen ennen kuin käyttäjä lähettää kyseisen tekstitiedoston releelle, jonka jälkeen asetustyökalu lähettää komennon releelle, joka käskee sitä ottamaan uudet asetukset käyttöön. Tämä onnistuu valitsemalla releen yläpalkista ”Commands” ja sen alta ”Write Configurations”.

Matriisieditori toimii siten, että sen näkymässä näkyy input-signaalit vasemmassa reunassa ja output-signaalit näkymän yläosassa (**Kuva 6.**). Matriisieditorin luonnissa käytetään struktia, johon on laitettu muistiin kaikki signaalit, mitkä kuuluvat kyseiselle matriisieditorille. Jokaisen input-signaalin kohdalla menee vaakasuora katkoviiva. Tämä katkoviiva on jono QWidget-olioita. Jokaisen output-signaalin kohdalla on erillinen QWidget-olio. Jotain tämän katkoviivan kohtaa klikkaamalla yhdistetään kyseinen input-signaali tiettyyn output-signaalin, riippuen siitä, mihin kohtaan käyttäjä on klikannut. Kun klikkaus tapahtuu, ohjelma katsoo mikä QWidget-olio on kyseisessä kohdassa, muuttaa tämän kuvaa, asettaa kyseisen kohdan seuraavaan tilaan sekä kirjoittaa tekstitiedostoon kaikki input-signaalit, jotka on yhdistetty johonkin output-signaaliin. Hiiren klikkaus tunnistetaan käyttämällä Qt:n mousePressEvent–tapahtumakäsittelijää. Se on kuin virtuaalinen funktio, joka suoritetaan kun hiirtä klikataan. Kuvien piirtämiseen käytetään QPainter–luokkaa.

Matriisieditorissa näkee myös signaalin tilan. Jos yhdistetyn kohdan ympyrän väri on punainen, on kyseisen input-signaalin arvo 0. Jos taas yhdistetyn kohdan ympyrän väri on vihreä, on kyseisen input-signaalin arvo 1. Silloin kun matriisieditorin tekstien värit ovat oranssilla se tarkoittaa, että matriisieditorissa on tehty muutoksia eikä niitä ole vielä lähetetty releelle. Kun muutokset lähetetään releelle, muuttuvat tekstin värit takaisin valkoiseksi.



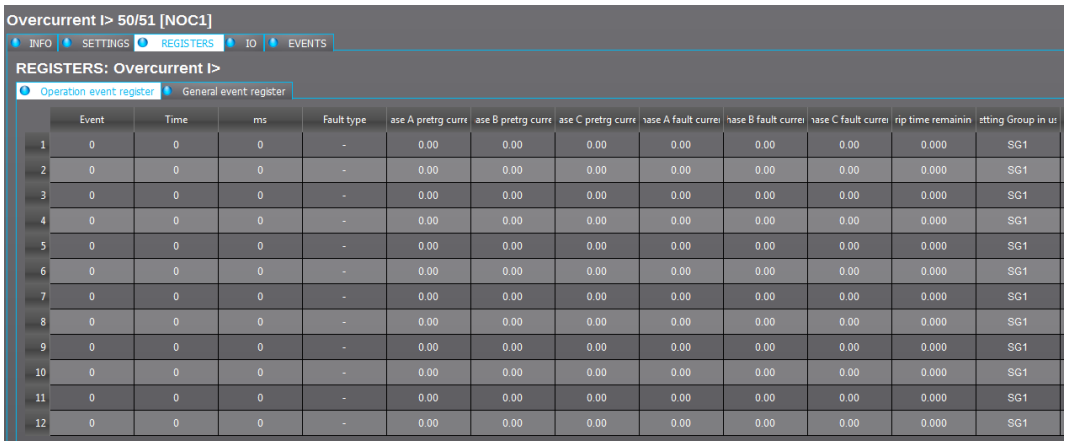
Kuva 5: *Kuvan yläosassa näkyy connectilla kytketyn inputin ja outputin arvot ja kuvan alaosassa näkyy latchilla kytketyn inputin ja outputin arvot.*



Kuva 6: *NOC1:sen START signaali yhdistetty latchilla OUT1 outputtiin sekä NOC1:sen TRIP signaali yhdistetty connectilla OUT2 outputtiin matriisieditorissa.*

4.2.5 Rekisterinäkymä

Rekisterinäkymässä näkyy kyseisen rekisterin 12 uusinta rekisteriä (**Kuva 7.**). Rekisterit laitetaan taulukkoon näkyviin. Rekisterinäkymässä on clear-toiminto, joka tyhjentää kyseiset rekisterit. Rekisterit ovat parametrejä, joita luetaan releen tietokannasta. Rekisteritaulukon luomiseen on käytetty Qt:n QTableWidget–luokkaa. Jokainen rekisterin sarake on toteutettu käyttäen QTableWidget-olioita.



Kuva 7: *Rekisterinäkymä.*

4.2.6 Mimiikkaeditori

Mimiikkaeditori on editori, jossa käyttäjä voi suunnitella releen mimiikkanäkymää (**Kuva 9.**). Mimiikkaeditorissa on 7x9 ruudukko, jossa on nappeja (**Kuva 8.**). Napit on tehty käyttäen Qt:n QPushButton-luokkaa. Nappia painamalla tulee esiin ikkuna, mistä käyttäjä voi valita jonkun kuvion tai objektin. Avautuva ikkuna on tehty käyttäen Qt:n QDialog-luokkaa. Kuvion tai objektin valittua käyttäjä painaa OK nappia, jolloin se tulee ruudukkoon näkyviin käyttäjän painaman napin kohdalle.

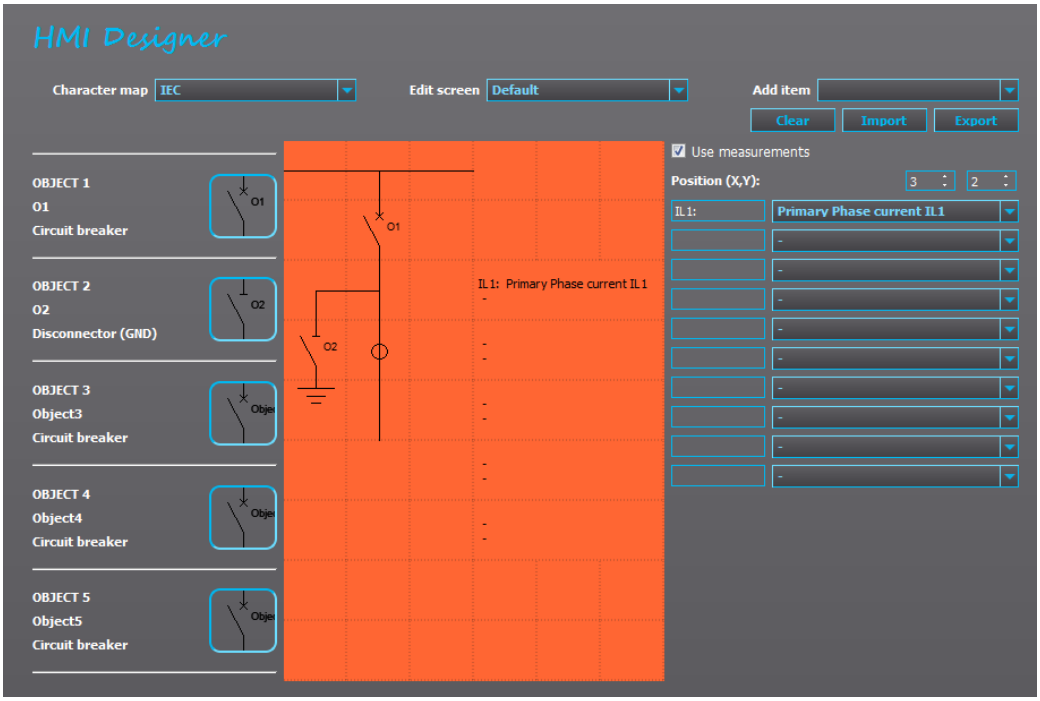
Kuviot piirretään vektoreilla. Vektoreille annetaan koordinaatit, mitkä sitten piirretään. Objekteilla on myös eri tilat. Näitä tiloja käytetään vain releessä. Jos objektin tila on auki, piirretään aukinainen objekti, jos objektin tila on kiinni, piirretään kiinni oleva objekti. Piirtäminen tapahtuu QPainter-luokkaa käyttäen ja kuvan näyttäminen tapahtuu QPixmap-luokkaa käyttäen.

Kun käyttäjä on suunnitellut haluamansa mimiikan, käyttäjän pitää tallentaa kyseinen mimiikka XML-tiedostoon. Tämä onnistuu painamalla ”Export”-nappia. Kun tätä nappia painetaan, ohjelma katsoo jokaisen ruudukossa olevan napin tilan, millainen kuvio tai objekti siinä on ja kirjoittaa sen XML-tiedostoon.

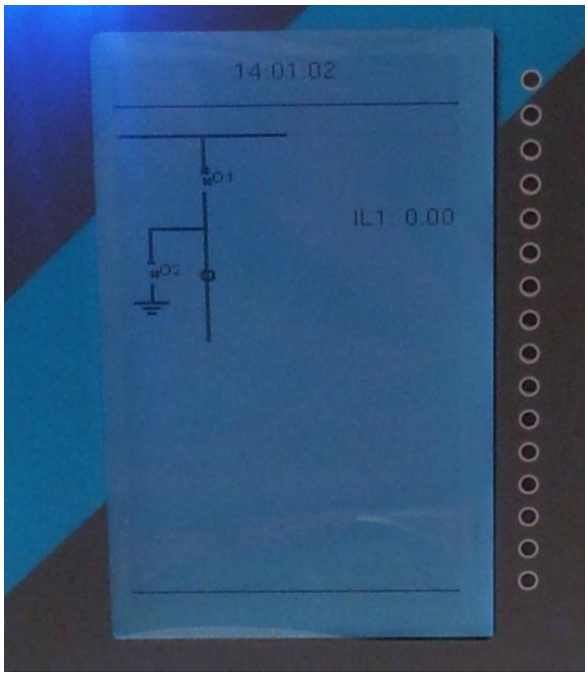
Mimiikkaeditorissa on myös ”Import”-nappi, joka lukee mimiikan XML-tiedostosta. Tämä tapahtuu myös automaattisesti kun käyttäjä avaa mimiikkaeditorin. Tiedoston avaaminen ja tallentaminen tapahtuu QFile-luokkaa käyttäen. Itse XML-tiedoston luku tapahtuu QDomDocument-luokan avulla.

Mimiikkaeditorin oikealla puolella on valintaruutu, jota ruksaamalla käyttäjä saa myös tiettyjä mittausarvoja näkyviin. Tämä valintaruutu on tehty QCheckBox-luokan avulla. Käyttäjä voi valita mihin kohtaan hän haluaa mittaukset näkyviin (QSpinBox), mitkä mittaukset laitetaan näkyviin (QComboBox) sekä tekstit näille mittauksille (QLineEdit).

Mimiikkaeditorissa on myös ”Clear”-nappi, joka tyhjentää ruudukon. Tämä ei kuitenkaan koske mittausarvoja.



Kuva 8: *Mimiikkaeditorissa suunniteltu mimiikka.*



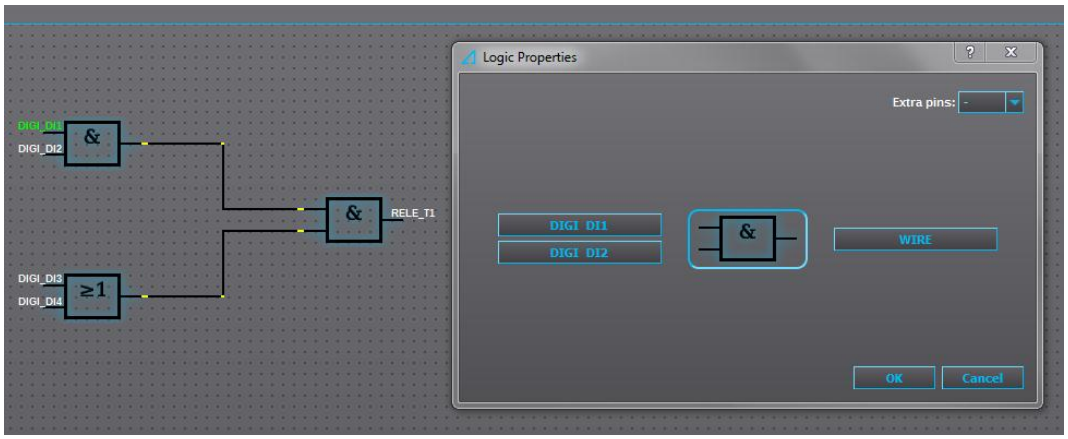
Kuva 9: *Mimiikkaeditorissa suunniteltu mimiikka releessä.*

4.2.7 Logiikkaeditori

Logiikkaeditorissa käyttäjä pystyy suunnittelemaan logiikkaa. Logiikkaeditorin näkymässä on 2 erillistä asettelualuetta. Toisessa on lista loogisia portteja ja toisessa näkymässä voidaan suunnitella logiikkaa. Porttilistasta voidaan vetää hiirellä eri portteja suunnittelualustalle. Logiikkaportit ovat QWidget-alaluokan olioita, joilla on jokaisen nastan signaali muistissa. Logiikkaporttien liikuttaminen hiirellä on toteutettu käyttäen Qt:n mousePressEvent-, dragMoveEvent-, dragEnterEvent- sekä dropEvent- tapahtumakäsittelijöitä.

Suunnittelualustassa voidaan yhdistää portteja toisiinsa. Klikkaamalla hiirellä jonkun portin nastasta luodaan johto, joka voidaan sitten yhdistää toiseen nastaan. Nämä johdot ovat QLabel-olioita, joihin on piiretty kuva käyttämällä QPainter- ja QPixmap-luokkia. Näille johdoille annetaan erikoinen seurantanimi, jotta tiedetään mikä johto on yhdistetty mihinkä kohtaan. Johtojen keskellä on keltainen piste. Tästä pisteestä hiirellä vetämällä, voidaan jakaa johto useaan eri osaan. Näin voidaan yhdistää 1 nasta useaan eri nastaan.

Klikkaamalla hiiren oikealla napilla suunnittelualustassa, kun hiiren kursori on jonkun portin kohdalla ja valitsemalla ”properties”, saadaan portin ominaisuusikkuna auki (**Kuva 10.**). Tässä ikkunassa voidaan asetella portin nastoille signaaleja. Painamalla jotakin nappia, joka menee jonkun nastan kohdalla, saadaan lista signaaleista näkyviin, josta voidaan sitten valita mikä signaali yhdistetään kyseiseen nastaan.



Kuva 10: *Logiikkaeditorissa suunniteltu logiikka sekä vasemman and-portin ominaisuudet.*

Jos signaalin arvo on 1, tulee tämän signaalin teksti näkymään vihreänä. Jos taas signaalin arvo on 0, tulee tämän signaalin teksti näkymään valkoisena. Jos joitakin portteja on yhdistetty johdoilla, musta johto tarkoittaa, että sen signaalin arvo on 0 ja vihreä johto tarkoittaa, että sen signaalin arvo on 1. Näitä arvoja luetaan releen tietokannasta. Kaikki loogiset operoinnit tapahtuvat releessä.

Logiikkaeditorissa on myös import- ja export-napit. Import-nappia painamalla luetaan logiikka tekstitiedostosta ja laitetaan se suunnittelualustalle näkyviin. Export-nappia painamalla tallennetaan suunnittelualustassa oleva logiikka tekstitiedostoon. Tähän tiedostoon ei tallenneta logiikkaporttien sijainteja. Tämä on syy, miksi tuotu logiikka ei näytä samanlaiselta kuin viety logiikka. Asetustyökalun pitää päätellä porttien sijainnit sen perusteella, missä järjestyksessä portit ovat tiedostossa sekä mihin toiseen porttiin on mikäkin portti yhdistetty. Tiedostosta lukeminen ja tiedostoon tallentaminen tapahtuu QFile- ja QTextStream-luokkien avulla.

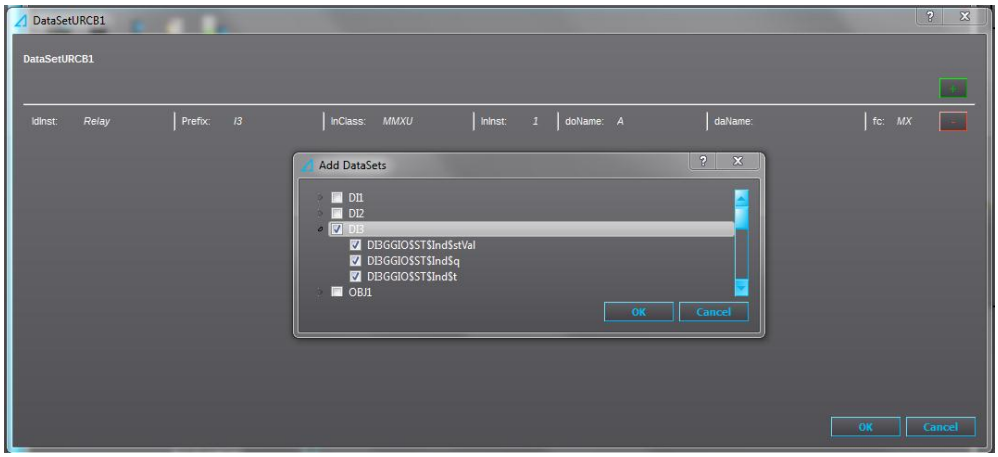
Logiikka voidaan viedä releelle suorittamalla ”Update logic”-komennon. Tämä komento löytyy ”Commands”-valikon alta, joka on asetustyökalun yläosassa. Tämä komento vie tekstitiedoston releelle FTP-protokollan avulla, jonka jälkeen asetustyökalu lähettää komennon releelle, joka käskee tätä ottamaan uuden logiikan käyttöön.

4.2.8 IEC 61850-editori

IEC 61850-editori on editori, jossa voidaan säätää kommunikaatioasetuksia. Editorissa on 2 puunäkymää (**Kuva 12.**). Vasemmanpuoleisessa on lista olioista ja oikeanpuoleisessa on olion tärkeämmät tiedot. Puunäkymät on tehty käyttäen QTreeWidget–luokkaa.

Editori toimii siten, että ensiksi avataan XML-tiedosto, jonka avulla luodaan vasemmanpuoleinen puunäkymä. QFile–luokkaa käytetään tiedoston avaamiseen ja QDomDocument–luokkaa XML-tiedoston lukemiseen. Tästä puunäkymästä löytyy attribuutteja joita voidaan muokata. Kun hiirellä klikataan jotakin vasemmanpuoleisen puunäkymän oliota, sen tärkeämmät tiedot tulevat oikeanpuoleiseen puunäkymään (**Kuva 12.**). Jos kyseisen olion arvoa voidaan muuttaa, olion teksti tulee olemaan vihreä. Olion arvoa voidaan muuttaa kaksoisklikkaamalla sen arvoa oikeanpuoleisessa puunäkymässä.

IEC 61850 sisältää ns. datasettejä. Jos klikataan vasemmanpuoleisesta puunäkymästä jonkun dataset-olion kohdalta hiiren oikealla, tätä datasettiä voidaan editoida valitsemalla ”Edit Dataset”. Datasettiä editoidaan siten, että joko lisätään datasettejä painamalla vihreää ”+”-nappia, tai sitten poistamalla lisättyjä datasettejä painamalla punaista ”-”-nappia (**Kuva 11.**). Nämä napit on tehty käyttäen QPushButton–luokkaa.

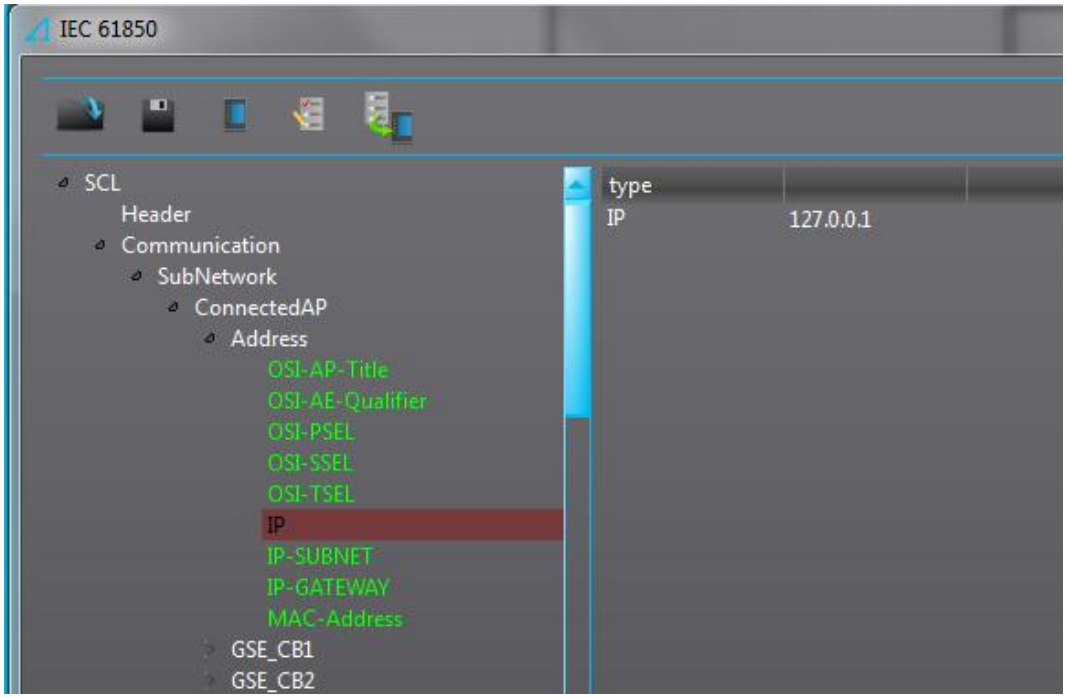


Kuva 11: *Datasetin editointi-ikkuna sekä datasetin lisäysikkuna.*

IEC 61850-editoriin on laitettu ns. helppokäyttötoimintoja. Löytyy ”Edit DataSets”-nappi, jota painamalla aukeaa valikko, josta voidaan valita jokin dataseteistä ja editoida sitä suoraan ilman, että käyttäjä joutuu seikkailemaan puunäkymässä. Editorista löytyy myös ”Configurations”–nappi. Tätä nappia painamalla aukeaa ikkuna, jossa voidaan säätää mm. kommunikaatio asetuksia. Tämä toiminto on myös tehty sitä varten, että käyttäjä ei joutuisi seikkailemaan puunäkymässä. Kaikki editorin yläreunassa olevat napit on tehty käyttäen QToolButton–luokkaa.

Kun käyttäjä on tehnyt haluamansa muutokset, hän voi tallentaa ne XML-tiedostoon ja lähettää ne releelle.

IEC 61850–editorin luokka on QDialog–luokan alaluokka.



Kuva 12: *IEC 61850-editori ja sen puunäkymät.*

4.3 Kommunikaatio

Asetustyökalulla voidaan ottaa yhteys releeseen. Tämä hoidetaan Qt:n QTcpSocket-luokan avulla. Tälle annetaan releen IP-osoite sekä portti jonka jälkeen yhteys muodostetaan. Yhteen releeseen voidaan muodostaa vain 1 yhteys kerrallaan, eli monta asetustyökalua ei voi olla yhteen releeseen samanaikaisesti yhteydessä.

Asetustyökalussa on ”Live edit”-tila. Jos releeseen on otettu yhteys, eikä tämä tila ole päällä, kaikki muutokset mitä tehdään eivät mene releelle. Releestä ei myöskään lueta mitään vielä. Vain ”Live edit”-tilassa kaikki muutokset mitä tehdään asetustyökalussa menevät suoraan releelle. Arvojen lukeminen releestä tapahtuu myös vasta sitten kun ”Live edit”-tila on päällä.

Yhteys muodostetaan omassa luokassa, joka pyörii omassa säikeessä. ”Live edit”-tilassa, tämä luokka lukee jatkuvasti releen tietokannasta tietoa ja päivittää ne tietokoneen RAM-muistiin. Releestä voidaan lukea maksimissaan 64 parametriä kerrallaan. Kun käyttäjä haluaa lukea jonkun arvon releestä, tämä ei koskaan tapahdu suoraan. Ohjelma kertoo kommunikaatioluokalle, että tämä arvo halutaan päivittää, jolloin kommunikaatio-luokka aloittaa päivittämään tätä arvoa RAM-muistiin. Kaikki arvot, jotka laitetaan näkyviin, luetaan aina RAM-muistista. Releeseen kirjoittaminen tapahtuu kuitenkin kirjoittamalla käyttäjän syöttämä arvo RAM-muistiin sekä releeseen.

Releeseen voidaan myös muodostaa FTP-yhteys. Tätä käytetään kun asetustyökalun pitää lähettää tiedostoja releelle tai kun asetustyökalun pitää hakea tiedostoja releestä. FTP-yhteys muodostetaan käyttämällä Qt:n QFtp-luokkaa. Tälle annetaan IP sekä portti, joka on aina 21. Tämän jälkeen kirjaudutaan FTP-palvelimen sisään.

Tämä oli vaihe, jossa Qt:n omat luokat ja niiden hienous tuli hyvin esiin. Oli helppoa muodostaa TCP–yhteys sekä FTP-yhteys käyttämällä Qt:n omia luokkia.

4.4 Testaus

Asetustyökalun testaus tapahtui jatkuvasti. Osa testauksesta voidaan toteuttaa pelkällä asetustyökalulla ja osaan tarvitaan relettä. Esimerkiksi logiikkaeditoria voidaan testata ilman relettä, kun taas kommunikaatiofunktioiden testaamiseen tarvitaan relettä.

Yrityksen työntekijät ovat käyttäneet asetustyökalua jo aika pitkään ja he ilmoittavat aina kun löytävät jotain virheitä ohjelmasta. Asetustyökalua kehitetään jatkuvasti, joten testaamista pitää tehdä koko ajan. Myös asiakkaille on jaettu asetustyökalua. Osa virheistä on ollut heidän ilmoittamia, joten hekin toimivat testaajina.

Projektissa käytetään myös versionhallintaa. Versionhallintaohjelmistona toimii Git. Asetustyökalun uusin versio lähetetään versionhallintaan kun asetustyökaluun tehdään jotain muutoksia. Tänne myös merkitään mitä muutoksia on tehty.

Tulevaisuudessa käyttäjät voivat ilmoittaa virheistä kertomalla mitä ohjelmistoversiota he ovat käyttäneet ja millainen virhe on tapahtunut. Tämä lisätään virhelistaan, johon merkitään aina kun jokin virheistä on korjattu. Tämä toimii siten, että aina kun tulee jokin virhe, annetaan sille ”lippu”. Tällä lipulla on koodi. Kun lipussa ilmoitettu virhe on korjattu, tämä lippu merkataan suljetuksi.

Ohjelmaan tulee muutoksia jatkuvasti. Jotkut raitkaisut eivät tyydytä, joten on etsittävä toinen raktaisu. Kun tämä ratkaisu on toteutettu, tätä testataan ja verrataan edelliseen. Jos tämä ratkaisu on parempi kuin edellinen, se otetaan käyttöön. Muulloin palataan takaisin edelliseen ratkaisuun, kunnes keksitään toinen ratkaisu sille.

5 LOPPUPÄÄTELMÄT

Projekti on onnistunut hyvin. Projektin alkuvaiheessa oli vaikea kuvitella, että asetustyökalu tulisi toimimaan näin hyvin. Alussa projekti lakkasi välillä etenemästä, koska ei ollut omaa osaamista tarpeeksi. Omien taitojen kehittyessä projekti eteni aina vain paremmin. Nyt projekti on hyvässä vaiheessa. Projekti tulee kuitenkin jatkumaan vielä pitkään, koska asetustyökalu on jatkuvassa kehityksessä.

Projekti on kehittänyt minun ohjelmointitaitoja suuresti ja tämä oli minun mielestäni tärkein osa tätä projektia. Projektin aikana on tullut opittua paljon Qt:sta sekä C++:sta. Kun projekti alkoi, Qt ei ollut ollenkaan tuttu. Tämän takia projektin alkuvaiheessa tuli tehtyä typeriä juttuja, joita on myöhemmin joutunut korjaamaan. Esimerkiksi osoitinmuuttujien käyttö oli alussa huonoa. Monesti osoittimia tungettiin turhiin paikkoihin. Jälkeenpäin nämä jouduttiin korjaamaan. Monesti tuli myös tehtyä eri asioita vaikeamman tavan kautta, koska Qt:n luokat eivät olleet tuttuja. Myös funktioiden järjestely jäi alussa kokonaan huomioimatta. Tämä kostautui myöhemmin, kun tuli huomattua, että funktioiden määrän lisääntyessä oli todella vaikeaa löytää oikea funktio koodista. Tämän takia myöhemmin piti lähteä järjestelemään funktioita uudestaan.

Myös kommenttien käyttö oli tosi laiskaa. Monesti ohjelmointivaiheessa on jätetty kommentit pois, koska on luultu, että se kohta koodista on ollut selvä. Myöhemmin kun on joutunut palaamaan takaisin samaan koodipätkään, on joutunut miettimään mitä kyseisessä kohdassa tapahtuu. Tämä oli erittäin tuskallista funktioissa, joissa oli suuri määrä koodirivejä. Piti lähteä seuraamaan tiettyjä muuttujia funktion aluista asti, että ymmärtäisi mikä niiden tarkoitus on. Tämä olisi pystytty välttämään laittamalla muutama kommentti siihen kohtaan.

Myös ”forward declaration” oli kokonaan uusi asia. Alussa ihmeteltiin tosi kauan, kun ei tiennyt mitä tehdä, jos oli 2 luokkaa joiden headerit tarvitsivat toisiaan. Jos yritti sisällyttää toisen headerin toisen headeriin sekä toisinpäin, tuli virhetilanne vastaan. Tämä ongelma pystyttiin ratkaisemaan forward declarationin avulla.

Tämä tarkoittaa sitä, että jätetään headereiden sisällytykset pois headereista ja ”esimääritetään” luokka headerissa (**Kuva 13.**).

a.h

```
1  #ifndef A_H
2  #define A_H
3
4  #include "b.h"
5
6  class A
7  {
8  public:
9      A();
10     B objectB;
11 };
12
13 #endif // A_H
```

a.h

```
1  #ifndef A_H
2  #define A_H
3
4  class B;
5
6  class A
7  {
8  public:
9      A();
10     B objectB;
11 };
12
13 #endif // A_H
```

Kuva 13: *Vasemalla puolella näkyy b.h-headerin sisällyttäminen a.h-headeriin. Tämä tuottaa ongelmia, jos b.h-headerissa sisällytetään a.h-headeri. Oikealla puolella näkyy ratkaisu forward declarationin avulla.*

Nyt kun on kokemusta tälläisistä projekteista, mitä toivon että olisin tehnyt toisin, on C++-kielen opiskelu. Koulussa ei opiskeltu C++-kieltä ollenkaan, joten näitä taitoja minulla ei juuri ollut. Tämä aiheutti monia alkeellisia ongelmia. Jos haluaa opetella Qt-ohjelmointia, kannattaa ensiksi oppia C++-kielen perusteet. Suosittelisin myös koululle, että C++-kieltä aloitettaisiin opettamaan koulussa. Monissa kursseissa käytettiin Javaa. Mielestäni tätä käytiin liikaa läpi.

LÄHTEET

Binner. 2005. Trolltech Releases Qt 4.0. Viitattu 13.6.2012 http://dot.kde.org/2005/06/28/trolltech-releases-qt-40

Blanchette, J. & Summerfield, M. C++ GUI Programming with Qt 4. Viitattu 13.6.2012 http://www.civilnet.cn/book/embedded/gui/qt4/pref04.html

Getting started with Qt. Aarhus University. Viitattu 13.6.2012 http://devkit8000.wikispaces.com/file/view/Getting+Started+with+Qt.pdf

Google Earth. Qt:n omat sivut. Viitattu 12.6.2012 http://qt.nokia.com/qt-in-use/story/app/google-earth

Qt Inteface. Videolan Wiki. Viitattu 12.6.2012. http://wiki.videolan.org/Qt_Interface

Qt in use. Qt:n omat sivut. Viitattu 12.6.2012 http://qt.nokia.com/qt-in-use/target/desktop

Schumacher, C. 2011. Qt and the Future of KDE. Viitattu 12.6.2012. http://dot.kde.org/2011/03/03/qt-and-future-kde

Skype. Qt:n omat sivut. Viitattu 12.6.2012. http://qt.nokia.com/qt-in-use/story/app/skype

Tanilkan, S. 2011. Qt 4.8.0 Released. Viitattu 13.6.2012 http://labs.qt.nokia.com/2011/12/15/qt-4-8-0-released/

Thelin, J. 2010. Cross Platform Qt. Viitattu 12.6.2012. http://www.slideshare.net/e8johan/cross-platform-qt-4159891#

What is Trolltech? Opentopia. Viitattu 13.6.2012 http://encycl.opentopia.com/term/Trolltech