TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotuotanto

Tutkintotyö

Jarkko Nieminen

AUTOMAATIOJÄRJESTELMÄN TIEDONKERUUOHJELMISTO

Työn ohjaaja Lehtori Tony Torp
Työn teettäjä Quatrottec, valvojana Automaatioinsinööri Pasi Pesonen
Tampere 2008

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotuotanto
Nieminen Jarkko Automaatiojärjestelmän tiedonkeruuohjelmisto
Tutkintotyö 25-sivua
Työn ohjaaja Lehtori Tony Torp
Työn teettäjä Quatrottec, valvojana Automaatioinsinööri Pasi Pesonen
Toukokuu 2008
Hakusanat wxWidgets, -SQL, InTouch

TIIVISTELMÄ

Automaatiojärjestelmän tiedonkeruuohjelmiston tehtävä on kerätä anturitietoja automaatiojärjestelmästä ja mahdollistaa kerättyjen tietojen analysointi jälkikäteen.

Työ toteutettiin C++-ohjelmointikielellä ja InTouch-sovelluskehittimellä. Ohjelmiston tietovarastona toimii Microsoft SQL-tietokanta, joka on asennettu samalle koneelle kuin itse ohjelmistokin.

Tätä työtä kirjoitettaessa työ on asiakkaalla testattavana ja palautteen pohjalta sitä kehitetään edelleen ja uusia ominaisuuksia lisätään.

TAMPERE-UNIVERSITY OF APPLIED SCIENCES
Computer Systems Engineering
Software Engineering
Nieminen Jarkko

Datacollection software for automation systems
Thesis
25-pages
Thesis Supervisor
lector Tony Torp
Commissioning Company
Quatrottec, Automation Engineer Pasi Pesonen
May 2008
Keyword
wxWidgets, -SQL, InTouch

ABSTRACT

Datacollection software is used to collect sensor readings from a automation system and to analyse the collected data afterwards. The software was developed with C++ software languane and InTouch IDE. Data are saved to Microsoft SQL database, which is installed on the same computer as the software. The software is currently being tested by client and new features will be added acording to the feedback.

SISÄLLYSLUETTELO

LYHENTEIDEN JA MERKKIEN SELITYKSET

CSV	Tiedostomuoto jolla tallennetaan yksinkertaista taulukkodataa.
ODBC	Microsoftin määrittelemä rajapinta (API) tietokannoille.
IDE	Ohjelma tai joukko ohjelmia, jolla ohjelmoija toteuttaa ohjelmistoa.
Muutto	Tuotantoerä
C++	Ohjelmointikieli
ER-kaavio	Tietokannan rakenteen kuvaava relaationmalli.
SQL	Kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja.
Visual Basic	Ohjelmointikieli
Tag	Nimitys muutujalle InTouch-ympäristössä
GUI	Graafinen käyttöliittymä

1 JOHDANTO

Monilla tuotantolaitoksilla kuten paperitehtailla otetaan laadunvarmistavia näytteitä valmistettavasta tuotteesta suhteellisen harvoin. Mikäli joltakin asiakkaalta tulee tieto, että jossakin tuotantoerässä valmistetussa tuotteessa on sekundaa, täytyy kaikki tuotteet, jotka on valmistettu samojen kahden näytteenonton välissä kuin virheellinen tuote, kutsua takaisin. Tällä tavalla toimittaessa saattaa mennä hukkaan myös hyvälaatuisia tuotteita, koska ei ole mitään keinoa tarkistaa, mitkä näytteenottojen välillä tuotetut tuotteet olivat huonolaatuisia ja mitkä hyvälaatuisia. Tämän ongelman ratkaisuksi kehitettiin tiedonkeruujärjestelmä, jolla olisi mahdollista analysoida tietoja myös näytteenottojen välillä, jotta vain vialliset tuotteet tarvitsisi kutsua takaisin.

Työn tavoitteena on toteuttaa automaatiojärjestelmän tiedonkeruuohjelmisto, jota tullaan käyttämään erilaisten automaatiojärjestelmien tiedonkeruuseen sekä kerättyjen tietojen analysointiin. Varsinainen tiedonluku automaatiojärjestelmään kytketyltä logiikalta ei kuulu tämän työn aihepiiriin.

Asiakasvaatimuksia ovat automaatiojärjestelmään kytkettyjen anturien arvojen näyttö sekä mahdollisuus jälkikäteen hakea anturien arvoja halutuilta ajoilta ja piirtää näistä histogrammeja. Lisäksi ohjelman piti tallentaa tietoja muutoista eli tuotantoeristä. Tietoja piti pystyä hakemaan myös tietyn tuotantoerän ajalta.

Luvussa 2 kuvataan lyhyesti järjestelmän vaatimukset. Luvussa 3 kerrotaan tässä ohjelmistossa käytetystä WxWidgets-käyttöliittymäkirjastosta. Luvussa 4 kerrotaan muista tässä ohjelmiston toteutuksessa käytetyistä työkaluista. Luku 5 kuvaa varsinaista sovelluksen toteutusta, ja luvussa 6 on esitetty sovelluksen käyttöliittymä. Luvussa 7 esitetään ohjelman jatkokehitysmahdollisuuksia, ja luvussa 8 on esitetty yhteenveto työstä.

2 AUTOMAATIONJÄRJESTELMÄN TIEDONKERUUOHJELMISTO

Ohjelmiston tarkoituksena on mahdollistaa automaatiolaitteistolta tulevien anturitietojen tallentaminen tietokantaan ja tarjota työkalut tietojen hakemiseen sekä analysointiin jälkikäteen. Ohjelmisto koostuu kolmesta osasta: tiedonkeruu, tiedonanalysointi ja tietokanta. Kaikkien osien pitää toimia Microsoft Windows -käyttöjärjestelmällä.

Tiedonkeruuohjelman tehtävä on näyttää enintään 8 järjestelmään kytketyn anturin lukemat ja tallentaa ne määrätyin väliajoin tietokantaan. Tiedonkeruuohjelma tallentaa myös muuttojen alkamis- ja loppumisajankohdat tietokantaan. Lisäksi tiedonkeruuohjelmasta täytyy pystyä säätämään anturitietojen tallennuksen tiheyttä.

Tietokannan tehtävä on toimia tietovarastona muutoille sekä kerätyille anturitiedoille. Tietokannan hakujen keston maksimiajaksi määriteltiin 10 s, mutta käytännössä hakujen kesto pyrittiin rajoittamaan korkeintaan muutamaan sekunttiin

Tiedonanalysointiohjelman tehtävä on tarjota tilaisuus analysoida kerättyjä tietoja jälkikäteen antamalla mahdollisuus hakea haluttujen anturien arvoja tiettynä aikavälinä ja tietyn tuotantoerän ajalta. Haetuista tiedoista pitää pystyä piirtämään kuvaajia, jonka toisella akselilla on aika ja toisella akselilla haettujen sensorien arvot. Kuvaajia pitää pystyä zoomaamaan ja skaalaamaan vapaasti. Lisäksi kuvaajan tietyn kohdan ajankohta ja sensorin arvo sinä ajankohtana pitää saada tarkasti selville.

3 WXWIDGETS-KÄYTTÖLIITTYMÄKIRJASTO

Wxwidgets on Lgpl-lisenssin alainen, alustariippumaton avoimen lähdekoodin käyttöliittymäkirjasto, josta on olemassa C++:n lisäksi implementaatiot myös monille muille ohjelmointikielille, kuten Pythonille, Javalle ja Perlille. Käyttöjärjestelmistä WxWidgetistä löytyy porttaukset yleisimmille PC-käyttöjärjestelmille eli Microsoft Windowsille, Mac OS:lle ja Linuxille. Mobiilikäyttöjärjestelmistä WxWidgetistä on olemassa porttaukset Microsoft Windows CE:lle sekä PALM OS:lle. Myös Symbian-käyttöjärjestelmälle on tulossa oma porttaus, mutta se oli tätä kirjoitettaessa vielä kehitysasteella/1/.

WxWidgets käyttää käyttöliittymäkomponettien piirtämiseen käyttöjärjestelmän omia elementtejä. Tämä tekee ohjelmista nopeampia ja käyttöjärjestelmän grafiikkaan paremmin istuvia.

WxWidgetissä on varsinaisten käyttöliittymäkomponettien lisäksi myös muita ominaisuuksia, kuten säikeet ja socket-yhteydet. Nämä lisäominaisuudet ovat myös alustariippumattomia, mikä mahdollistaa WxWidgets-ohjelmien kääntämisen useille käyttöjärjestelmille ilman muutoksia lähdekoodiin.

Tärkein syy WxWidgetsin valintaan tähän ohjelmistoon oli sen ilmaisuus myös kaupallisissessa käytössä sekä siihen valmiina olevat laajat valikoimat käyttöliittymäkomponentteja.

WxWidgetsin pääasiallinen tapahtumakäsittelytapa on tapahtumataulukoiden (event table) käyttäminen. Jokainen luokka, joka on periytetty WxEvtHandler-luokasta, voi sisältää tapahtumataulukon. Tapahtumataulukko on staattinen rakenne, jossa on yhdistetty tietyn käyttöliittymäkomponentin lähettämä tapahtuma luokassa määriteltyyn tapahtumankäsittelyfunktioon.

Tapahtumankäsittelyfunktioiden pitää olla void-tyyppisiä ja niillä pitää olla yksi parametri, joka on saman tyyppinen kuin funktioon tapahtumataulukossa yhdistetty tapahtuma. Mikäli luokka käyttää tapahtumataulukkoa, täytyy se kertoa kääntäjälle kutsumalla DECLARE_EVENT_TABLE()-makroa luokan määrittelyssä. Alla on esitetty esimerkki tapahtumia käyttävästä luokasta, jossa on määritelty tapahtumankäsittelyfunktiot OnQuit, OnSize ja OnButton.

```
class MyFrame : public wxFrame
{
public:
// Constructor
MyFrame(const wxString& title);

// Event handlers
void OnQuit(wxCommandEvent& event);
void OnSize(wxSizeEvent& event);
void OnButtonOK(wxCommandEvent& event);
private:
// This class handles events
DECLARE_EVENT_TABLE()
};
```

Jotta käyttöliittymäkomponentit voisivat lähettää tapahtumia, täytyy niitä luotaessa niille antaa oma tunniste, jota käytetään tapahtumataulukossa kyseisen käyttöliittymäkomponentin lähettämien tapahtumien yhdistämiseen tapahtumankäsittelyfunktioihin. Alla olevassa esimerkissä luodaan WxButton-tyyppinen olio, jonka rakentajan toinen parametri on sille annettava tunniste.

```
wxButton* button = new wxButton(this, wxID_OK, wxT(“OK”),
wxPoint(200, 200));
```

Alla on tapahtumataulukon luomiseen käytetty koodi. BEGIN_EVENT_TABLE- ja END_EVENT_TABLE-makroilla määritetään kääntäjälle tapahtumataulukon alku ja loppu. Muut makrot yhdistävät jokainen yhden tapahtuman tapahtumankäsittelyfunktioon. EVT_BUTTON-makro yhdistää aiemmassa

koodiesimerkissä luodun painonapin myframe-luokan OnButtonOK-funktioon, jota kutsutaan aina kun nappia painetaan.

```
// Event table for MyFrame
BEGIN_EVENT_TABLE(MyFrame, wxFrame)

EVT_MENU (wxID_EXIT, MyFrame::OnQuit)
EVT_SIZE ( MyFrame::OnSize)
EVT_BUTTON (wxID_OK, MyFrame::OnButtonOK)

END_EVENT_TABLE()
```

Tapahtumataulukot ovat staattisia eikä niitä voi muuttaa ajoaikaisesti. Mikäli ohjelmassa olevan painonapin toimintaa täytyy muuttaa ohjelman suorituksen aikana, täytyy käyttää wxEvtHandler-luokasta löytyviä Connect- ja Disconnect-funktioita. Alla on esitetty Connect-funktion syntaksi. Funktiolle välitetään parametereina tapahtuman lähettävän käyttöliittymäkomponentin tunnus, tapahtuman tyyppi sekä osoitin funktioon, jota kutsutaan kun tapahtuma havaitaan.

```
frame->Connect( wxID_EXIT,
wxEVT_COMMAND_MENU_SELECTED,
wxCommandEventHandler(MyFrame::OnQuit) );
```

4 KEHITYSYMPÄRISTÖ JA TYÖKALUT

Tässä kappaleessa on esitetty lyhyesti työssä käytetyt työkalut ja kerrottu syyt niiden valintaan.

4.1 Ohjelmointikielen valinta

Ohjelmointikieleksi valittiin C++, koska työn tekijällä oli siitä eniten kokemusta. Myös C++-ohjelmien nopeus oli tärkeä seikka ohjelmointikieltä valittaessa, koska tässä ohjelmassa jouduttaisiin käsittelemään suuriakin tietomääriä.

4.2 InTouch

InTouch on automaatiojärjestelmien valvomo-ohjelmistojen tekemiseen tarkoitettu ohjelma. InTouchissa on monia valmiita käyttöliittymäkomponetteja, joista ohjelman käyttöliittymä voidaan rakentaa helposti ”drag and drop” -menetelmällä. Intouch-ohjelmiin voi myös skriptata lisätoiminnallisuuksia hieman Visual Basicia muistuvalla skriptikielellä. Intouchilla on myös mahdollista lukea anturin lukemia automaatiojärjestelmään kytketyltä logiikalta ja käyttää tietokantaa ODBC-ajurin kautta, mitkä olivat kaksi syytä InTouchin käyttöön tässä ohjelmassa.

InTouch-ohjelmistoa voi käyttää vain Windows-käyttöjärjestelmässä.

4.3 Wxdev-C++

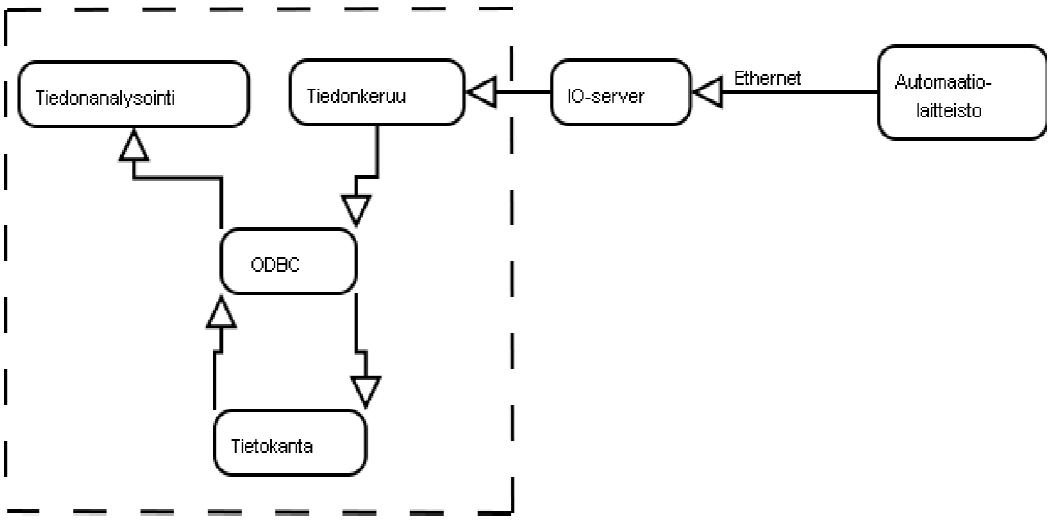
Wxdev-C++ on Dev-c++-kääntäjään perustuva IDE, joka on tehty erityisesti Wxwidgets-ohjelmien kehittämiseen. Ohjelma sisältää GUI-editorin sekä debuggauksessa tarvittavat työkalut. Muita vaihtoehtoja olisivat olleet code::blocks ja WxFormBuilder, joista jälkimmäinen olisi vaatinut vielä erikseen kääntäjän.

4.4 Microsoft SQL server

Microsoft SQL Server päätettiin valita tietojen tallennuspaikaksi, koska InTouchin OBDC-rajapinta tuki virallisesti vain sitä ja Oraclen tietokantaa. Microsoft SQL-serveristä on myös olemassa ilmaisversio, jossa tietokannan koko on rajoitettu.

5 SOVELLUKSEN TOTEUTUS

Kuvassa 1 on esitetty tiedonkeruujärjestelmän rakenne. Tämän opinnäytetyön alue on ympyröity katkoviivalla. Järjestelmään kytketyt anturit ovat kiinni automaatiolaitteistossa, josta niiden arvot lukee ethernetin yli IO-server. IO-server on osa InTouch-ohjelmaa.



Kuva 1 **Järjestelmän rakenne**

Ohjelmisto on jaettu kahteen osaan: Intouchilla tehtyyn tiedonkeruuohjelmaan ja C++-ohjelmointikielellä tehtyyn tiedonanalysointiohjelmaan. Tähän päädyttiin, koska työn tekijällä oli enemmän kokemusta C++:sta, jolla oli lisäksi mahdollista tehdä monipuolisempi ohjelma kuin melko rajoittuneella InTouchilla. Tietokantaa käytetään anturilukemien sekä muuttojen varastointiin. Sekä tiedonkeruu- että tiedonanalysointiohjelma kommunikoivat tietokannan kanssa ODBC:n kautta.

5.1 Tiedonkeruuohjelma

Tiedonkeruuohjelman rakenne melko yksinkertainen, ja se koostuu muutamasta globaalista aliohjelmasta, jotka muodostavat ohjelman toiminnallisuuden.

Ohjelmassa joka sensorilla on oma tagi, jota vähennetään sekunnin välein, kun sen arvo saavuttaa nollan tallennetaan kyseisen sensorin lukema tietokantaan, minkä jälkeen tagin arvoksi asetetaan asetustiedostossa määritelty tallennusväli.

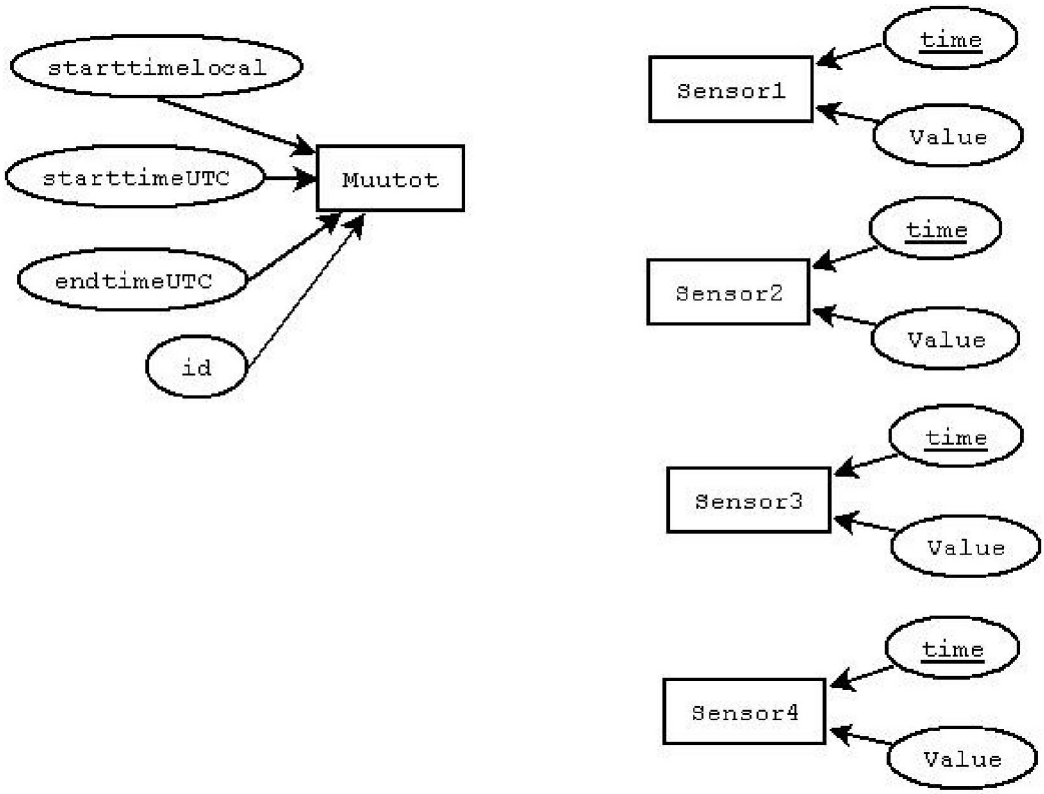
Varsinainen tiedon tallennus tietokantaan tehdään DataToSql-aliohjelmalla, joka saa parametrina tallennettavan tiedon ja anturin lukeman. DataToSql käyttää tiedon tallentamiseen InTouchin omaa OBDC-kirjastoa.

Konfiguraatiotiedostojen käsittelyyn käytetään ReadConfigFile- ja WriteConfigFile-aliohjelmia. ReadConfigFile-aliohjelmalla luetaan ohjelman alussa sensorien tiedot ohjelmaan. WriteConfigFile-aliohjelmalla voidaan asetukset tallentaa konfiguraatiotiedostoon, mikäli käyttäjä muokkaa niitä.

Trendikuvaajien näyttöön käytetään InTouchista valmiina löytyvää trendline-käyttöliittymäkomponenttia.

5.2 Tietokanta

Kuvassa 2 on esitetty ohjelman käyttämän tietokannan er-kaavio. Muutot-taulussa on varastoituna tiedot muutoista viimeisen kahden vuoden ajalta. Starttimelocal-ominaisuus kertoo, mikä oli paikallinen aika, kun muutto alkoi, ja sitä käytetään tiedonhaussa hakemaan muutot tietyn vuorokauden ajalta. StarttimeUTC ja endtimeUTC kertovat muuton alku- ja loppuajankohdat UTC-aikana. Id on muuton numero, joka kasvaa yhdellä, aina kun tauluun lisätään uusi muutto ja nollautuu joka päivä kello kuusi vuoronvaihdon yhteydessä.



Kuva 2 **Tietokannan er-kaavio**

Varsinaiset sensorilukemat ovat varastoituna sensor-tauluhin, siten että jokaisella sensorilla on oma taulunsa. Kuvassa 1 on esitetty selkeyden vuoksi vain neljä sensor-taulua, mutta todellisuudessa niitä on yhtä monta kuin sensoreitakin. Sensor-taulujen time-ominaisuuteen on datetime-tyyppisenä tallennettu sensorilukeman ajankohta UTC-aikana ja value-ominaisuuteen varisnainen sensorin arvo.

Sensorilukemat olisi periaatteessa voinut tallentaa myös yhteen tauluun, mutta silloin tiedonhaku ei olisi ollut yhtä nopeaa, koska yleensä kerrallaan haetaan vain muutaman sensorin tiedot ja yhdestä suuresta taulusta niiden hakeminen olisi hitaampaa. Useamman taulun käyttö mahdollistaa myös sen, että eri sensorit voivat tuottaa liukulukujen lisäksi esim. kokonaislukuja, joiden tallentaminen liukulukumuodossa kasvattaisi tietokannan kokoa turhaan.

Hakujen nopeuttamiseksi sensor-tauluissa on time-kentästä muodustuva indeksi, josta on lisäksi sekin hyöty, että sensorilukemat palautuvat valmiiksi aikajärjestyksessä ilman että tarvitsee tehdä mitään lajitteluja SQL-hauissa tai jälkikäteen tiedonanlysointiohjelmassa.

Tietokantapalvelimena toimi Microsoft SQL server express edition. Se on myös kaupallisessa käytössä ilmainen ohjelma, jossa tietokannan koko on rajoitettu 4 GB:n kokoon, joka riittää järjestelmän ensimmäisen version tarpeisiin, jossa järjestelmään on kytketty vain 8 anturia. Mikäli anturimäärää aiotaan kasvattaa, täytyy SQL-tietokannaksi valita jokin kooltaan rajoittamaton tietokanta.

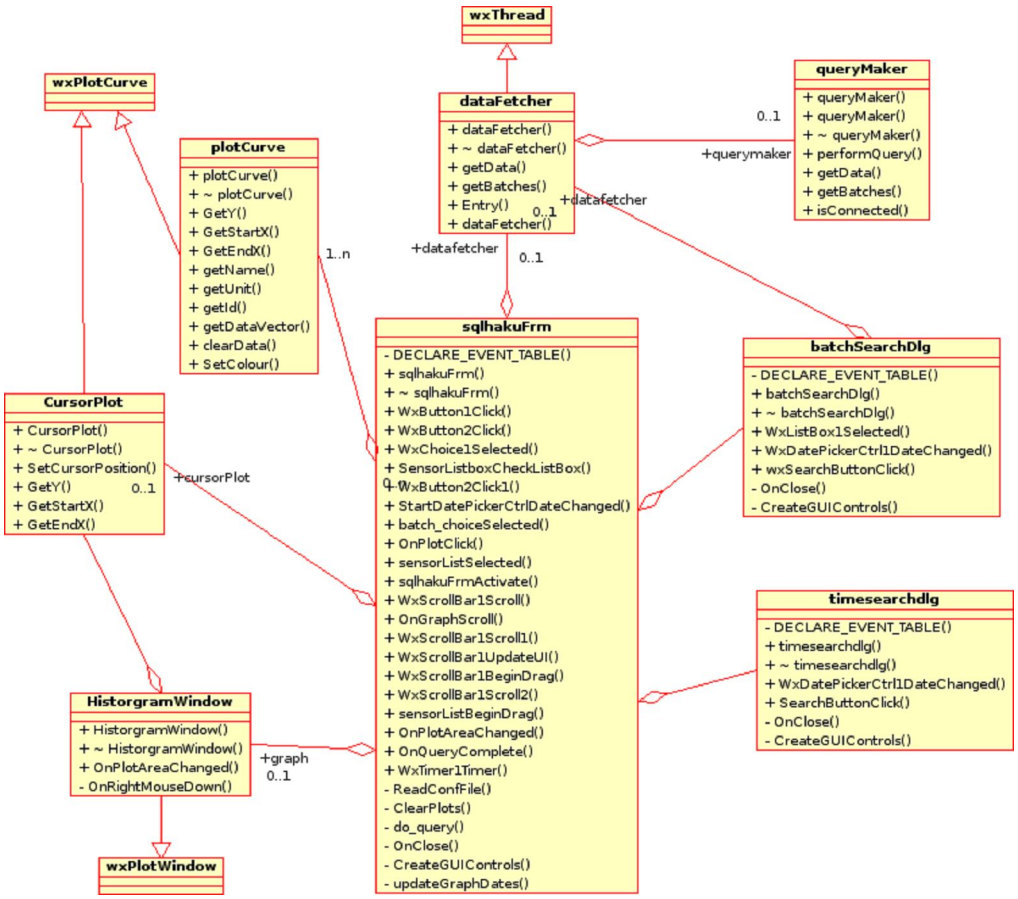
5.3 Tiedonanalysointiohjelma

Tässä luvussa kerrotaan sovelluksen tiedonanalysointiohjelman toteutuksesta.

5.3.1 Luokkakaavio

Kuvassa 3 on esitetty tiedonanalysointiohjelman luokkakaavio. Ohjelman suurin luokka on sqlhakuForm, joka muodostaa batchSearchDlg- ja timesearchdlg-luokkien kanssa ohjelman käyttöliittymän sekä ohjelman toimintalogiikan. DataFetcher- ja queryMaker-luokkia käytetään tiedonhakuun tietokannasta. PlotCurve-, CursorPlot- ja HistogramWindow-luokkia käytetään histogrammin

näyttämiseen.



Kuva 3 **Tiedonanalysointiohjelman luokkakaavio**

5.3.2 Tiedonhaku tietokannasta

Tiedonhakuun tietokannasta käytetään datafetcher- ja querymaker-luokkia. Datafetcher-luokka on käyttöliittymän rajapinta tiedonhakuun.

Käyttöliittymä tekee sensoritiedonhaun kutsumalla datafetcher-luokan getData-aliohjelmaa, jolle välitetään parametreina aikaväli, jolta sensoritiedot haetaan sekä STL-vektori, joka sisältää sensoreita edustavat cursorplot-oliot. GetData-aliojelmaa kutsuttaessa Datafetcher-olio käynnistää uuden säikeen, jossa se Querymaker-luokkaa käyttäen tekee tiedonhaun. Querymaker-luokka tekee varsinaiset sql-kyselyt tietokantaan käyttäen datalayer-ODBC-kirjastoa.

5.3.3 Sensoritietojen näyttö

Histogrammin näyttöön käytetään kuvaajien piirtoon käytettyä wxPlotWindow-käyttöliittymäkomponenttia. Varsinainen sensoridata sijaitsee PlotCurve-tyyppisissä olioissa, joihin sensoridata on tallennettua STL-vektoriin, siten että jokainen vektorin alkio vastaa anturin arvoa tietyn vektorin indeksiä vastaavan sekunnin aikana. Esim. vektorin 3600. alkion arvo on sensorin arvo tunti (3600 s) sensorin ensimmäisen arvon jälkeen. Tällaiseen tietorakenteeseen päädyttiin, koska se on nopea, mikäli histogrammi pitäisi piirtää uudestaan. Muistinkäytön kannalta optimaalisempi rakenne olisi ollut jonkinlainen tietorakenne, jossa varsinaisen sensorilukeman lisäksi tallennettaisiin myös sensorilukeman ajankohta, jolloin esim. 5 sekunnin välein tietokantaan tallennetuista sensorin arvoista tarvisisi tietorakenteeseen panna 1/5-osa sensorilukemia verrattuna nykyiseen tietorakenteeseen. Käytännössä kuitenkin säästöt muistinkäytössä olisivat vähäiset, koska myös sensorin lukeman ajankohdan tallentaminen vie muistia ja tietokantaan tallennetaan tietoa suurimmalta osalta sensoreita vain muutaman sekunnin välein, joten varsinaisten tietokannasta löytyvien sensorilukemien väliin jää vain vähän ”ylimääräisiä” sensorilukemia.

PlotCurve-luokka perii WxPlotcurve-rajapintaluokan, joka mahdollistaa sen näyttämisen kuvaajana WxPlotWindow-tyyppisessä käyttöliittymäkomponentissa. PlotCurve toteuttaa joukon rajapintaluokan vaatimia funktioita, joista tärkein on GetY, joka palauttaa kuvaajan arvon tietyllä hetkellä.

Cursorplot on luokka, jota käytetään histogrammia hiirellä klikattaessa käytetyn pystysuoran kursorin piirtämiseen. WxPlotWindow-luokasta ei löytynyt valmiina tällaisen kursorin piirtämisen mahdollistavaa toiminnallisuutta, joten se päätettiin toteuttaa erillisellä luokalla, joka on itseasiassa vain histogrammiin piirrettävä kuvaaja.

5.4 Konfiguraatiotiedostot

Sekä tiedonkeruu- että analysointiohjelma käyttävät samoja konfiguraatiotiedostoja. Niiden muotona on CVS, johon päädyttiin, koska InTouch tukee CSV-tiedostoja suoraan ja C++:lla CSV:n luku on helppo toteuttaa. Konfiguraatiotiedostoja on kaksi kappaletta: config.csv ja intervals.csv.

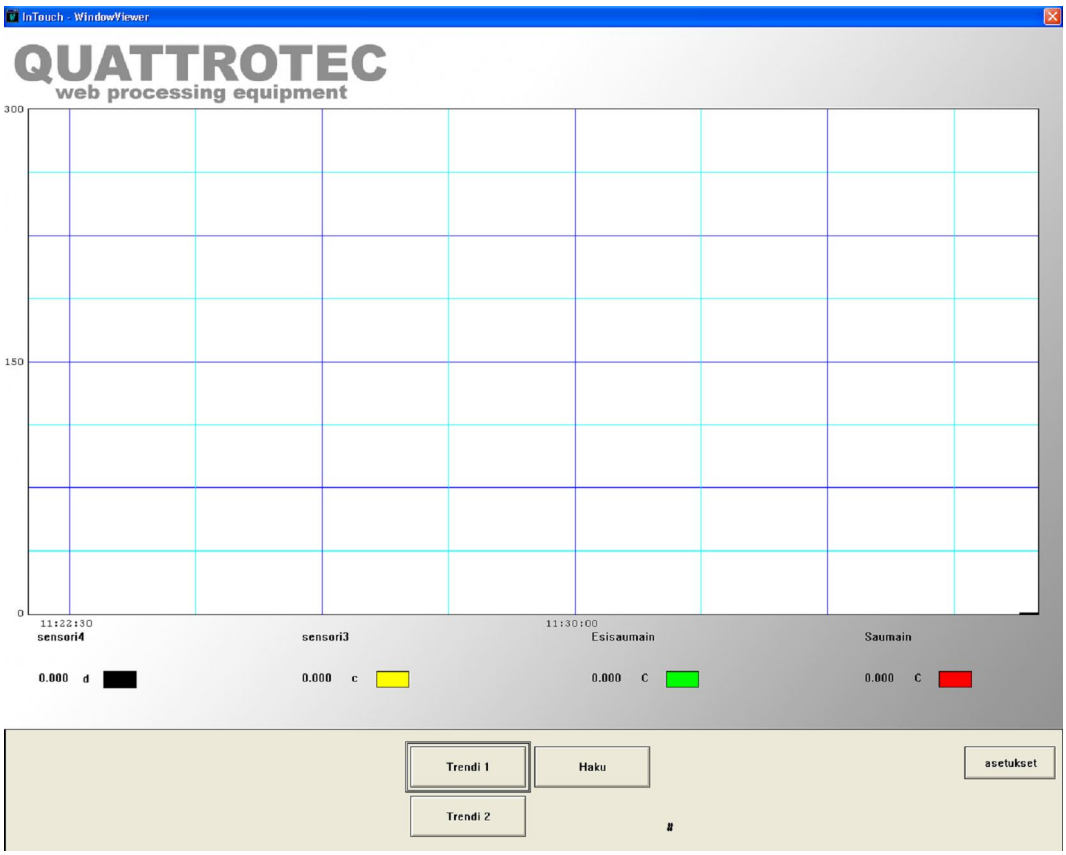
Config.csv:n ensimmäisellä rivillä on sensoreiden lukemien yksiköt pilkulla erotettuna. Toisella rivillä on antureiden nimet, kolmannella rivillä minimiarvot, jotka anturit voivat saada, ja neljännellä rivillä ovat maksimiarvot. Intervals.csv-tiedostossa on yksi rivi, jolla ovat antureiden näytteenoton aikavälit sekunteina.

6 SOVELLUKSEN KÄYTTÖLIITTYMÄ

Tässä kappaleessa on esitetty sovelluksen eri näytöt ja kerrottu lyhyesti niiden toiminnasta.

6.1 Anturitietojen näyttö

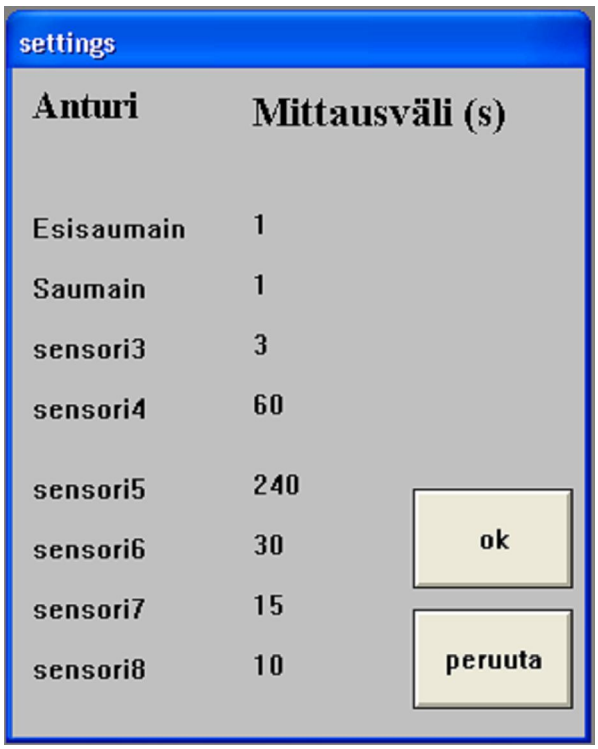
Kuvassa 4 on kuvattu tiedonkeruuohjelman anturitietojen näyttö. Ruudulla on suuri trendikuvaaja, josta näkyy neljän anturin arvojen kehitys ja tämänhetkiset arvot. Trendikuvaajia voi olla ohjelmassa useampia, ja niiden välillä voi liikkua ruudun alalaidassa olevilla napeilla. Tiedonanalysointiohjelman voi käynnistää haku-napista. Asetukset-napista avautuu oma ruutunsa josta voi muuttaa tiedonkeruuohjelman asetuksia.



Kuva 4 **Anturilukemien näyttö**

6.2 Astetusruutu

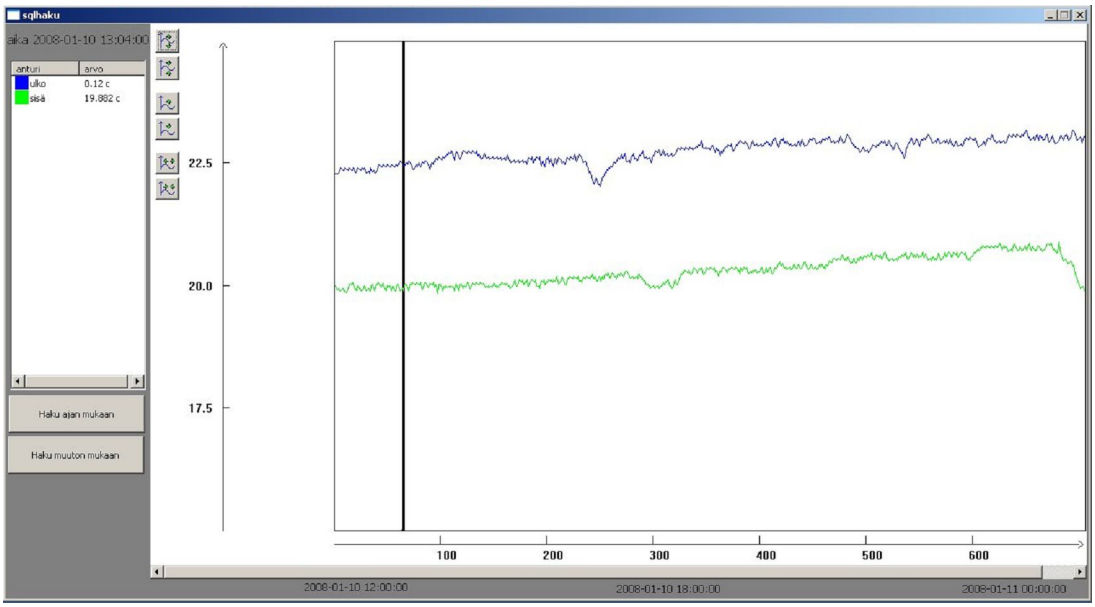
Kuvassa 5 on esitetty tiedonkeruuohjelman asetusruutu. Sen alulla voi muuttaa sensorien lukemien tallennusväliä sekunteina. Ok-napin painaminen tallentaa muutokset konfiguraatiotiedostoon, kun taas Peruuta-napin painaminen sulkee asetusruudun tallentamatta muutoksia.



Kuva 5 **Asetusruutu**

6.3 Tietojen analysointinäyttö

Kuvassa 6 on esitetty tiedonanalysointiohjelman päänäyttö.
Histogrammikuvaajasta näkee valittujen anturien kehityksen tietyllä aikavälillä. Histogrammia hiirellä klikkaamalla ohjelma piirtää valittuun kohtaan pystysuoran kursorin ja näyttää näytön vasemmassa reunassa anturien arvot kursorin kohdalla. Histogrammia voi zoomata ja skaalata sen vasemmalla puolella olevilla napeilla.



Kuva 6 **Tietojen analysointinäyttö**

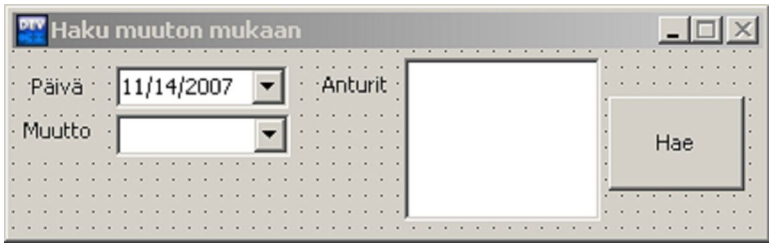
6.4 Haku ajan mukaan

Kuvassa 7 on esitetty dialogi, jolla voi tehdä hakuja tietyltä aikaväliltä. Alkupäivä- ja Loppupäivä-kohdista voi valita haun alku- ja loppupäivän, ja alkuaika ja loppuaika kohdista vastaavasti alku- ja loppuajan. Kellonaikoja ei tarvitse kirjoittaa sekunnin tarkkuudella, vaan käyttäjä voi halutessaan kirjoittaa myös pelkät tunnit tai tunnit ja minuutit. Dialogin oikeassa reunassa olevasta listasta voi valita ne sensorit, joiden tiedot haetaan.

Kuva 7 **Hakuajanmukaan-dialogi**

6.5 Haku muuton mukaan

Kuvassa 7 on esitetty dialogi, jolla voi tehdä hakuja tietyn muuton ajalta. Kun päivä kohdasta valitaan jokin päivä, muutto kohtaan tulee lista kyseisen päivän muutoista, joista käyttäjä voi valita muuton jonka ajalta haku tehdään. Dialogin oikeassa reunassa olevasta listasta voi valita sensorit, joiden tiedot haetaan.



Kuva 8 **Hakumuutonmukaan-dialogi**

7 JATKOKEHITYSSUUNNITELMAT

Työtä tehtäessä tuli esille muutama jatkokehitysajatus. Microsoft SQL server vaihdetaan mahdollisesti Postgres SQL-serveriin, koska siitä on olemassa kaupallisessa käytössä ilmainen versio, jossa ei ole tietokannan kokorajoituksia. Tiedonkeruuohjelmaan lisätään mahdollisesti hälytykset, jolloin käyttäjä saa ilmoituksen, mikäli jonkin anturin arvo menee määrättyjen raja-arvojen yli tai ali. Myös tietojen analysointia voidaan monipuolistaa lisäämällä tiedonanalysointiohjelmaan mahdollisuus tulostaa erilaisia raportteja esim. jonkin anturin minimi- ja maksimiarvoista sekä keskihajonnasta. Ohjelmaan saatetaan lisätä myös varmuuskopiointitoiminto, koska tällä hetkellä se puuttuu kokonaan.

8 YHTEENVETO

Työn lopputuloksena syntyi valmis ohjelmisto, joka täytti sille asetetut vaatimukset. Sisällöllisesti työ osoittautui kiinnostavaksi sen sisältäessä monipuolisia osa-alueita, kuten käyttöliittymän ja tietokannan suunnittelua.

Vaikka tietokannoista työn tekijällä olikin perusteet hallussa jo ennen työn aloittamista, myös niistä opittiin paljon uutta esim. eri tyyppisistä indekseistä ja niiden vaikutuksesta hakuaikoihin erityyppisillä hauilla.

Ohjelmiston suuritöisin osa oli tiedonanalysointiohjelman tekeminen. Siihen sisältyi useita osa-alueita, esim. käyttöliittymän suunnittelu ja toteuttaminen. Tiedonanalysointiohjelman työmäärä lisääntyi siksikin että työntekijällä oli vain vähän aikaisempaa kokemusta WxWidgetsistä, joten sen joutui opiskelemaan sen melko lailla alusta asti. WxWidgets-käyttöliittymäkirjasto tulikin työtä tehtäessä tutuksi ja sitä opittiin käyttämään monipuolisesti.

WxWidgets osoittautui toimivaksi käyttöliittymäkirjastoksi mutta sen dokumentaatio oli joiltakin osiltaan puutteellista. Erityisesti tiedonanalysointiohjelman kuvaajan piirtoon käytetyssä Wxplotwindow-käyttöliittymäkomponentissa oli useita täysin dokumentoimattomia ominaisuuksia, joista osa toimi ja osa ei.

WxDev-C++ osoittautui vielä melko keskeneräiseksi ja epävakaaksi kehitystyökaluksi. Se kaatuili usein ja vaikka tietoa ei varsinaisesti menetetty en itse kyseistä kehitystyökalua enää muissa projekteissa käyttäisi ainakaan vielä tässä kehitysvaiheessa.

Jälkikäteen ajateltuna tiedonkeruuohjelman toteuttamiseen C++-ohjelmointikieltä ja WxWidgets-käyttöliittymäkirjastoa paremmin työn toteuttamiseen olisi ehkä sopinut C#-ohjelmointikieli, jonka avulla työn ohjelmointiosuudesta olisi päässyt vähän helpommalla suorituskyvyn kuitenkaan suuremmin kärsimättä.

LÄHTEET

Sähköiset lähteet

1 WxWidgets S60 porttaus [www-sivu]. Saatavissa: http://www.wxwindows.org/wiki/index.php/

Painetut lähteet

2 J.Smart, K.Hock, Cross-Platform GUI Programming with wxWidgets, Pearson Education, 2006, 744 s.