Timo Hietala

SÄÄASEMA

Opinnäytetyö
Keski-Pohjanmaan Ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Marraskuu 2009

KESKI-POHJANMAAN AMMATTIKORKEAKOULU
MELLERSTA ÖSTERBOTTENS YRKESHÖGSKOLA

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö **Ylivieskan yksikkö**	**Aika** 20.11.2009	**Tekijä/tekijät** Timo Hietala
Koulutusohjelma Tietotekniikka		
Työn nimi Sääasema		
Työn ohjaaja Ritva Saviluoto		**Sivumäärä** [36 + 45 liitettä]
Työelämäohjaaja Hannu Puomio		

Työssä tehtiin LabVIEW-ohjelmointiympäristön avulla ohjelmisto, joka mittaa jännitettä antureilta sekä tallentaa suureet määrätyin väliajoin tietokantaan. Mitattuja sääsuureita pystytään katsomaan myös selaimen avulla.

Sääasema koostuu kolmesta ohjelmakomponentista pääohjelmasta, verkkokäyttöliittymästä sekä asiakassovelluksesta.

Asiasanat
Flex, LabVIEW, MySQL, Sää

CENTRAL OSTROBOTHNIA
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

CENTRAL OSTRO-BOTHNIA UNIVERSITY OF APPLIED SCIENCES	**Date** 20.11.2009	Author Timo Hietala
Degree programme Information technology		
Name of thesis Weather Station		
Instructor Ritva Saviluoto		**Pages** [36 + 45 attachments]
Supervisor Hannu Puomio		

In the thesis a software which measures voltages at the sensors and records measured quantities to the database in designated intervals, was made with the help of LabVIEW programming environment. Measured weather quantities can be viewed with a web browser.

The weather station consists of three program components which are the main program, the web interface and the client application.

Key words
Flex, LabVIEW, MySQL, Weather

LYHENTEET

AI	Analog Input
DAQmx	Data AcQuisitation
ECMA	European Computer Manufacturers Association
Flex	Adobe Systemsin ylläpitämä avoimen lähdekoodin kehitysympäristö
GNU	Gnu's Not Linux
J2EE	Java 2 platform, Enterprise Edition
LabVIEW	National Instrumentsin kehittämä graafinen ohjelmointiympäristö
MySQL	MY Structured Query Language
MXML	Macromedia eXtensible Markup Language
NI	National Instruments
ODBC	Open DataBase Connectivity
RDBMS	Relational DataBase Management System
SQL	Structured Query Language
WWW	World Wide Web
XML	eXtensible Markup Language

TIIVISTELMÄ
ABSTRACT
LYHENTEET

SISÄLLYS

1 Johdanto

Tämän työn tarkoituksen oli tehdä sääasemasovellus käyttäen LabVIEW-ohjelmointiympäristöä sekä siihen liitettyä MySQL-tietokantaa. Työn pohjana käytettiin sääkonetyötä (Hietala T. 2009). Miksi haluttiin tehdä sitten sääasema? Kaupallisia sovelluksia on markkinoilla melko yleisesti, mutta niissä on tiettyjä puutteita, kuten jos halutaan lisätä antureita jälkeenpäin järjestelmään sekä tiedon välittäminen internettiin tietokantaa käyttäen. Haluttiin toteuttaa sääaseman ohjelmisto käyttämällä National Instrumentsin kehittämää LabVIEW ohjelmistoa.

Työn aloituksen jälkeen kävi hyvin pian selväksi, että ohjelmointi pitää tehdä vähintään kahdessa osassa, koska ohjelman toiminnot piti saada hyvin pitkälle myös internetin puolelle näkyviin. Pelkästään yhtä ohjelmointiympäristöä käyttämällä ei työtä pystyisi saattamaan loppuun eli joutuisi käyttämään varmasti ainakin kahta ohjelmointikieltä. Johtuen edellä mainitusta opeteltavaksi tuli uusi ohjelmointikieli Flex, jota kutsutaan myös MXML-kieleksi. Flex ohjelmointia käytetään internetin puolelle sijoitettavan asiakasohjelman ohjelmoinnissa. Asiakassovelluksen avulla on helppo katsella sääaseman arvoja suhteellisen reaaliaikaisesti. Sovellukseen haluttiin myös aikaisemmin mitattujen arvojen hakeminen netin yli.

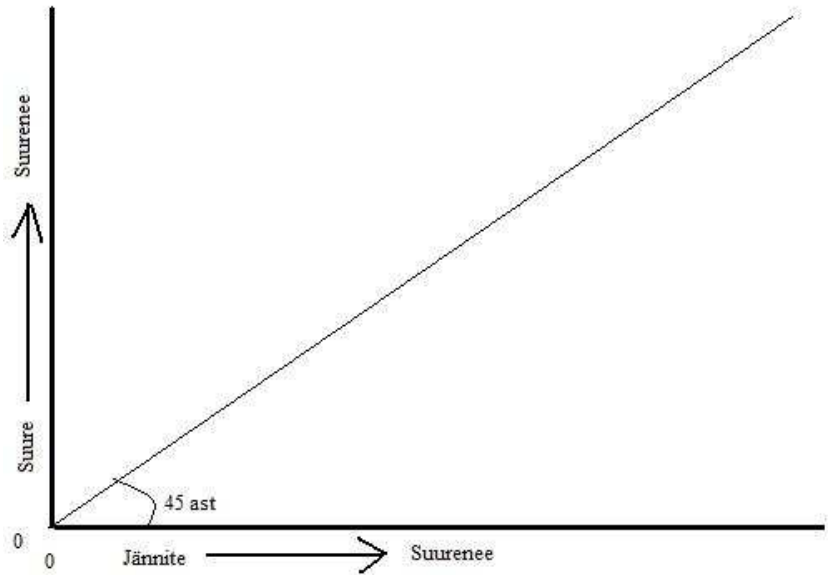
2 Sääaseman anturit

2.1 Yleiset määritelmät

Yleisesti sanottuna anturit ovat teknologiaa, joilla suureita voidaan tarkastella toisessa muodossa. Esimerkiksi halutaan mitata lämpötilaa, sen voi tuntea iholla ja nähdä miten vesi muuttaa muotoaan lämpötilan mukaan. Mutta miten sen arvon voi mitata? On kehitettävä teknologia, jolla pystytään mittaamaan sen arvo. Perinteinen elohopealämpömittari on myös anturi, koska siitä näkee suureen arvon, mikä lämpötila on tällä hetkellä. Yleisesti pääteltynä anturit ovat itse asiassa aika oiva keksintö.

Ohjelmaan liitettävien säätilan antureiden pitää täyttää seuraavat määritelmät, jotta laitteistoa voidaan käyttää: Anturi tai muunnoslaite tuottaa jännitesignaalia, jonka arvo vaihtelee välillä 0 ja 10 volttia, kun anturin mittaama suure muuttuu, niin jännitteen pitää vaihtua samanaikaisesti vastaamaan mitattua suuretta.

Anturit, jotka liitetään ohjelman käsiteltäväksi, pitää olla suhteellisen suorasti muuttuvia. Suhteellisella tarkoitetaan, että jännitteen arvo muuttuu tasaisesti suureen arvon muuttuessa. Kuviosta 1 nähdään miten suureiden arvot pitäisi muuttua, eli kun jännite kasvaa niin suureen arvokin kasvaa.



Kuvio 1. Havainnointi kuvio anturin toiminnasta.

2.2 Kytkennät sääasemaan

Anturit sijoitetaan ulkotiloihin, kunkin anturin valmistajan määritelmien mukaan. Antureiden mittausjänniteulostulo tuodaan johtimia pitkin kytkentäalustan läheisyyteen. Johtimet liitetään NI 6034E mittauskortille, käyttämällä National Instrumentsin tekemää kytkentäalustaa CB-68LP. CB-68LP:n kytkentöjen numerointi, vastaa täsmälleen kuviossa 2, olevaa NI 6034E mittauskortin numerointia.

Kytkennät kytkentäalustalle määriteltiin valmiiksi, jotta sääarvojen mittausohjelmisto voitaisiin tehdä. Tuulennopeuden anturin mitattava jännitearvo tuli sijoittaa AI0 liittimeen. Tuulensuunnan jännitteen mittaus tapahtuu liittimestä AI1. Kosteuden anturin mitattava jännitearvo kohdasta AI2. Lämpötilan jännitteen mittausliittimestä AI3. Ilmanpaineen anturin jännitetieto mitataan AI4 liittimestä. Sademäärän anturin jännitteen mittausliittimestä AI5. Lisätietoa porteista on kuviossa 2, joka esittää NI 6034E mittauskortin liittimen kytkentöjä. CB-68LP kytketään mittauskorttiin NI 6034E, käyttämällä NI R6868 kaapelia. Mittauskortti NI 6034E pitää olla asennettuna tietokoneeseen ennen antureiden kytkentää.

Kuvio 2. 6034E mittauskortin liitinjärjestys (NI 6034E/6035E/6036E Family Specifications).

Määriteltyjä kytkentöjä pitää noudattaa, jotta ohjelma voisi määritelmän perusteella näyttää oikeat mittaustulokset. Esimerkiksi jos lämpötilan mittaus kytketään johonkin muuhun liittimeen kuin AI3 vaikka AI5:een, tällöin ohjelma lukee jännitteen arvon väärin ja ei toimi toivotulla tavalla. Sääasemaohjelmaan tehdään varotoimenpide tällaisten tilanteiden varalle, tarkemmin anturin määrittelystä järjestelmään lisätietoa on liitteessä 4.

2.3 Tuulennopeus

Tuulennopeuden mittaukseen käytetään yleisesti anemometrejä. Anemometrejä on monenlaisia kuppianemometrejä, tuulimyllyn tapaisia anemometrejä, kuumalanka anemometrejä, lasermittaukseen perustuvia anemometrejä ja paljon muita. (WikipediaF)

Yleisesti käytetyt tuulennopeutta mittaavat anemometrit ovat yleensä ilman vauhdin mittaukseen perustuvia. Harvinaisempia anemometrejä ovat ilmanpaineen mittaukseen perustuvat laitteet. Ilmanpaineen mittaukseen perustuvat anemometrit ovat harvinaisia koska niiden mittausvirhe voi olla huomattavan iso ja suuret nopeuden vaihtelut jäävät yleensä mittaamatta antureiden hitauden takia. (WikipediaF)

Tuulennopeuden anturi oli valmiina työn aloitusvaiheessa, joten tässä työssä käytetään tuulennopeuden mittaukseen kyseistä anturia (Hietala T. 2009). Anturina käytetään tässä työssä Vaisalan tekemää kuppianemometria, tyypiltään WAA15A. Anturi on optoelektroninen anemometri. Anturin pyörimiskynnys on alle 0,5 m/s, mittausalue on tähän työhön säädetty 0 – 51,2 m/s. Anturia ei voi liittää suoraan järjestelmään, sillä anturi antaa ulostulona jännitepulssin. Ulostulona olevan jännitemuodon takia, signaalin muuntamisessa tasaiseksi jännitteeksi käytetään Vaisalan erityisesti tarkoitukseen tekemää mittamuunninta WAT12.

2.4 Tuulensuunta

Tuulensuunnan mittauksessa anturi on yleensä sellainen jonka toimintasäde akselinsa ympäri on 360 astetta. Tuulensuuntaa mitataan yleisesti tuuliviireillä, jotka näyttävät visuaalisesti mistä päin tuulee. Tuuliviiri yleensä sijoitetaan korkeimmalle paikalle, jotta saadaan mahdollisimman tarkka tulos tuulensuunnasta, ilman häiritseviä tuulen pyörteitä esimerkiksi rakennusten aiheuttamia vaikutuksia. Modernit tuulensuunnan mittauslaitteet ovat yleensä vain pelkkä nuoli joka osoittaa tuulensuunnan. (WikipediaG)

Työssä käytetään suunnanmittaukseen anturia, jota käytettiin aikaisemmassa sääkonetyössä (Hietala T. 2009). Anturia käytetään, koska näin vältytään uuden anturin hankinnalta. Anturi on Vaisalan valmistama tuulilevy WAV15A. Anturi antaa kulman suuntatietona 6-bittisen Gray-koodin. Gray-koodi perustuu binäärikoodiin, jossa numeron muuttuessa vain yksi bitti muuttuu (Gray Code). Suoraan ei anturia voida kytkeä järjestelmään, koska anturi ei toteuta yleisiä määrittelyjä. Ulostulosignaalin takia joudutaan käyttämään Vaisalan tekemää mittamuunninta WAT12, joka muuntaa Gray-koodin haluttuun sähköiseen muotoon.

2.5 Kosteus

Kosteusantureita käytetään mittaamaan suhteellista kosteutta. Ilmankosteuden mittaukseen käytetään yleensä anturia, jonka mittausväli voi vaihdella 0-100 % välillä. Suositeltava kosteusanturityyppi sääasemaan on elektroninen kosteusanturi. Elektronisista kosteusantureista kaksi yleisintä perustuvat resistanssin tai kapasitanssin muuttumiseen. Kapasitiivinen anturi käyttää vaihtojännitettä mittaamaan kapasitanssin muutosta, kuinka paljon vettä ilmassa on. Resistiivinen anturi käyttää polymeeristä johdinta, jonka avulla voidaan määrittää johtavuuden muutoksia kosteuden muuttuessa. Käyttökohteita kosteusantureille on teollisuudessa, saunoissa ja museoissa. (WikipediaH)

2.6 Lämpötila

Lämpötilan mittaukseen käytetään yleensä resistiivisiä antureita, platinametallikela on yksi esimerkki resistiivisestä anturista. Resistiiviset anturit yleensä antavat paremman tarkkuuden ja toistettavuuden kuin termoelementit. Resistiiviset anturit ovat yleensä tehty platinasta, koska sen resistanssin muutos on hyvin lineaarinen lämpötilan muutoksen kanssa. Termoelementit ovat antureita, jotka generoivat pientä jännitettä ja jota voidaan mitata. Termoelementin jännite muuttuu, kun resistiivisen anturin resistanssi muuttuu lämpötilan mukaan. (Aumala O. 2003)

Yleisimmät kaupalliset platinametallianturit ovat 100 ohmisia 0 asteen lämpötilassa. Asteen muutos lämpötilassa yleensä aiheuttaa 0,385 ohmin muutoksen resistanssissa eurooppalaisen standardin mukaan. Amerikkalaisessa standardissa käytetään puhtaampaa platinaa kuin Euroopassa ja yhden asteen muutos aiheuttaa 0,392 ohmin muutoksen resistanssissa. (WikipediaI)

2.7 Ilmanpaine

Ilmanpaineen yksikkönä käytetään hehto Pascalia eli hPa:ta. Ilmanpaineen mittaamiseen käytetään useasti barometriä. Barometriä käytetään sään ennustamisessa, korkea ilmanpaine ennakoi korkeapainetta, joka tavallisesti ennakoi hyvin kirkasta säätä ainakin kesäaikaan. Matala ilmanpaine ilmoittaa että myrskyt ovat hyvin todennäköisiä. Ilmanpaineen mittausta käyttävät meteorologiset laitokset tuottamaan yhdessä monien verkkoon liitettyjen antureitten avulla säädiagrammeja. (WikipediaJ)

Ilmanpaineen mittauksen yhteydessä useasti mitataan myös tuuliolosuhteita. Jolloin yhdessä monien eri paikoissa olevien asemien verkostojen avulla voidaan tuottaa lyhytaikaisia sääennusteita. Yleisimpiä ilmanpainemittarityyppejä ovat vesipohjaiset anturit, elohopeaanturit sekä ilman nestettä olevat anturit (WikipediaJ). Suositeltava anturityyppi sääaseman käyttöön on Vaisalan valmistama sähköinen anturi, joka sisältyy ilman nestettä oleviin antureihin.

2.8 Sademäärä

Sademäärän mittaukseen voidaan käyttää tähän sääasemaan vain sellaista anturia, joka mittaa mm/s yksikkönä sademäärän eli tavalliset pidemmältä ajalta mittaavat anturit eivät käy. Sademäärän mittaamiseen on monta eri tapaa, tapoja ovat perinteinen mittaustapa, sateen painon mittaaminen, sadepisaran tippumisen mittaaminen sekä optinen sademäärän mittari. (WikipediaK)

Perinteisellä tavalla puutarhaan jätetään astia vuorokauden ajaksi keräämään sadetta, vuorokauden kuluttua mitataan kuinka paljon on satanut. Sateen painon mittaaminen perustuu kahden peräkkäisen mitatun painon erotukseen, jolloin saadaan sataneen veden paino, jonka perusteella voidaan laskea myös sen määrä. Sadepisaran tippuminen perustuu yhden vesipisaran tipahtamiseen, jolloin elektronisesti mitataan pisaran lähteminen kerääjästä. Optinen mittaus on samantapainen sadepisaran tippumisen kanssa, mutta pisaran tipahtaminen mitataan laser-säteellä. Kun laser-säteen tielle osuu pisara, niin anturi lähettää tiedon tapahtumasta. (WikipediaK)

Sademäärän anturin lisääminen järjestelmään arvioidaan olevan kaikista vaikein toteuttaa. Miksi näin? Antureiden määrittelyn perusteella jokaiseen sademäärän anturiin pitäisi tehdä

muunnoselektroniikkaa tai olisi kehitettävä kokonaan uusi anturi, jota voidaan käyttää tähän tarkoitukseen.

2.9 Yhteenvetoa antureista

Tiivistelmänä sääasemaan suunnitelluista antureista voidaan kertoa seuraavaa: Koska jokaisen anturin pitää täyttää yleiset määritelmät, niin antureita jotka käyvät suoraan sääasemaan on rajallinen määrä. Tietenkin jos anturi käyttää jotakin muuta viestintätapaa ulostulossaan kuin jännitteen välillä 0-10 volttia, pitää ennen anturin kytkemistä järjestelmään tehdä väliin muunnospiiri, joka muuntaa anturilta tulevan tiedon 0-10 voltin jännitteeksi. Anturin pienin mahdollinen suure, joka voidaan mitata, tulisi olla mahdollisimman lähellä 0 volttia. Päinvastaisesti ajateltuna lähempänä 10 volttia oleva arvo pitää olla suurin mitattavissa oleva arvo.

Sääasemassa oli tämän opinnäytetyön teko hetkellä tuulennopeuden ja suunnan mittaamisen soveltuvat anturit, jotka olivat sijoitettuna sähkövoimatekniikan laboratorion katolle. Ensimmäisinä viikkoina tuulenmittausasema siirrettiin korkeimmalle paikalle C-rakennuksen katolle. Tuuliasema siirrettiin, koska näin ollen anturit saisivat mitattua tuulta ilman häiritseviä ilmapyörteitä. Ilman siirtoa anturit olisivat antaneet väärää tuulitietoa, jonka seurauksena mittaustuloksia ei voisi pitää luotettavina. Suunnitelmissa oli lisätä lämpötila-anturi työn kuluessa paikoilleen, mutta sitä ei ehditty lisätä tämän työn teon aikana ohjelmistoon. Anturi voidaan lisätä paikoilleen myöhempänä ajankohtana ohjelmistoon.

3 MySQL-tietokanta

MySQL on relaationalinen tietokannan hallintajärjestelmä, jolla on yli 6 miljoonaa asennusta erilaisiin käyttötarkoituksiin. Relaationalinen tietokannan hallintajärjestelmä eli englanniksi sanottuna lyhenteenä RDBMS perustuu relaatiomalliin, jonka esitteli brittiläinen tietokonetutkija Edgar F. Codd ollessaan IBM yhtiön palveluksessa. (WikipediaE)

MySQL on monisäikeinen palvelin. Se tarkoittaa, että yhteyden muodostuessa aloitetaan joka kerta uusi palvelinprosessi. Yhteydet MySQL:n eivät jaa prosesseja, jos prosessi päättyy odottamatta tai ylikuormittaa palvelimen käyttämällä runsaasti muistia, vain yksi prosessi sulkeutuu eikä koko palvelin kaadu. (Meloni J. C. 2003 s. 12)

MySQL on käytettävästä käyttöjärjestelmästä riippumaton, koska ohjelman lähdekoodi on vapaasti saatavilla verkossa GNU General Public License alaisena. Ohjelman koodaus on toteutettu pääasiassa C/C++ ohjelmointikieltä käyttäen, SQL-komentojen jäsentäjä on toteutettu käyttäen yacc-sovellusta ja lex-ohjelman yhdistelmää. (WikipediaE) MySQL:ä käytetään tässä työssä, koska se on ilmainen tietokantaohjelmisto.

MySQL:n merkittävämpiä saavutuksia ovat ensimmäinen sisäinen julkaisu 23. toukokuuta 1995, samana vuonna MySQL AB niminen yhtiö perustettiin. Sittemmin Sun Microsystems osti MySQL AB:n 26 helmikuuta 2008, jonka jälkeen tietokannan kehitys on ollut Sun Microsystemsin vastuulla. (WikipediaE)

4 Ohjelmointiympäristöt

4.1 LabVIEW

LabVIEW on National Instrumentsin kehittämä graafinen ohjelmointiympäristö, joka on ollut markkinoilla vuodesta 1986 ja on tällä hetkellä markkinajohtaja teollisuudessa. (NI, What is Labview?) Tämän työn teossa käytetään pääasiassa LabVIEWiä ohjelmiston tekoon. Graafisessa ohjelmointiympäristössä on muutamia hyviä puolia: Ohjelman toimintojen virheiden löytäminen on helpompaa kuin tekstipohjaisessa ympäristössä. Ei tarvitse kirjoittaa tuhansia riviä koodia kuin esim. C++-kielessä, vaan riittää kun sijoittaa ohjelman toiminnan algoritmit graafisesti.

Hyvien ominaisuuksien lisäksi LabVIEW:n avulla tehtyyn sovellukseen liittyy muutamia huonoja asioita: Ohjelmaa ei voi rakentaa ohjelmointiympäristön avulla niin, että se toimisi ilman minkäänlaisia suoritusympäristöä eli konekieliseksi sitä ei voi tehdä suoraan. Mikäli pelkkä luotu exe-tiedosto kopioidaan toiselle koneelle, niin se ei toimi. Ohjelman pystyy kyllä liittämään asennuspakettiin, johon saadaan suoritusympäristöt mukaan, jolloin asennuspaketin avulla ohjelma toimii jokaisessa koneessa. Ominaisuudet, mitkä toimivat ohjelmointiympäristössä, eivät välttämättä toimi käännetyssä ympäristössä, kun koodista tehdään esimerkiksi verkkopalvelu vi.

Ohjelman toimintaan liittyvä käsite vi on lyhennys englannin kielen sanoista virtual instrument. LabVIEW ohjelmia sanotaan vi:ksi, koska ohjelmat matkivat usein oikeita instrumentteja, vaikka ne ovat todellisuudessa virtuaalisia instrumentteja. (What is VI)

Huonojen ominaisuuksien takia LabVIEW:tä ei voi suositella ohjelmointiohjelmaksi, jota käytetään päivittäin tuottamaan ohjelmia ja sitten jaellaan päivityksiä käyttäjille. Hyvien ominaisuuksien puolesta ohjelmaa voidaan käyttää esim. kun halutaan testata jonkin elektroniikan toimivuus ilman, että kirjoitetaan jopa tuhansia rivejä koodia. Yksi hyöty LabVIEW:a verrattuna toisiin kehitysympäristöihin on erittäin laaja ohjelmistotuki instrumentointilaitteistoille. Ajurit ja käsitteelliset kerrokset ovat saatavilla monille eri instrumentointilaitteille ja väylille. (WikipediaA)

Tehokkuuden näkökulmasta katsottuna LabVIEW sisältää kääntäjän, joka tuottaa natiivin koodin kulloinkin käytössä olevalle suoritinalustalle. Graafinen koodi käännetään suoritettavaksi konekieliseksi koodiksi tulkitsemalla koodin syntaksia ja käännöstä. LabVIEWin syntaksi on hyvin voimakkaasti vahvistettu muokkausprosessissa, joka käännetään suoritettavaksi konekieliseksi koodiksi, kun ohjelman suoritusta pyydetään tai ohjelman tallennuksen yhteydessä. Jälkimmäisessä tapauksessa suorittava ohjelma ja lähdekoodi on merkitty yhteen tiedostoon. Ohjelma suoritetaan LabVIEW:n ajon aikaisen apuohjelman avulla, joka sisältää monia valmiiksi käännettyjä koodeja, jotka on määritelty graafisessa ohjelmoinnissa. Ajonaikainen apuohjelma vähentää ohjelman käännösaikaa ja tuo vakio rajapinnan useille eri käyttöjärjestelmille, graafisille järjestelmille, laitteistokomponenteille ja niin edelleen. (WikipediaA)

4.2 Adobe Flex

Asiakassovellus on tehty Flex Builderilla. Miksi valittiin Flex Builder asiakassovelluksen tekoon? Syitä on pari kappaletta. Ensiksi Flex ympäristössä on hyvin monia valmiiksi määriteltyjä funktioita, joiden avulla ohjelman kirjoittaminen ei vie niin paljon aikaa kuin esim. C/C++-kielessä. Ei tarvitse kirjoittaa perusfunktioita, riittää kun käyttää valmiiksi määriteltyjä funktioita, niin ohjelman pituus ei ole suuri. Toinen asia miksi Flex valittiin, liittyy hyvin paljon Internetin puolelle sijoitettuihin Web Service vi:den käyttöön. Flexillä on helppo kirjoittaa sovelluksia, jotka käännetään sitten Flash sovelluksiksi, tarkoittaen että käyttäjän ei tarvitse asentaa koneellensa mitään muita lisäosia kuin yleisesti käytetty Flash player.

Adobe Flex ohjelmistorajapinta on käyttöjärjestelmäriippumaton, jolla voi luoda hyvin korkealaatuisia internet-sovelluksia ja jotka toimivat samalla tavalla kaikissa yleisissä selainmerkeissä. (Adobe, Adobe Flex Framework Technologies)

Adobe Flexin historia alkaa maaliskuusta vuonna 2004 jolloin Macromedia sisällytti julkaisuun ohjelmistokehitysalustan, työkalun sekä J2EE integrointisovelluksen, joka tunnettiin nimellä Flex Data Services. Sittemmin Adobe osti Macromedian vuonna 2005, jonka jälkeen tulleet Flexin julkaistut ohjelmat eivät enää vaatineet kallista lisenssiä Flex Data Services:iin, joka erotettiin Flex alustasta erilliseksi tuotteeksi ja nimettiin LiveCycle Data

Services. Helmikuussa 2008 Adobe julkaisi Flex 3-ohjelmistokehitysalustan avoimena lähdekoodina, joka tunnetaan nimellä Mozilla Public License. (WikipediaB)

Adobe Flex lyhyesti sanottuna on MXML-ohjelmointikielen ja Actionscriptin yhdistelmä. Mikä MXML-ohjelmointikieli sitten on? MXML on XML pohjainen käyttöliittymämerkintäkieli, joka esiteltiin ensimmäisen kerran maaliskuussa vuonna 2004 silloisen Macromedian toimesta.(An overview of MXML: The Flex markup language) Actionscript on alun perin Macromedian kehittämä scriptauskieli, joka perustuu ECMAScriptiin, ECMAScript on ECMA Internationalin standardoima scriptauskieli, jonka standardin järjestysnumero on ECMA-262. (WikipediaD)

5 Ohjelman suunnittelu

Työn alussa tehtiin suunnitelma, minkälainen ohjelman pitäisi olla. Heti aluksi ohjelmointi jaettiin kahteen osaan LabVIEW ohjelmointiin sekä Web-ohjelmointiin, joista LabVIEW-ohjelmointi tulisi olemaan melko iso osa työtä.

LabVIEW-ohjelmointia käsitellään tässä kahtena osana pääohjelmana sekä nettikäyttöliittymäohjelmointina. Asiakassovelluksen eli Web-ohjelmoinnin toteutustavasta ei vielä ensimmäisinä päivinä suunnittelua ollut juuri tietoa.

Yleisesti ohjelman toiminnoiksi haluttiin saada seuraavat asiat: Ohjelman pitää tallentaa ennalta määrätyn ajan välein mittaustuloksia tietokantaan, varmuuskopioida koko tietokanta tiedostoon, poistaa kaikki tietokannassa olevat tiedot, hakea varmuuskopioista tiedot tietokantaan, antureiden lisäys ohjelmaan sen teon jälkeen, näyttää ”reaaliaikaisia” mittaustuloksia internetissä, hakea internetin kautta aikaisemmin mitattuja sääarvoja halutulta ajan jaksolta, ohjelman pitää olla käytettävästä koneesta riippumaton. Haluttujen toimintojen lisäksi ohjelman tulisi olla riittävän helppokäyttöinen.

5.1 Pääohjelma

Pääohjelman suunniteltiin toimimaan palvelinkoneessa niin että se on koko ajan päällä, kun säätietoja mitataan. Suunnitteluvaiheessa, joka kesti hyvin lyhyen aikaa todettiin, että aiemmin tehty sääkonetyö ei kelvannut suoraan ohjelman pohjaksi, joten jouduttaisiin tekemään koko ohjelma uusiksi alusta asti. Aivan alusta asti ohjelmaa ei tarvinnut aloittaa, sillä sääkonetyössä oli hyviä aliohjelmia, joita pystyisi käyttämään myös uudessa pääohjelmassa. Sääkonetyössä olleet tietokantaan kirjoittavat sekä lukevat aliohjelmat olisivat erityisesti tähän tarkoitukseen sopivia, joten niitä voitiin käyttää myös tässä tarkoituksessa. (Hietala T. 2009)

Ensimmäisinä suunnittelupäivinä ohjelmoinnin vaatimusmäärittelyjä käydessä läpi, ohjelmistorajoitukset huomioiden päätettiin, mitkä määrittelyjen osat joudutaan toteuttamaan serverillä olevalla ohjelmistolla, joka on koko ajan päällä. Vaatimusmäärittelyjä katsoessa kävi selväksi, mitkä ominaisuudet piti pääohjelman suorittaa. Ominaisuudet ovat:

- Tallentaa ennalta määrätyin väliajoin tietokantaa mittaustulokset, tähän tapaukseen valittiin 1 minuutin väliajoin tapahtuva tallennus.
- Varmuuskopioida tiedot tietokannasta tiedostoon, toteutus olisi yksinkertainen.
- Hakea varmuuskopioidusta tiedostossa olevat arvot ja ladata ne saadata-tietokantaan.
- Antureiden lisäys ohjelmaan, joka tulisi tapahtumaan ohjelman teon jälkeen.
- Ohjelman tulee olla käytettävästä koneesta riippumaton.

Ohjelman ominaisuuksien määrittelyjen jälkeen huomattiin yksi merkittävän asian, koska ohjelma pitää olla käytettävästä koneesta riippumaton, joudutaan tekemään rutiinit, jotka kertovat ohjelman suorituksen sijainnin tiedostojärjestelmässä.

5.2 Nettikäyttöliittymä

Nettikäyttöliittymänä tarkoitetaan tässä tapauksessa LabVIEW:lle erikoista verkkopalvelu-vi:tä. Verkkopalvelu-vi tarkoittaa kevytrakenteista sovellusta, joka käyttää National Instrumentsin kehittämää arkkitehtuuria nimeltä RESTfull. Verkkopalvelu on suhteellisen tuore lisäys LabVIEW:in. Verkkopalveluksi tekeminen LabVIEWissä on ollut olemassa versioista 8.6 asti. Miksi verkkopalvelu on lisätty vasta näin myöhään? Yksi selitys valmistajan mukaan on seuraava: Verkkopalvelu toiminto lisättiin versioon 8.6 vastauksena asiakkaiden pyynnöstä saada avoimempi ja standardimpi tapa kommunikoida verkon yli vi:hin. (Web Services in LabVIEW)

Katsottaessa ohjelmoinnin vaatimusmäärittelyä nettikäyttöliittymän toteutettavaksi tuli kaksi asiaa. Tavoitteena oli muodostaa linkki verkon ja pääohjelman välille ja hakea pääohjelmasta mittaustiedot ns. ”reaaliaikaisesti”, joka ei tässä tapauksessa voinut olla miten-kään paljon reaaliaikainen. Miksi ei sitten voinut olla reaaliaikainen? Syitä ei ole kuin yksi. Reaaliaikaisesti tehtynä ohjelma tulisi syömään hirveästi palvelinkoneen resursseja, joten käytännön syistä ei olisi kannattavaa tehdä reaaliaikaista. Lisäksi reaaliaikaisuus ei olisi muutenkaan mahdollista, johtuen laitteistossa tapahtuvasta viiveestä. Nettikäyttöliittymän kautta piti olla haettavissa tietokantaan jo tallennettuja mittausarvoja, niin että käyttäjälle tarjottiin halutulta aikaväliltä tehtyä tiedostoa. Jälkimmäisen nettikäyttöliittymän vaatimusmäärittelyä arvioiden, tässä jälkimmäisessä menee eniten aikaa.

Suunnitelmana oli tehdä nettikäyttöliittymä kahdessa osassa viimeisimmän mittaustuloksen hakeminen sekä halutun aikavälin tapahtumien lataaminen tiedostona käyttäjän koneelle. Viimeisimmän mittaustuloksen hakeminen määriteltiin tapahtuvan seuraavasti: Asiakassovellus tekee pyynnön palvelimen osoitteeseen ”/web/web”, jolloin asiakassovellus saa tiedon viimeisimmästä tietokantaan tallennetusta arvosta.

Halutun aikavälin hakeminen tietokannasta tiedostoon määritelmä on seuraava: Asiakassovellus tekee pyynnön palvelimen osoitteeseen ”/web/data”, johon asiakassovellus tekee pyynnön käyttäen alkuaikaa sekä loppuaikaa esimerkiksi näin: Käyttäjä valitsee asiakassovellukseen alkuajan sekä loppuajan, jotka tässä esimerkissä ovat seuraavat: Alkuaika on 1.10.2009 klo 2.30 ja loppuaika on 4.10.2009 klo 5.00. Asiakassovellus muuttaa ajat näin alkuaika on ”01%10%2009klo02&30” ja loppuaika on ”04%10%2009klo05%00”, jonka jälkeen asiakassovellus tekee pyynnön nettikäyttöliittymälle ”/web/data”, johon on lisätty alkuaika sekä loppuaika näin ”/web/data/01%10%2009klo02&30/04%10%2009klo05%00” vastauksen nettikäyttö liittymän pitää antaa tiedostonimi asiakassovellukselle, jonka avulla se voi hakea tiedoston nettiserverin juurihakemistosta.

5.3 Asiakassovellus

Ennen työn aloittamista pohdittavaksi tuli, millä tekniikalla verkon puolelle sijoitettava ohjelman tulisi toimia. Ensi pohdinnan jälkeen yritettiin php:tä asiakassovelluksen tekemiseen, mutta hyvin pian aloituksen jälkeen ja php:n käyttöön liittyvien internet sivujen lueskelun jälkeen kävi selväksi, että php ei tukenut LabVIEW:n palvelinta, joten jouduttiin ajatus php-ohjelmointikielenä hylkäämään.
Ensimmäisen suunnitelman kaaduttua piti kehittää jokin muu tekniikka asiakassovellukseen. Vähän ajan kuluttua National Instrumentsin nettisivuilla tuli vastaan Adobe Flex, -tekniikka, jolla saisi luotua asiakassovelluksen, joka kävisi tähän tarkoitukseen. Löytyipä sivuilta myös esimerkki, miten tehdä asiakassovellus ja joka toimisi nettikäyttöliittymän kanssa. (Web Services in LabVIEW) Suhteellisen nopeasti esimerkin lähdekoodin tarkastelun jälkeen kävi selväksi että pelkästään LabVIEW-ohjelmoinnilla ei lopputulokseen päästä.

Asiakassovelluksen määrittelyt ovat: Sovelluksen pitää olla mahdollisimman helppo käyttöinen, joten jo nettikäyttöliittymän suunnittelussa asia jouduttiin ottamaan mietintään.

Nettikäyttöliittymän suunnitelman mukaisesti asiakassovelluksen pitää muuttaa käyttäjän valitsemat alku ja loppupäivämäärien arvot oikeaan muotoon. Asiakassovelluksen pitää hakea myös nettikäyttöliittymän kautta viimeksi kirjoitetut arvot tietokannasta eli tietojen hakemisessa käytetään nettikäyttöliittymän suunnitelman mukaisesti ”/web/web”-käskyä, jonka avulla saadaan vastauksena tietokannasta viimeiseksi mitatut arvot. Käskyn jälkeen vastauksena saadut arvot pitää lisäksi käsitellä asiakassovelluksessa oikealla tavalla, jotta kaikki arvot näytetään oikein.

Asiakassovelluksen määrittelyjen arvioinnin jälkeen, oletetaan ohjelmoinnin olevan hyvin yksinkertainen. Joten Flex-ohjelmointi jätetään ohjelmistojen teossa viimeiseksi asiaksi.

6 Ohjelmointi

6.1 Pääohjelma

Suunnitelman mukaisesti käytetään hyväksi sääkonetyön aliohjelmia. Erityisesti ohjelman tietokannasta tiedostoon kirjoittava aliohjelma oli hyvä pohja koko tietokannan varmuuskopiointi vaatimukseen. Myös tiedostosta lukeva ja tietokantaan tallentava aliohjelma sääkonetyöstä oli hyödyllinen. (Hietala T. 2009)

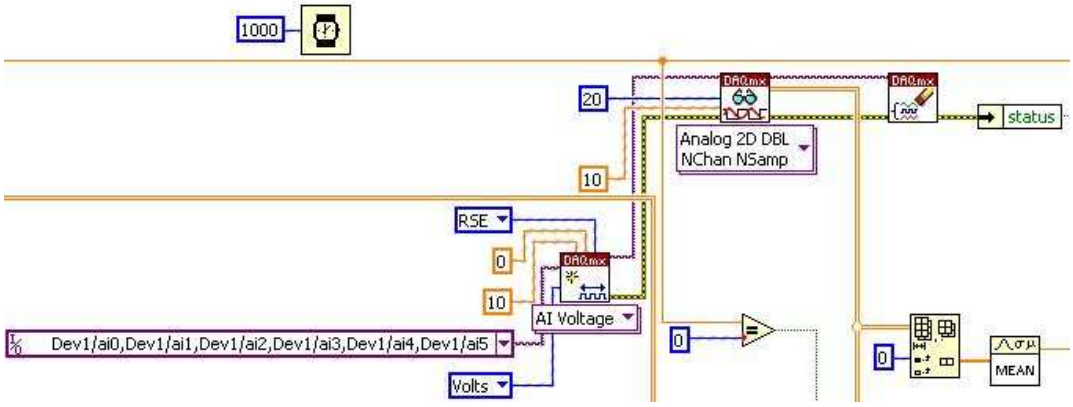
Ohjelman alkutoimintojen luomisen jälkeen kehitetään silmukka, jossa ohjelmaa pyörii niin kauan kuin se lopetetaan. Ratkaisu löytyi while-silmukasta jonka suunniteltiin käynnistyvän vasta kun alkutoiminnot oli suoritettu. While-silmukan sisälle sijoitettiin if-komponentin ja komponentin tilaa määrittämään sijoitettiin nappi, josta saa huoltotilan päälle. Huoltotilaan laitetaan seuraavat toiminnot: ’Lataa tietokanta tiedostoon’,- ’Lataa tietokanta tiedostosta’,- ’Tuhoa tietokanta’,- ja ’Määritä kaavat’ painikkeet. Huoltotilan ollessa pois päältä ohjelma pyörii, joka sekunti päivittäen mitatut arvot näkyville käyttöliittymään. Ohjelman toimittua 10 sekuntia päivitetään current -tietokantaan tämän hetkiset mittausarvot. 60 sekunnin välein ohjelma lisää uuden rivin saadata -tietokantaan. Huoltotilaan pystytään näin ollen siirtymään vajaan sekunnin sisällä napin painamisesta. Huoltotilassa pystytään tekemään tarvittavia varmuuskopioita tietokannasta sekä kirjoittamaan varmuuskopioista tietokanta uudestaan, mikäli tieto jostain syystä katoaa. Varmuuskopioiden tekemisessä ja lukemisessa käytetään sääkonetyöstä tuttuja aliohjelmia (Hietala T. 2009). Huoltotilaan siirtymisessä on pari huonoa puolta: Tietokannat eivät päivity, joten jos tuona aikana asiakassovelluksesta katsotaan säätietoja, niin ne eivät päivity.

Tietokantojen päivitystoimintojen ja huoltotilan lisäämisen jälkeen pohdittiin, millä tavalla ohjelmaan lisättäisiin uusia antureita ohjelman teon jälkeen. Ohjelmointi osoittautui yllättävän vaikeaksi, koska mistä saada muunnoskaavat signaalille sen teon jälkeen. Aikaa meni huomattavan paljon ennen ratkaisun oivaltamista. Lopulta löytyi ratkaisu miten lisätä uusia antureita. Ratkaisuna antureiden lisäämiseen jäljestäpäin käytetään huoltotilan ’määritä kaavat’ -painiketta. Kun huoltotilan ’määritä kaavat’ -painiketta painetaan, niin esiin hyppää dialogi, johon pystyy määrittämään jäljestäpäin jännitteet ja suurearvot millä anturi toimii. Tarkemmin anturin lisäyksestä ohjelmaan on selvitetty liitteessä 4. Anturin lisäysdialogi käyttää tietojen tallennukseen erikseen luotua formula.ini tiedostoa. Ohjelman

käynnistyksessä tai huoltotilan puolella käynnin jälkeen käydään aina lukemassa formula.ini tiedostosta tiedot, mitkä antureista on määritelty paikoilleen, jonka perusteella kehitetty algoritmi muuntaa mitatut arvot anturin suureiksi.

Algoritmi koostuu 4 tai 6 eri tiedosta, jotka saadaan formula.ini tiedostosta, kun anturit on määritelty ensin ohjelmaan. Algoritmi olettaa pienimmän jännitteen olevan pienin suure, jonka anturi pystyy mittaamaan. Vastaavasti suurin ilmoitettu jännite vastaa suurinta mitattua suuretta. Toiminta algoritmilla on seuraavaa: Pienin jännite oletetaan 0 pisteeksi ja suurin jännite maksimipisteeksi. Algoritmi perustuu matemaattisesti geometriaan. Jaetaan maksimisuure maksimijännitteellä, jolloin saadaan suhdeluku. Suhdelukua käytetään ratkaisemaan mitattua suuretta, kun mitattu jännite tiedetään. kerrotaan suhdeluku mitatulla jännitteellä, jolloin saadaan vastauksena suure, joka on mitattu. Algoritmin takia antureiden antamat jännitesignaalit pitää olla 45 asteen kulmassa suhteellisesti toimivia, kuten luvussa 2.1 on määritelty. Mikäli anturi antaa jännitettä muuten kuin suhteellisesti, ohjelman antama suure ei ole tällöin oikea.

Antureilta tulevien signaalien lukemiseen käytetään LabVIEWin DAQmx-palikoita. DAQmx – palikat määriteltiin lukemaan kaikki signaalit kerralla, kuten kuviosta 3 näkyy. Lisäksi arvot luetaan 20 kertaa peräkkäin, joista otetaan niiden keskiarvo, jotta yhden mittauksen virhe ei vaikuttaisi kovin paljon lopputulokseen. Mittausten keskiarvo näin ollen muunnetaan algoritmin avulla suureeksi.



Kuvio 3. Signaalin luku.

Tämän jälkeen siirryttiin testaamaan pääohjelman toimivuutta. Testauksessa ei löytynyt suurempia ongelmia. Yksi pieni ongelma tosin löytyi. Mikäli ohjelma siirrettiin johonkin toiseen hakemistoon, niin ohjelma ei toiminut oikein. Virhe liittyi ohjelman tapaan käsitellä tiedostoa eli siihen piti keksiä ratkaisu. Ongelma ratkesi tekemällä aliohjelma, joka etsi hakemiston mistä ohjelma suoritettiin. Aliohjelma liitettiin osaksi jokaiseen tiedostoja käsittelevään osaan, ongelman korjaamiseksi.

Nettikäyttöliittymän tekemisen aikoihin merkittävä virhe paljastui nettikäyttöliittymässä, jonka johdosta pääohjelman suunnitelmaan sekä ohjelmointiin tuli muutoksia, jotta saatiin toimimaan asiakassovelluksessa tietojen haku tietyltä aikaväliltä. Eri asioita kokeiltaessa haettiin ratkaisua, saisiko ohjelman tekemään mitenkään yhteistyötä muiden ohjelmien kanssa. Ratkaisu löytyi parin viikon etsinnän jälkeen. Pääohjelmaan piti tehdä funktio, joka käynnistyy vain, kun jaetun muuttujan arvo vaihtui todeksi. Jotta jaetun muuttujan toimintoja pystyi muuttamaan nettikäyttöliittymästä, piti jaettu muuttuja julkaista verkossa. Funktion, joka suoritettiin vain kun muuttuja muuttui todeksi, toiminta oli seuraava: Suoritettiin tietokannasta tiedot haku, jotta tiedettiin miltä ajanjaksolla tiedosto piti luoda määriteltyyn server -hakemistoon tietojärjestelmässä. Tietokannasta haetuista tiedoista näkyi alku- ja loppupäivämäärä kellonajalla sekä tiedosto, mihin tiedot tallennetaan. Tietokannasta saadata haetaan alku- ja loppupäivämäärän perusteella aikaisemmat mitatut tiedot ja tallennetaan haetun tiedostonimen mukaisesti tiedot tiedostoon. Ohjelmoinnin loppuvaiheessa pääohjelmasta käännettiin suoritettava exe -tiedosto.

6.2 Nettikäyttöliittymä

Sääkonetyö oli hyvä pohja aloittaa ohjelmointi. Sääkonetyössä oli pari hyvää kevyt rakenteista ohjelmaa, joiden avulla nettikäyttöliittymän ohjelmointi pääsi hyvään alkuun. Suunnitelman mukaan ”/web/web” käskyn piti hakea tietokannasta viimeisimmät tallennetut arvot current -tietokannasta, sääkonetyössä oli samantyyppinen ratkaisu ”/weather/get” funktion palauttamana tietona. (Hietala T. 2009)

Tarkemmin asiaa katsottuna alkuperäisestä suunnitelmasta poiketen kävi selväksi, että eihän ohjelmia voi käyttääkään tähän tarkoitukseen, sillä tarkoituksena oli hakea current-tietokannasta viimeisin mitattu arvo, eikä suoraan mitata sitä mittauskortilta. Toinen sääkone työssä käytetty toiminto, säätietojen haku halutulta aikaväliltä, ei myöskään käynyt

suoraan. (Hietala T. 2009) Aikaisemmin tehtyjen ohjelmapalikoiden soveltumattomuuden seurauksena, joudutaan tekemään nettikäyttöohjelmat alusta asti kokonaan uusiksi.

Ensimmäisenä ohjelmoitavana nettikäyttöliittymänä on ohjelma, joka tulostaa verkkoon suunnitelman mukaisesti viimeisimmät arvot current-tietokannasta eli ohjelma, joka vastaa käskyyn ”/web/web”. Ohjelman teko on hyvin yksinkertaista käyttämällä LabVIEW:ä löytyviä tietokantakomponentteja. Tietokannasta saatu data on 2d-array muotoista, joten oli tarpeellista luoda aliohjelma, joka erottelee 2d-arrayna olevan tiedon ja muuntaa sen tekstiksi. Muunnettu teksti on tällöin ulostulona current-tietokannasta ja jokainen arvo on eroteltuna kuvion 4 osoittamalla tavalla.

```
Nopeus=2;
Suunta=131;
Kosteus=0;
Lampotila=0;
Ilmanpaine=0;
Sademaara=0;
Aika=19.10.2009 klo 13:43;
```

Kuvio 4. ”/web/web” nettikäyttöliittymän ulostulo.

Toisen sovelluksen aloitus tapahtui ensimmäisen ohjelman teon jälkeen. Toisella sovelluksella tarkoitetaan tässä ohjelmaa, joka vastaa kuvioissa 5 olevaan kyselyyn antamalla Excel-taulukon tiedostonimen, joka sisältää 1.10.2009 klo 2.30 ja 4.10.2009 klo 5.00 välisen ajan tiedot taulukossa.

```
/web/data/01%10%2009klo02&30/04%10%2009klo05%00
```

Kuvio 5. Kysely nettikäyttöliittymälle.

Aluksi lähdettiin tekemään sovellusta niin, että haettiin halutun ajanjakson tiedot tietokannasta, jonka jälkeen LabVIEW:n report generation toolkittiä hyväksi käyttäen luotiin Excel-tiedosto tietokannasta tulleilla tiedoilla. Luodun Excel tiedoston nimi annettiin vastauksena verkon puolelle jonka asiakassovellus haki palvelimelta. Tähän asti kaikki meni suunnitelmien mukaan, lukuun ottamatta yhtä seikkaa. Tiedostokoko oli aivan liian pieni verrattuna normaaliin Excel-tiedostoon. Alkuarvio mahdollisesta ongelmasta perustui luuloon

sen johtuvan tietokannan tyhjyydestä haetulle aikajaksolle, mutta tarkemman analyysin jälkeen voitiin sanoa, että LabVIEWin report generation kit ei toimi tässä tapauksessa.

Tuosta päästiinkin sitten varsin ison ongelman ratkaisuun. Ison ongelmasta teki pari asiaa, ensimmäinen oli, että verkkopalveluksi muutettu vi ohjelma ei toiminutkaan oikein, kun se oli sijoitettu palvelimeen. Toisena ongelmana oli, että ihan LabVIEWin perusmuunnokset tekstistä numeroksi eivät myöskään toimineet verkkopalveluna. Joten yhtäkkiä ongelma olikin suuri. Ratkaisua ongelmaan haettiin pari viikkoa, tutkimalla eri ratkaisuja ongelmiin.

Tarkemman tutkimisen jälkeen, verkkopalvelu osoittautui niin kevyt rakenteiseksi, että sillä ei pysty luomaan mitään muuttujia eikä varaamaan muistia. Niin yritettiin käyttää muuttujia, jotka varaisivat pääohjelman avulla muistia, mutta joita pystyisi muuttamaan toisesta sovelluksesta. Eri muuttujien kokeilujen jälkeen LabVIEW:ä löytyi jaettu muuttuja, nimellä shared variable. Eri vaihtoehtojen jälkeen testattiin, mitkä muuttujan variaatiot toimisivat verkkopalvelu-vissä. Niitä löytyi vain yksi, minkä arvoa pystyi muuttamaan, ja se oli boolean muuttuja. Verkkopalvelun avulla pystyy kyllä lukemaan kaikkia muita muuttujia, mutta jostain syystä ei kirjoittamaan. Verkkopalvelussa käytettävät jaetut muuttujat piti lisäksi olla verkkoon julkaistuja muuttujia, jotta niitä pystyi käyttämään.

Nettikäyttöliittymän muuttujaongelman ratkaisun löydyttyä, jouduttiin muuttamaan voimakkaasti ohjelman toimintatapaa. Ensisijainen muutos oli lopettaa tietokannasta haku verkkopalvelu-ohjelmassa ja muuttaa sitä kevytrakenteisemmaksi. Siispä verkkopalvelusovellus kirjoitti suoraan tietokantaan tiedot haettavan ajanjakson sekä tiedostonimen, johon kirjoittaa ajanjakson tiedot. Ohjelman saadessa haettavan ajanjakson tiedot asiakassovellukselta, ohjelma kehitti Excel tiedostonimen käyttämällä satunnaislukugeneraattoria väliltä 0-10000, jonka jälkeen ohjelma kirjoitti tietokantaan aloitusajan, lopetusajan sekä tiedostonimen. Tietokantaan kirjoittamisen jälkeen ohjelma vaihtoi boolean muuttujan tilan tosi arvoon, muuttumisen jälkeen pääohjelma hakee tietokannassa olevat arvot ja teki niiden perusteella Excel tiedoston ja vaihtoi suorituksen jälkeen boolean muuttujan arvon epätodeksi. Boolean muuttujan vaihtuminen epätodeksi antoi ohjelmalle tiedon, että tiedosto oli valmis. ’Tiedosto valmis’ -komennon suorittamisen jälkeen sovellus kirjoittaisi tiedostonimen asiakassovellukselle, joka antaa käyttäjälle kehotuksen ladata tiedosto koneellensa.

6.3 Asiakassovellus

Asiakassovelluksen ohjelmointi aloitettiin siinä vaiheessa, kun nettikäyttöliittymä on valmis testattavaksi. Ohjelmoinnin perustana käytettiin tehtyä suunnitelmaa, johon sovellus tulisi perustumaan. Sovelluksen teko Flex ympäristössä on helppoa. Flex ympäristö ei ollut ennestään tuttu, joten edessä olisi uuden ohjelmointikielen opettelu, ennen kuin pystyttiin tekemään asiakassovellusohjelma. Kielen opettelu tapahtuu esimerkiksi lukemalla Flexiä käsittelevä kirja, jonka avulla ohjelman teko pääsi alkuun. (Adobe Flex training from the source)

Ohjelmointikieli kun oli hallinnassa, aloitettiin ohjelmointi, joka oli suhteellisen helppoa. Asiakassovelluksen tekeminen tapahtui käyttämällä Adoben tekemää Flex Builder sovelluskehitintä, koska aina on helpompaa kirjoittaa koodia, kun kehitin ilmoittaa koodin virheistä ohjelmaa käännettäessä. Ensimmäisenä työstettiin ohjelman ominaisuutta hakea halutun ajanjakson tietoja. Ohjelmoinnin aloituksen jälkeen löytyi internetistä valmis Flex komponentin, jonka toimintatapa oli samantyyppinen, joka ohjelmalta vaaditaan, komponentti näkyy kuviossa 6.

Kuvio 6. Internetistä löydetty Flex komponentti (Datetime).

Komponentti löytäminen oli hyvä ratkaisu, mutta sitä ei pystynyt käyttämään sellaisenaan. Miksi ei pystytty käyttämään? Koska ei haluttu, että komponentissa joutuisi valitsemaan yhdysvaltalaiseen tyyliin, oliko kyseessä aamu vai iltapäivä. Niinpä komponentti piti saada sellaiseksi, joka ymmärtäisi 24 tunnin aikamuotoa. Hyvä asia löydössä oli se, että komponentin lähdekoodi oli vapaasti saatavilla, niinpä komponentin koodi haettiin Flex Builderiin ja ruvettiin muokkaamaan komponenttia. Muokkauksen tuloksena komponentin ulkoasu muuttui hyvin erilaiseksi ja enää ei tarvinnut valita, oliko kyseessä aamu tai iltapäivää, vaan komponentti ymmärsi myös 24 tunnin aikamuodon. Muutetun komponentin lähdekoodi löytyy liitteestä 3, joka on myös laitettu vapaasti jakeluun alkuperäisen komponentin alkuperäisen tekijän työtä kunnioittaen. Kuviosta 7 näkyy muunnettu komponentti.

Kuvio 7. Muunnettu komponentti.

Komponentin toimivuuden testaamisen jälkeen asiakassovellus alkoi muodostua. Seuraava osa oli tehdä ohjelma muuttamaan käyttäjän asettaman alku ja loppuajan lähettäminen palvelimelle. Ensin kehitettiin algoritmi muuttamaan komponenteilta saadut tiedot palvelimen ymmärtämään muotoon. Ratkaisu oli käytettävän Flex ympäristön takia helppo toteuttaa, niinpä ratkaisun tekoon ei mennyt kauankaan aikaa. Ratkaisu toteutettiin aliohjelmilla jotka löytyvät liitteestä 1 nimellä alkuaik sekä loppuaik.

Päivämäärien määrittelyjen ja ’hae arvot ajalta’ -painikkeen painamisen jälkeen lähetettiin ajanjakso palvelimelle, joka vastauksena antaa tiedostonimen, jonka perusteella ohjelman piti hakea tiedosto palvelimen juurihakemistosta ja antaa käyttäjälle kehotuksen ladata tiedosto koneellensa. Ohjelman ratkaisu oli helppo toteuttaa, kuten liitteestä 2 näkyvässä lähdekoodissa näkyy.

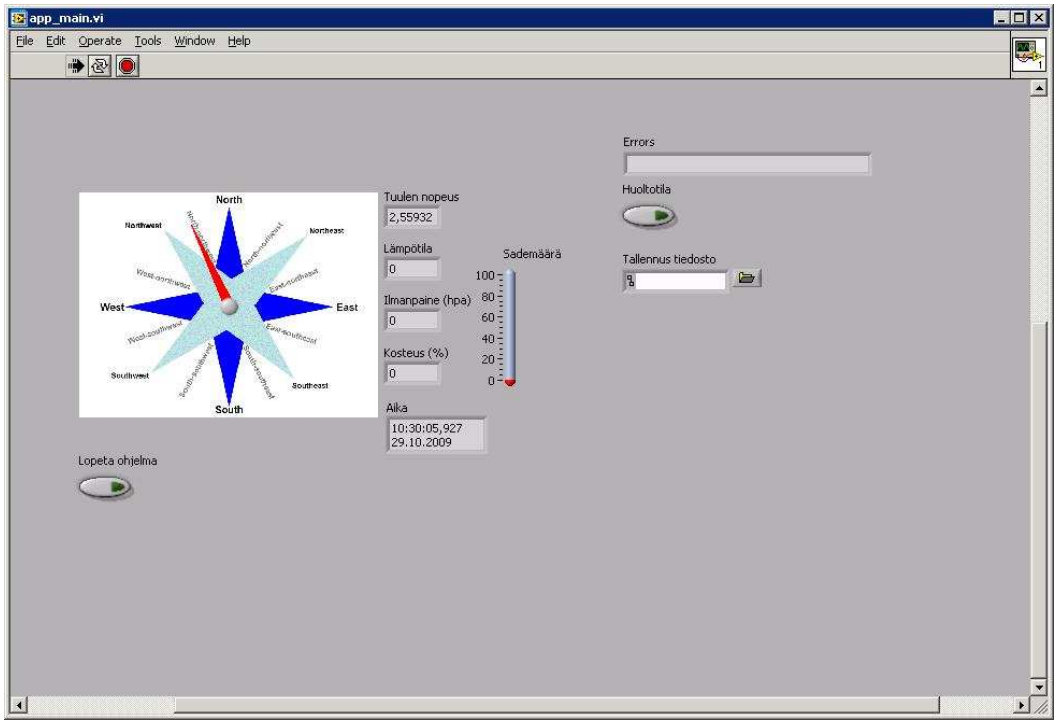
Asiakassovellus oli tässä vaiheessa puoleksi valmis, enää piti lähettää palvelimelle käsky ”/web/web”, jonka vastauksena ovat viimeisimmät mitatut sääarvot. Ohjelman piti enää erotella vastauksesta saadut tiedot ja näyttää ne käyttäjälle. Ratkaisuna ongelmaan kehitettiin aliohjelma, joka etsii halutun merkkiyhdistelmän ja ennen seuraavan ’;’-merkin välissä olleen tekstin, aliohjelmalta saatu vastaus on siis mitattu sääarvo. Aliohjelmalta saatu vastaus näytetään tällöin suoraan käyttäjälle, joka voi halutessaan tarkastella sääsuureita.

Sääarvojen näyttämisen ja tiedostonhakutoimintojen lisäämisen jälkeen oli ohjelma melko lailla valmis, mutta ei aivan, sillä ohjelmaan lisättiin yksi lisätoiminto. Lisätoiminnon tehtävänä oli hakea automaattisesti 20 sekunnin välein uudet arvot käyttäjän nähtäväksi. Toiminnon toteuttaminen Flexillä ei ollut hankalaa, sillä Flexissä on muuttujatoiminto timer eli ajastus. Ajastin-toiminnon avulla tehtävän toteuttaminen ohjelmassa oli helppoa, kuten liitteessä 2 olevasta aliohjelmasta initial näkyy. Kaiken lisäksi initial ohjelma käynnistyy automaattisesti kun käyttäjä aukaisee ohjelman. Toteutus oli helppo lisätä Flexin eli MXML-koodiin <mx:Application kohtaan alavalikko creationComplete ja sen suoritettavaksi ohjelma initial, kuten liitteessä 1 näytetään. Ohjelmointi oli lisätoiminnon lisäämisen jälkeen valmis.

7 Tulokset

7.1 Ohjelmisto

Lopputuloksena syntyi ohjelmisto, jossa on 3 eri sovellusta. Ensimmäiseksi käsitellään pääohjelmaa, joka toimii palvelimella. Pääohjelman tarkoituksena on toimia linkkinä laitteiston, verkkoon liitetyn käyttöliittymän sekä tietokannan välillä. Pääohjelman nimettiin hieman omaperäisellä nimellä server_prog. Ohjelma vastasi hyvin suunniteltuja tavoitteita, miten ohjelman pitäisi toimia.



Kuvio 8. Pääohjelma käynnissä.

Kuviossa 8 näkyy kuvakaappaus tehdystä ohjelmasta, josta selviää että ohjelma on graafisesti hyvin yksinkertainen. Pääasia oli että ohjelmatoiminnot ovat selkeät sekä toimivuus on sujuvaa. Ohjelma päivittää joka sekunti kyllä mittausarvot palvelimen ylläpidon puolella, mutta verkkoon tiedot päivittyvät vain 10 sekunnin välein.

Pääohjelman graafisuus ei ole merkittävässä osassa kuten kuviossa 8 näkyy, sillä käyttäjät eivät pääse näkemään verkon puolelta sitä. Graafisuuden ei ollessa merkittävä, jätettiin ohjelma hyvin perusrakenteiseksi ja karuksi. Ainoastaan ohjelman ylläpidosta vastaava henkilö pääsee käyttämään ohjelmaa, tämä on tehty mitattujen sääarvojen turvallisuutta varten. Server_prog ohjelman tarkempia käyttötietoja löytyy liitteestä 4.

Nettikäyttöliittymä jakaantuu kahteen eri sovellukseen, Tietokannasta haku ohjelmaan sekä viimeisimmän mitatun sääarvojen-hakuohjelmaan. Sääarvojen-hakuohjelma tehtiin hyvin yksinkertaiseksi ja kevytrakenteiseksi. Toiminta oli suoraviivaista, ohjelma haki verkosta tulleella komennolla /web/web tietokannasta viimeisimmät mitatut sääarvot, jonka jälkeen ohjelma purki tietokannasta saadun 2 ulottuvuuksisen datan yksinkertaiseksi tekstiksi. Tekstiksi purun jälkeen ohjelma tulosti lähteväksi tekstiksi esimerkiksi kuviossa 9 olevan tekstin.

```
Nopeus=2;
Suunta=131;
Kosteus=0;
Lampotila=0;
Ilmanpaine=0;
Sademaara=0;
Aika=19.10.2009 klo 13:43;
```

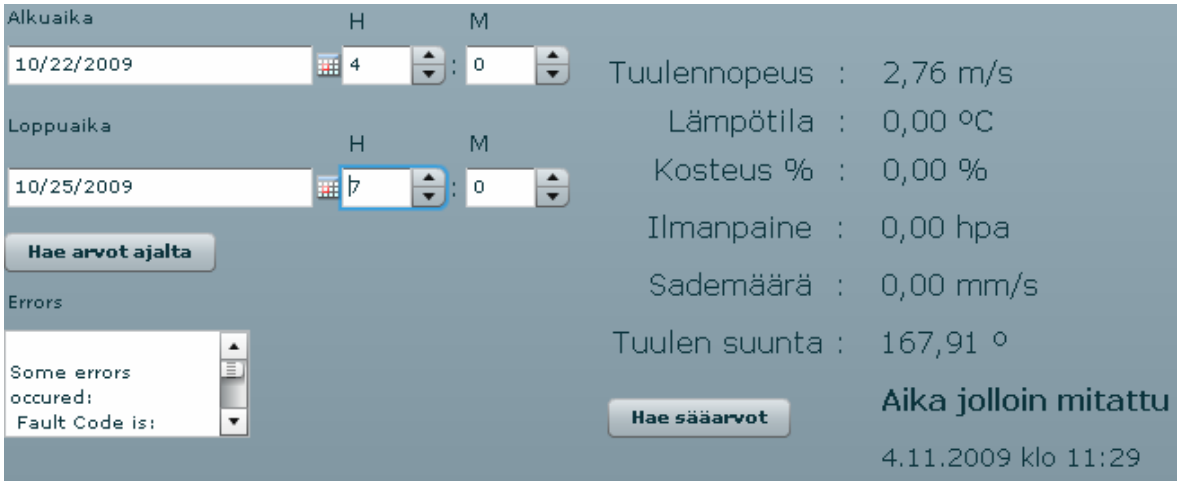
Kuvio 9. Sääarvojen hakuohjelman vastaus annettuun suorituspyyntöön.

Hakuohjelman vastaus on tarkoituksen mukainen, josta asiakassovellus hakee kunkin sääarvon tiedot ja näyttää ne käyttäjälle. Tulosta arvioidessa on mielessä, että vastauksen olisi voinut näyttää suoraan käyttäjälle ilman monimutkaisen etsintä-aliohjelman tekoa asiakassovellukseen. Kuitenkin kyseinen etsintärutiini tehtiin, koska käyttäjäsovelluksen piti olla yksinkertainen ja helppo lukea. Asiakassovelluksen etsintäaliohjelma löytyy liitteestä 2 nimellä find.

Toinen nettikäyttöliittymän sovellus, jota kutsutaan nimellä tietokannasta haku, oli hieman työläs toteuttaa. Lopputuloksena syntyi ohjelma, joka kävi varsin hyvin käytettävään tarkoitukseen, vaikkakin työn teossa vierähti enemmän aikaa kuin alun perin oli arvioitu. Ohjelma vastasi oikeanlaiseen pyyntöön antamalla tiedostonimen, jonka avulla asiakassovellus pystyi hakemaan tiedoston palvelimelta. Ohjelma kun vaati oikeanlaisen pyynnön, niin sen takia tiedoston haku piti toteuttaa asiakassovelluksella. Esimerkiksi haetaan tietokannasta mittausarvot aikaväliltä 20.10.2009 klo 13.00 – 25.10.2009 klo 12.00. Nettikäyttöliittymälle tulevan pyynnön pitää tällöin olla ”/web/data/20%10%2009klo13&00/25%10%2009klo12&00”, kuten viestijonosta käy selväksi käyttäjältä ei voinut odottaa kyseisen lausekkeen kirjoittamista palvelimen osoiteri-

viin. Tämän takia oli tarpeellista kehittää asiakassovellus muuntamaan käyttäjän visuaalisesti antaman tiedon palvelimen käyttöliittymän ymmärtämään muotoon.

Asiakassovelluksen lopputulos oli visuaalisesti kuvion 10 mukainen. Käyttäjä määritti graafisesti alkupäivämäärän ja ajan sekä loppupäivämäärän ajan, miltä väliltä halusi saada mittaustiedot koneellensa tiedostona.



Kuvio 10. Asiakassovelluksen käyttöliittymä.

Ohjelma hakee automaattisesti viimeisimmät mitatut sääarvot käyttäjän tarkasteltavaksi, lisäksi 20 sekunnin välein tapahtui automaattinen sääarvojen uudelleen haku palvelimelta. Asiakassovellus oli käyttäjäystävällinen, koska käyttääkseen ohjelmaa ei tarvinnut osata monimutkaisia symboleja hakeakseen viimeisimmät säätiedot. Riittää kun käytäjä menee asiakassovellukseen, jolloin ohjelma haki itse koko ajan viimeisimmät tiedot.

Mikäli ohjelma kohtasi virheen hakiessaan säätietoja palvelimelta, voidaan virheet havaita asiakassovelluksessa olevasta pienestä Errors valikosta. Kuviosta 11 havaitaan miten ohjelma reagoi, kun ohjelmaa ajetaan eri koneella, kuin mihin se on tarkoitettu.

Kuvio 11. Asiakassovellus virheilmoituksella.

Työn tuloksena oli siis syntynyt 4 ohjelman yhdistelmä, joka toimi mainiosti keskenään. Mutta vastasiko ohjelmisto työn alussa syntynyttä määrittelyä? Suoraan sanottuna ohjelmisto vastaa suurelta osin työn tarkoitusta.

7.2 Ohjelman mittauslaitteisto vaatimukset

Työn tuloksena syntynyttä ohjelmistoa ei voi asentaa mihin tahansa tietokoneeseen, vaan ohjelma voidaan asentaa vain tietokoneeseen, joka täyttää oheiset määritelmät: Tietokoneessa on oltava National Instrumentsin tekemä mittauskortti 6034E väylässä Dev1, mittauskortin ajurit on oltava myös asennettuna oikein, jotta sitä voidaan käyttää ohjelmassa. Tietokone johon ohjelma asennetaan pitää olla myös MySQL-tietokanta v 5.1 tai uudempi asennettuna. MySQL-connector ohjelmisto, joka tarjoaa liitynnän Windows pohjaiseen tietokoneeseen käyttäen ODBC tietokantarajapintaa pitää myös olla asennettuna ennen kuin ohjelma voidaan edes asentaa. Ohjelma voidaan kyllä asentaa koneeseen ilman tietokantoja, mutta tällöin ohjelma saattaa kaatua tai ainakin esittää vakavia virheilmoituksia ja ohjelman suoritus ei toimi oikein. Ohjelma käyttää ODBC rajapintaa muodostaakseen yhteyden tietokantaan. Lisäksi jotta ohjelma voi muodostaa yhteyden tietokantaan, on palvelimen ylläpitäjän annettava MySQL-tietokannan pääkäyttäjän salasana ohjelman tietoon, kuten liitteessä 4 on kerrottu. Käyttöjärjestelmänä pitää olla Windows XP tai uudempi. Tietokoneessa on lisäksi oltava verkkokortti asennettuna.

Vaatimusten täytyttyä voidaan ohjelma asentaa koneeseen. Tietokoneen on oltava suhteellisen nopea, jotta ohjelma toimisi oikein. Mikäli tietokone on hidas, niin se ei käsittele tar-

peeksi nopeasti internetin puolelta tulevia käskyjä tehdä tiedosto palvelinhakemistoon tietokannasta. Netinpuolelta tulevat käskyt eivät tällöin kykene suoriutumaan tarpeeksi nopeasti, aiheuttaen virheilmoituksen käyttäjälle että tiedostoa ei löydy.

8 Johtopäätökset ja pohdinta

Lopputuloksena syntyi sääaseman ohjelmisto, jonka toteutustapa oli hieman erikoinen. Yleensä sääasema sovelluksia tehdään C/C++ -kielillä, mutta tämä työ tehtiin LabVIEW-ympäristöön perustuen. Poikkeuksena LabVIEW-ohjelmointiin on asiakassovellus. Asiakassovellusta ei voinut toteuttaa LabVIEW:ä, siispä se täytyi tehdä eri tavalla. Edessä oli ennestään tuntemattoman uuden MXML-ohjelmointikielen opettelu sekä ohjelmointiympäristön opettelu. Ennen asiakassovelluksen tekoa meni muutama päivä tutustuessa sekä omaksuessa uuden ohjelmointiympäristön ominaisuudet.

Tavoitteena oli tutkia mahdollisuutta lisätä määriteltyinä olleet anturit jäljestäpäin järjestelmään. Vastauksena ongelmaan kehitettiin ratkaisu, jonka avulla määritetään tulevat mittausanturit järjestelmään. Tarkemmin toteutuksesta on kerrottu liitteessä 4, lyhyesti sanottuna käyttäjä määrittelee ohjelmalle minimi ja maksimiarvon, joiden avulla järjestelmä laskee kulloinkin määritellyn suureen ja näyttää arvon sekä asiakassovelluksessa että pääohjelmassa.

Kokonaisuutena sanottuna ohjelmassa olisi ehkä vielä kehittämistä. Sademäärän anturin sijoittaminen ohjelmaan tulisi luultavasti olemaan vaikea, käytetyn suureen mm/s myötä ja anturin olisi annettava tieto jännitteenä. Muutoin ohjelma toimii arvion mukaan hyvin vakaasti eli häiriöitä ei pitäisi syntyä.

Uuden ohjelmointikielen opettelu oli helppoa, kun luki ohjelmointikielen kirjaa samalla kun opettelin ohjelmointia. Lisäksi Internetistä löytyi paljon apuja työn tekemiseen, joskaan ei tarpeeksi, työssä jouduttiin monesti kokeilemaan eri ratkaisuja ilman että tiedettiin toimisivatko ne.

Sääaseman ohjelmiston työstäminen oli loppujen lopuksi aika haastavaa, kun eteen tuli odottamattomiakin ongelmia. Lisäksi käytettävissä olleen laitteiston vuoksi mikäli käyttäjiä oli useita hakemassa tiedostoa ohjelmiston avulla, niin ohjelma saattoi antaa melko useasti virheilmoituksia. Ohjelmisto toimivuutta parantaisin heti ensimmäisenä nopeammaksi vaihtamalla palvelimen laitteiston huomattavan paljon nopeammaksi.

Jatkossa ohjelmistoa voidaan parantaa, mikäli jostain ohjelmakomponentista löydetään jokin virhe. Asiakassovellusta kannattaisi parantaa, tekemällä visuaalisesti näyttävämpiä komponentteja esim. lämpötila ja tuulen suunnan suureet, voisi näyttää paremmin graafisesti.

Tulevaisuudessa sääasemaan voi liittää lämpötila,- ilmanpaineen,- kosteuden,- ja sademäränanturit, muuttamatta ohjelman koodausta. Palvelimen voisi tuoda julkisesti näkyville internettiin, jonka välityksellä säätiedot näkisi jokaisesta internettiin kytketystä koneesta.

Lähteet

Painetut

Julie C. Meloni, 2003, MySQL Trainer Kit, IT Press

Olli Aumala, 2003, Mittaustekniikan perusteet, Suomi, Otatieto

Matthew Boles, Michael Labriola, James Talbot, Jeff Tapper. 2008, Adobe Flex 3 Training from the source, Yhdysvallat, Adobe Press

Painamattomat

Hietala, Timo. 2009, LabVIEWillä toteutettu sääasema

Sähköiset

Adobe Systems Incorporated: Adobe Flex Framework Technologies, WWW-dokumentti, Luettu 29.10.2009, Saatavissa: http://labs.adobe.com/technologies/flex/

An overview of MXML: The Flex markup language: WWW-dokumentti, Tekijä Christophe Coenraets, Luettu 3.11.2009, Saatavissa: http://www.adobe.com/devnet/flex/articles/paradigm.html

Datetime: WWW-dokumentti, Luettu 31.10.2009, Tekijä: Joel Hooks, Saatavissa: http://joelhooks.com/2008/10/11/flex-date-and-time-datetime-picker-control/

Gray Code: WWW-dokumentti, Tekijä: Weisstein, Eric W, WWW-dokumentti, Luettu 6.11.2009, Saatavissa: http://mathworld.wolfram.com/GrayCode.html

National Instruments: NI 6034E/6035E/6036E Family Specifications, WWW-dokumentti, Luettu 28.10.2009, Saatavissa: http://www.ni.com/pdf/manuals/370721c.pdf

National Instruments: What is Labview?, WWW-dokumentti, Luettu 29.10.2009, Saatavissa: http://www.ni.com/labview/whatis/

National Instruments: Web Services in LabVIEW, WWW-dokumentti, Luettu 30.10.2009, Saatavissa: http://zone.ni.com/devzone/cda/tut/p/id/7350

What is VI, WWW-dokumentti, Luettu 6.11.2009, Tekijä: University of Sydney, Saatavissa: http://www.eelab.usyd.edu.au/labview/vi.html

WikipediaB: WWW-dokumentti, Luettu 30.10.2009, Saatavissa: http://en.wikipedia.org/wiki/Adobe_Flex

WikipediaC: WWW-dokumentti, Luettu 30.10.2009, Saatavissa: http://en.wikipedia.org/wiki/MXML

WikipediaD: WWW-dokumentit, Luettu 30.10.2009, Saatavissa:
http://en.wikipedia.org/wiki/ECMAScript sekä http://en.wikipedia.org/wiki/ActionScript

WikipediaE: WWW-dokumentti, Luettu 1.11.2009, Saatavissa:
http://en.wikipedia.org/wiki/MySQL

WikipediaF: WWW-dokumentti, Luettu 1.11.2009, Saatavissa:
http://en.wikipedia.org/wiki/Anemometer

WikipediaG: WWW-dokumentti, Luettu 1.11.2009, Saatavissa:
http://en.wikipedia.org/wiki/Wind_vane

WikipediaH: WWW-dokumentti, Luettu 3.11.2009, Saatavissa:
http://en.wikipedia.org/wiki/Hygrometer

WikipediaI: WWW-dokumentti, Luettu 1.11.2009, Saatavissa:
http://en.wikipedia.org/wiki/Resistance_thermometer

WikipediaJ: WWW-dokumentti, Luettu 1.11.2009, Saatavissa:
http://en.wikipedia.org/wiki/Barometer

WikipediaK: WWW-dokumentti, Luettu 1.11.2009, Saatavissa
http://en.wikipedia.org/wiki/Rain_gauge

Liitteet

Liite 1. Asiakassovelluksen lähdekoodi.

Liite 2. Pääohjelman graafinen koodi.

Liite 3. Nettikäyttöliittymän graafinen koodi.

Liite 4. Sääaseman käyttöohje.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" lay-
out="absolute" xmlns:ns1="*" creationComplete="initial()" viewSour-
ceURL="srcview/index.html">
 <mx:HTTPService id="Labview" url="{values}"
  requestTimeout="15"
  result="resultHandler(event)"
  fault="faultHandler(event)"
 />
 <mx:HTTPService id="Lab_xml" url="{value2}"
  result="resulthand(event)"
  fault="faulthand(event)"
  requestTimeout="15".
 />
 <mx:Script>
  <![CDATA[
   import mx.events.FlexEvent;
   import mx.controls.Text;
   import mx.rpc.events.FaultEvent;
   import mx.rpc.events.ResultEvent;
   import mx.collections.ArrayCollection;
   import mx.controls.LinkButton;
   import mx.controls.FormItemLabel;

   [Bindable]public var values:String =
"/web/data/01%01%2008klo13&00/01%02%2008klo13&00";
   [Bindable]public var value2:String = "/web/web", link:String="";
   [Bindable]public var omaData:String="";    ...
   [Bindable]public var alkuaika:String="01%01%2008klo13&00";
   [Bindable]public var loppuaika:String="01%02%2008klo13&00";
   [Bindable]public var tiedot:String="";
   public var apu:String="", apu2:String="", apu3:String="",
apu4:String="";
   private var avu:int=0, ajoitus:Timer,delayer:Timer;
   //public var tes-
tii:String="Nopeus=2;Suunta=131;Kosteus=0;Lampotila=0;Ilmanpaine=0;Sadema
ara=0;Aika=19.10.2009 klo 13:43;";

   public function setValues():void{

    values="/web/data"+"/"+alkuaika+"/"+loppuaika;
    Labview.send();
   }
   public function haearvot():void{
    Lab_xml.send();
   }
   private function initial():void{
    ajoitus=new Timer(20000);
    ajoitus.addEventListener(TimerEvent.TIMER,aika_aika);
    ajoitus.start();
    haearvot();
   }
   private function delayt():void{
    delayer=new Timer(3000);
    delayer.addEventListener(TimerEvent.TIMER,go);
    seekh.enabled=false;
    delayer.start();.
   }
   private function go(evf:TimerEvent):void{
    navigateToURL(new URLRequest(link),'_blank');
    delayer.stop();
```

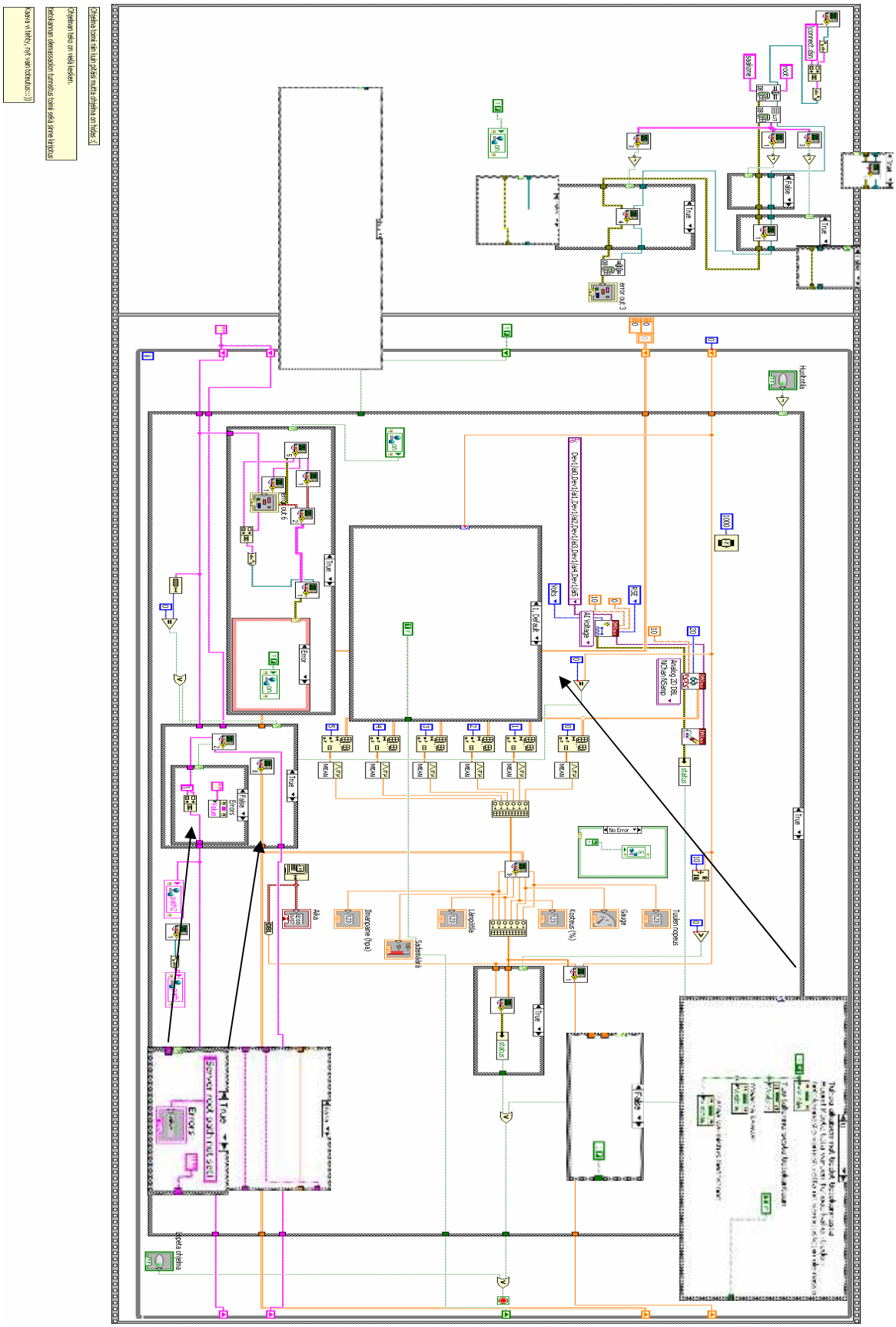
```
 seekh.enabled=true;
}
private function aika_aika(evr:TimerEvent):void{
 haearvot();
}
private function resulthand(evet:ResultEvent):void{
 tiedot=evet.result.toString();
 //var ku:String="";
 nop.text=find(tiedot,"Nopeus=")+" m/s";
 suu.text=find(tiedot,"Suunta=")+" º";
 kos.text=find(tiedot,"Kosteus=")+" %";
 lam.text=find(tiedot,"Lampotila=")+" ºC";
 ilma.text=find(tiedot,"Ilmanpaine=")+" hpa";
 sad.text=find(tiedot,"Sademaara=")+" mm/s";
 aika.text=find(tiedot,"Aika=");
}
/*public function test(*tied:String):void{
 var oo:String="";
 nop.text=find(testii,"Nopeus=")+" m/s";
 suu.text=find(testii,"Suunta=")+" º";
 kos.text=find(testii,"Kosteus=")+" %";
 lam.text=find(testii,"Lampotila=")+" ºC";
 ilma.text=find(testii,"Ilmanpaine=")+" hpa";
 sad.text=find(testii,"Sademaara=")+" mm/s";
 aika.text=find(testii,"Aika=");
}*/
public function find(src:String, finder:String):String{
 var temp:String="",testk:String="",mof:String="";
 var i:int=0,tmp:int=0,add:int=0;
 var hhhh:Boolean=false,kl:Boolean=false;
 temp="",testk="",i=0,tmp=0,add=0,hhhh=false,kl=false;
 mof=src;
 i=src.length;
 tmp=finder.length;
 for(var o:int=0;o<(i-tmp);o++){
  testk="";
  for(var j:int=0;j<tmp;j++){
   testk=testk+mof.charAt(j+o);
  }
  kl=checker(testk,finder,tmp);
  if(kl==true)
  {
   do{
    temp=temp+mof.charAt(add+o+tmp);
    add++;
   }while(mof.charAt(add+o+tmp)!=';');
   hhhh=true;
   return temp;
  }
 }
 return "erroe";

}
private function checker(str1:String,str2:String,len:int):Boolean{
var th:int=0;
 for(var u:int=0;u<len;u++){
  if(str1.charAt(u)==str2.charAt(u)){
   th++;
  }
 }
 if(th==len){
```

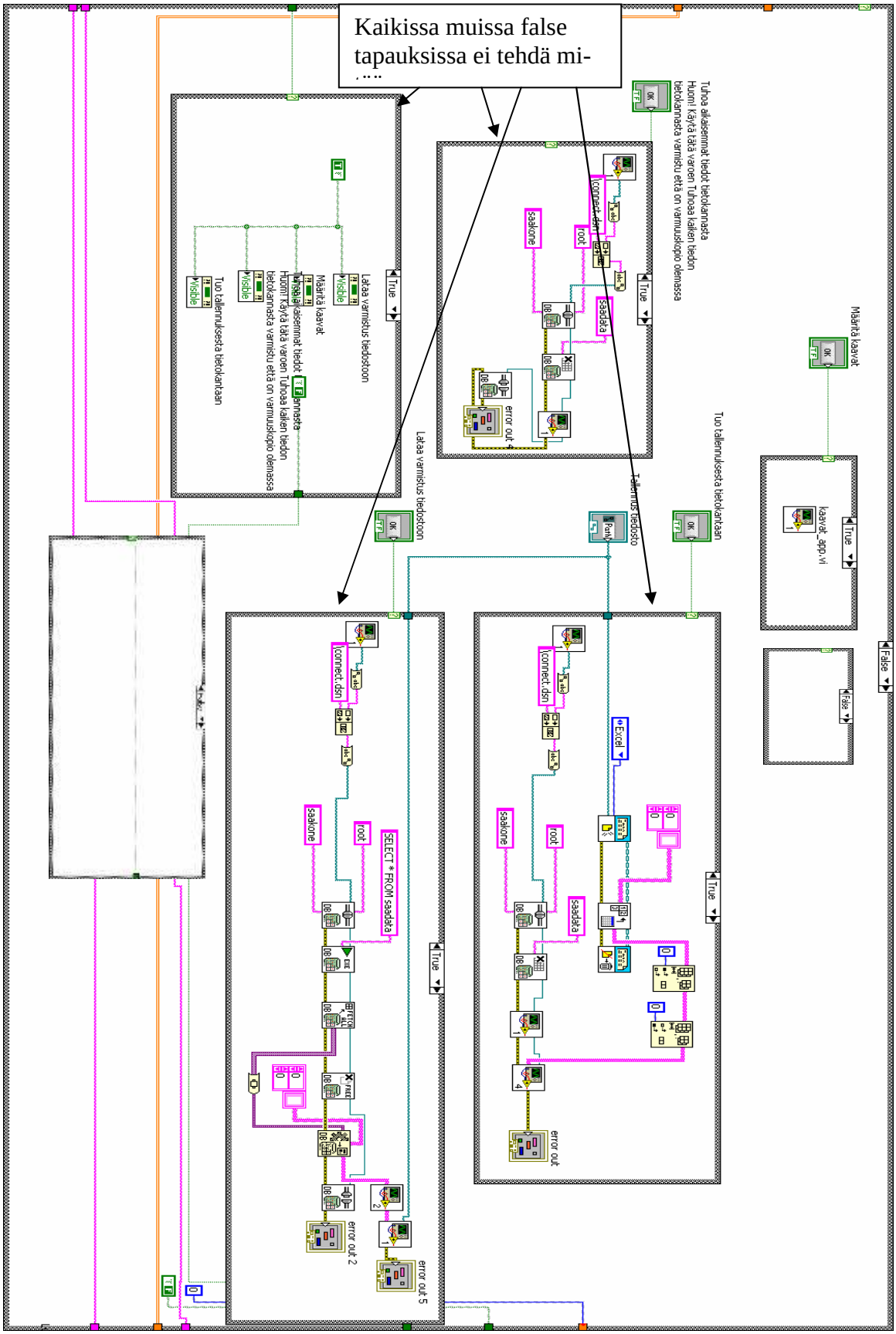
```
  return true;
 }
 else{
  return false;
 }
}
private function faulthand(evt:FaultEvent):void{
 var errors:String = "\nSome errors occured:";
 errors+="\n Fault Code is: \n"+evt.fault.faultCode;
 errors+="\n Fault Detail is: \n" +evt.fault.faultDetail;
 errors+="\n Fault String is: \n" +evt.fault.faultString;
 list.text=errors;
}
public function hae():void{
 alkuaik();
 loppuaik();
 setValues();
}
private function loppuaik():void{
 avu=lopetusaika.selectedDate.date.valueOf();
 if(avu < 10)
 {
  apu2="0"+avu.toString();
 }
 else
 {
  apu2=avu.toString();
 }
 avu=lopetusaika.date.displayedMonth.valueOf()+1;
 if(avu < 10)
 {
 apu="0"+avu.toString();.
 }
 else
 {
  apu=avu.toString();
 }
 loppuaika=apu2+"_"+apu+"_"+lopetusaika.date.displayedYear.toString();
 if (lopetusaika.tunti.value < 10)
 {
  apu3="0"+lopetusaika.tunti.value.toString();
 }
 else
 {
  apu3=lopetusaika.tunti.value.toString();
 }
 if (lopetusaika.minuutti.value < 10)
 {
  apu4="0"+lopetusaika.minuutti.value.toString();
 }
 else
 {
  apu4=lopetusaika.minuutti.value.toString();
 }
 loppuaika=loppuaika+"klo"+apu3+"&"+apu4;
}
private function alkuaik():void{
    avu=aloitusaika.selectedDate.date.valueOf();
 if(avu < 10)
 {
  apu2="0"+avu.toString();.
```

```
        }
        else
        {
         apu2=avu.toString();
        }
        avu=aloitusaika.date.displayedMonth.valueOf()+1;
        if(avu < 10)
        {
         apu="0"+avu.toString();.
        }
        else
        {
         apu=avu.toString();
        }
        alkuaika=apu2+"_"+apu+"_"+aloitusaika.date.displayedYear.toString();
        if (aloitusaika.tunti.value < 10)
        {
         apu3="0"+aloitusaika.tunti.value.toString();
        }
        else
        {
         apu3=aloitusaika.tunti.value.toString();
        }
        if (aloitusaika.minuutti.value < 10)
        {
         apu4="0"+aloitusaika.minuutti.value.toString();
        }
        else
        {
         apu4=aloitusaika.minuutti.value.toString();
        }
        alkuaika=alkuaika+"klo"+apu3+"&"+apu4;
       }

       private function resultHandler(evt:ResultEvent):void{
        omaData=evt.result.toString();
        var b:Boolean=false;
        var c:String="";
        for (var i:int=0;i<omaData.length;i++)
        {
         if(omaData.charAt(i)=='=')
         {
          b=true;
         }
         else if(b==true)
         {
          c=c+omaData.charAt(i);
         }
        }
        link=c;

if(link.charAt(0)=='e'&&link.charAt(1)=='r'&&link.charAt(2)=='r'&&link.ch
arAt(3)=='o'&&link.charAt(4)=='r'){
         list.text="Error occurred in retrieving data:\n"+link;
        }
        else{
         delayt();.
        }
       }
       private function faultHandler(evt:FaultEvent):void{
        var errors:String = "\nSome errors occured:";
```

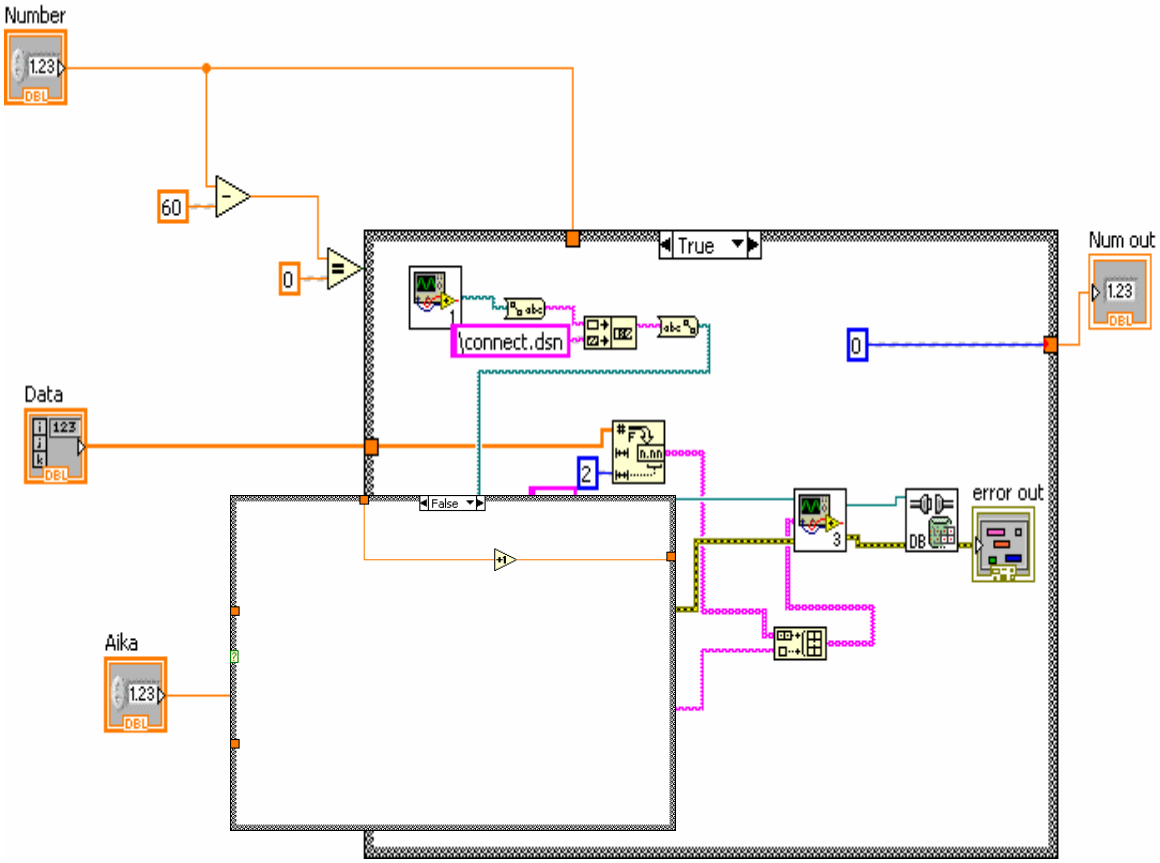
```
    errors+="\n Fault Code is: \n"+evt.fault.faultCode;
    errors+="\n Fault Detail is: \n" +evt.fault.faultDetail;
    errors+="\n Fault String is: \n" +evt.fault.faultString;
    list.text=errors;
   }
  ]]>
 </mx:Script>
 <mx:Label x="10" y="11" text="Alkuaika" width="139"/>
 <mx:Label x="10" y="71" text="Loppuaika" width="137"/>
 <mx:TextArea x="10" y="193" width="137" id="list" height="60"/>
 <mx:Text x="10" y="169" text="Errors"/>
 <mx:Button x="10" y="139" id="seekh" label="Hae arvot ajalta"
click="hae()"/>
 <ns1:datetime x="10" y="29" id="aloitusaika" width="327" />
 <ns1:datetime x="10" y="97" id="lopetusaika" width="327"/>
 <mx:Label x="345" y="39" text="Tuulennopeus  : " fontSize="16"
id="tuulennop"/>
 <mx:Label x="378" y="66" text="Lämpötila  : " fontSize="16"
id="lampotila"/>
 <mx:Label x="366" y="125" text="Ilmanpaine  : " fontSize="16"
id="ilmanpaine"/>
 <mx:Label x="370" y="93" text="Kosteus %  : " fontSize="16"
id="kosteus"/>
 <mx:Label x="367" y="157" text="Sademäärä  : " fontSize="16"
id="sademaara"/>
 <mx:Button x="346" y="231" label="Hae sääarvot" click="haearvot()"/>
 <mx:Label x="347" y="189" text="Tuulen suunta :" fontSize="16"/>
 <mx:Label x="497" y="39" text="0" fontSize="16" id="nop"/>
 <mx:Label x="497" y="66" text="0" fontSize="16" id="lam"/>
 <mx:Label x="497" y="93" text="0" fontSize="16" id="kos"/>
 <mx:Label x="496" y="125" text="0" fontSize="16" id="ilma"/>
 <mx:Label x="496" y="157" text="0" fontSize="16" id="sad"/>
 <mx:Label x="496" y="189" text="0" fontSize="16" id="suu"/>
 <mx:Label x="496" y="253" id="aika" fontSize="14" width="217"/>
 <mx:Label x="496" y="221" text="Aika jolloin mitattu" fontSize="16"
fontWeight="bold" width="175"/>
 <mx:Label x="200" y="80" text="H" fontSize="12"/>
 <mx:Label x="267" y="79" text="M" fontSize="13"/>
 <mx:Label x="200" y="11" text="H" fontSize="13"/>
 <mx:Label x="267" y="11" text="M" fontSize="13"/>
</mx:Application>
```

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Tämä koodissa oleva komponentti otettu sivulta 1.10.2009 klo 12.00
http://joelhooks.com/2008/10/11/flex-date-and-time-datetime-picker-
control/
koodia on muokattu jonkin verran että toiminnallisuus on saatu tähän tar-
koitukseen
sopivaksi.-->
<mx:HBox xmlns:mx="http://www.adobe.com/2006/mxml" paddingBottom="2" pad-
dingLeft="2" paddingRight="2" paddingTop="2" horizontalGap="0" verticalA-
lign="middle" height="34" width="518">
 <mx:DateField x="10" y="10" id="date" change="handleChange()"
width="186"/>
 <mx:NumericStepper x="108" y="10" id="tunti" change="handleChange()"
minimum="0" maximum="23" stepSize="1" enabled="true"/>
 <mx:Text x="174" y="10" text=":" height="22" fontSize="14"/>
 <mx:NumericStepper x="193" y="10" id="minuutti" change="handleChange()"
minimum="0" maximum="59" stepSize="1" enabled="true"/>
<mx:Metadata>
 [Event(name="change", type="flash.events.Event")]
</mx:Metadata>
<mx:Script>
 <![CDATA[
  import mx.controls.DateField;
  import mx.controls.NumericStepper;
  [Bindable] private var _selectedDate:Date = new Date();
  public function get selectedDate():Date
  {
   this.validateNow();
   return this._selectedDate;
  }
  [Bindable] public function set selectedDate(value:Date):void
  {
   this._selectedDate = value
   this.date.selectedDate = this._selectedDate
   this.tunti.value = value.getHours()
   this.minuutti.value = value.getMinutes()
   this.validateNow();
  }
  override public function validateProperties():void
  {
   super.validateProperties();
  }
  public function handleChange():void
  {
   var selDate:Date = this.date.selectedDate;
   var date:Date = new Date(
   selDate.getFullYear(),
   selDate.getMonth(),
   selDate.getDate(),
   tunti.value,
   minuutti.value)
   this.selectedDate = date;
   this.invalidateProperties();
   this.validateNow();
   this.dispatchEvent(new Event("change"));
  }
 ]]>
</mx:Script>
</mx:HBox>
```

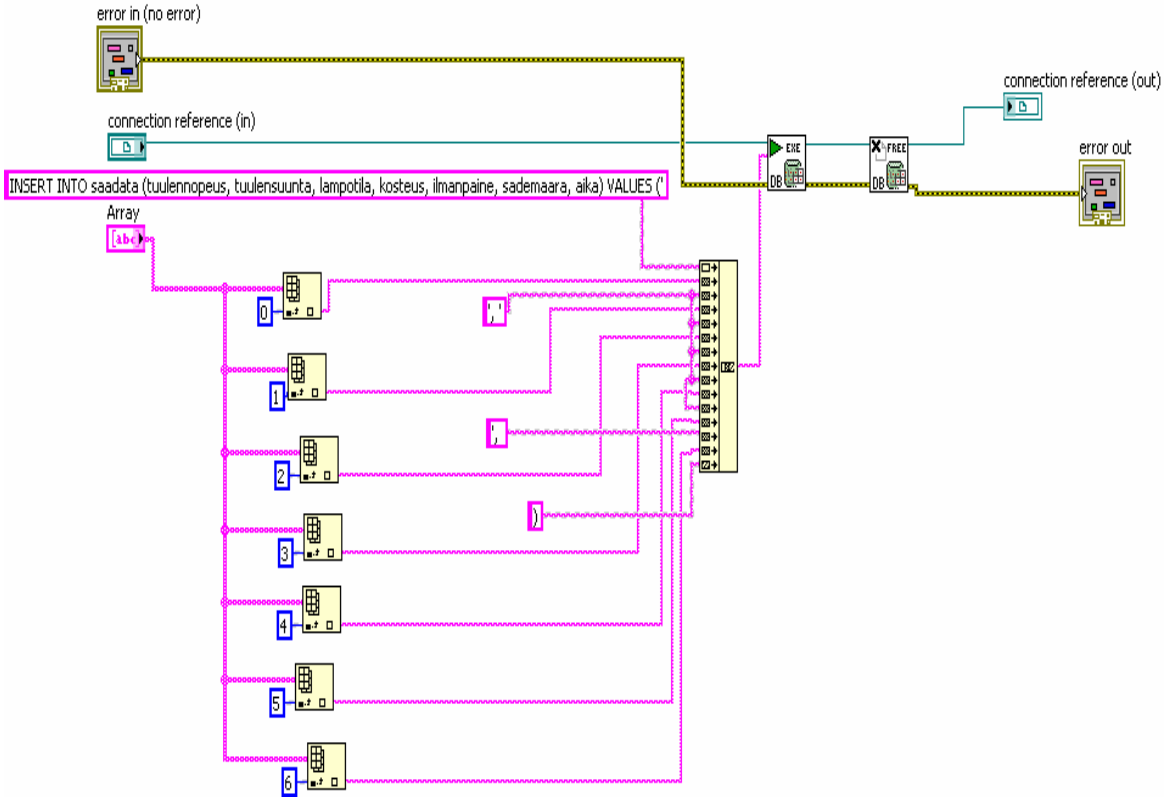



Kaikissa muissa false tapauksissa ei tehdä mi-
Tuhoa alkaisemmat tiedot tietokannasta
Huom! Käytä tätä varoen Tuhoaa kaiken tiedon tietokannasta varmistu että on varmuuskopio olemassa
Määritä kaavat
Lataa varmistus tiedostoon
Tuo tallennuksesta tietokantaan
connect.dsn
saakone
root
saadata
SELECT * FROM saadata
Excel
error out
error out 2
error out 5
kaavat_app.vi
True
False

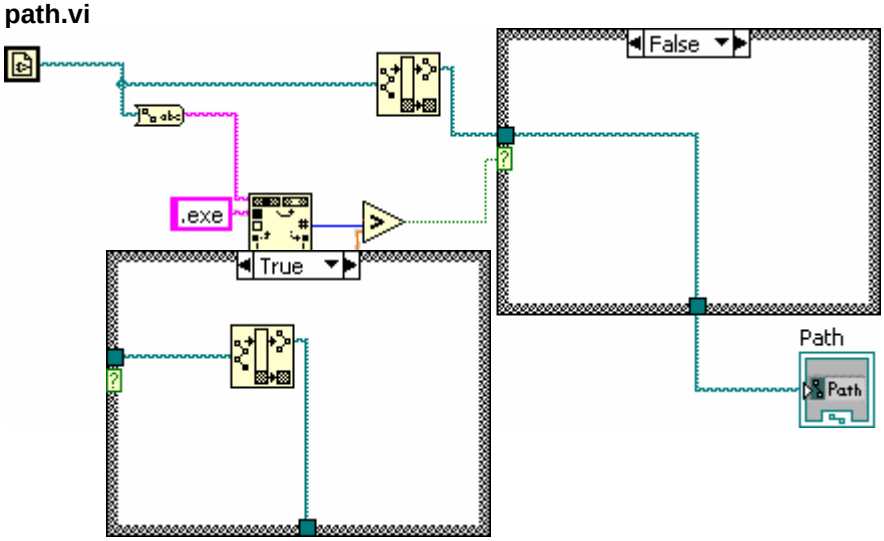
app_main_apu.vi



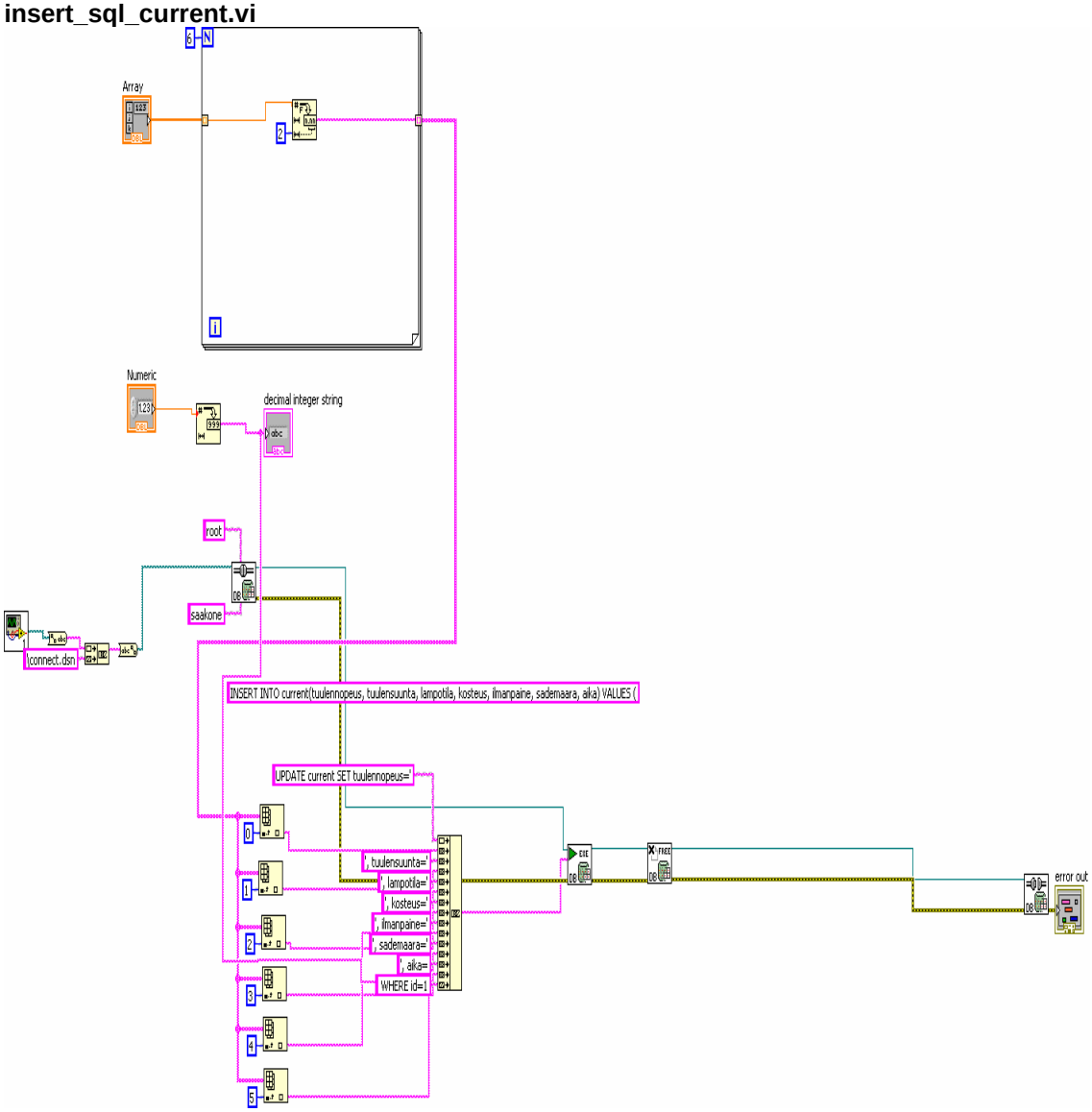
insert_sql.vi



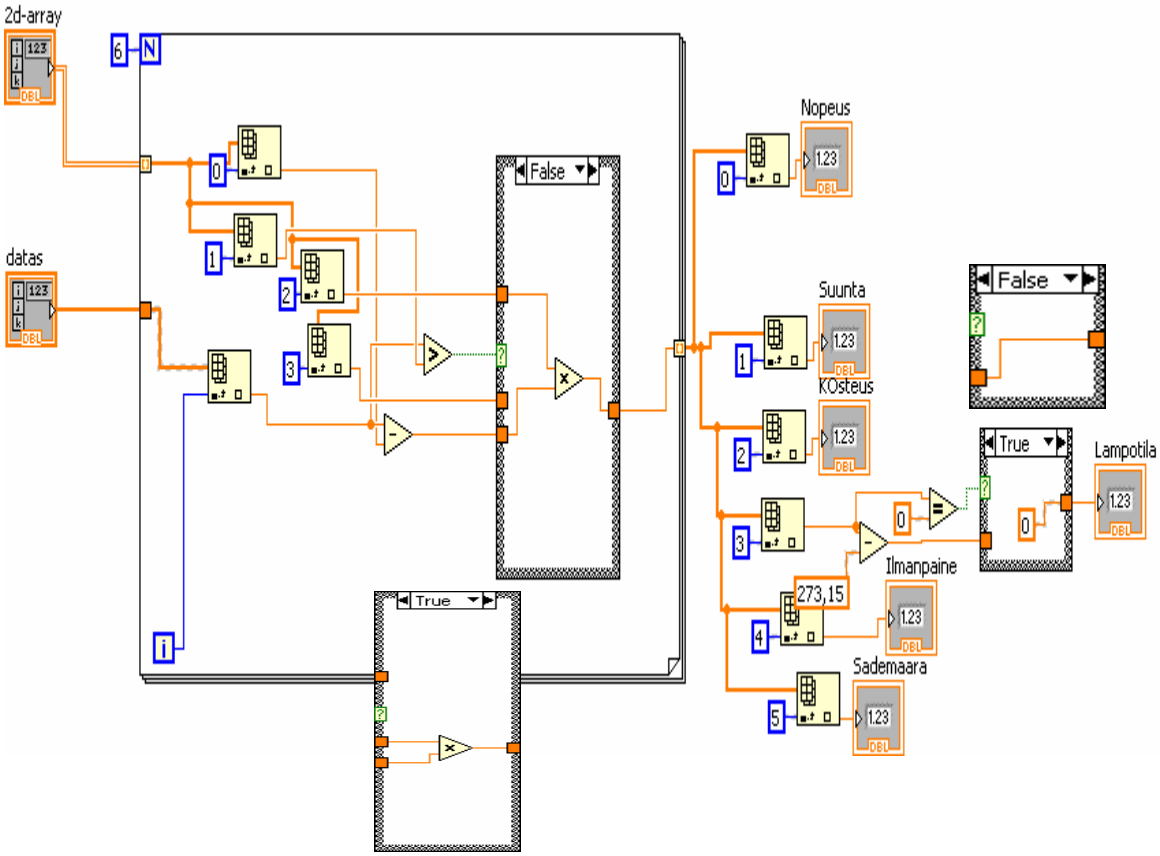
path.vi



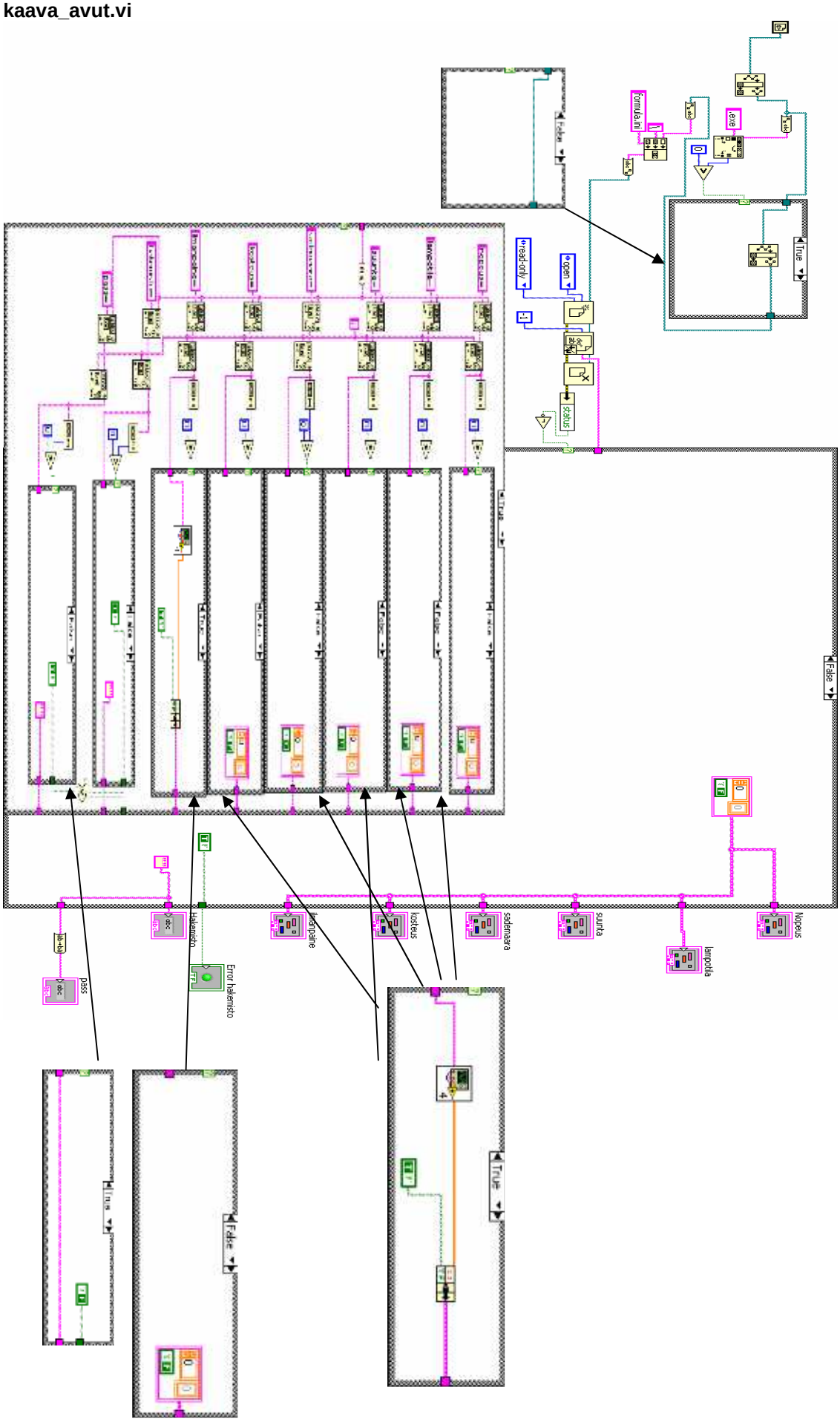
insert_sql_current.vi



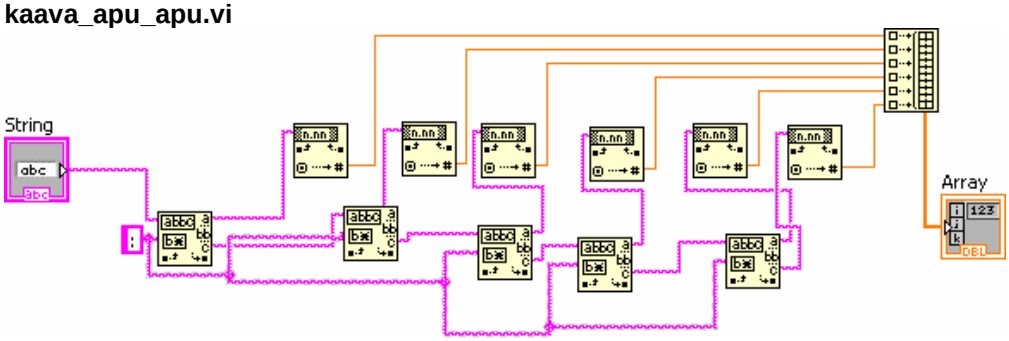
kaava_main_calc.vi



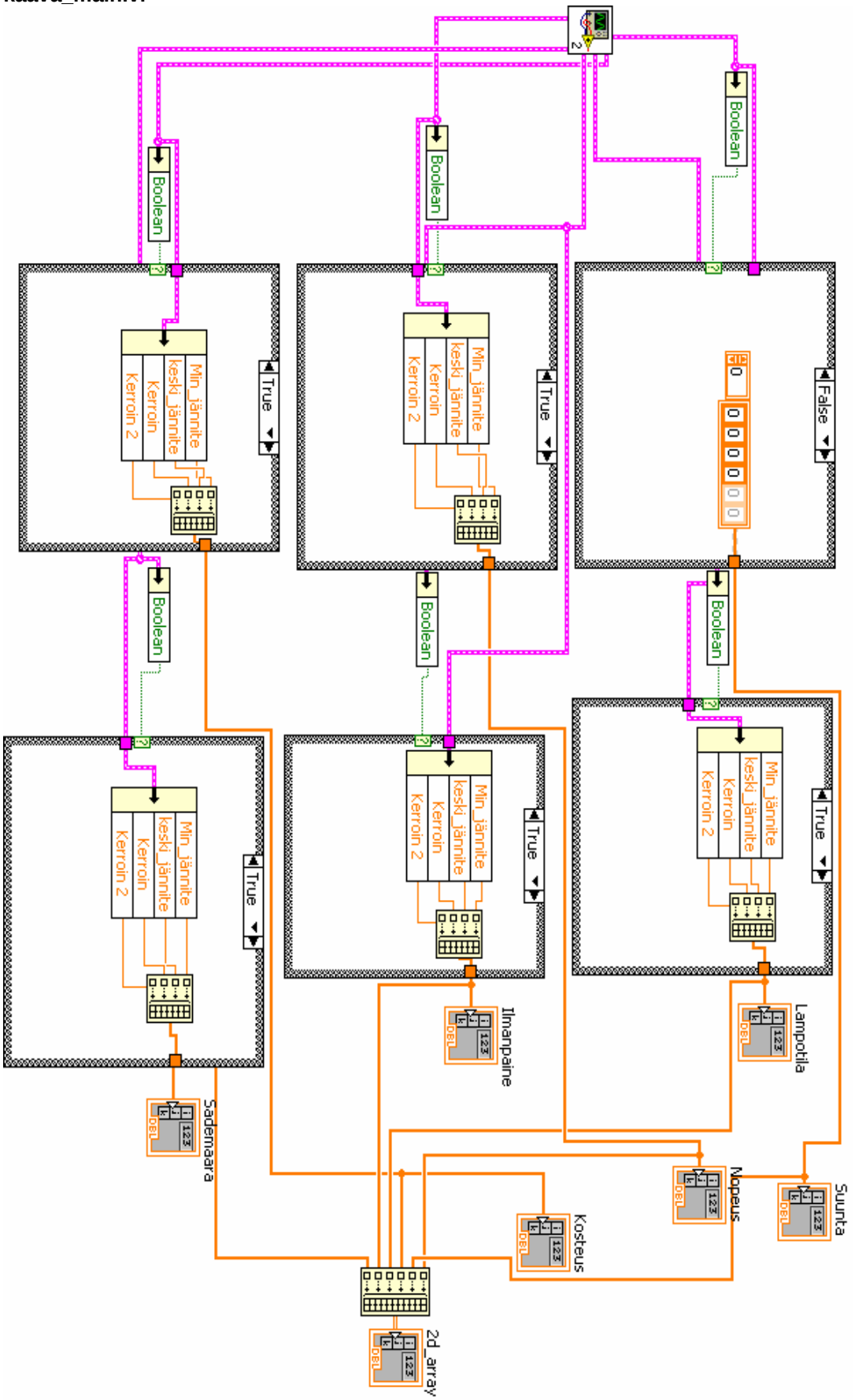
kaava_avut.vi



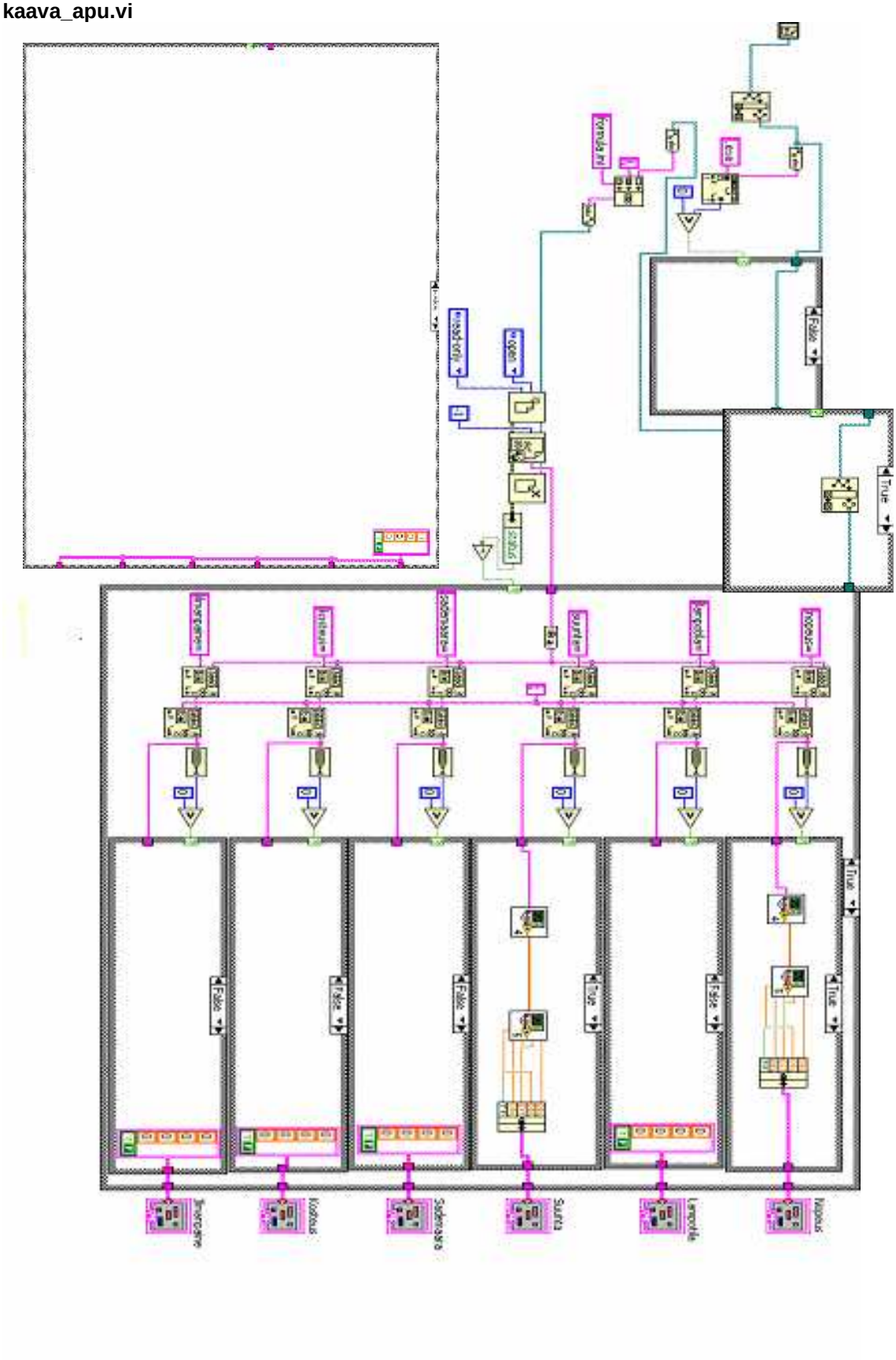
kaava_apu_apu.vi



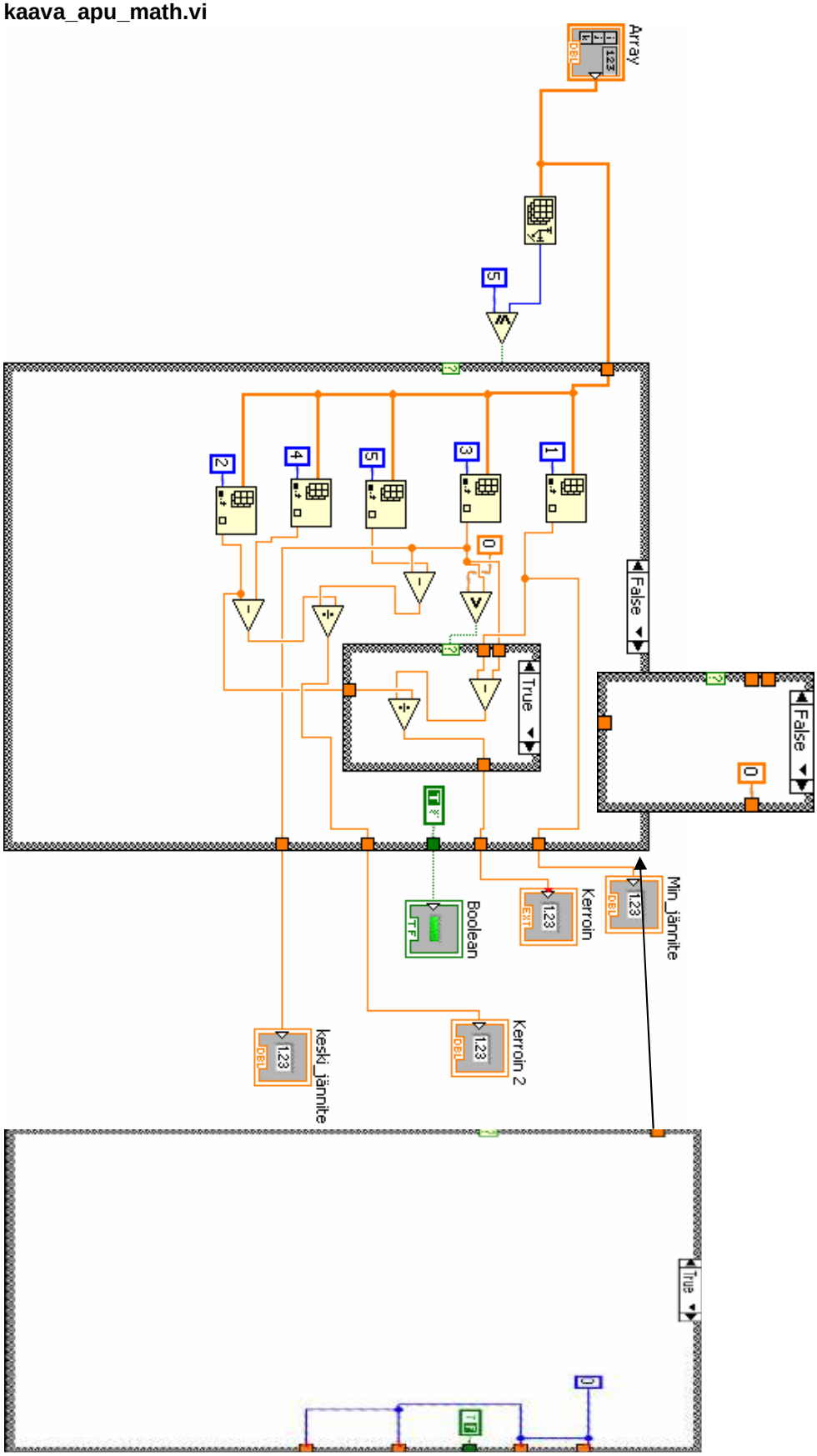
kaava_main.vi



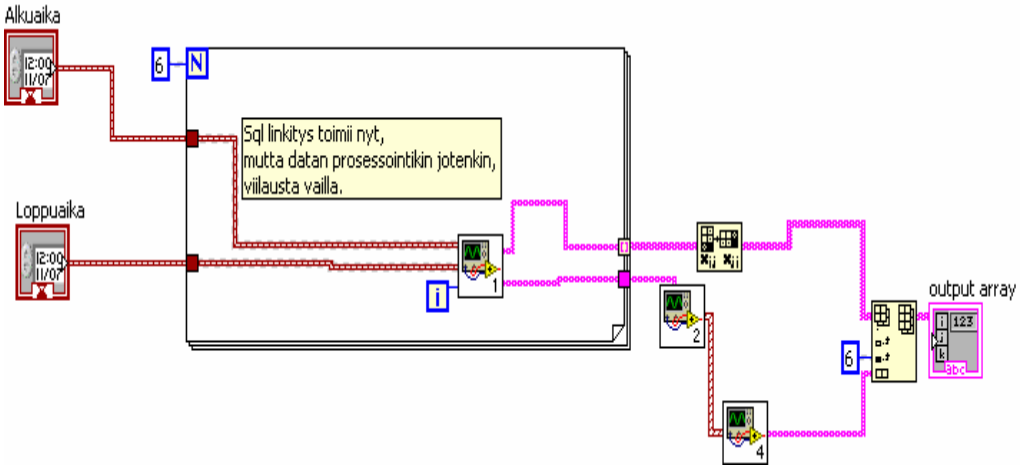
kaava_apu.vi



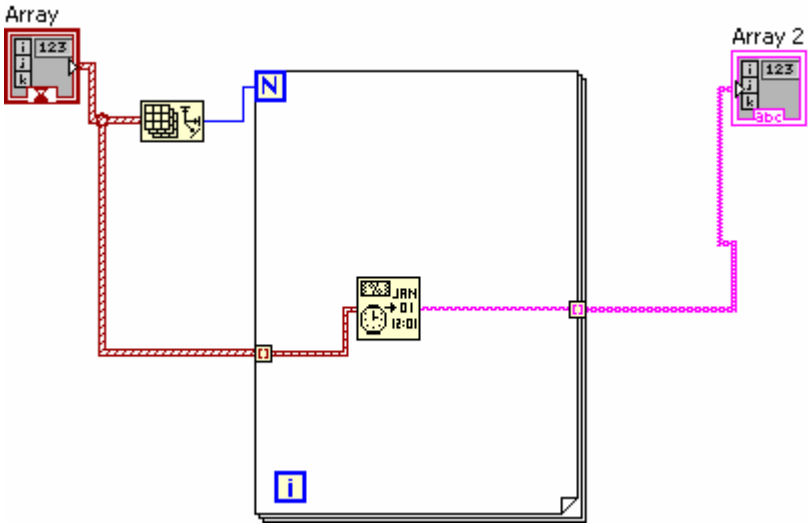
kaava_apu_math.vi



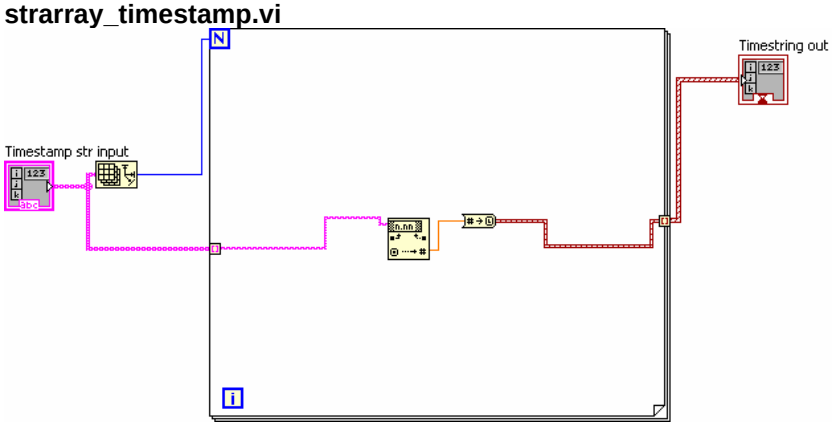
hae_arvot.vi



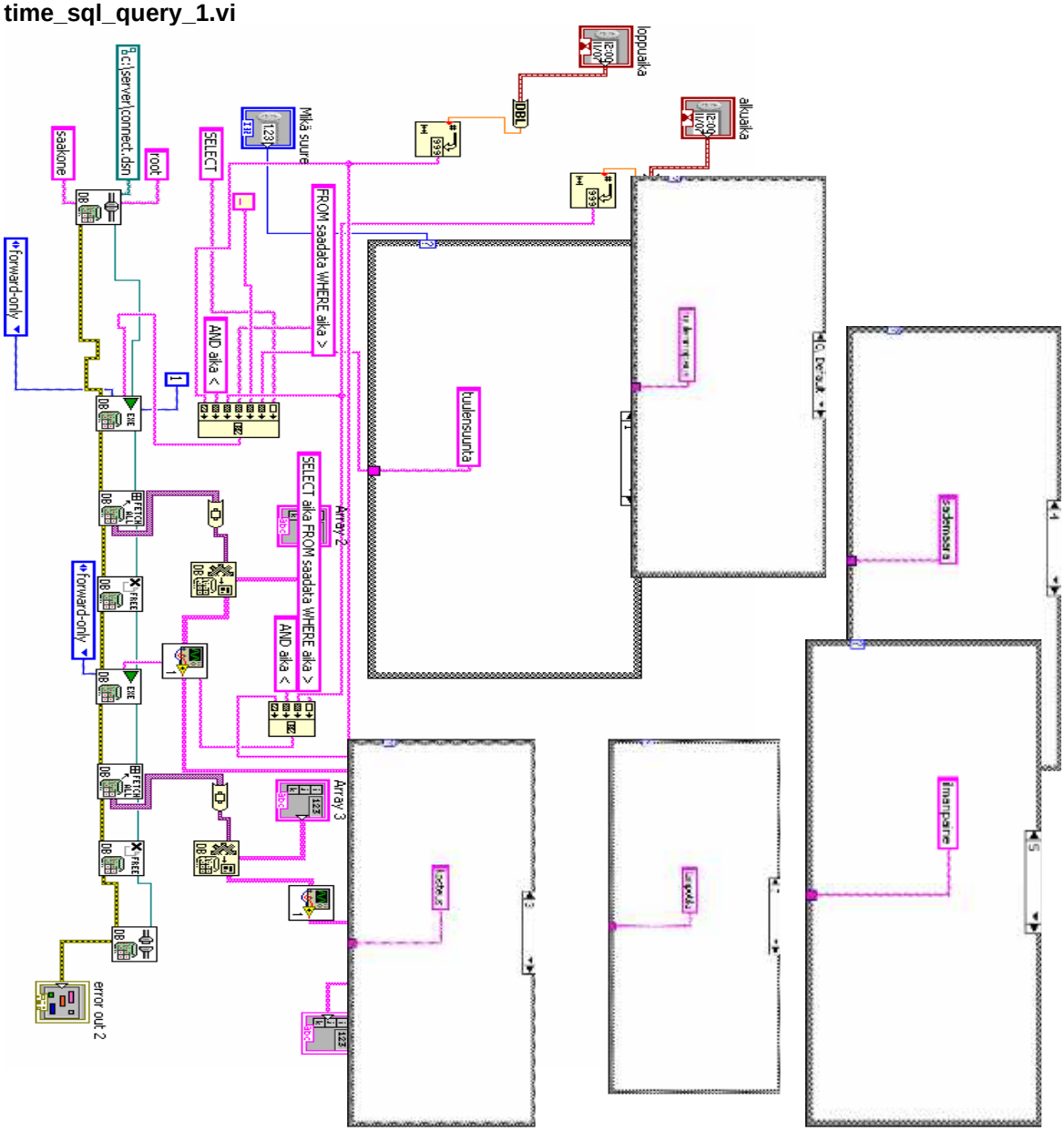
timestmparr_strarr.vi



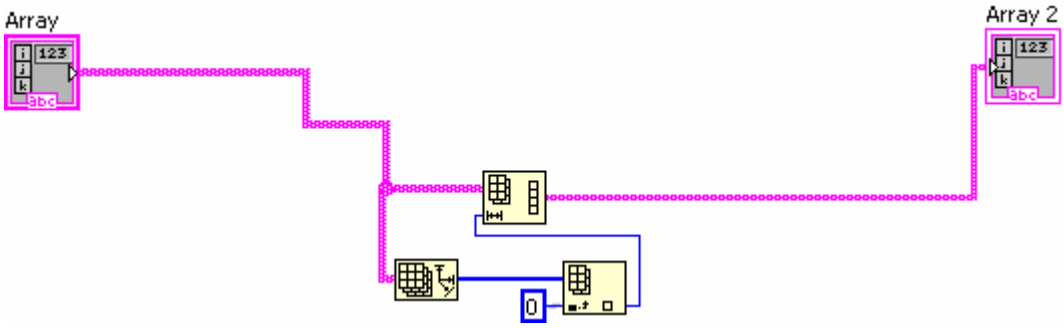
strarray_timestamp.vi



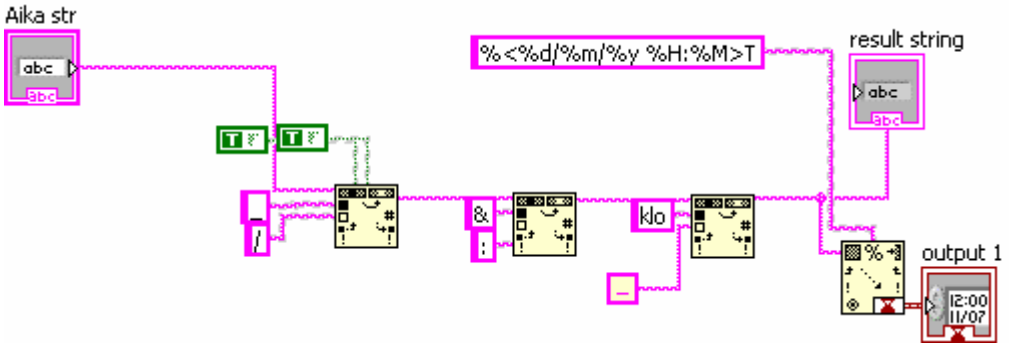
time_sql_query_1.vi



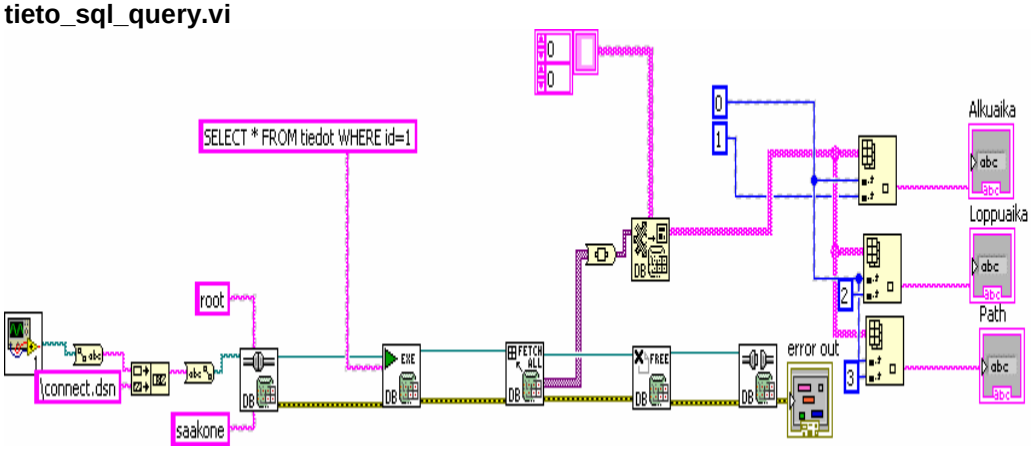
2d_to_1d_array.vi

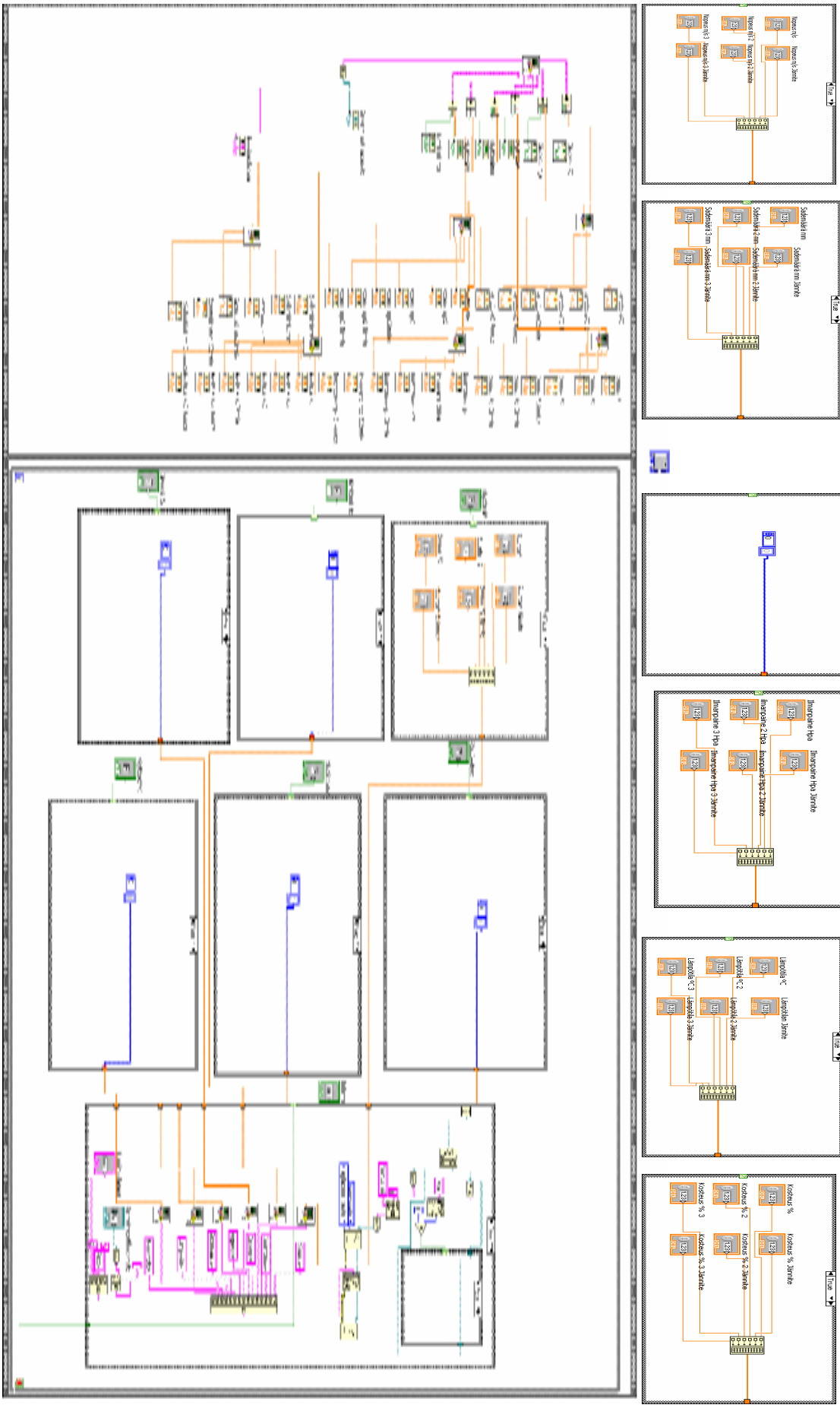


str_to_time.vi

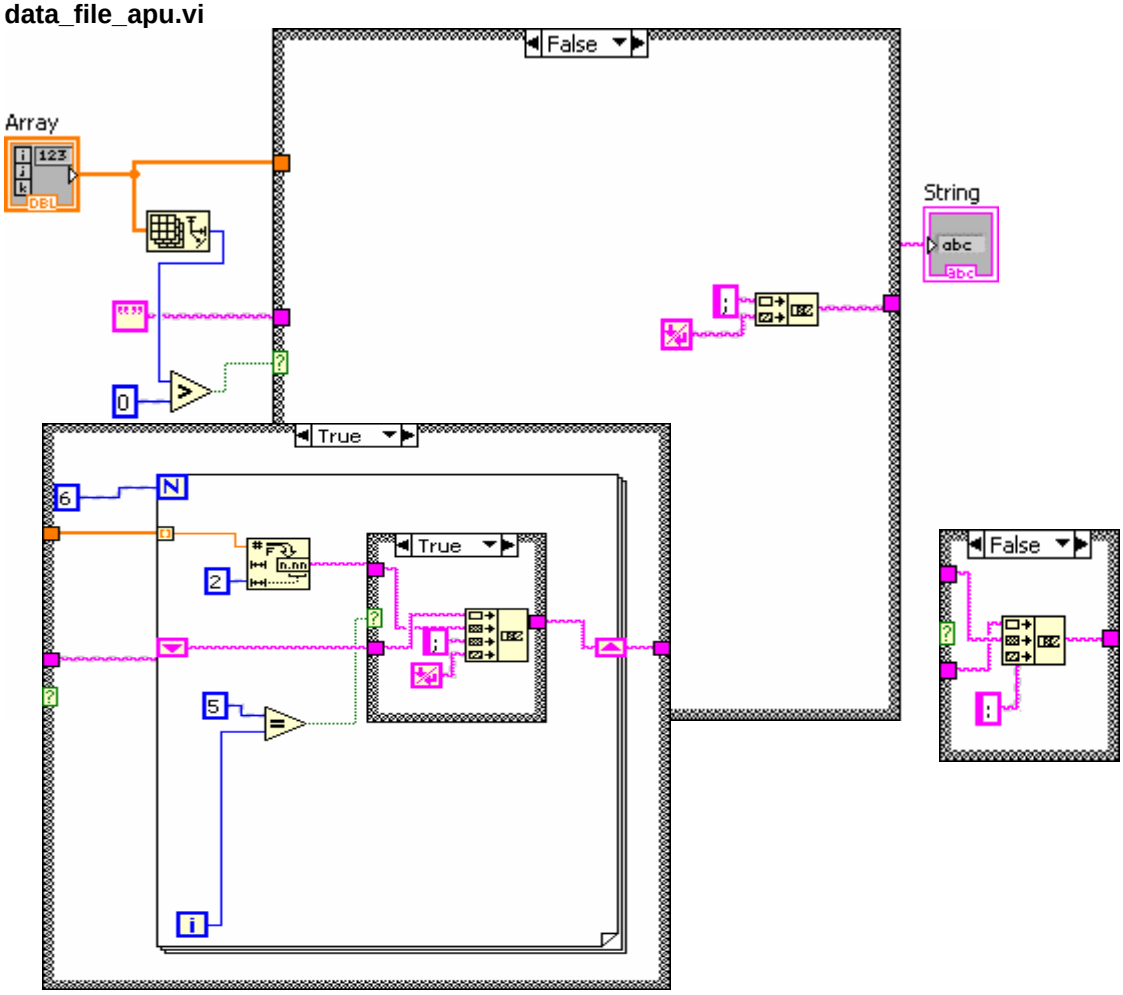


tieto_sql_query.vi

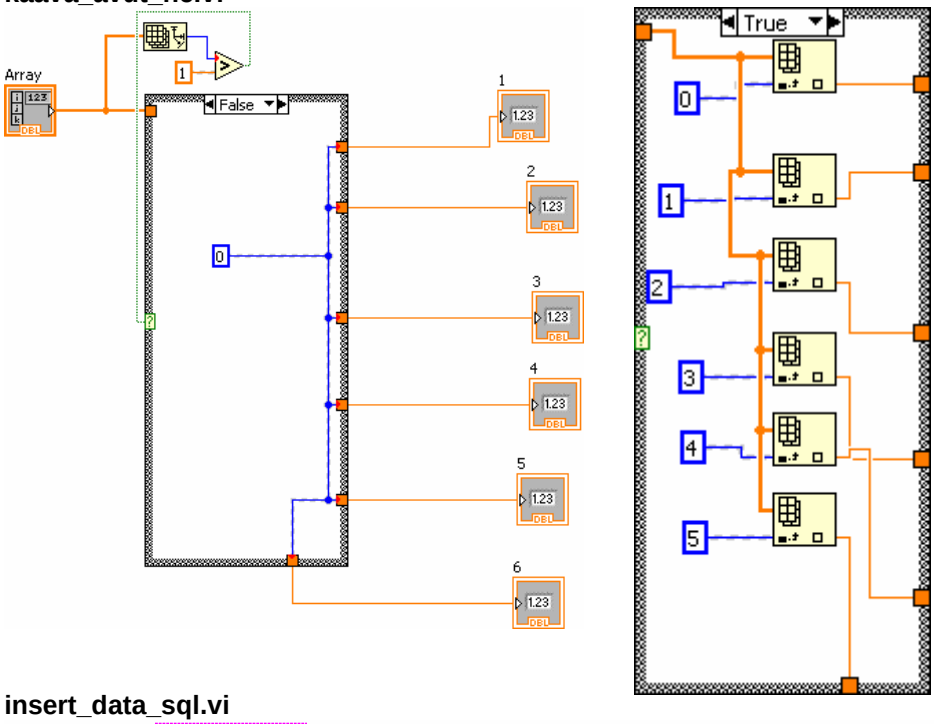




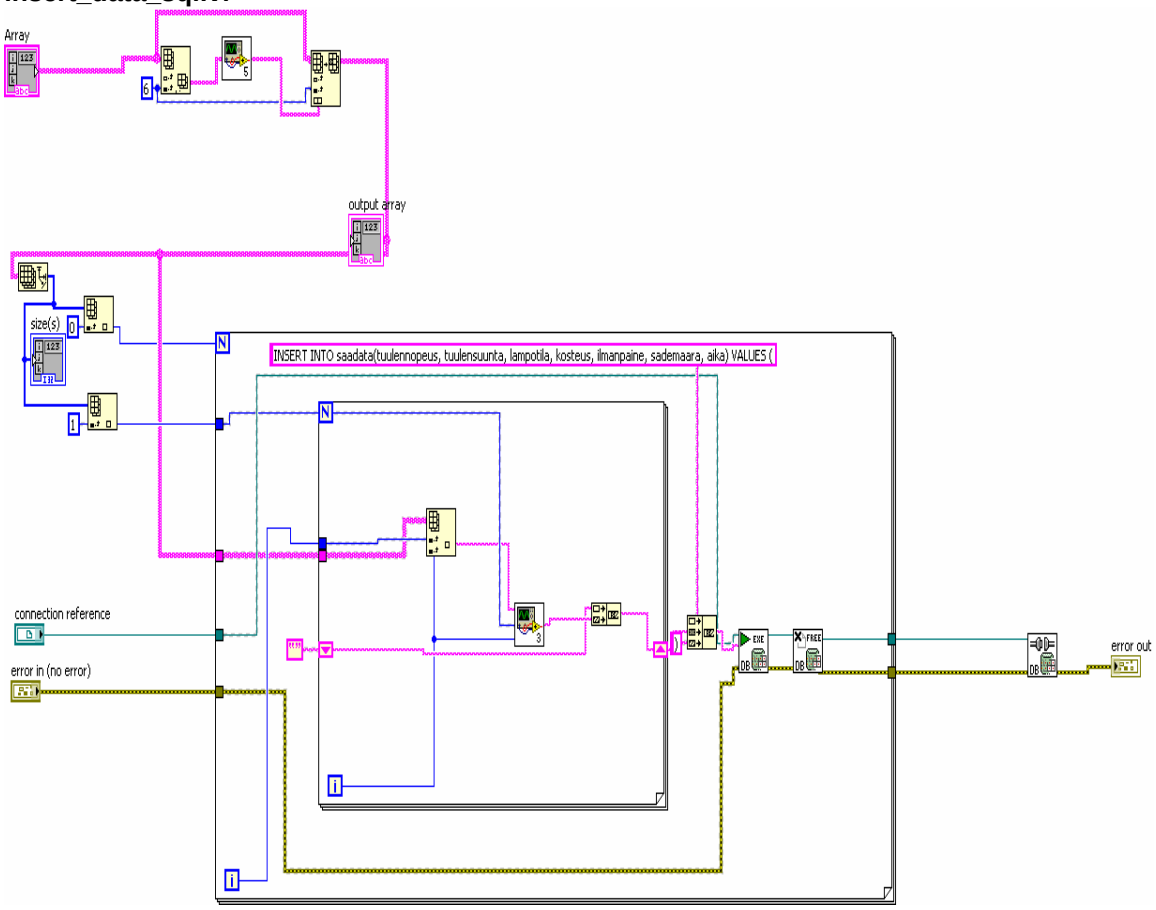
data_file_apu.vi



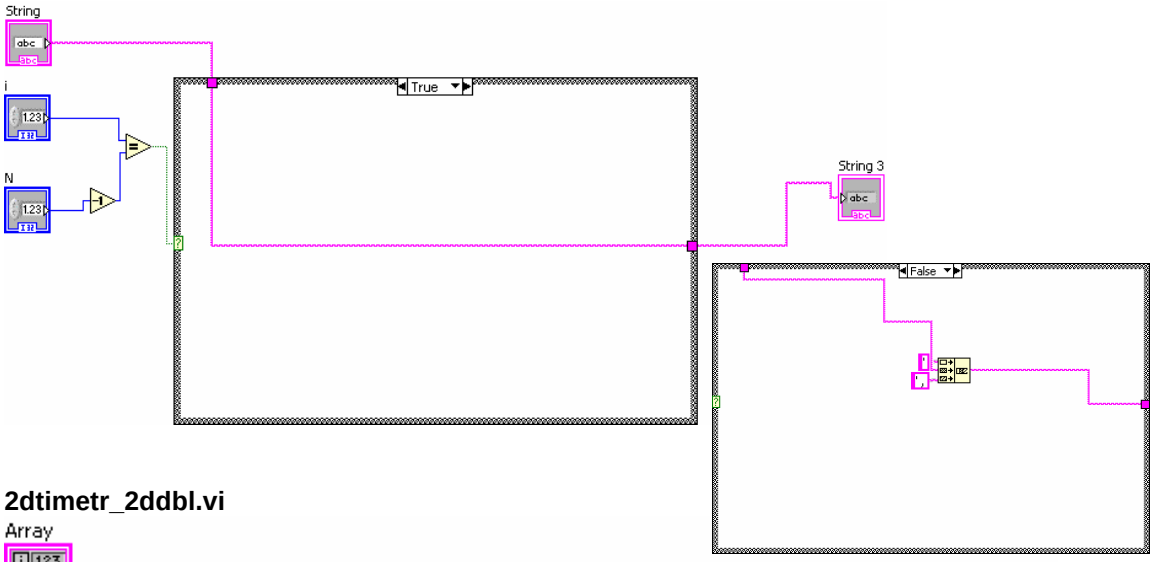
kaava_avut_he.vi



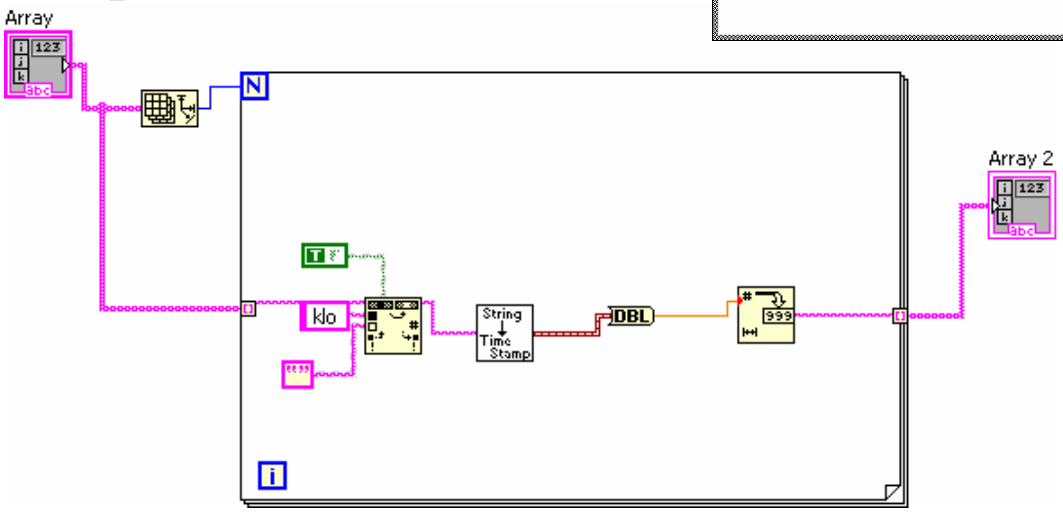
insert_data_sql.vi



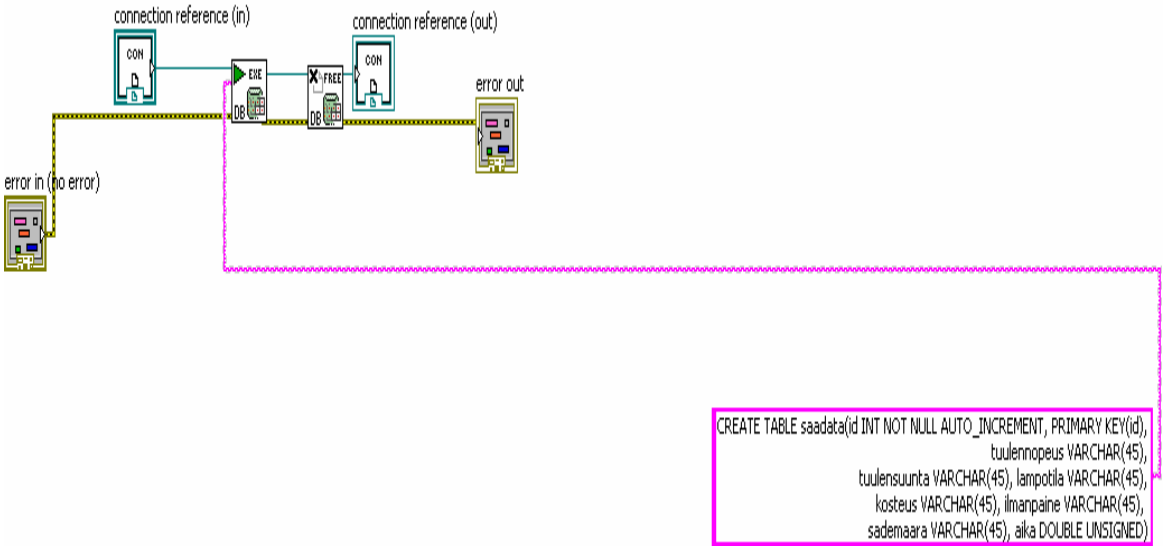
str_apu.vi



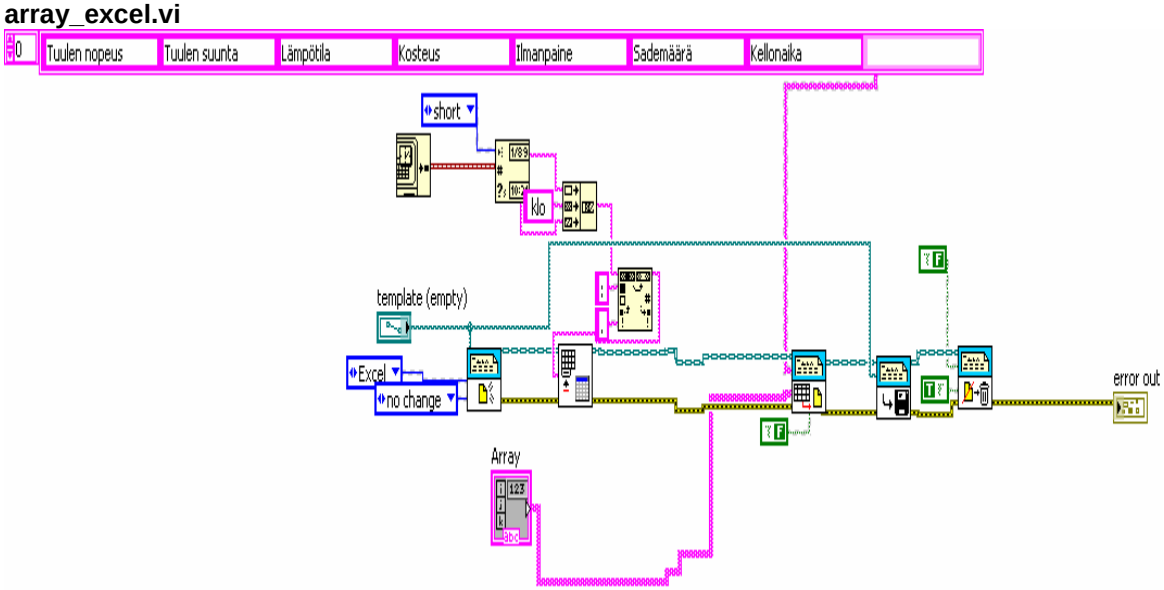
2dtimetr_2ddbl.vi



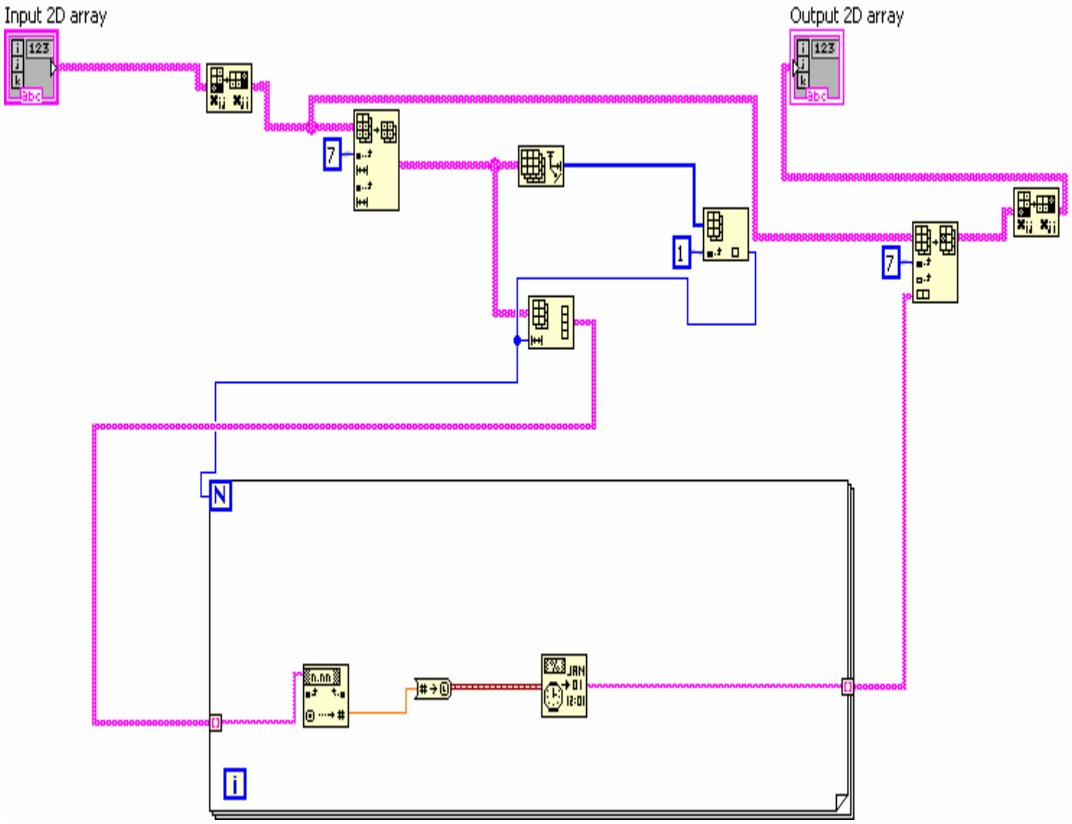
create_table.vi



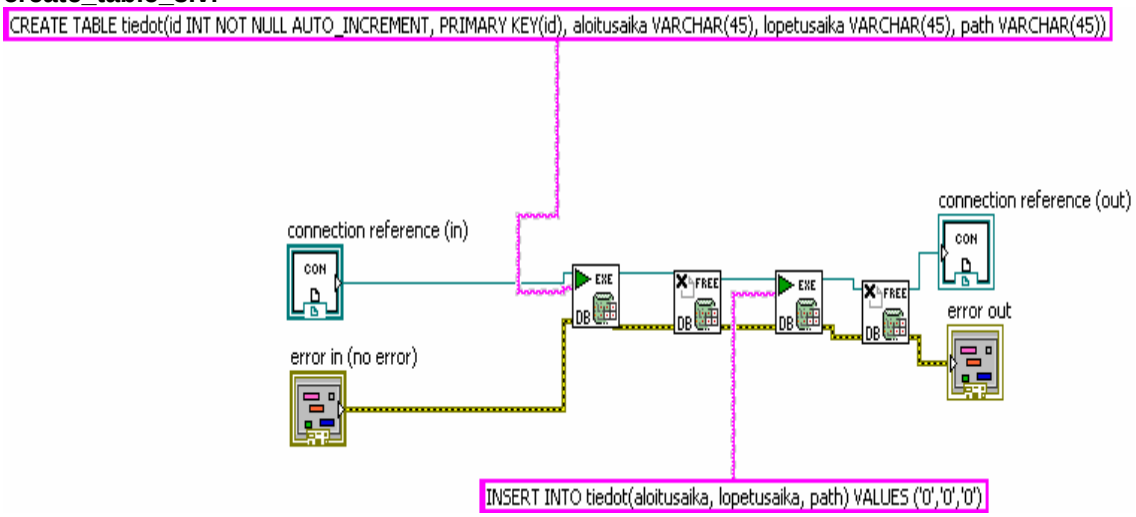
array_excel.vi



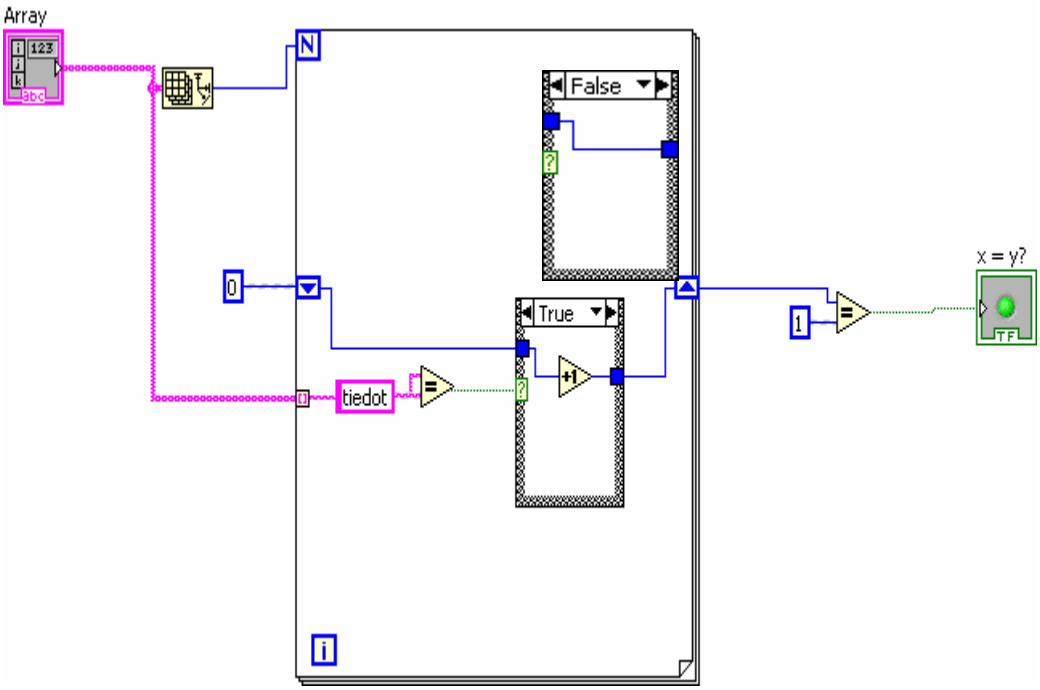
2darr_trans_timstmp_2d.vi



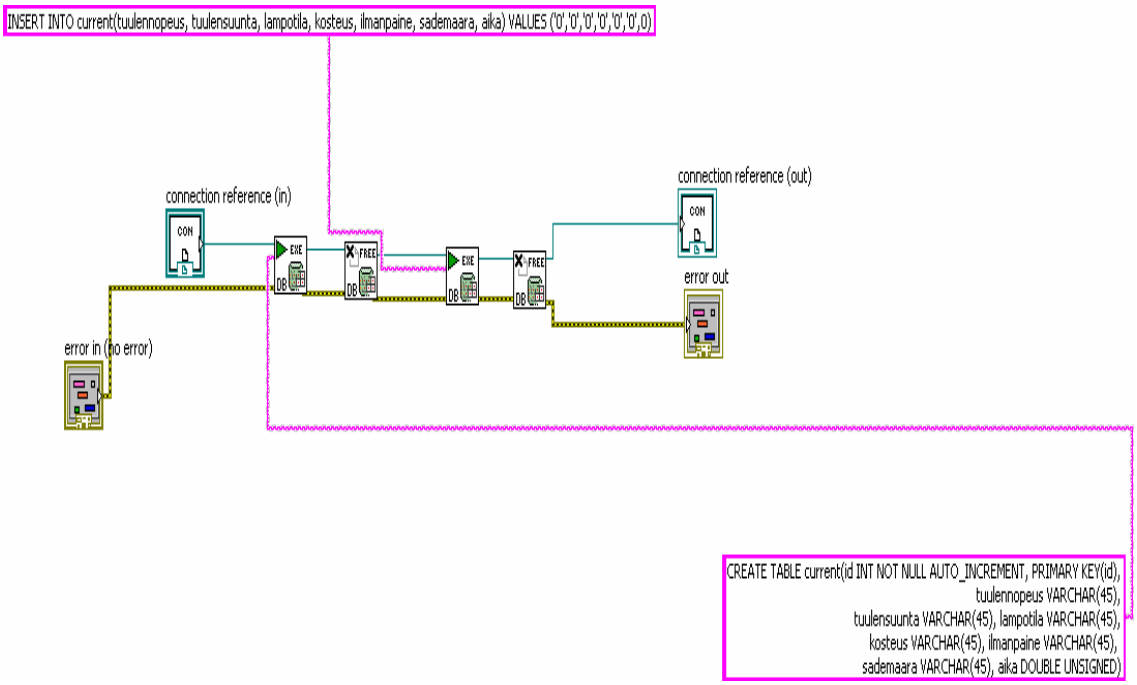
create_table_3.vi



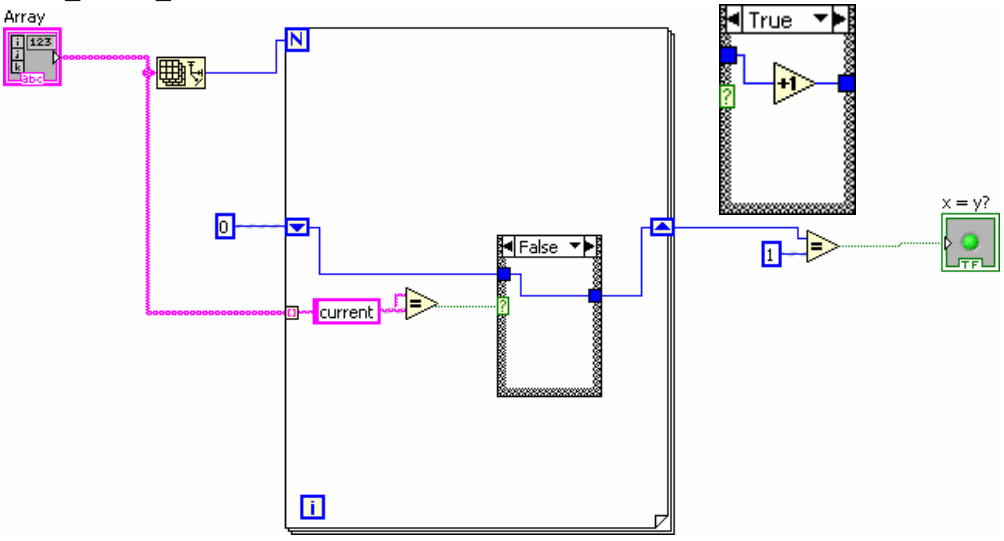
table_exists_3.vi



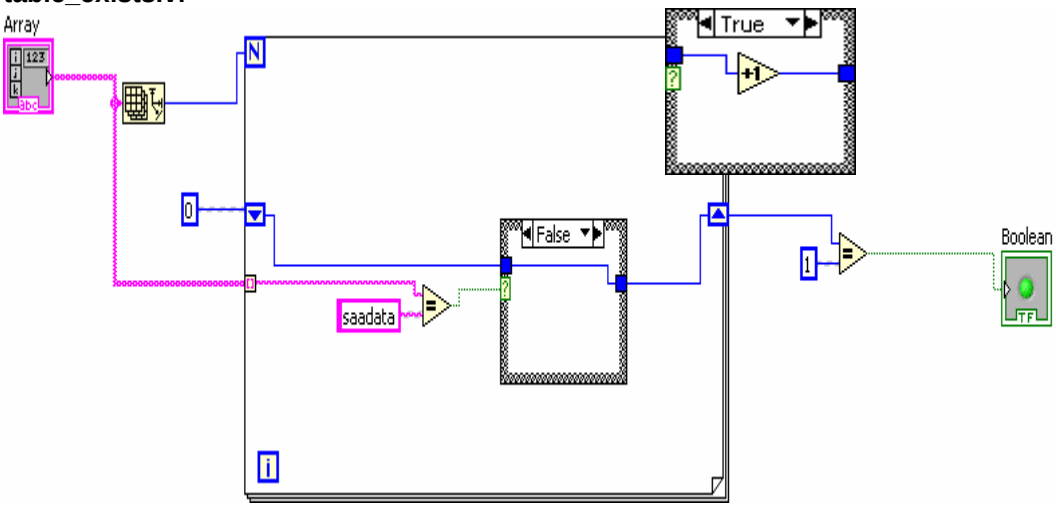
create_table_2.vi



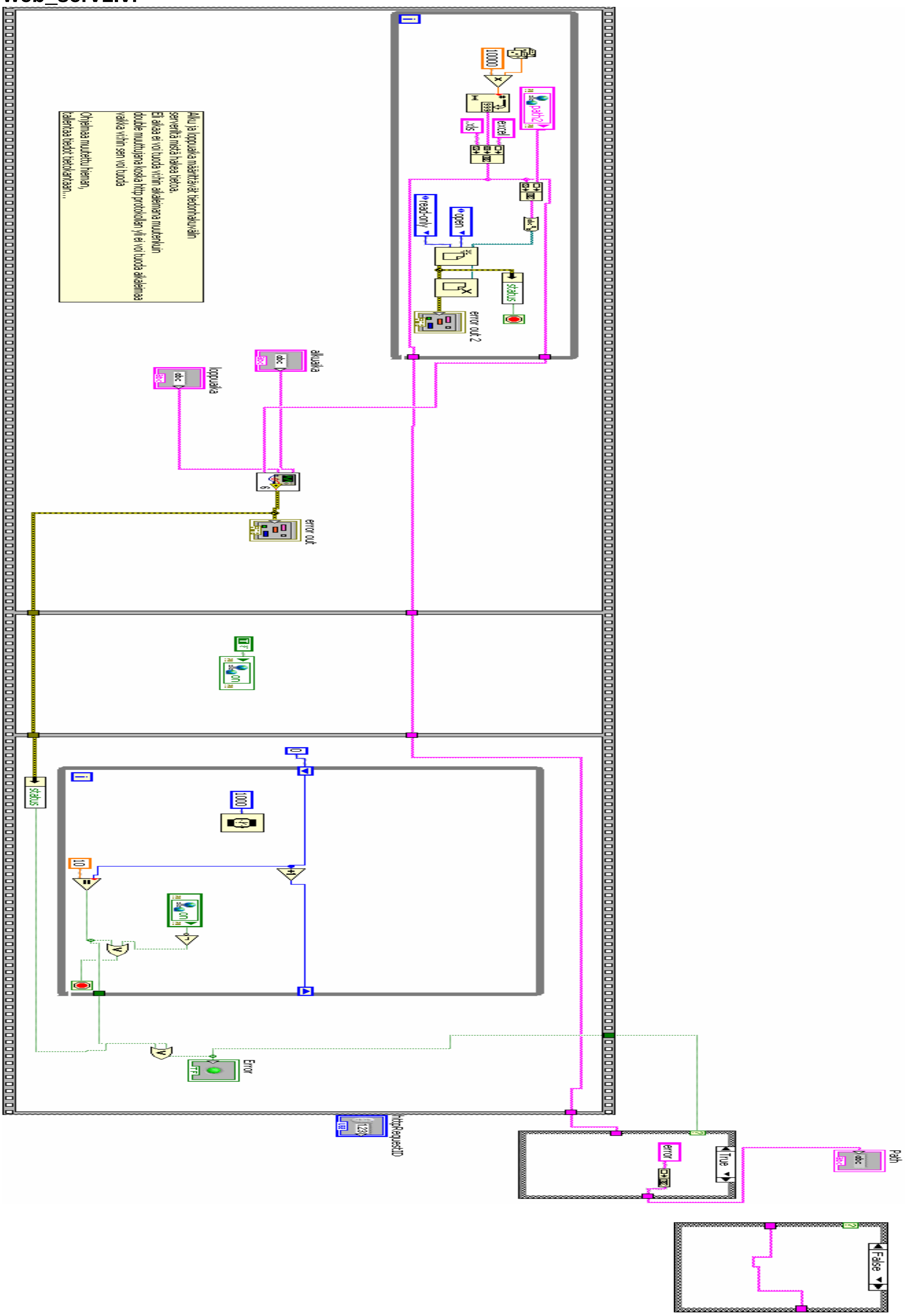
table_exists_2.vi



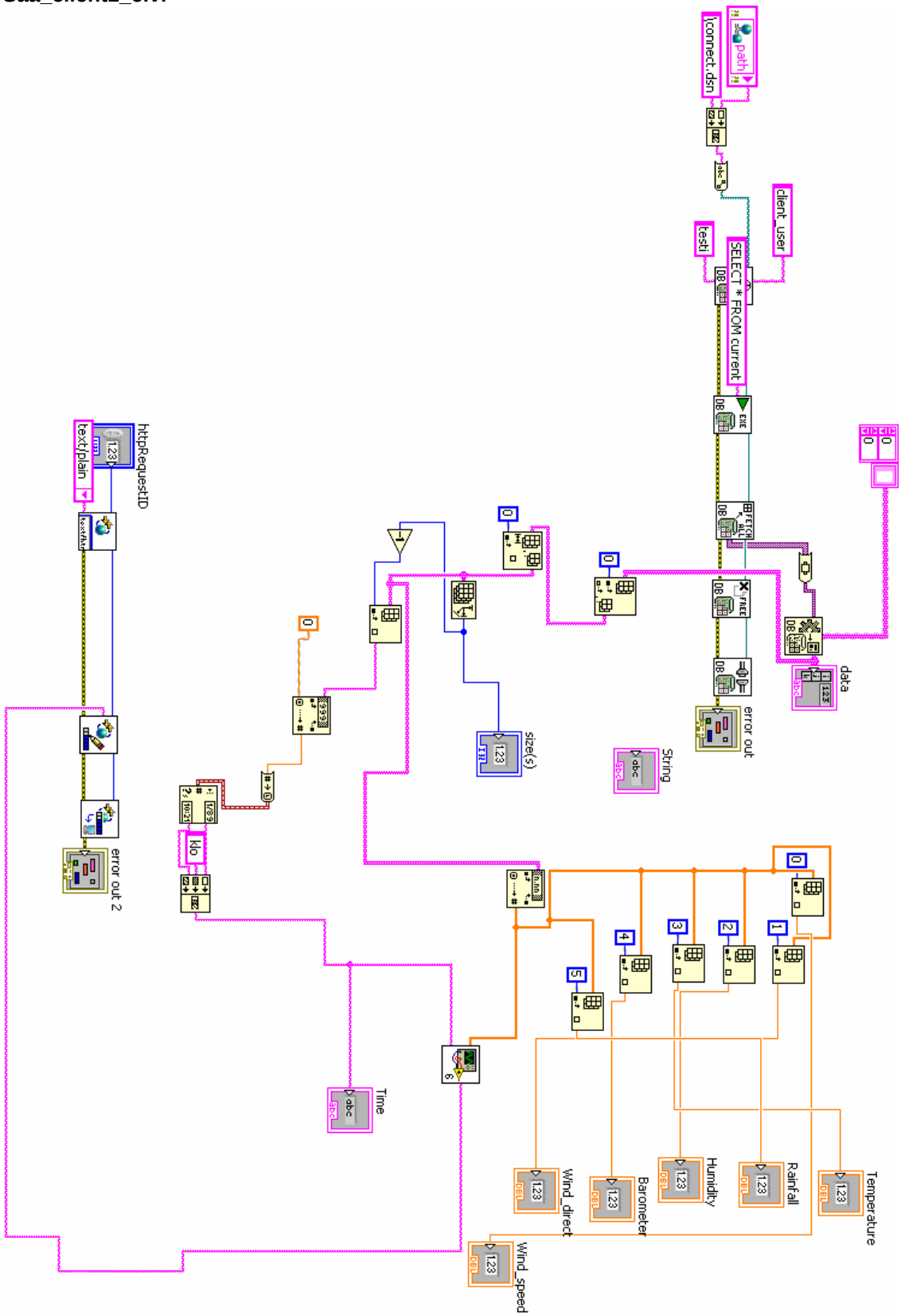
table_exists.vi



web_serv1.vi


alkuaika
loppuaika
error out
error out 2
status
read-only
open
excel
.xls
path2
10000
1000
10
Error
httpRequestID
error
True
False
Path

Saa_client1_c.vi



Sääasema sovelluksen käyttöohje

Tehnyt:
Timo Hietala

Sisällysluettelo

1. Vaatimukset

Ohjelman laitteistovaatimukset ovat:

- Tietokoneessa on oltava NI 6034E mittauskortti asennettuna ajureiden kanssa.
- Verkkokortti, joka on liitettynä käytettävänä olevaan verkkoon.
- Windows XP tai uudempi.
- MySQL-tietokanta asennettavassa koneessa tai saatavilla verkkoyhteyden päässä.
- MySQLi-connector ohjelmisto valitulle MySQL versiolle on asennettu, ennen ohjelmiston asennusta tietokoneeseen.

Mikäli ohjelma aiotaan asentaa tietokoneeseen ilman asennuspakettia, pitää lisäksi olla saatavilla LabVIEW version 8.6.1 development suiten tai uudemman version levyt.

2. Asennus

2.1 Paketin kanssa

Ennen asennusta on luotava C: asemalle server hakemisto, muutoin asennus saattaa epäonnistua. Lisäksi on varmistuttava siitä että ohjelmiston vaatimuksen toteutuvat.
Edellä mainitun jälkeen voidaan aloittaa varsinainen asennus.
Asennuspaketin kanssa ohjelmiston asentaminen on helppoa. Avaa asennuspaketti ja seuraa näyttöön tulevia ohjeita. Asennuksen yhteydessä tulee myös valikko tallennuskansioista, johon voi halutessaan kirjoittaa oman hakemiston, mutta suositeltavaa on käyttää oletushakemistoa.

Paketti asentaa automaattisesti myös verkkopalvelun, jolloin sen asentamisesta erikseen ei tarvitse huolehtia.

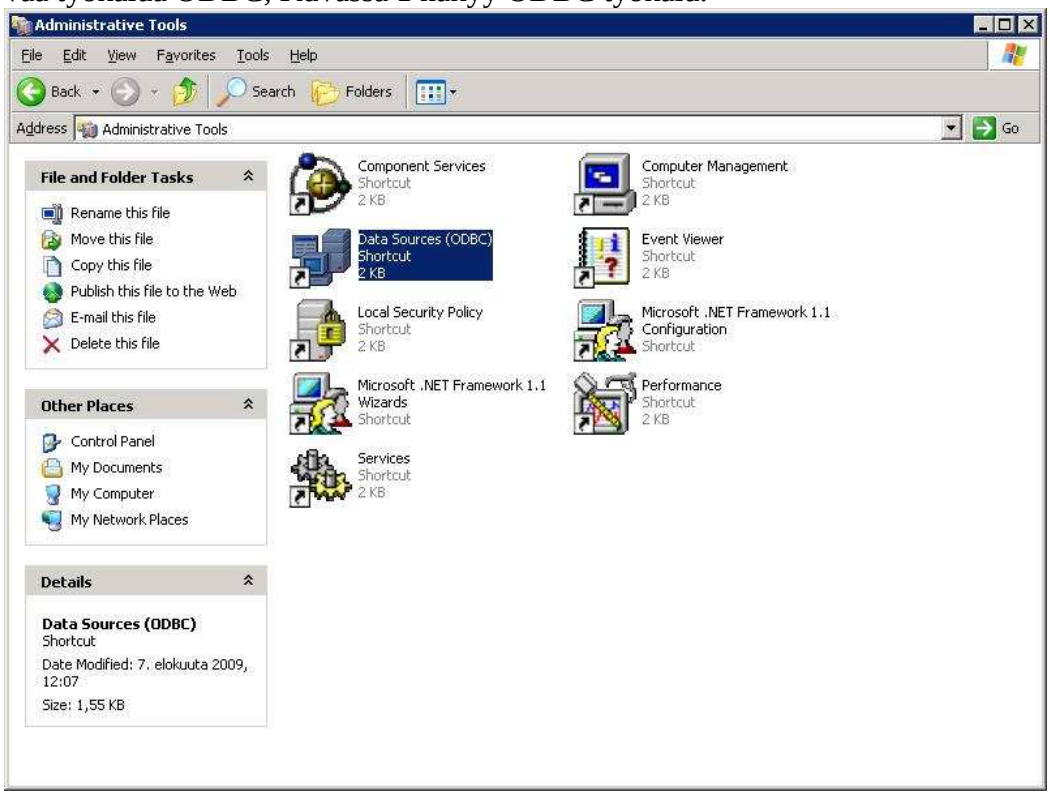
Ainoat tiedostot mitkä pitää erikseen asentaa liittyvät asiakassovellukseen, eli zip-paketista saaasema.zip on purettava kaikki tiedostot aiemmin luotuun server hakemistoon.

Nyt ohjelma on viimeistelyä vaille käynnistettäväksi, mikäli alla olevan **HUOM!** kohdan määrittely ei päde, niin pääohjelma voidaan käynnistää asennetusta hakemistosta. Ohjelman nimi on server_adv ohjelman käynnistyksen jälkeen, ohjelmisto on nyt asennettu.

HUOM! alla oleva pitää tehdä vain, jos MySQL-tietokanta on jossakin muussa koneessa kuin asennettavassa koneessa tai MySQL-tietokannan portti on jokin muu kuin 3306.

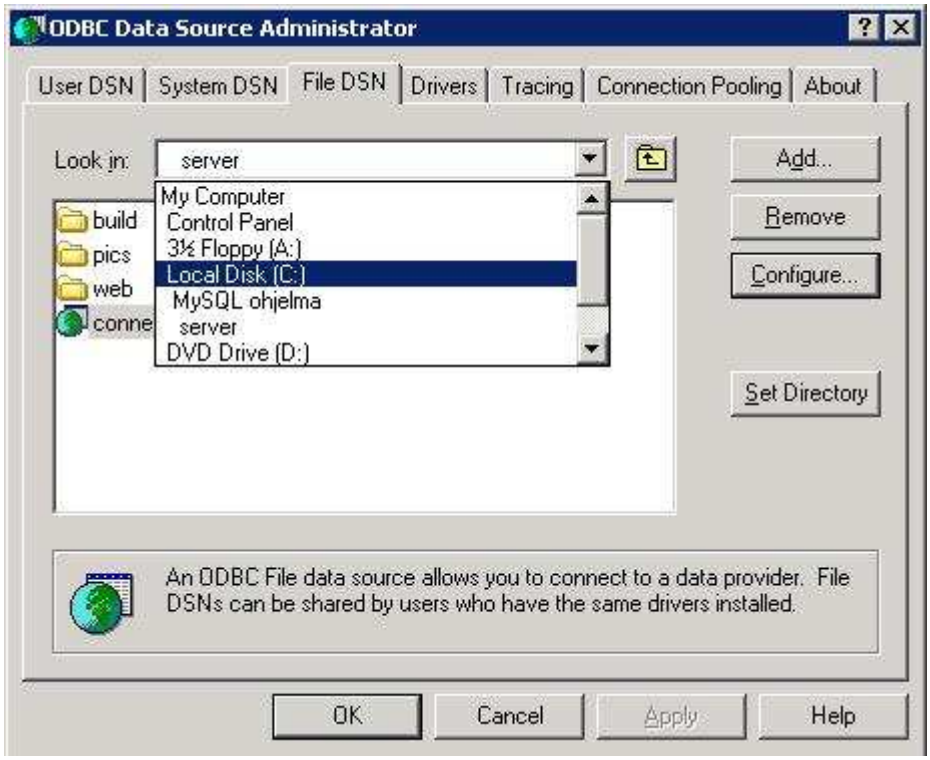
Edellä mainittujen vaiheiden jälkeen pitää vielä muuttaa connect.dsn -tiedosto oikeaksi, connect.dsn -tiedosto löytyy samasta kansioista, mihin ohjelma asen-

nettiin. Muuttaminen tapahtuu käyttämällä Windowsin ohjauspaneelista löytyvää työkalua ODBC, Kuvassa 1 näkyy ODBC työkalu.



Kuva 1. ODBC työkalu.

Avataan työkalu, jolloin näkyviin tulee ODBC-valikko. Valitaan File DSN kuten kuvassa 2 näytetään. Valitaan hakemisto mihin ohjelma on asennettu kohtaan look in. Tämän jälkeen näkyviin pitäisi tulla connect.dsn tiedosto. Valitaan Configure tai tuplaklikataan connect.dsn tiedostoa.



Kuva 2. ODBC valikko.
Tuplaklikkauksen tai configuren painon jälkeen tulee kuvan 3 mukainen dialogi.

Kuva 3. MySQL dialogi.

Mikäli MySQL tietokannan sijainti tai jokin muu kuvassa näkyvä tieto on muuttunut, päivitetään tiedot. Tietojen päivittämisen jälkeen kokeillaan toimiiko MySQL tietokanta painamalla Test -painiketta. Mikäli yhteys muodostuu, niin tiedot ovat oikein. Mikäli ei yhdisty niin jotakin on vialla, jos kaikki tiedot ovat oikein. Tällöin ongelma on joko MySQL tietokannassa, yhteydessä tai palomuurissa ym. sellaisessa. Joka tapauksessa mikäli ongelmia on, niin ohjelmakaan ei tällöin toimi, joten turha käynnistää sitä.
Mikäli yhteys muodostui oikein, voidaan poistua ODBC valikoista painamalla jokaiseen dialogin OK painiketta, jolloin tiedosto tallentuu.

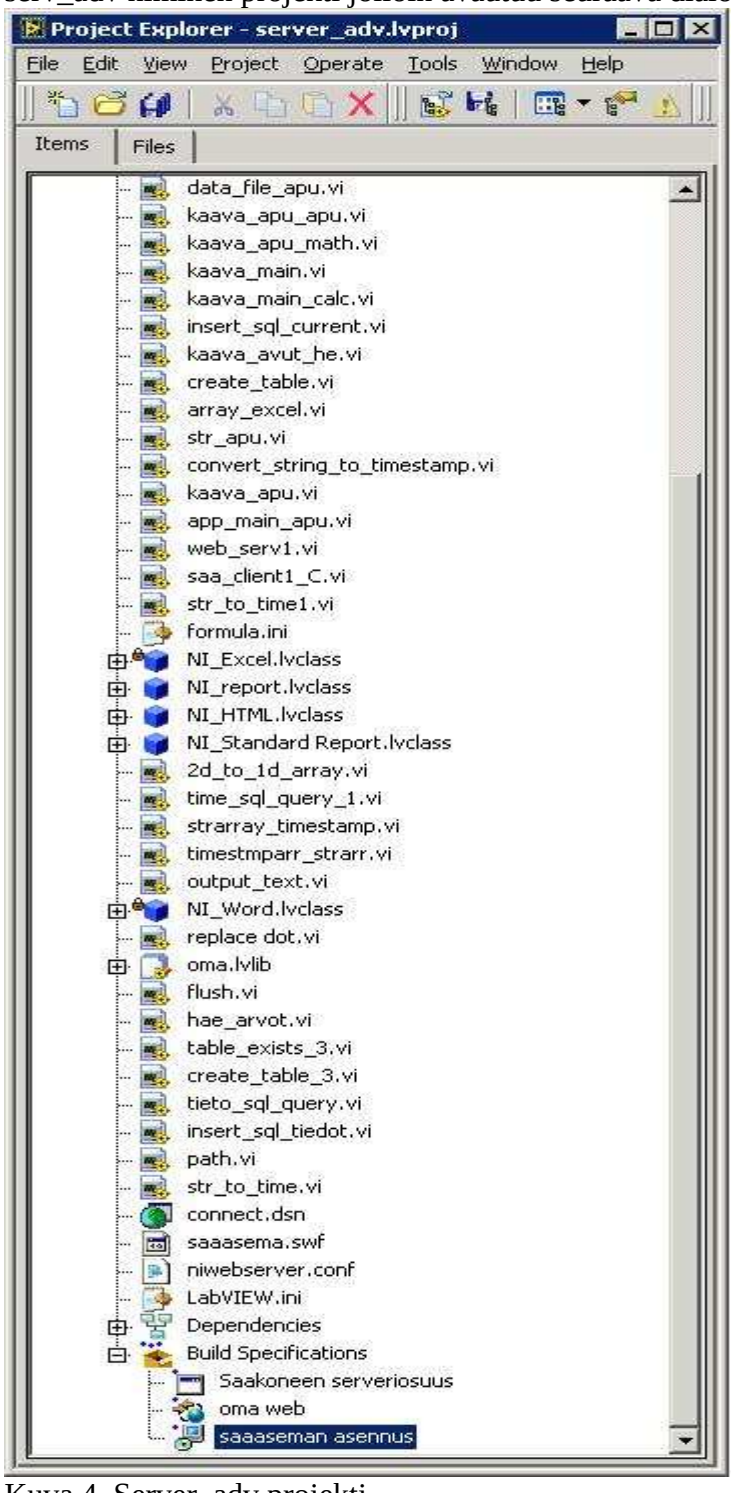
2.2 Ilman pakettia

Ennen asentamista koneen on täytettävä ohjelmiston vaatimukset. Vaatimusten täytyttyä voidaan ohjelmisto asentaa.
Ilman pakettia ohjelmiston asentaminen on hieman työlästä, joten tätä tapaa ei ole suositeltavaa käyttää. Asentaminen menee seuraavasti:

1. Käytettävään koneeseen on asennettava LabVIEW 8.6.1 sisältäen web servicen, database ja daqmx palikat.

2. Puretaan zip-paketista server.zip kaikki tiedostot johonkin hakemistoon.

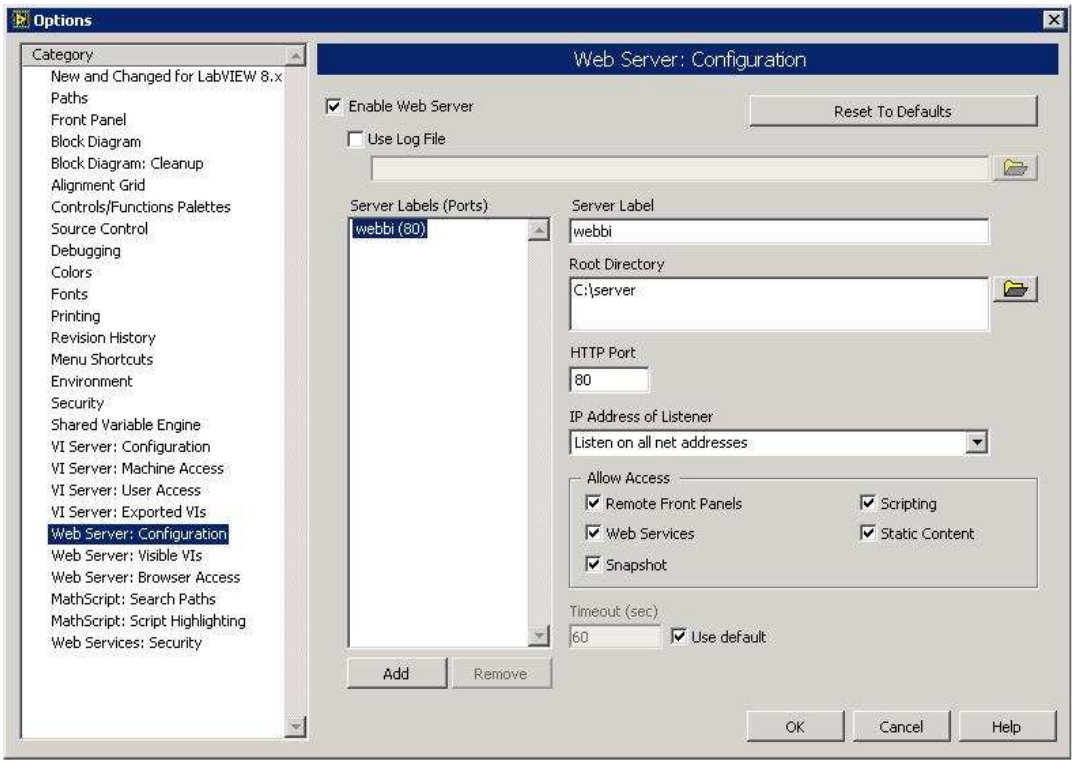
3. Avataan labviewin avulla hakemistosta mihin server.zip paketti purettiin serv_adv niminen projekti jolloin avautuu seuraava dialogi kuva 4.



Kuva 4. Server_adv projekti.

4. Tässä välissä luodaan käytettävälle koneelle C: aseman juureen server kansio. Johon puretaan saaasema.zip kansion sisältö kokonaan.

5. Mennään tools valikkoon ja määritetään LabVIEWin web palvelin ominaisuus päälle Kuvan 5. osoittamalla tavalla, jonka jälkeen hyväksytään asetukset painamalla Ok-painiketta.



Kuva 5. LabVIEWin options valikko.

6. Vieritetään server_adv projektin valikko aivan alas ja aukaistaan build specifiction ryhmä auki kuvan 4. osoittamalla tavalla. Klikataan hiiren oikealla napilla oma web kohdan päällä jolloin avautuu valikko, josta valitaan Deploy vaihtoehto.

7. Mikäli Deploy toiminto onnistui, voidaan siirtyä eteenpäin.

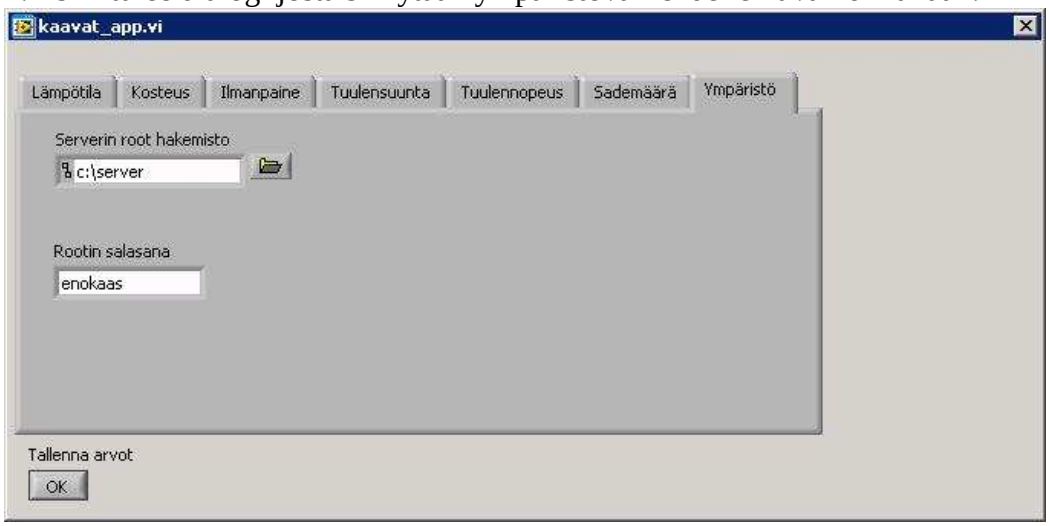
8. Luodaan käytettävälle koneelle pikakuvake, joka viittaa ohjelmaan joka löytyy alihakemistosta mihin server.zip paketti purettiin. Pikakuvake sijoitetaan windowsin käynnistä-valikkoon ohjelmat ja alihakemisto käynnistys. Lisäksi LabVIEW ohjelmisto pitää määrittää käynnistymään automaattisesti lisäämällä myös sen pikakuvake käynnistysvalikkoon.

8 kohtaisen listan jälkeen ohjelma on nyt melkein asennettu käsin. Nyt pitää enää MySQL tietokanta määritellä connect.dsn tiedostoon. Määrittely tapahtuu edellisen osion 1.1 kohdan **HUOM! alla oleva** määrittelemällä tavalla paitsi connect.dsn tiedoston alkulähteenä käytetään alahakemisto build, mihin server.zip paketti purettiin.

Edellisen tehdyn jälkeen pääohjelma on valmis käynnistettäväksi, ohjelma löytyy alahakemistosta build puretun kansion server.zip osoittamasta hakemistosta. Klikataan server_adv.exe sovellusta jolloin pääohjelma käynnistyy ja ohjelmisto on näin ollen asennettu.

2.3 Asennuksen jälkeen

Kun ohjelmisto on asennettu, pitää määritellä pääohjelmaa muutamia asioita mitkä pitää ennen käyttöönottoa tehdä. Ensimmäinen asia on määritellä server_adv ohjelmalle palvelinhakemisto sekä MySQL-tietokannan pääkäyttäjän (root) salasana ohjelmaan seuraavasti:
1. Ohjelmassa siirrytään huoltotilaan ja painetaan määritä kaavat painiketta.
2. Esiin tulee dialogi josta siirrytään ympäristövälilehdelle kuvan 6 mukaan.



Kuva 6. Ympäristö dialogi.

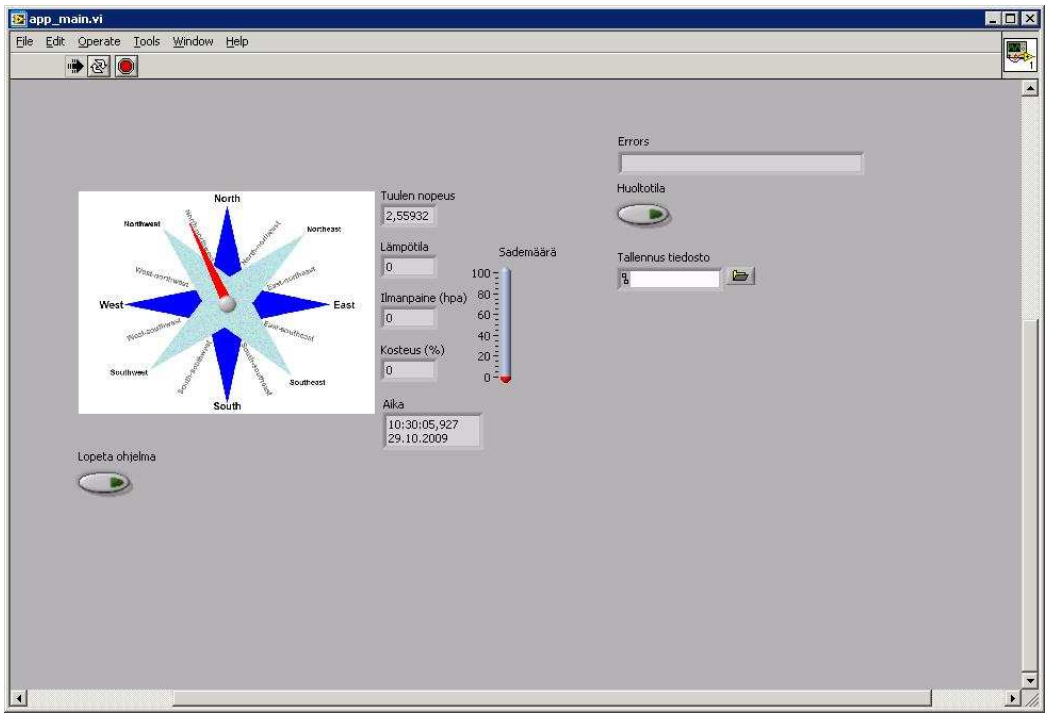
3. Määritetään Serverin root -hakemisto kohtaan hakemisto, missä palvelinhakemisto on (asiakassovelluksen kansio).
4. Dialogi kohtaan rootin salasana kirjoitetaan mysql-tietokannan pääkäyttäjän eli rootin salasana. **HUOM.** Salasanaa ei kirjoiteta suoraan vaan käännetyssä järjestyksessä. Esimerkiksi salasana on historia, niin se kirjoitetaan airotsih dialogiin.
5. Poistutaan dialogista painamalla tallenna arvot painiketta.
6. Poistutaan huoltotilasta.

Oheisen jälkeen ohjelmisto on valmis käytettäväksi.

3. Käyttö

3.1 Palvelinohjelmisto

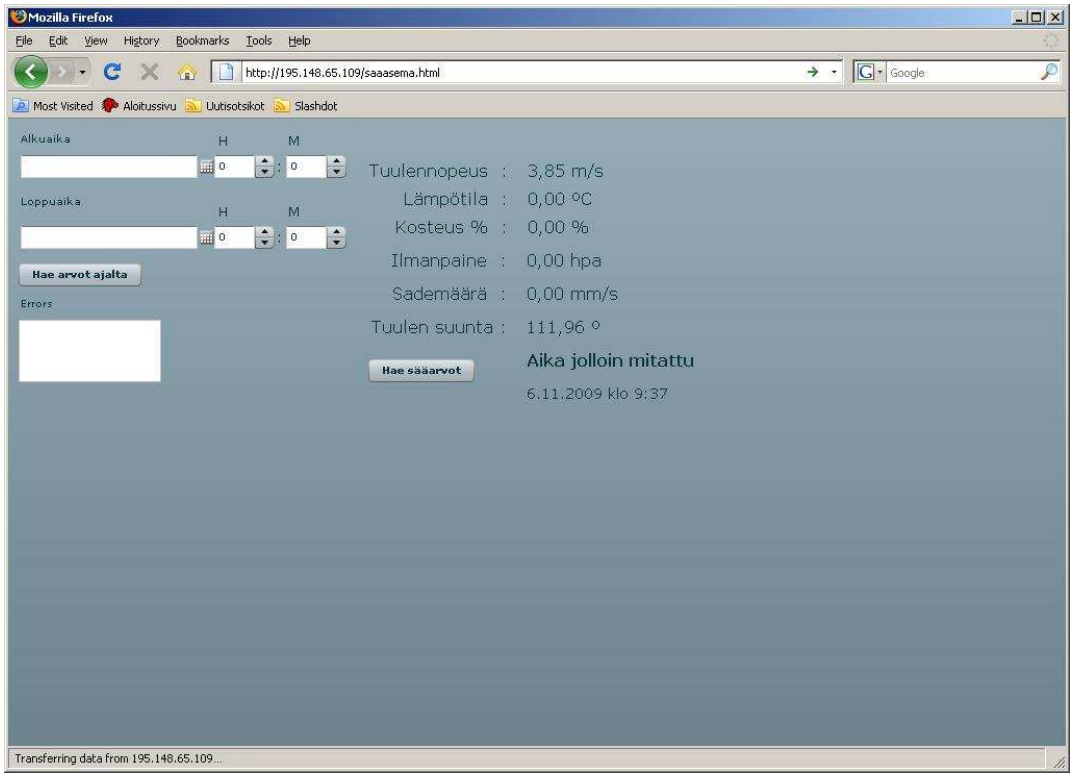
Palvelin ohjelmisto pyörii itsekseen taustalla, joten sitä ei tarvitse käyttää muuta kuin ylläpitotilanteessa tai jos halutaan vain katsella sekunnin välein tapahtuvia mittauksia. Normaalisti ohjelmistolle ei tehdä mitään muuta, kuin voidaan katsella sen toimintaa. Ohessa oleva kuva 7 näyttää ohjelmiston toimintaa, kun ohjelma toimii.



Kuva 7. Ohjelmisto toiminnassa.

3.2 Asiakassovellus

Asiakassovelluksen käyttö on hyvin yksinkertaista. Minkä tahansa Internet selaimen avulla pystytään katselemaan sovellusta, ainoa vaatimus on, että tietokoneessa, millä ohjelmaa katsellaan, on Adoben Flash Player versio 9 tai uudempi. Asiakassovellus näyttää tältä kuva 8.



Kuva 8. Asiakassovellus.

Sovelluksen avulla voidaan hakea aiemmin mitattuja sääarvoja, olettaen että palvelinohjelmisto on ollut toiminnassa tuona aikana. Hakeminen tapahtuu seuraavasti:

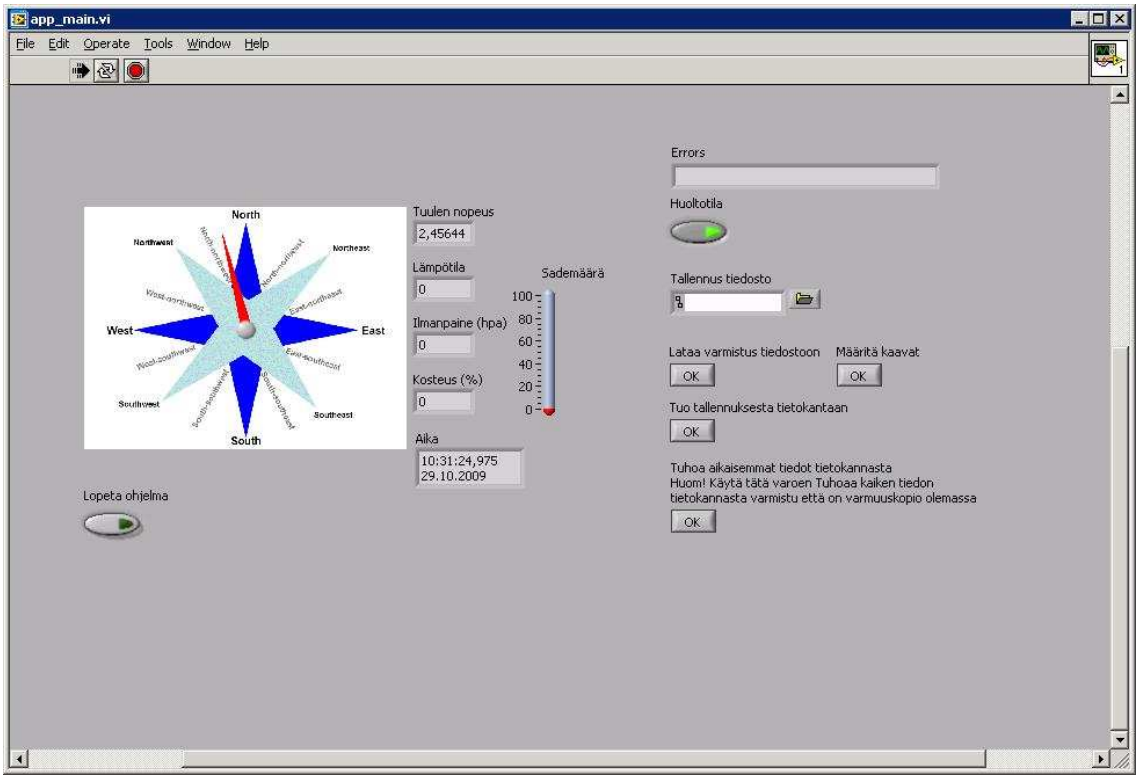
1. Alkuaika kohtaan laitetaan mitattujen tietojen alkuajan kohta kellonajan kanssa, H merkitsee tuntia ja M minuuttia lisäksi kuvakkeen avulla merkitään aloituspäivämäärä.

2. Loppuaika kohta määritellään samalla tavalla kuin alkuaika kohta, paitsi kellonaika ja päivämäärä kuvaavat loppuaikaa.

3. Alku ja loppuajan merkitsemisen jälkeen klikataan ’Hae arvot ajalta’ -painiketta kerran. Klikkauksen jälkeen odotetaan muutamia sekuntteja, jonka jälkeen ohjelma kehottaa lataamaan tiedoston. Kyseisessä tiedostossa on haetun ajanjakson mittaustulokset. **HUOM! mikäli selaimessa on ponnahdusikkunoiden esto päällä voi tiedoston lataamisikkunan ilmestyminen katketa siihen. Suositeltavaa onkin kytkeä esto pois päältä asiakassovelluksen käyttämälle palvelimelle.**

4. Sääaseman ylläpito

Huoltotilaan siirrytään painamalla Huoltotila painiketta pääohjelmasta server_adv, jolloin tulee huolto painikkeet näkyville kuva 9.



Kuva 9. Huoltotila

HUOM! huoltotilan aikana eivät mittaustulokset päivity eikä asiakassovellukselta pysty tekemään mittaustulosten hakua.

Huoltotilasta poistutaan vapauttamalla Huoltotila painike, jolloin ohjelman toiminnot jälleen toimivat.

4.1 Varmuuskopiointi

Joskus kun on esimerkiksi puoli vuotta kulunut, että ohjelma on ollut päällä, on hyvä varmuuskopioida tietokanta tiedostoon. Varmuuskopiointi tapahtuu huoltotilassa seuraavasti:

1. Valitaan tallennus tiedosto kohtaan tiedostonnimi ja paikka, mihin varmuuskopiointi tiedosto tallennetaan.
2. Klikataan Lataa varmistus tiedostoon painiketta, jonka klikkauksen jälkeinen toiminto saattaa kestää useita minuutteja riippuen kuinka iso tietokanta on ja kuinka nopea palvelinkone on.
3. Toiminnon jälkeen varmuuskopiointi on valmis.

Tarvittaessa voidaan mittaustulokset tuoda varmuuskopioidusta tiedostosta tietokantaan seuraavasti:

1. Valitaan tiedosto kohtaan tiedostonnimi ja paikka mistä varmuuskopiointi tiedostosta on kyse.
2. Klikataan Lataa tallennuksesta tietokantaan painiketta, jonka klikkauksen jälkeinen toiminto saattaa kestää useita minuutteja riippuen kuinka iso tiedosto on ja kuinka nopea palvelin kone on.
3. Toiminnon jälkeen varmuuskopioinnista tuonti on valmis.

4.2 Anturin lisääminen/poistaminen

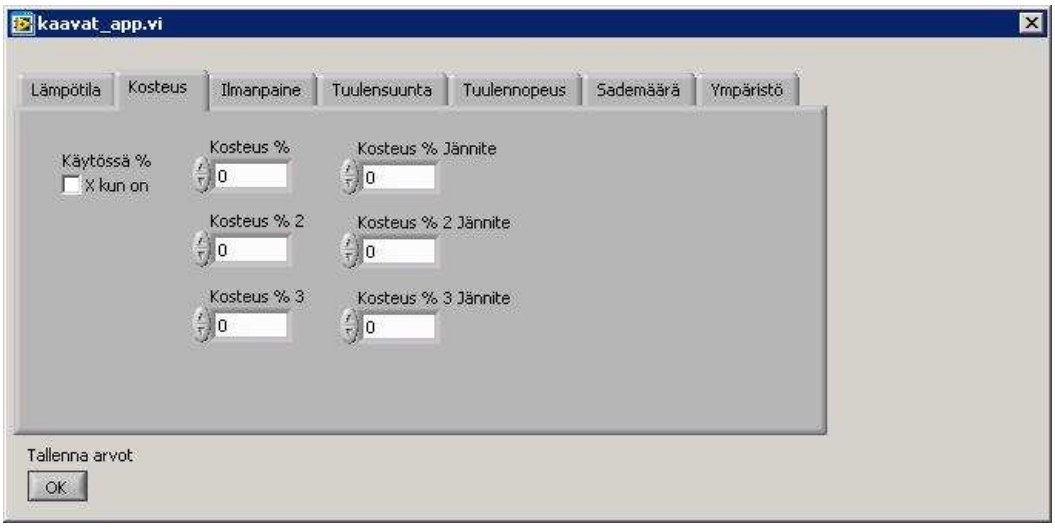
Anturin lisäämiseen on määritelty seuraavat portit: AI0 = tuulennopeuden anturi, AI1 = tuulensuunnananturi, AI2 = Ilmankosteuden anturi, AI3 = lämpötilan anturi, AI4 = Ilmanpaineen anturi, AI5 = Sademäärän anturi.

Aina kun anturi lisätään, pitää server_adv ohjelmaan määritellä anturin pienin suure ja jännite, minkä anturi pystyy mittaamaan. Myös suurin anturin mitattavissa oleva suure ja jännite pitää antaa ohjelmaan. Tarvittaessa voidaan ohjelmaan antaa kolmas tieto minimi ja maksimiarvon väliltä, jolloin suureen laskennan virhettä voidaan yrittää minimoida.

Lämpöanturin lisääminen on ainoa poikkeus ohjelman anturien lisäyksessä, poikkeus liittyy käytettävään suureeseen jonka pitää olla kelvin.

Esimerkiksi kosteuden anturin lisäys ohjelmaan toimii seuraavasti:

1. Liitetään anturi ensin fyysisesti mittauskorttiin.
2. Klikataan huoltotilassa Määritä kaavat painiketta jolloin kuvan 10 mukainen dialogi ilmestyy



Kuva 10. Dialogi.

3. Valitaan dialogin välilehdeksi Kosteus jolloin tulee kuvan 8 mukainen valikko.
4. Määritetään ilman numeroa olevaan suureeseen anturin pienin mittaukseen pystyvä suure, lisäksi ilman numeroa olevaan kohtaan anturin antama jännite kyseiselle suureelle.
5. Määritetään numeroon 3 olevaan suureeseen anturin suurin mittaukseen pystyvä suure, lisäksi numeroon 3 olevaan kohtaan anturin antama jännite kyseiselle suureelle.
6. Tämän jälkeen anturin lisäys on puoliksi valmis. Laitetaan käytössä kohtaan ruksi ja poistutaan dialogista painamalla tallenna arvot painiketta.
7. Anturin lisäys on valmis. Poistutaan huoltotilasta, jolloin uuden anturin antama suure näkyy siinä kohdassa minkä anturi lisättiin. Uuden anturin antama mittaustulos päivittyy myös automaattisesti asiakassovelluksen näyttöön.

Halutessa voidaan ohjelmalle antaa ’määritä kaavat’ -painikkeen tuottamassa kohdassa kolmas tiedetty arvo numeroituun kohtaan 2, joka on minimin ja maksimin väliltä. Tällöin ohjelman tuottamaan suureen tarkkuus kasvaa.

Anturin poistaminen

Anturin poistaminen on yksinkertaista. Klikataan määritä kaavat painiketta ja poistetaan käytössä kohdasta ruksi poistetun anturin valikosta. Jonka jälkeen painetaan tallenna arvot painiketta, jonka jälkeen ohjelma ei enää huomioi anturia.

HUOM! Mikäli määritelty anturi on väärässä portissa, niin ohjelma ei huomioi sitä, vaan mittaa suureen arvoa määritellystä anturista vaikka anturi ei ole siinä. Tällä tavalla mitattua suuretta ei voi pitää luotettavana, itse asiassa suure on tällöin 100% väärin.

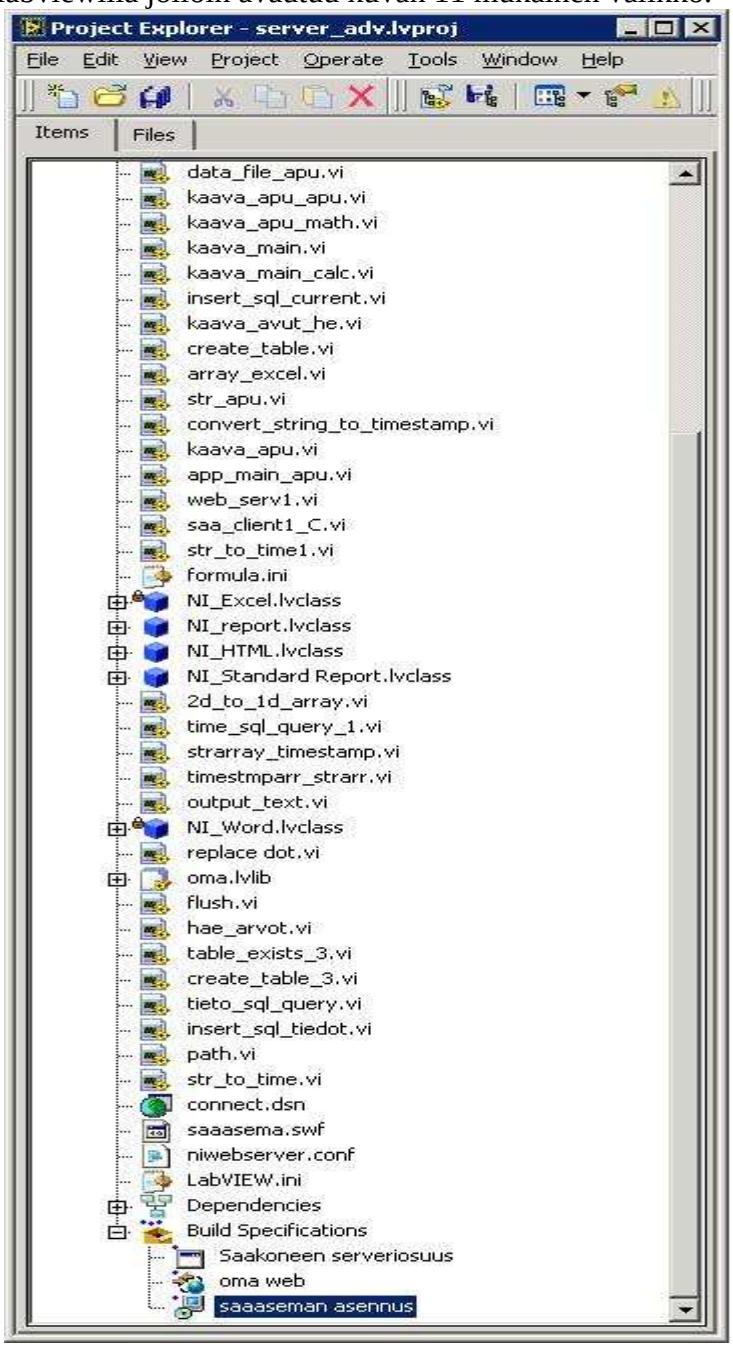
4.3 Muuta tietoa

Mikäli asiakasohjelmaa käytetään tietojen hakuun, niin ohjelman määriteltyyn server hakemistoon kerääntyy ajan myötä paljon excel tiedostoja. Onkin suositeltavaa aina huollon yhteydessä poistaa kaikki xls-päätteiset tiedostot server hakemistosta käsin, menemällä komentokehotteeseen ja server hakemistoon käyttämällä komentoa del *.xls.

Huoltotilassa on lisäksi toiminto, jonka avulla voidaan tuhota aikaisemmin mitatut arvot. Toimintoa pitää käyttää varoen, jos varmuuskopiota tietokannasta ei ole, niin kaikki aikaisemmin mitatut tiedot menetetään.

5. Sääasema ohjelmiston asennuspaketin luominen

Sääasema paketin luominen on helppoa, avataan server_adv niminen projekti labviewillä jolloin avautuu kuvan 11 mukainen valikko.



Kuva 11. Server_adv projektin valikko.

Vieritetään valikko alas asti kuvan 10 osoittamalla tavalla avaten ’build specifications’ -ryhmä. Ryhmän avaamisen jälkeen klikataan oikealla ’saaaseman asennus’ -kohtaa ja valitaan build. (Ennen tämän vaiheen suorittamista pitää olla labviewin version 8.6.1 levyt käytettävissä) Build -kohdan klikkauksen jälkeen seuraa näyttöön tulevia ohjeita. Kun LabVIEW on luonut asennuspaketin, näyttöön tulee lisäksi ilmoitus missä kansiossa paketti sijaitsee, josta se voidaan kopioida talteen sekä käyttää sitä asentamaan ohjelmisto toiseen koneeseen tämän ohjeen mukaan.