VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Toni Puukko

SULAUTETUN PROSESSINSEURANTA-LAITTEEN KÄYTTÖLIITTYMÄ

Tekniikka ja liikenne
2010

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan opetusohjelma

TIIVISTELMÄ

Tekijä	Toni Puukko
Opinnäytetyön nimi	Sulautetun prosessinseurantalaitteen käyttöliittymä
Vuosi	2010
Kieli	suomi
Sivumäärä	35 + 2 liitettä
Ohjaaja	Jukka Matila ja Pekka Liedes

Työssä on aiheena osa Wapice Oy:n asiakasprojektin prosessinseurantalaitteen uudistettua ohjelmistoa. Työssä toteutetaan sulautetussa laitteessa ajettavan ohjelmiston käyttöliittymä. Prosessinseurantalaite voidaan konfiguroida seuraaman erilaisia prosesseja. Asiakasprojektiin kuuluu myös PC konfiguraatiotyökalu. Työkalulla luotu konfiguraatio määrittelee laitteella seurattavat arvot ja miten ne esitetään käyttäjälle. Mitattavia arvoja on yleensä satoja.

Koska arvoja on niin monta, on käyttöliittymä jaettu sivuihin. Käyttöliittymässä toteutetaan mekanismi, joka rakentaa konfiguraatiossa määritellyt sivut. Käyttöliittymän avulla säädetään myös sulautetun laitteen asetuksia. Käyttöliittymän toteuttamiseen käytetään käyttöliittymäkirjastoa, jota pääasiassa käytetään PC-ohjelmien käyttöliittymien toteuttamiseen. Tämän takia kaikkia projektissa käytettäviä käyttöliittymäkirjaston komponentteja täytyy muokata, jotta niitä voidaan käyttää uudella laitteella. Joitakin täysin uusia komponentteja täytyy myös toteuttaa.

Uusi versio toteutettiin käyttäen Qt kirjastoa. Ohjelmistosta on jo toimitettu asiakkaalle ensimmäinen versio. Ohjelmiston jatkokehitys on vielä käynnissä.

Avainsanat käyttöliittymät, C++, ohjelmointi, sulautettu tietotekniikka

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Toni Puukko
Title	User Interface for an Embedded Process Monitoring Device
Year	2010
Language	Finnish
Pages	35 + 2 Appendices
Name of Supervisor	Jukka Matila and Pekka Liedes

The subject of this thesis is the implementation of a graphical user interface for an embedded process monitoring device. The subject is part of a project for a customer of Wapice Oy. The process monitoring device can be configured to monitor different processes. As a part of the customer project a configuration application that runs on a PC has also been created. The configuration defines the process parameters that are monitored and defines how the parameters are presented to the user. The number of defined parameters is usually in the hundreds.

Because there are so many parameters, the user interface has been divided into pages. The pages are defined and configured with the PC application. The pages are constructed in the user interface and presented to the user. The user interface is also used to modify the settings of the embedded device. The user interface is implemented by using a library that is primarily used to implement user interfaces on a PC. Some functionality needs to be added to the components that are used in the user interface to make them usable on the embedded device.

The user interface was implemented using Qt as the graphical user interface library. First version of the software has already been completed. Further development is still ongoing.

Keywords C++, programming, embedded devices, user interfaces

SISÄLLYS

TIIVISTELMÄ
ABSTRACT
LIITELUETTELO
KÄSITELUETTELO
1 JOHDANTO....7
1.1 Alustan kuvaus....7
1.2 Yleistä käyttöliittymäkirjastoista....8
1.3 Käyttöliittymäkirjastoa käyttävän ohjelman rakenne....10
2 KIRJASTOJEN ARVIOINTI....13
2.1 Qt....13
2.2 Gtkmm....16
3 KÄYTETYT TEKNOLOGIAT....18
3.1 Kehitysympäristö....18
3.2 CMake....19
4 KÄYTTÖLIITTYMÄN MÄÄRITTELY....20
4.1 Komponentit....20
4.2 Pääikkuna....21
4.3 Sivut....22
4.4 Painikkeet....22
4.5 Tila- ja hälytysrivit....23
4.6 Muokattava tekstikenttä....23
4.7 Spinbox....24
4.8 Rivikomponentti....24
4.9 Navigointipuu....25
5 TOTEUTUS....26
5.2 Pääikkuna....26
5.3 Käyttöliittymän kytkeminen laitteeseen....27
5.4 Sivut....27
5.5 Komponentit....28

5.6 Painikeryhmät.....29
5.7 Navigointi.....31
5.8 Tekstikenttä ja muokattavat komponentit.....33
6 TESTAUS.....34
7 JOHTOPÄÄTÖKSET.....35

LIITTELUETTELO

LIITE 1 Luokkakaavio

LIITE 2 Sekvenssikaavio

KÄSITELUETTELO

Komponentti (en. widget tai control) on yksi käyttöliittymän osa. Yleensä komponentti esittää käyttäjälle tavan muokata tai tarkastella tietoa käyttöliittymässä.

Avoin lähdekoodi tai vapaa ohjelmisto on ohjelmiston kehitysmenetelmä, jossa ohjelman lähdekoodi on vapaasti saatavilla ja muokattavissa. Ohjelmaa ja siitä tehtyjä muokattuja versioita saa myös vapaasti levittää.

GNU LGPL on GNU-projektin pääasiassa kirjastoille tarkoitettu avoimen lähdekoodin ohjelmistolisenssi. Se takaa kirjaston ohjelmakoodin avoimuuden, mutta ei vaadi kirjastoa käyttäviltä sovelluksilta mitään.

Templatet ovat C++:n abstraktiomekanismi jolla voidaan tehdä luokista uudelleenkäytettävämpiä. Niitä käyttämällä voidaan samaa luokkaa käyttää monen eri tyyppisen tiedon käsittelyyn.

Käyttöliittymäkirjasto on kokoelma luokkia, aliohjelmia ja ohjelmia graafisen käyttöliittymän kehittämistä varten.

1 JOHDANTO

Työn aiheena on osa Wapice Oy:n asiakasprojektin prosessinseurantalaitteen uuttaa ohjelmistoa. Wapice Oy on vuonna 1999 perustettu 150 työntekijän ohjelmistoyritys joka tuottaa ohjelmistoja teollisuuden tarpeisiin. Projektissa on tarkoituksena toteuttaa ohjelmisto uutta prosessinseurantalaitetta varten. Laitteeseen toteutetaan järjestelmäohjelmisto ja käyttöliittymä. Laitteen konfiguroimista varten luodaan PC-konfiguraatiotyökalu. Laitteen seuraamassa prosessissa on yleensä satoja sensoriarvoja. Tämän työn aiheena on projektissa luotava sulautetun laitteen käyttöliittymä.

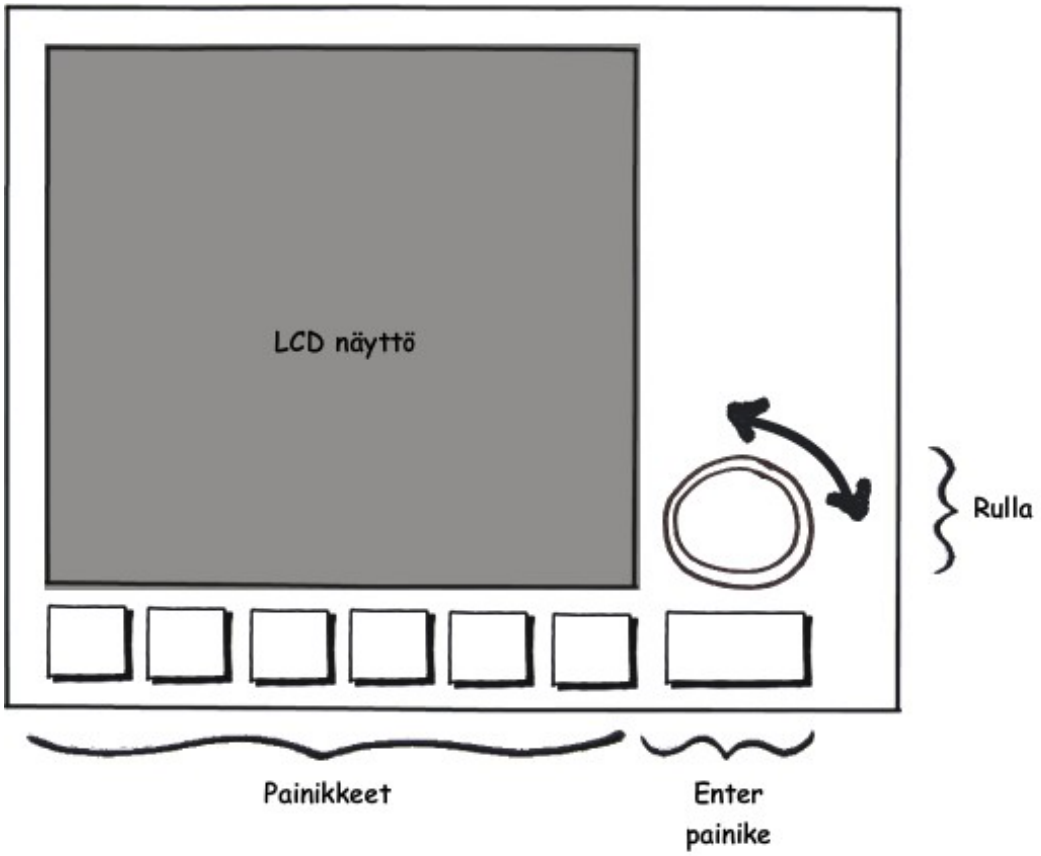
Koska seurattavia prosessiarvoja on satoja, on käyttöliittymä jaettu sivuihin. Sivut määritellään PC-työkalulla luotavassa konfiguraatiossa. Konfiguraatiossa on määritelty sivuilla olevien komponenttien sijainnit, koot ja muut mahdolliset ominaisuudet. Konfiguraation lukeminen tai sen rakenne ei kuulu tämän työn piiriin. Mutta se, miten luetusta konfiguraatiosta syntyy sulautetussa laitteessa käytettävä käyttöliittymä kuuluu.

Sulautettu laite asettaa ympäristönä rajoituksia ja haasteita käyttöliittymän suunnittelulle ja käytettävyydelle. Uudessa ohjelmistoversiossa kiinnitetään myös huomiota prosessiarvojen havainnolliseen esitystapaan. Sulautetun laitteen käyttöliittymän toteutukseen käytetään käyttöliittymäkirjastoa, joka on suunniteltu PC-ohjelmien käyttöliittymien toteuttamiseen. Tämän takia täytyy kirjaston tarjoamista komponenteista tehdä sulautetussa laitteessa käytettäviä.

1.1 Alustan kuvaus

Ohjelmaa ajetaan sulautetussa laitteessa jonka käyttöliittymänä toimivat 10 painiketta ja pyöritettävä rulla jota voi myös painaa. Laitteen prosessorina on 400 MHz PPC prosessori. Laitteessa on 2 ethernet liityntää, USB-liitin ja CAN väylä. Käyttöjärjestelmänä laitteessa on Denx 4.1, joka on sulautetuille laitteille tarkoitettu Linux jakelu. Näyttönä laitteessa on 6,5 tuumainen LCD-näyttö. Näytön resoluu-

tio on 640x480. Keskusmuistia 128 Mt. Laitteessa on myös sarjaportti ja USB 1.1 liitännät.



Kuva 1: Laitteessa olevat painikkeet ja rulla

1.2 Yleistä käyttöliittymäkirjastoista

Käyttöliittymäkirjasto on kokoelma luokkia, aliohjelmia ja ohjelmia graafisen käyttöliittymän kehittämistä varten. Olio-ohjelmoinnin ajatusmaailma sopii käyttöliittymien toteuttamiseen erittäin hyvin, koska eri komponentit voidaan esittää yksittäisinä luokkina. Käyttöliittymät rakennetaan uudelleenkäytettävistä komponenttiluokista, jotka käyttävät toisia komponenttiluokkia osinaan. Tämä johtaa siihen, että komponenteista voidaan muodostaa komponenttihierarkia. Hierarkiassa kaikki komponentin sisällä käytettävät komponentit ovat komponentin lapsia tai lapsen lapsia jne.

Yleisimpiä käyttöliittymäkirjastoja ovat Microsoft Foundation Classes (MFC), Swing, Gtk+ ja Qt. Microsoft Foundation Classes ei tämän sovellu ohjelman toteuttamiseen koska se on käytettävissä vain Microsoft Windows alustalla. Swing on saatavilla alustalle, mutta sen suorituskyky ei ollut testattaessa riittävä. Työn toteuttamiseen parhaiten soveltuvat vaihtoehdot olivat Qt ja Gtk+.

1.2.1 Qt

Qt on C++ kehitysalusta, jota Trolltech on kehittänyt vuodesta 1991. Nokia osti Trolltechin vuonna X. Uusin versio on 4.7.1. Qt tarjoaa suurimman osan sovelluskehityksessä tarvittavista kirjastoista kuten käyttöliittymä-, tietokanta ja XML kirjastot. Sulautettuja laitteita varten on tarjolla Qt Extended. Qt:n voi hankkia kahdella eri lisenssillä, GNU LGPL:llä tai kaupallisella lisenssillä. Qt:n dokumentaatio on kattava ja jokaisen uuden version mukana julkaistaan myös päivitetty dokumentaatio. Qt:lla toteutetut sovellukset on helppo siirtää eri alustoille. Käyttöliittymien ulkoasu on yhdenmukainen alustan alkuperäisen ulkoasun kanssa. Qt:ta käyttäviä sovelluksia ovat mm. Google Earth, Skype ja KDE työpöytäympäristö (Wikipedia: Qt (framework)).

1.2.2 Gtk+ ja Gtkmm

Gtk+ on alun perin GIMP kuvankäsittelyohjelmaa toteutettu käyttöliittymäkirjasto. Kehitys aloitettiin vuonna 1997. Uusin versio 2.20. Gtk+ on toteutettu C-ohjelmointikielellä, jossa ei normaalisti ole tukea olio-ohjelmoinnille. Gtk+ käyttää GLib kirjastoa, joka tarjoaa olio-ohjelmointiin tarvittavat työkalut. GLib kirjasto sisältää kirjastot säikeiden hallintaan, erilaisten datarakenteiden käsittelyyn ja datatyyppien käsittelyyn. Suurin Gtk+:n käyttäjä on GNOME työpöytäympäristö ja siihen liittyvät sovellukset. Gtkmm on GTK+-kirjaston C++ rajapinta joka sovittaa GLib kirjaston oliomallin C++ oliomalliin. Gtk+:n ja gtkmm:n lisenssinä on GNU LGPL (Wikipedia: GTK+).

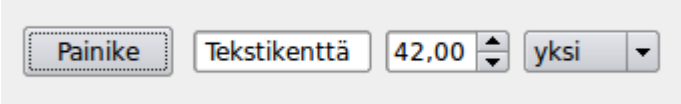
1.3 Käyttöliittymäkirjastoa käyttävän ohjelman rakenne

Perinteisesti tietokoneohjelmaa ajatellaan sekvenssinä käskyjä, jotka tietokone suorittaa. Ohjelma jolla on graafinen käyttöliittymä toimii käyttöliittymää alustettaessa samalla tavalla. Käyttöliittymän alustuksessa komponentit asetellaan ja signaalit yhdistetään. Alustuksen jälkeen ohjelman kontrolli luovutetaan käyttöliittymäkirjastolle.

Jotta käyttöliittymän toimintaa voitaisiin kontrolloida ajon aikana on käyttöliittymäkirjastoissa signaalimekanismi. Esimerkki signaalista on signaali joka lähetetään kun painiketta on painettu. Signaalien mukana voidaan myös lähettää dataa. Eri kirjasoilla on omat toteutuksensa signaaleista. Eri kirjastojen signaalit eivät toimi keskenään.

1.3.1 Komponentit

Komponentit ovat käyttöliittymän rakennuspalasia. Komponentti näyttää tietyn tyyppistä tietoa ja antaa käyttäjälle mahdollisuuden muokata sitä. Jokaista komponenttia vastaa käyttöliittymäkirjastossa luokka. Esimerkkejä komponenteista ovat painikkeet ja pudotusvalikot. Kaikissa kirjastoissa peruskomponentit ovat toiminnaltaan samat. Tässä projektissa käytetään pääasiassa painikkeita, valintalaatikoita ja tekstikenttiä.



Kuvio 1: Painike, tekstikenttä, spinbox ja valintalaatikko

1.3.2 Signaalit

Jotta käyttöliittymässä olevasta painikkeesta tapahtuisi jotain täytyy ohjelman tietää, että painiketta on painettu. Käyttöliittymäkirjastoissa tämä on signaali mekanismin vastuulla. Käyttöliittymäkirjastoissa oleville komponenteille on lisätty tietyt oletussignaalit. Esimerkiksi painikkeissa on määritelty signaali joka lähetetään aina kun painiketta painetaan. Ohjelman täytyy myös kertoa kirjastolle mitä ohjelman kuuluu tehdä kun signaali vastaanotetaan. Tämä tehdään yhdistämällä signaali johonkin ohjelmassa olevaan funktioon tai metodiin. Signaalit yhdistetään yleensä metodeihin ennen käyttöliittymän käynnistämistä. Yhteyksiä voidaan kuitenkin katkaista ja muodostaa myös käyttöliittymän ollessa käynnissä. Tätä voidaan käyttää esimerkiksi kun halutaan vaihtaa painikkeeseen yhdistettyä toimintoa valintalaatikon valinnan perusteella.

1.3.3 Teemat

Teema on kirjastosta erillinen kokoelma asetuksia ja mahdollisesti ohjelmakoodia, jotka muuttavat kirjastolla toteutettujen ohjelmistojen ulkonäköä. Teemat määrittelevät yleensä seuraavia asioita:

- Fontin ja sen koon
- Komponenttien värit
- Komponenttien muodon

Eri kirjastot toteuttavat teemat omalla tavallaan. Gtk+:-kirjastossa teemat koostuvat konfiguraatiotiedostosta ja piirtomoottorista(en. Drawing engine, ohjelmakoodi joka piirtää komponentit). Qt-kirjastossa teemat perivät QStyle luokan jossa määritellään miten eri komponentit piirretään. Kirjastot käyttävät komponenttien piirtämiseen piirtokirjastoja. Piirtokirjastot sisältävät peruspiirtorutiinit(esim. eri muotojen piirtäminen näytölle) ja niissä olevan alustakohtaisen koodin. Qt:n piirtokirjaston nimi on arthur ja Gtk+:n Gdk.

1.3.4 Asettelu

Kirjastoissa on myös työkalut komponenttien asettelua varten. Yleisimpiä asetteluluokkia ovat horisontaalinen, vertikaalinen ja ruudukkoasettelu. Horisontaalisessa asettelussa komponentin sisällä olevat komponentit järjestetään vasemmalta oikealle ja vertikaalisessa ylhäältä alaspäin. Ruudukkoasettelussa ensin määritellään rivien ja sarakkeiden määrä, ja asetellaan komponentit taulukon koordinaattien avulla.

Monille kirjastoille on myös graafinen työkalu käyttöliittymän suunnittelua varten. Näissä työkaluissa käyttäjä voi asetella käyttöliittymäkomponentteja paletista ikkunoihin miten haluaa, tai käyttämällä eri asetteluja. Tässä projektissa asettelujen käyttö on rajoittuu sivuihin, joita ei generoida konfiguraatiosta.

1.3.5 Tekstikenttä

Tekstikenttä on yhden rivin korkuinen tekstinsyöttökomponentti. Sitä käytetään yleensä lomakkeissa. Nykyaikaisissa kirjastoissa tekstikentissä on useita ominaisuuksia ja käyttötarkoituksia. Tekstikenttää voidaan käyttää mm. salasanan syöttöön jolloin käyttäjän kirjoittamaa tekstiä ei näytetä.

Syötettävä teksti voidaan myös pakottaa johonkin tiettyyn muottiin. Esimerkiksi tietyn pituiseksi tai siten että se voi sisältää vain numeroita.

1.3.6 Valintalaatikko

Valintalaatikko (en. Combobox) on yleensä toteutettu tekstikenttäkomponenttina, johon on lisätty pudotusvalikko. Pudotusvalikko avataan laatikon vasemmassa reunassa olevasta painikkeesta. Pudotusvalikon elementit määritellään listana jota voidaan muokata myös ajon aikana. Vaihtoehdot voidaan lukita tai käyttäjän voidaan antaa kirjoittaa omia. Tässä projektissa valintalaatikoita on pääasiassa käytetty järjestelmäasetussivuilla.

2 KIRJASTOJEN ARVIOINTI

Kirjastojen arvioinnissa kiinnitettiin erityisesti huomiota kirjastojen yleisyyteen, ulkonäköön, ulkonäön muokattavuuteen ja näppäinoikotieominaisuuksiin. Ulkonäön tulee olla riittävän moderni. Kirjaston pitää noudattaa olio-ohjelmoinnin periaatteita. Komponenttien ulkonäön muokkaamisen tulee olla yksinkertaista ja helppoa.

2.1 Qt

Tässä osiossa käsitellään Qt:n ominaisuuksia ja niiden soveltamista käytäntöön.

2.1.1 Qt:n signaalit

Qt lisää C++ ohjelmointikieleen kielioppia signaalimekanismiaan varten. Lähetettävät signaalit määritellään signaaleiksi. Signaaleja vastaanottavat metodit määritellään sloteiksi. Slot metodia kutsutaan kun siihen yhdistetty signaali lähetetään. Yhteen slottiin voidaan yhdistää monia signaaleja. Signaaleiksi määritellyt metodit määritellään luokassa mutta niille ei kirjoiteta toteutuksia. Sloteiksi määritellyt metodit täytyy toteuttaa. Luokissa joissa käytetään Qt:n signaaleja ei voi käyttää templateja.

Esimerkkikoodi luokasta jossa määritellään signaali ja slot.

```
class MyWidget : public QWidget {

    Q_OBJECT

public:

    MyWidget(QWidget *parent = 0);

    ~MyWidget();

public slots:
```

```
    void MySlot(int n);
signals:
    void MySignal(int n);
}
```

Signaalien mukana voidaan lähettää dataa antamalla signaalille argumentteja. Signaalilla ja slotilla täytyy olla samantyyppiset argumentit, jotta vastaanottaja ymmärtäisi datan. Argumenttien yhteensopivuus tarkistetaan ajon aikana. Jos tarkistus epäonnistuu yhteyttä ei muodosteta.

Signaalien yhdistäminen tapahtuu connect funktion avulla. Argumentteina sille annetaan signaalin lähettäjä, lähettäjässä määritelty signaali, signaalin vastaanottaja ja vastaanottajassa määritelty slot metodi. Esimerkki connect -funktiokutsusta:

```
connect(mywidget, SIGNAL(MySignal(int)), mywidget,
SLOT(MySlot(int)));
```

Signaalin lähettäminen tehdään ohjelmakoodissa näin:

```
emit MySignal(i);
```

Kun signaali lähetetään näin on lähettäjänä olio jonka metodissa koodi on. Signaalille annettu argumentti muuttuja ”i” välitetään kaikille signaalin vastaanottaville slot metodeille argumenttina.

2.1.2 Qt ulkonäön räätälöinti

Qt:n teemat toteuttavat QStyle luokan eli sen avulla voidaan määritellä koko sovelluksen tyyli. Komponenteille voidaan asettaa QStyle, mutta se ei ole suositeltavaa.

Qt:ssa jokaisella komponentilla on palette() (Qt 4.7: Qwidget Class Reference) metodi joka palauttaa QPalette olion. Tämä olio määrittelee eri värit joita komponentti käyttää. Värien sijasta voidaan myös käyttää kuvia tai liukuvärejä.

Qt tyylit tuo Qt:ssa jo olemassa olevat ulkonäön muokkausominaisuudet helpommin saataville yksinkertaisen kielen kautta. Sillä voi määritellä ulkonäön jokaiselle komponentille erikseen tai yhdelle suuremmalle komponenttikokonaisuudelle. Tyyleillä on helppo testata erilaisia ulkonäköjä.

Esimerkkinä Qt tyyli joka asettaa tyylin kaikille QPushButton komponenteille.

```
QPushButton {

    color: white;

    background: black;

}
```

Jos tämä Qt tyyli asetettaisiin sovelluksen päätyyliksi, asettaisi se kaikkien QPushButton painikkeiden tekstin valkoiseksi ja taustan mustaksi.

Kuvio 2: Vasen painike ilman tyyliä, oikeassa käytetään yllä määriteltyä tyyliä

2.1.3 Qt:n näppäinoikotieominaisuudet

Näppäinoikoteiden avulla saadaan näppäimistön näppäimet lähettämään signaaleja. Yksi näppäinoikotie vastaa yhtä näppäintä. Qt:ssa voidaan määritellä näppäinoikotiet QShortcut luokan avulla. Näppäinoikotie voidaan määritellä merkkijonona (Esim. ”Shift+Tab”).

Näppäinoikotiet voidaan asettaa toistaviksi tai ei toistaviksi. Toistava toistaa activated() (Qt 4.7.1: Qshortcut Class Reference) signaaliaan niin kauan kuin näppäintä pidetään pohjassa. Ei toistava lähettää signaalinsa vain kerran näppäintä painettaessa.

2.2 Gtkmm

Tässä osassa käsitellään gtkmm-kirjaston ominaisuuksia ja miten niitä käytetään käytännössä.

2.2.1 Gtkmm-kirjaston signaalit

Gtkmm käyttää alun perin sitä varten kehitettyä libsigc++-kirjastoa signaaleja varten. Libsigc++-kirjasto ei käytä minkäänlaista C++:n lisättyä syntaksia, joten se on paremmin yhteensopiva uusien ohjelmointikieliominaisuuksien kanssa.

Gtkmm-kirjastossa yhdistetään signaalit ohjelmakoodissa tällä tavalla:

```
button.signal_clicked().connect(sigc::ptr::fun(&on_button_clicked)
);
```

Tämä yhdistää painikkeen(button) signaalin (signal_clicked) funktioon on_button_clicked.

2.2.2 Gtk+ ulkonäön räätälöinti

GTK+:n ulkonäön muokkaus tapahtuu teemojen avulla. GTK+ teemat koostuvat gtkrc-tiedostosta ja piirtomoottorista. Moottori on ohjelmakoodi jossa määritellään eri komponenttien muodot. Yhden moottoriin voidaan perustaa monia teemoja luomalla uusia gtkrc-tiedostoja. Gtkrc-tiedostossa määritellään komponenttien fontit, värit ja muut ominaisuudet. Piirtomoottorit voivat määritellä lisäparametreja komponenttien ulkonäölle.

2.2.3 Gtkmm näppäinoikotiet

Gtk+:ssa komponenteille lisätään oikoteitä add_accelerator (gtkmm: Gtk::Widget Class Reference) metodilla. Niitä voidaan antaa ikkunalle tai tietyille komponenteille.

Mikäli oikotielle määritellään polku(esim. "<MainWindow>/File/Open"), silloin

loppukäyttäjä voi asettaa näppäimen itse.

3 KÄYTETYT TEKNOLOGIAT

Projektin käyttöliittymäkirjastoksi valittiin Qt. Qt:ssa ulkonäön muokkaaminen on helppoa Qt tyylien avulla. Gtk:n ulkonäön räätälöinti on rajallista ilman koodin kirjoittamista. Wapice Oy:llä on myös enemmän kokemusta Qt:sta.

3.1 Kehitysympäristö

Koska ohjelmisto käyttää samaa Qt versiota joka on käytössä työasemissa voidaan käyttöliittymää myös kehittää työasemassa. Ohjelmiston testatusta sulautetussa laitteessa varten on palvelimella käännösympäristö valmiina. Qt:n komponentteja käytetään työssä muuttamalla niiden näppäinoikoteitä ja ulkonäköä alustalle sopiviksi.

3.1.1 Qt:n kehitystyökalut

Joitakin Qt:n liittyviä työkaluja :

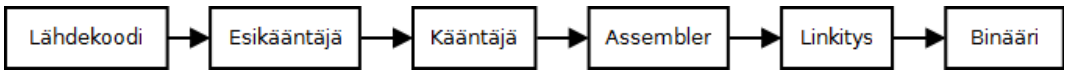
- qmake - käännösprosessin ohjaustyökalu (Build tool)
- Qt designer - käyttöliittymän suunnittelutyökalu
- Qt Linguist - Käännöstyökalu

Qt Designer on graafinen työkalu käyttöliittymän suunnitteluun. Käyttäjä valitsee paletista komponentteja ja asettelee ne haluamallaan tavalla ikkunaan. Qt designeria käytetään tässä projektissa joidenkin asetussivujen toteuttamiseen.

Qt Linguist on työkalu jolla ohjelmisto käännetään eri kielille. Ohjelmisto poimii lähdekoodista kaikki merkkijonot käännettäviksi. Kun näin käännettyä ohjelmaa ajetaan käyttöjärjestelmä kertoo sille mitä kieltä halutaan käyttää.

Qt lisää C++:n joitakin ominaisuuksia joita siinä ei normaalisti ole. Tämän takia ohjelman kääntäminen tapahtuu hieman eri tavalla. Normaalisti C++ ohjelman

käännösprosessi menee näin:



Kuva 2: C++ käännösprosessi

Koska Qt lisää C++:n omaa syntaksiansa täytyy prosessiin lisätä yksi vaihe lisää. Ennen esikäännöstä ajetaan Qt:n oma esikääntäjä joka muuntaa Qt:n oman syntaksin normaalin esikääntäjän ymmärtämään muotoon.

3.2 CMake

CMake on alustariippumaton ohjelman kääntämisen ohjaustyökalu. CMake:n kehitys aloitettiin vuonna 2000 ja nykyinen versio on 2.8.3. Se pystyy luomaan käännöstiedostot Windows tai Unix ympäristöissä samasta konfiguraatiosta monelle eri kehitystyökalulle.

QMakea ei käytetä tässä projektissa koska vain käyttöliittymä käyttää Qt:ta. CMake soveltuu paremmin koko projektin kääntämiseen. Konfiguraatiotiedostossa täytyy erotella lähdekooditiedostot joissa on QObject luokasta periytyviä luokkia. Ne annetaan Qt:n esikääntäjälle ennen kääntämistä. CMake ohjaa myös Qt Designerilla luotujen käyttöliittymien integroimista ohjelmistoon.

4 KÄYTTÖLIITTYMÄN MÄÄRITTELY

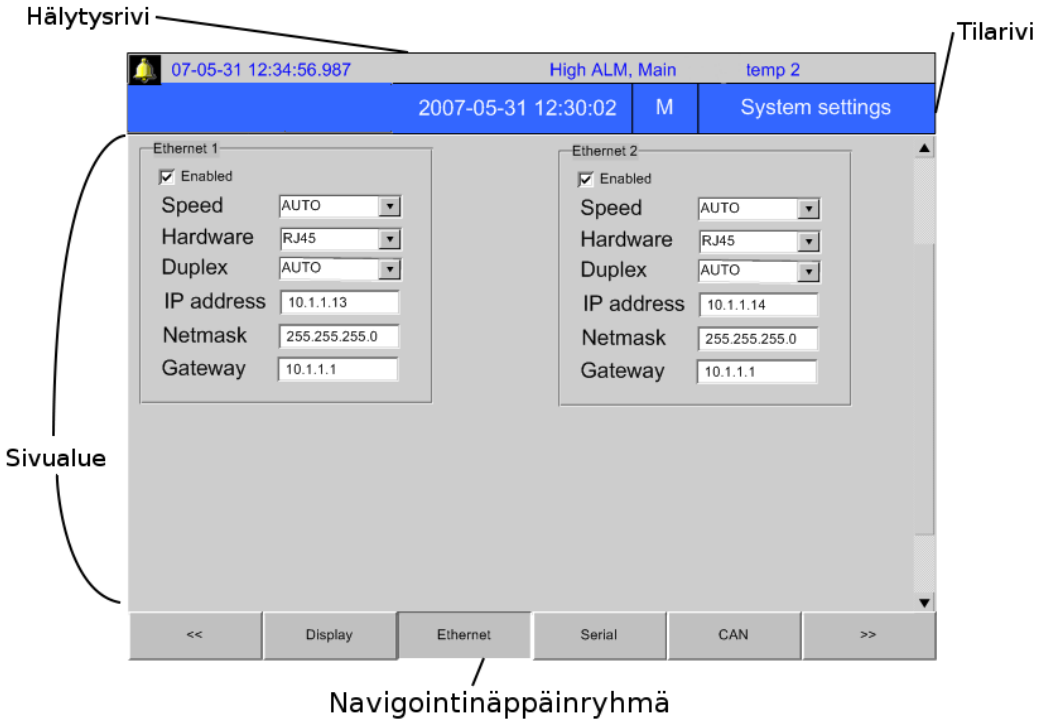
Käyttöliittymä ladataan PC:llä tehdystä konfiguraatiosta. Konfiguraatio luodaan Wapice Oy:n toteuttamalla konfiguraatiotyökalulla. Työkalulla luodaan tämän projektin ohjelmistoa varten konfiguraatio. Konfiguraatiosta saadaan tarvittavat tiedot käyttöliittymän rakentamiseen ja eri datalähteiden lukemiseen.

4.1 Komponentit

Koska ohjelmaa ajetaan sulautetussa laitteessa täytyy kirjastossa olevista komponenteista tehdä laitteella käytettäviä. Käyttöliittymäkonfiguraatiossa on määritelty useita komponenttityyppejä. Jotkin komponentit täytyy toteuttaa itse ja toiset voidaan toteuttaa muokkaamalla käyttöliittymäkirjastosta löytyviä vakiokomponentteja.

PC-konfiguraatiotyökalu ja laitteessa ajettava käyttöliittymä on kehitetty samassa projektissa. Joten konfiguraatiosta ei löydy komponentteja joita käyttöliittymä ei tue.

4.2 Pääikkuna



Kuvio 3: Pääikkuna

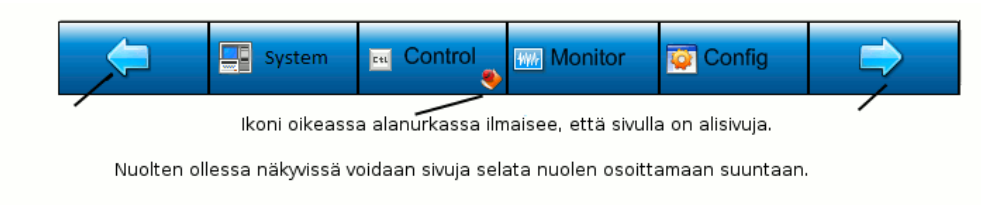
Käyttöliittymän pääikkunan keskeisimmät osat ovat sivualue ja painikeryhmä. Sivualue näyttää konfiguraatiossa määritellyt sivut ja painikeryhmän avulla käyttäjä voi käyttää komponenttien ominaisuuksia. Valittua komponenttia eli fokusta vaihdetaan pyörittämällä rullaa ja komponentti aktivoidaan painamalla rullaa. Käyttöliittymän muut osat ovat näytön yläosassa oleva hälytysrivi ja statusrivi. Hälytysrivillä näkyy uusin hälytys ja tilarivillä näytetään kellonaika, tämän hetkisen konfiguraation nimi, käyttäjätaso ja tällä hetkellä avatun sivun nimi. Sivualueen alapuolella on painikeryhmä jonka painikkeet vastaavat näytön alapuolella olevia painikkeita.

4.3 Sivut

Suurimman osan näyttötilasta käyttää alue jossa näytetään auki oleva sivu. Sivuja on kahta eri tyyppiä: Sivuja joiden sisältö on täysin konfiguroitavissa ja sivuja joille on määritelty rooli.

Sivua voidaan vaihtaa painikkeiden avulla tai puukomponentista. Sivut luetaan ohjelman käynnistyessä konfiguraatiosta. Sivut on ryhmitelty puurakenteeseen. Esimerkiksi asetussivut ovat samassa ryhmässä.

Näytöllä olevaa sivua vastaava painike on korostettu painikerivissä. Painikkeet vastaavat niitä sivuja, jotka ovat tällä hetkellä valitun sivun lapsia tai samalla tasolla olevia sivuja mikäli valitulla sivulla ei ole lapsia.



Kuvio 4: Navigointipainikkeet

4.4 Painikkeet

Laitteessa näytön alapuolella sijaitsevia kuutta nappia käytetään näytöllä olevien painikkeiden aktivointiin. Nappien toiminnallisuus on kontekstiriippuvaista. Esimerkiksi tekstikenttää muokattaessa napeista syötetään kirjaimia tai selatessa sivuja napeista voidaan vaihtaa näkyvää sivua.

4.4.1 Navigointipainikkeet

Tämä painikeryhmä on käytössä ohjelman käynnistyessä. Navigointipainikeryhmä avulla liikutaan sivuhierarkiassa. Painiketta painettaessa näytetään näytöllä olevan sivun lapsisivut. Jos sivulla ei ole lapsia näytetään ylemmän tason sivut. Vasemmalla nuolipainikkeella liikutaan puussa ylöspäin mikäli painikkeet on selattu al-

kuun. Jos sivulla on lapsia, näytetään napeissa oikeassa alareunassa ikoni.

4.4.2 Tekstinsyöttöpainikkeet

Tekstikenttiin tekstin syöttäminen tapahtuu tällä painikeryhmällä. Syöttämiseen käytettävät painikkeet toimivat samaan tapaan kuin kännykän näppäimistö, eli painikkeissa on valittavana esimerkiksi 3 numeroa(esim. 123). Mikäli samaa painiketta painetaan toisen kerran vaihtuu kursorin kohdalla oleva merkki.

Tässä painikeryhmässä ei käytetä selausominaisuuksia. Painikeryhmän ensimmäinen painike vaihtaa muista painikkeista syötettävät merkit tietyn tyyppisiksi. Tyyppeinä käytetään aakkosia, numeroita ja erikoismerkkejä. Saatavilla olevia merkkityyppejä voidaan rajoittaa tekstikentän käyttötarkoituksen mukaisesti.

4.5 Tila- ja hälytysrivit

Näytön yläreunassa on tilarivi, jossa näkyy tämän hetkinen käyttäjätaso, kello ja näkyvissä olevan sivun nimi. Tilarivin yläpuolella on hälytysrivi jossa näkyy uusin hälytys.

4.6 Muokattava tekstikenttä

Muokattava tekstikenttä on komponentti, jossa on yksi rivi tekstiä. Käyttäjä voi kirjoittaa siihen vapaasti. Tekstikentän muokkaaminen aloitetaan valitsemalla kenttä ja painamalla laitteen rullaa. Tämän jälkeen painikeryhmä vaihtuu muokkausriviksi. Rullasta siirretään kursoria ja näytön alapuolella olevasta painikeryhmästä valitaan syötettävä merkki.

Tekstikentistä voidaan tehdä räätälöityjä versioita eri tarkoituksiin. Esimerkiksi tekstikentän hyväksymiä merkkejä voidaan rajoittaa tai rajata teksti tietyn pituiseksi. Näitä ominaisuuksia on käytetty mm. IPv4 osoitteen muokkaamiseen tarkoitettujen tekstikenttien yhteydessä.

IPv4-osoite tekstikenttä on jaettu neljään osaan. Osoitteen jokaista tavua vastaa-

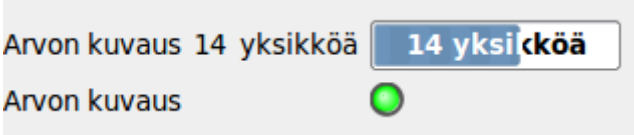
vaan. Jokainen osa toimii kuten yksittäinen tekstikenttä. Jokainen yksittäinen osoitteen osa hyväksyy vain arvoja välillä 0-255.

4.7 Spinbox

Spinbox on numeroarvon syöttämiseen tarkoitettu komponentti joka koostuu tekstikentästä ja kahdesta painikkeesta. Spinbox pitää ensin aktivoida painamalla rullaa sen kohdalla ja tämän jälkeen arvoa voidaan muokata pyörittämällä rullaa. Myötäpäivään pyörittäminen kasvattaa ja vastapäivään pyörittäminen pienentää numeroarvoa. Arvo muuttuu nopeammin mikäli pyöritys on yhtäjaksoista.

4.8 Rivikomponentti

Rivikomponentti on taulukon näköinen komponentti jossa on listattuna prosessiarvoja. Rivejä voidaan valita pyörittämällä rullaa. Valitulla rivillä on musta reuna. Painamalla rullaa kun rivi on valittuna saadaan näkyviin ponnahdusikkuna jossa on tarkempia tietoja arvosta sekä sen raja-arvot.



Kuvio 5: Rivikomponentti, jossa sensoririvi ylhäällä ja merkkivalorivi alhaalla

4.8.1 Merkkivalorivi

Rivikomponentin merkkivalorivi esittää on/off arvon merkkivalona. Mikäli merkkivalo on päällä sensoriarvo on totta ja mikäli ei se on epätosi. Rivillä on myös arvon kuvaus.

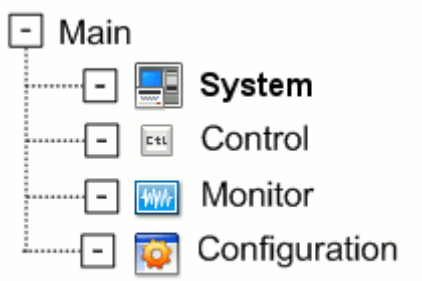
4.8.2 Sensoririvi

Rivikomponentin sensoririvi sisältää kuvauksen arvosta, sensorin arvon, yksikön

ja arvon pylväsgraafina. Pylväsgraafin taustaväri muuttuu mikäli arvo on asetettujen rajojen ulkopuolella.

4.9 Navigointipuu

Navigointipuu on tälle ohjelmalle keskeinen komponentti, koska siihen on sijoitettu kaikki valittavat sivut. Navigointipuun saa näkyviin pitämällä rullaa pohjassa 2 sekuntia. Puun jäsen valitaan rullaa pyörittämällä ja rullasta painamalla valittu sivu avataan. Painamalla laitteen oikeassa alanurkassa olevaa painiketta näytetään jäsenen lapset.



Kuvio 6: Navigointipuu

5 TOTEUTUS

Ohjelmisto on jaettu kahteen osaan: käyttöliittymään ja domainiin. Käyttöliittymä osassa käytetään Qt:ta ja sen käyttöliittymätoimintoja. Domainissa Qt kirjastoja ei käytetä. Koska projekti aloitettettaessa Qt:a ei vielä saanut GNU LGPL lisensoituna.

Käyttöliittymä ladataan PC:llä tehdystä konfiguraatiosta. Käyttöliittymä saa tarvittavat tiedot käyttöliittymän rakentamiseen domainilta tietorakenteina.

5.1.1 Domain ja konfiguraatio

Suurin osa käyttöliittymästä luetaan konfiguraatiosta. Konfiguraatiossa on määritelty eri sivut joita käyttöliittymässä näytetään ja komponenttien sijainnit. Sivuilla on myös käyttöoikeudet. Suurin osa määrittelyistä on mittausarvoja ja niiden esitystapa. Konfiguraation luku ja datarakenteet on määritelty domainissa. Konfiguraation tallennusmuoto ja sen lukeminen eivät kuulu tämän työn piiriin.

Ohjelman käynnistyessä konfiguraatio luetaan muistiin ja esitetään ohjelmalle Configuration luokan instanssina. Configuration luokka sisältää pääikkunan komponenttien asettelun, tyylimäärittelyt ja sivut. Yksittäisen sivun asetukset esitetään PageData luokan avulla. Yksi PageData olio sisältää lapsisivujen PageData oliot, komponenttimäärittelyt ja asetussivudatan.

Domainissa on apuluokkia, joita sivukomponentit käyttävät järjestelmän asetusten muokkaamisen.

5.2 Pääikkuna

Kun ohjelma käynnistetään näytetään pääikkuna ja siinä avattuna konfiguraatiossa määritelty kotisivu. Pääikkuna on toteutettu MainWindow luokkana, joka perustuu QWidget luokkaan. Pääikkunassa luodaan instanssit käyttöliittymän pääkomponenteista. Jäseninä mm. ButtonBoxDirector, PageArea ja StatusRow. Tässä luo-

kassa yhdistetään signaalit, joiden avulla eri komponentit viestivät keskenään.

5.3 Käyttöliittymän kytkeminen laitteeseen

Laitteen painikkeiden ja rullan avulla luodaan virtuaalinen näppäimistö. Rullan myötäpäivään pyörittäminen vastaa sarkain näppäintä ja vastapäivään pyöritys vaihto+sarkain painallusta. Koska näillä näppäimillä valitaan komponentti PC sovelluksissa, ei sovellukselle tarvitse kertoa näitä näppäinoikoteitä erikseen. Näytön alapuolella olevat painikkeet taas vastaavat funktionäppäimiä. Funktionäppäimet on kytketty painikkeisiin QShortcut luokan avulla.

Rullan painaminen lähettää rivinvaihdon. Komponentteja joiden sisältöä käyttäjä voi muokata on useita. Näissä komponenteissa rivinvaihtoa käytetään muokkaamisen aloittamista ja lopettamista varten. Tähän käytetään QShortcut luokkaa. QShortcutin ominaisuuksiin kuuluu, että samaan aikaan ei voi olla kahta eri QShortcut oliota jotka vastaavat samaa näppäintä aktiivisena. Jos kaksi samaa näppäintä vastaavaa QShortcut oliota ovat aktiivisena samaan aikaan kumpikaan ei ota vastaan painallusta. Tämä voisi aiheuttaa ongelman mikäli samalla sivulla olisi kaksi komponenttia, jotka käyttävät samaa näppäintä. Ongelma voidaan välttää asettamalla QShortcut olio tilaan, jossa se on aktiivinen vain silloin kun komponentti on valittuna.

5.4 Sivut

Suurin osa käyttöliittymän toiminnallisuudesta on sivuilla. Sivuja on kahta päätyyppiä. Komponenteista koostuvia ja esimääriteltyjä.

Sivujen konfiguraatiossa esimääritellyt sivut on määritelty vain sivutyyppinä. Esimääriteltyjä sivuja käytetään pääasiassa laitteen asetusten muokkaamiseen(mm. verkko-, aika- kieliasetuksien).

Komponenteista koostuvat komponentit rakennetaan yhdeksi komponentiksi ja näytetään käyttäjälle. Komponenteista koostuvia sivuja pääasiassa sensoriarvojen

esittämiseen.

5.4.1 Sivujen rakentaminen

Sivut avataan puulta tulevan signaalin seurauksena PageArea komponentissa joka osaa näyttää sivun konfiguraatiodatan perusteella. Kun sivua vaihdetaan edellisen sivun komponentit häviävät muistista. Komponentit luodaan aina uudestaan kun sivu avataan. Samaa sivua ei kuitenkaan voi avata kahta kertaa peräkkäin.

PageArea komponentissa oleva PageBuilder luokka luo domainin tarjoamista datarakenteista niitä vastaavia käyttöliittymäelementtejä. PageBuilder luokassa on jäseninä ComponentBuilder ja FixedPageBuilder oliot. ComponentBuilder luokka rakentaa sivulla olevat komponentista. FixedPageBuilder päättelee mikä sivukomponentti täytyy rakentaa sille annetusta sivusta.

Komponenteista koostuvista sivuista rakennetaan sivulla olevia komponentteja vastaava QWidget. Mikäli sivulla on enemmän kuin 1 lapsikomponentti, luodaan CompositeComponent(QWidget) jonka lapsikomponenteiksi kaikki sivun käyttöliittymäkomponentit luodaan.

5.4.2 Sivukomponentit

Sivukomponentit ovat sivuja joiden sisältö on esimääritelty. Esim. verkkoasetus-, yksikköjärjestelmä- ja lokisivut on toteutettu sivukomponentteina. Asetussivuilla on domainissa määritelty mitä arvoja niillä on ja miten ne tallennetaan tai otetaan käyttöön.

5.5 Komponentit

Komponentit esitetään konfiguraatiolle ComponentData luokkaan perustuvina olioina. ComponentData luokassa on määritelty komponenteille seuraavia parametreja.

- datalähde
- koko
- sijainti
- tekstin väri
- taustakuva tai taustaväri

Komponenttien sijainnin ja koon määrittämiseen käytetään PositionData luokkaa. DataSource(datalähde) luokka on rajapinta sensoriarvon lukemiseen. Se sisältää myös raja-arvot.

Komponenttien geometria määritellään suhteessa koko ruutuun. Qt:ssa taas komponenttien sijainnit ovat aina suhteessa isäntäkomponentin sijaintiin. Tämän takia syvemmällä komponenttihierarkiassa olevia komponentteja luotaessa täytyy olla tiedossa siirtymä ikkunan vasemmasta yläkulmasta. Siirtymä annetaan PageBuilder oliolle kun ohjelma käynnistetään.

5.6 Painikeryhmät

Painikeryhmä on yksi projektin monimutkaisimmista komponenttityypeistä. Painikeryhmiä on kahden tyyppisiä; listaa esittäviä ja tarkoitukseen räätälöityjä.

Painikeryhmiä joissa tieto on listamuodossa on toteutettu tiedon selaus. Selaaminen tapahtuu käyttämällä painikeryhmän vasenta ja oikeaa painiketta. Näiden painikkeiden ikoneiksi asetetaan nuolet. Kun selaus nuolen osoittamaan suuntaan ei ole mahdollista nuoli piilotetaan ja sen tilalla näytetään ensimmäinen tai viimeinen tieto.

Listapohjaisissa painikeryhmissä voidaan päättää jääkö viimeksi painettu painike pohjaan. Tätä käytetään mm. navigointipainikkeissa ilmaisemaan auki olevaa sivua.

5.6.1 Painikeryhmien hallinta

Painikeryhmiä hallitsee ButtonBoxDirector luokka. Se vaihtaa näkyvissä olevaa painikeryhmää tarpeen mukaan ja huolehtii että vain yksi painikeryhmä on näkyvissä kerrallaan. Kun näkyvissä olevaa painikeryhmää vaihdetaan täytyy ButtonBoxDirectorin layout luoda uudestaan. Muuten painikeryhmä piirretään väärin.

Seuraava ohjelmakoodi on otettu ButtonBoxDirectorin Editing(IEditable* sender, bool bEditing) metodista. Koodissa pyydetään editoitavalta komponetilta nappirivi ja näytetään se.

```
if(bEditing){
      m_CurrentEdit = pEditable->GetButtonBox();
      if (m_CurrentEdit != NULL) {
      // If we don't create a new layout we won't be able to
remove the old
      // buttonbox from the layout if the page is changed while
the edit buttonbox is visible
      m_CurrentEdit->setFocusPolicy(Qt::NoFocus);
      QVBoxLayout * editLayout = new QVBoxLayout;
      editLayout->setSpacing(0);
      editLayout->setContentsMargins(0,0,0,0);
      m_CurrentEdit->SetConfig(&GetButtonBoxData());
      m_CurrentEdit->setParent(this);
      editLayout->addWidget(m_CurrentEdit);
      //layout()->addWidget(m_CurrentEdit);
      if(m_CurrentGlobal != NULL) {
                  m_CurrentGlobal->hide();
      }
      delete layout(); // Need to remove this before we can
                              //install the new one
      setLayout(editLayout);
      m_CurrentEdit->show();
}
```

5.6.2 Komponenttien painikeryhmät

Muokattavat komponentit toteuttavat IEditable rajapinnan. Rajapinta määrittelee metodin ButtonBox *GetButtonBox(), joka palauttaa painikeryhmän jolla komponenttia muokataan. ButtonBoxDirector vapauttaa painikeryhmän muistin kun komponentin muokkaus lopetetaan. Painikkeiden kautta annettu syöte annetaan muokattavalle komponentille rajapinnan kautta.

5.6.3 Globaalit nappirivit

Globaalit nappirivit on määritelty ButtonBoxDirector luokassa jäsenmuuttujina, joten ne tuhotaan vasta kun ohjelma suljetaan. Suurimman osan ajasta näkyvissä on navigointinappirivi ja joillakin sivuilla on omat nappirivinsä.

5.7 Navigointi

Sivujen navigointi perustuu puukomponenttiin. Kun käyttäjä valitsee puukomponentista elementin. Puukomponentti lähettää signaalin sivualueelle ja navigointinappiriville. Sivualue rakentaa ja näyttää sivun jonka jälkeen navigointinappirivi päivitetään.

5.7.1 Navigointipuu

Puukomponentti perustuu QTreeWidget luokkaan. Navigointipuu toteutetaan NavigationTreeWidget luokassa. Tässä luokassa määritellään puukomponentissa käytetyt näppäinoikotiet ja puun rakentaminen datan perusteella. Yhdistetty signaaleilla NavigationButtons ja PageArea luokkiin. Signaalit yhdistetään pääikkunassa. Navigointipuuta esitetään omassa dialogissaan. Dialogi näytetään ja piilotetaan pääikkunassa määritetyllä tavalla.

Puu rakentuu konfiguraatiossa olevasta puurakenteesta. Jokaista sivua vastaa NavigationTreeItem, jonka jäsenenä on domainista saatu sivudata. Sivudata esitetään PageData luokan avulla. NavigationTreeItem on räätälöity QTreeWidgetItem

luokkaan perustuva luokka.

Konfiguraatiossa on määritelty kotisivu, joka avataan kun ohjelma käynnistyy. Tämä toiminto täytyi siirtää pääikkunan showEvent(3 Qt dokumentaatio) funktioon koska muuten sivulle olisi yritetty siirtyä ennen kuin puu on rakennettu.

5.7.2 Navigointipainikeryhmä

Painikkeet joita käytetään navigointiin. Nämä painikkeet ovat näkyvissä suurimman osan ajasta. Navigointipainikeryhmä toteutetaan NavigationButtons luokassa.

Navigointipainikeryhmällä on datanaan lista NavigationTreeItem olioita. Lista sisältää sillä hetkellä valitun jäsenen lapset tai samalla tasolla olevat jäsenet. Lista päivitetään sivua vaihdettaessa vastaanotettavan signaalin perusteella.

Ryhmän yksittäisellä painikkeella on tekstinään sitä vastaavan sivun otsikko. Jos sivulla on lapsisivuja näytetään painikkeen oikeassa alanurkassa ikoni. Viimeksi valittua sivua vastaava painike pysyy pohjassa(kuten ”Ethernet” painike kuvassa Kuvio 3).

5.7.3 Nappirivin ja puun integrointi

Nappirivi ja puu on yhdistetty signaaleilla siten että molemmat ovat aina samassa tilassa. Sivua vaihdetaan vain silloin kun puun valinta muuttuu. Sivun valitseminen painikeryhmästä aiheuttaa sen että puun valinta asetetaan. Mikä johtaa siihen, että sivu vaihdetaan. Liite 2 esittää nappirivin päivityksen sekvenssikaaviona.

5.8 Tekstikenttä ja muokattavat komponentit

Editoitavalle komponentille luodaan aina uusi painikeryhmä kun muokkaus aloitetaan. Kun komponenttia muokkaaminen aloitetaan, pyytää ButtonBoxDirector uutta nappiriviä komponentilta. Muokattavat komponentit toteuttavat IEditable rajapinnan. Esimerkki muokattavasta komponentista on muokattava tekstikenttä, joka toteutetaan luokassa LineEdit.

5.8.1 LineEdit

Pohjaluokka muokattaville tekstikentille. Perustuu QLineEdit luokkaan ja se toteuttaa IEditable rajapinnan. Tässä luokassa toteutetaan tekstikenttien muokkaamiseen käytetyt näppäinoikotiet ja muokkaustilan päälle/pois mekanismi. Kun tekstikentän muokkaaminen aloitetaan tai lopetetaan lähetetään signaali ButtonBoxDirectorille.

LineEdit luokkaan perustuvia luokkia on luotu joihinkin erikoistapauksiin kuten IPv4 osoitteen muokkaamiseen.

6 TESTAUS

Komponenttien testaus tapahtuu käsin etukäteen laaditun testauskäytännön mukaisesti.

Navigointipainikeryhmää testattaessa tarvitaan konfiguraatio jossa on monimutkainen puu. Painikkeiden viimeisen painikkeen oikein piilottamista täytyy testata puun juuressa ja syvemmällä puussa. Kun nappitekstejä on enemmän, vähemmän tai saman verran kuin nappeja.

Muokattavia komponentteja testattaessa täytyy testata, että painikkeet tulevat näkyviin ja piilotetaan oikein. Puun näyttämistä pitää kokeilla kun komponentin sisältöä muokataan.

Listakomponentista täytyy testata seuraavia asioita: sarakkeiden leveydet menevät ovat konfiguraation mukaisia, valinnan ilmaiseva kehys toimii oikein. Sensoririvistä täytyy testata raja-arvot.

Sivujen rakentamista voidaan testata tekemällä konfiguraatio jossa on testattava sivu ja testattavat komponentit. Komponenttien sijoittamista täytyy testata sijoittamalla komponentteja sivun eri kohtiin.

7 JOHTOPÄÄTÖKSET

Käyttöliittymän testaaminen on työlästä käsin. Testaamiseen voisi käyttää jotain automaatiotyökalua. Domainin puolella voisi käyttää Qt:ta. Koska domain ei käytä Qt:ta joudutaan tekemään käyttöliittymän ja domainin rajapinnassa työläitä tyyppimuunnoksia. Painikeryhmien kehittämiseen käytettiin paljon aikaa. Painikeryhmien toiminta olisi voinut olla yksinkertaisempaa.

LÄHTEET

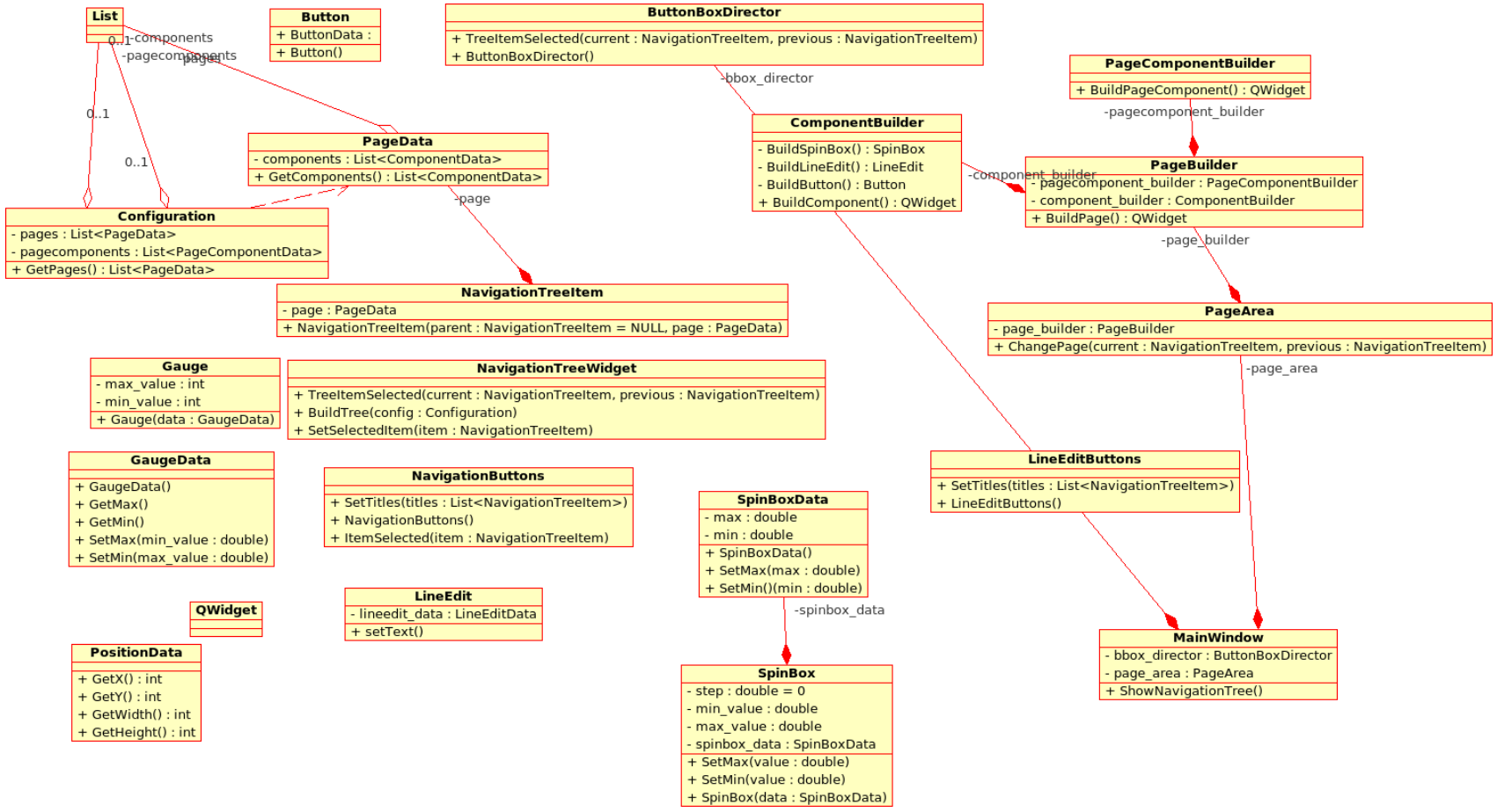
gtkmm: Gtk::Widget Class Reference [online]. [viitattu 30.11.2010]. Saatavilla www-muodossa:
<http://library.gnome.org/devel/gtkmm/2.22/classGtk_1_1Widget.html>

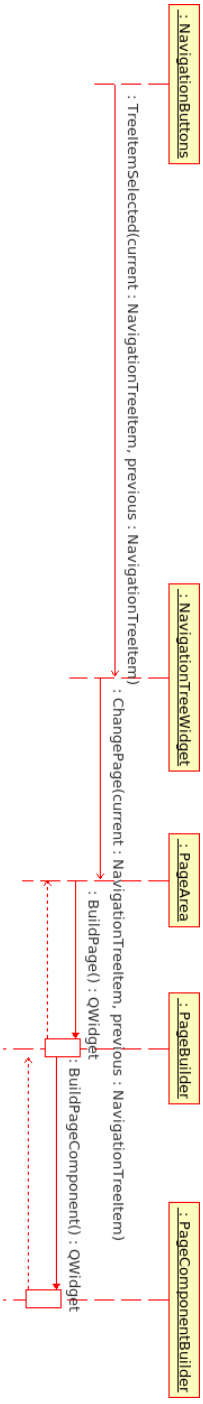
Qt 4.7: QWidget Class Reference [online]. [viitattu 29.11.2010]. Saatavilla www-muodossa: <http://doc.trolltech.com/main-snapshot/qwidget.html#palette-prop>

Qt 4.7.1: QShortcut Class Reference [online]. [viitattu 30.11.2010]. Saatavilla www-muodossa: <http://doc.qt.nokia.com/4.7/qshortcut.html#activated>

Wikipedia: GTK+ [online]. [viitattu 29.11.2010]. Saatavilla www-muodossa: <http://en.wikipedia.org/wiki/Gtk%2B>

Wikipedia: Qt (Framework) [online]. [viitattu 29.11.2010]. Saatavilla www-muodossa: <http://en.wikipedia.org/wiki/Qt_(toolkit)>


List
Button
+ ButtonData :
+ Button()
ButtonBoxDirector
+ TreeItemSelected(current : NavigationTreeItem, previous : NavigationTreeItem)
+ ButtonBoxDirector()
PageComponentBuilder
+ BuildPageComponent() : QWidget
-components
-pagecomponents
-page
0..1
-bbox_director
-pagecomponent_builder
PageData
- components : List<ComponentData>
+ GetComponents() : List<ComponentData>
ComponentBuilder
- BuildSpinBox() : SpinBox
- BuildLineEdit() : LineEdit
- BuildButton() : Button
+ BuildComponent() : QWidget
-component_builder
PageBuilder
- pagecomponent_builder : PageComponentBuilder
- component_builder : ComponentBuilder
+ BuildPage() : QWidget
-page_builder
Configuration
- pages : List<PageData>
- pagecomponents : List<PageComponentData>
+ GetPages() : List<PageData>
NavigationTreeItem
- page : PageData
+ NavigationTreeItem(parent : NavigationTreeItem = NULL, page : PageData)
PageArea
- page_builder : PageBuilder
+ ChangePage(current : NavigationTreeItem, previous : NavigationTreeItem)
-page_area
Gauge
- max_value : int
- min_value : int
+ Gauge(data : GaugeData)
NavigationTreeWidget
+ TreeItemSelected(current : NavigationTreeItem, previous : NavigationTreeItem)
+ BuildTree(config : Configuration)
+ SetSelectedItem(item : NavigationTreeItem)
GaugeData
+ GaugeData()
+ GetMax()
+ GetMin()
+ SetMax(min_value : double)
+ SetMin(max_value : double)
NavigationButtons
+ SetTitles(titles : List<NavigationTreeItem>)
+ NavigationButtons()
+ ItemSelected(item : NavigationTreeItem)
LineEditButtons
+ SetTitles(titles : List<NavigationTreeItem>)
+ LineEditButtons()
SpinBoxData
- max : double
- min : double
+ SpinBoxData()
+ SetMax(max : double)
+ SetMin()(min : double)
QWidget
LineEdit
- lineedit_data : LineEditData
+ setText()
-spinbox_data
PositionData
+ GetX() : int
+ GetY() : int
+ GetWidth() : int
+ GetHeight() : int
SpinBox
- step : double = 0
- min_value : double
- max_value : double
- spinbox_data : SpinBoxData
+ SetMax(value : double)
+ SetMin(value : double)
+ SpinBox(data : SpinBoxData)
MainWindow
- bbox_director : ButtonBoxDirector
- page_area : PageArea
+ ShowNavigationTree()


: NavigationButtons
: NavigationTreeWidget
: PageArea
: PageBuilder
: PageComponentBuilder
: TreeItemSelected(current : NavigationTreeItem, previous : NavigationTreeItem)
: ChangePage(current : NavigationTreeItem, previous : NavigationTreeItem)
: BuildPage() : QWidget
: BuildPageComponent() : QWidget