SYMBOLIKIRJASTOJEN LUONTI WINCC -YMPÄRISTÖÖN

Opinnäytetyö

Jaakko Tolonen

Lokakuu 2010

Automaatiotekniikka

Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) TOLONEN, Jaakko	Julkaisun laji Opinnäytetyö	Päivämäärä 12.10.2010
	Sivumäärä 41	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (x)
Työn nimi SYMBOLIKIRJASTOJEN LUONTI WINCC -YMPÄRISTÖÖN		
Koulutusohjelma Automaatiotekniikan koulutusohjelma		
Työn ohjaaja(t) HÄKKINEN, Veli-Matti		
Toimeksiantaja(t) JEEC Oy Jarmo Pekkanen		

Tiivistelmä

Työn tarkoituksena oli luoda teollisuuden komponentteja käsittävä symbolikirjasto WinCC – ohjelmistoon, tilaavan yrityksen sovellussuunnitteluvaiheen tehostamiseksi ja kustannussäästöjä silmällä pitäen. Tilaajana toimi jyväskyläläinen automaatioalan suunnitteluyritys JEEC Oy.

Symbolikirjastoon kuuluvat moottorit, venttiilit, mittaukset ja säätöpiirit. Objektit ovat Visual Basic -ohjelmoituja kokonaisuuksia ohjauksineen ja lukitusikkunoineen. Kirjaston tarkoitus on vähentää työmäärää WinCC -valvomoa suunniteltaessa ja rakennettaessa, sekä pienentää datansiirrossa käytettävien tagien määrää. Kirjaston avulla suunnittelija pystyy luomaan ohjattavia laitekokonaisuuksia automaatiojärjestelmien tapaan pelkästään laitepositioiden määrittelyllä.

Toimilohkot ja piiri-ikkunat todettiin testausvaiheessa toimiviksi ja selkeiksi loppukäyttäjää ajatellen. Kyseisellä toimilohkokirjastolla tagien määrä keskikokoisissa projekteissa putoaa huomattavasti tehokkaamman datankäsittelyn ansiosta.

Avoimen Visual Basic – koodin ansiosta kirjaston laajennettavuus ja kehitettävyys on lähes rajaton. Piirinäyttöihin voidaan lisätä enemmän prosessidataa asiakkaan toiveiden mukaan. Työhön sisältyvät Visual Basic – koodit antavat myös hyvän pohjan valvomosuunnittelun tehostamiselle jatkossa, kun esimerkiksi massatoimenpiteet suuremmissa valvomonäyttökokonaisuuksissa voidaan hoitaa yhdellä ohjelmakoodilla.

Avainsanat (asiasanat)

valvomosuunnittelu, sovellussuunnittelu, visual basic, toimilaite, kirjasto, wincc

Muut tiedot

Liitteenä esimerkit VBS koodeista, 16 sivua.

Author(s)
TOLONEN, Jaakko

Type of publication
Bachelor´s Thesis

Date
12102010

Pages
41

Language
Finnish

Confidential

( ) Until

Permission for web publication
(x)

Title
DESIGNING AND DEVELOPING A SYMBOL LIBRARY OF REGULATING UNITS AND INSTRUMENTS FOR WINCC ENVIRONMENT

Degree Programme
Automation technology

Tutor(s)
HÄKKINEN, Veli-Matti

Assigned by
JEEC Ltd.
Jarmo Pekkanen

Abstract

The purpose of thesis was to create a fully tested symbol library containing the most common components used in industry. The library is to be used with WinCC operating software. The main issue was to decrease the time used to build operating systems for projects and lower the costs caused by license payments. The thesis was assigned by JEEC Ltd, a company that offers automation and electrification designing services.

The library contains icons, control windows and loop windows for motors, valves, measures and controllers. The symbols are fully functional for handling interlocks, controlling and reading regulating unit states and driving modes. Visual Basic Scripting has been used for adding dynamics for symbols. The more efficient way of handling data between PLC and operating software reduces the amount of used tags and therefore lowers the costs.

The results were monitored by comparing the thesis to an older project in the company, with the assumption that the project would have been created using a symbol library. The amount of used tags would have dropped to half, which directly affects the costs.

Because of the open source code of symbols and control windows, developing and improving is easy. Designers can add project based information to the loop windows and make windows for controlling sequences.

Keywords
regulating unit, wincc, visual basic, application engineering, library

Miscellaneous
Examples of VBS codes attached, length 16 pages.

SISÄLTÖ

KUVIOT

1 TAVOITTEET JA TOIMEKSIANTAJA

1.1 Toimeksiantaja

Työn toimeksiantajana toimi jyväskyläläinen vuonna 2009 perustettu JEEC Oy, jonka toimialaan kuuluu automaatio- ja sähkösuunnittelu. Yrityksen ydinosaamiseen kuuluu automaatiosuunnittelu alalajeineen kuten esisuunnittelu, perussuunnittelu, instrumentointi- ja sähkösuunnittelu. Suunnittelun osaamisalaan kuuluvat sekä ohjelmoitavat logiikat, että eri valmistajien automaatiojärjestelmät. Teollisuudenaloista voidaan mainita mm. selluteollisuus, paperiteollisuus, kemia ja ympäristöteknologia sekä metalli- ja energiateollisuus.(JEEC Oy, 2010.)

1.2 Toiminnalliset tavoitteet

Automaation sovellussuunnittelun yksi keskeisiä työvaiheita on valvomosuunnittelu. Valvomonäytöt ovat se osa valmista prosessia, jonka kanssa loppukäyttäjä joutuu toimimaan työssään joka päivä. Valvomo on loppukäyttäjän rajapinta itse tuotantoprosessiin, oli kyse sitten kappaletavara-automaatiosta tai prosessiautomaatiosta. Tästä syystä hyvin suunniteltuun valvomoon on syytä kiinnittää huomiota; sen tulisi olla mahdollisimman informatiivinen, helppokäyttöinen ja kuitenkin tehokas. Operaattorin tulisi pystyä reagoimaan prosessin mahdollisiin virhetilanteisiin nopeasti, jolloin ainoa tiedonlähde prosessin tilasta käyttäjälle on valvomonäyttö.

Tässä opinnäytetyössä pyrittiin rakentamaan sekä suunnitteluryhmää, että tuotteen loppukäyttäjää palvelevia, valmiiksi ohjelmoituja valvomosymboleja. Työhön on kerätty yleisimmät teollisuuden komponentit kuten moottorit ja venttiilit, ja niille on rakennettu valmiit objektit sekä ohjausikkunat. Valvomo-objekteja tullaan käyttämään SIMATIC WinCC –PC pohjaisen valvomo-ohjelmiston kanssa.

Kehitysvaiheessa on pyritty koko ajan pitämään silmällä sekä suunnitteluajan lyhentämistä, että loppukäyttäjää palvelevaa kokonaisuutta. Jos tulevaisuudessa automaatiosuunnittelijalla on käytössään valmis kirjasto erilaisia teollisuuden komponentteja joiden näyttöobjektit on testattu ja

havaittu toimiviksi, nopeutuu suunnittelutyö huomattavasti ja valvomoiden rakentaminen on helpompaa. Yksi kehitystyötä eteenpäin ajanut voima on työajan tehostamisen ohella puhtaat kustannussäästöt joita käsitellään myöhemmin tässä raportissa.

1.3 Henkilökohtaiset tavoitteet

Osaksi työnkuvaani harjoitteluaikana toimeksiantajayrityksessä kuului erilaisten toimintojen kehittäminen. Tavoitteena oli suunnittelutyön kehittäminen entistä joustavammaksi, jolloin rutiininomaiset työvaiheet nopeutuisivat. Tällöin voidaan asettaa suuremmat resurssit uuden järjestelmän tai sovelluksen kehittämistyölle. Myös opinnäytetyön aihe lokeroitui tähän työnkuvaan. Opinnäytetyöprosessin kautta pyrinkin pääsemään kiinni tavalliseen suunnittelutyöhön, ja havaitsemaan sen kehityskohteita. Toki työn kuluessa tulivat tutuksi suunnittelutyössä käytettävät työkalut ja ohjelmistot sekä ennestään tuntemattomat ohjelmointikielet.

2 SUUNNITTELUPROSESSI JA SEN TYÖKALUT

2.1 Automaation sovellussuunnittelu

Sovellussuunnittelu käsittää ohjattavan laitteen tai laitekokonaisuuden ohjaamiseen käytettävän sovelluksen määrittämistä, tuottamista ja testaamista. Lähtökohdat sovellussuunnittelulle luo esisuunnittelu. Esisuunnittelussa syntyneitä dokumentteja ovat muun muassa:

- instrumenttipiiriluettelot
- toimintakuvaukset
- moottoriluettelot
- toimintakaaviot
- lukituskaaviot
- sekvenssi- ja ajokaaviot
- hälytys- ja raporttimäärittelyt

Olennaisena osana määrittelyihin kuuluvat myös aikataulut ja hankintarajamäärittelyt. (Pekkanen 2006, 35-36.)

Sovellussuunnittelu sisältää ohjelmiston rungon, eli peruspiirien lisäksi käyttöliittymän, ylemmän tason ohjelmistojen kuten raportoinnin, valvomon ja rajapintojen suunnittelun mahdollisiin muihin järjestelmiin. (SAS 2001, 50-51.)

2.2 Valvomosuunnittelu

Yleisesti sovellussuunnittelun aikana toimittaja esittää ostajan hyväksyttäväksi operointilaitteiden mallit ja – koot, sekä operointitavat (PSK 4602, 1996, 6). Toimittaja myös laatii ehdotuslistan prosessiajojen aikaisista hälytystoiminnoista. Standardin (PSK 4602, 1996, 6) mukaan ehdotuksiin tulee sisältyä seuraavat toiminnot:

- Luokittelu ja kiireellisyys, henkilöstöryhmien mukaan
- Erottelukyky
- Maskaukset ja jatkuvasti toistuvien hälytysten analysointi
- Hälytysten esittäminen käyttäjälle kirjoittimella, näyttöpäätteen eri näyttötyypeissä
- Voiko käyttäjä ohjata sitä, millaisia hälytyksiä näytetään
- Annetaanko tietoa hälytysten esiintymisajasta
- Käsitelläänkö hälytykset loogisesti niihin johtaneiden syiden määrittelemiseksi? Suodatetaanko hälytykset laitoksen käyntitilan perusteella?
- Annetaanko käyttäjälle tietoa hälytyksen syystä?
- Mitä kirjauksia säilytetään hälytyksistä tai muista ohjaus- ja instrumentointijärjestelmän tapahtumista? Tallennetaanko ne paperille vai magneettisesti? Miten kauan kirjauksia säilytetään?
- Miten sekvenssiohjausten toteutumattomat syyt esitetään näyttöpäätteiden näytöllä.

Tärkeimpänä dokumenttina erityisesti valvomosuunnitteluvaiheessa toimii prosessi- ja instrumenttikaavio (PI-kaavio) joka käsittää kokonaisuudessaan suunniteltavan prosessin laitteineen ja mittauksineen, PI –kaaviosta nähdään

myös, mitkä mittaukset ja toimilaitteiden ohjaukset tulee näkyä ohjaavan järjestelmän päässä eli valvomossa.

Valvomosuunnittelu sisällytetään sovellussuunnitteluun. Yleisesti määritellään näyttöihin ensin staattiset objektit, kuten putkilinjat, säiliöt ja muut rakenteet. Tämän jälkeen näyttöjen ulkoasu hyväksytetään asiakkaalla, jonka jälkeen voidaan lähteä suunnittelemaan dynaamisia näyttöobjekteja. (Pekkanen 2006, 35-36.) Valvomonäyttöjen suunnittelu ja rakentaminen vie verrattaen lyhyen ajan sovellussuunnitteluprosessin kokonaisajasta, valmiit ja testatut valvomo-objektit nopeuttavat vaihetta entisestään.

2.3 PC –pohjainen valvomo eli SCADA

2.3.1 Mikä on SCADA?

SCADA koostuu sanoista *Supervisory Control and Data Acquisition* ja tarkoittaa siis hallinta- ja tiedonkeruuohjelmistoa, käyttäjällä ei kuitenkaan ole aivan täysiä ohjausoikeuksia prosessiin. Pikemminkin ohjelmistoja voidaan pitää prosessin ylimmän tason elimenä jolla on tarpeenmukaiset hallintaoikeudet ohjattaviin laitteisiin. Järjestelmät ovat itsenäisiä, juuri valvomototeutuksiin kehitettyjä ohjelmistoja. Viime vuosien aikana SCADA –ohjelmistot ovat kehittyneet tehokkaammiksi ja entistä laaja-alaisemmiksi. Aikaisemmin yritykset muokkasivat niitä omiin tarpeisiinsa sopiviksi jolloin ohjelmistoista tuli monimutkaisia eikä toteutuksessa noudatettu yhteistä linjaa. (Daneels & Salter, 399)

2.3.2 Laitteistoarkkitehtuuri

Valvomototeutus voidaan hoitaa muutamalla eri tavalla. Riippuen laitoksen koosta, ohjelmiston sisältäviä pc – koneita voidaan sijoittaa tehtaan eri tiloihin useampi jolloin ne toimivat client – periaatteella. Tällöin järjestelmä sisältää erillisiä data server – yksiköitä jotka keskustelevat esimerkiksi ohjelmoitavan logiikan tai automaatiojärjestelmän kanssa. Nämä data serverit välittävät tiedot PC – koneiden valvomo-ohjelmistoon. Järjestelmä voidaan koota joko väyläperusteiseksi, tai ethernet – verkon avulla. (Daneels & Salter, 399)

Tämän opinnäytetyön toteutuksessa yksi PC toimii sekä data serverinä, että valvomokoneena. Yhden koneen toteutuksessa tietokone liitetään suoraan ohjelmoitavan logiikan ethernet – linkkiin. Valvomo-ohjelmistolle (tässä tapauksessa SIMATIC WinCC) annetaan logiikan IP – osoite jolloin järjestelmät pystyvät keskustelemaan keskenään.

2.3.3 Sovellusarkkitehtuuri

Järjestelmän data server – yksiköt hoitavat keskustelemisen ohjattavan järjestelmän kanssa. Ne suorittavat kyselyitä logiikalta tulojen, lähtöjen ja hälytysten suhteen, sekä välittävät ohjaustiedot client – koneelta ohjattavaan järjestelmään. Serverit sisältävät eri yksiköt muun muassa prosessidatankeruulle, hälytystietojen käsittelylle ja tallennukselle sekä raporttien luomiselle. Laajemmissa järjestelmissä jokaiselle tehtävälle on oma serveriyksikkö, pienemmissä taas toiminnot hoidetaan saman yksikön alaisuudessa. (Daneels & Salter, 339–340) Esimerkiksi WinCC – ohjelmisto käsittää itsessään kaikki edellä mainitut tehtävät.

2.3.4 Liitynnät ohjattaviin laitteisiin

Kuten mainittua, PC – valvomon Data Serverit suorittavat kyselyitä ohjattavan järjestelmän suuntaan. Suunnittelija voi valvomoa rakentaessa määritellä kyselyiden välisen ajan jokaiselle mittaukselle tai ohjaukselle erikseen (Daneels & Salter, 340). Esimerkiksi pinnankorkeuden mittaus on harvassa laitoksessa niin kriittinen kuin virtausnopeuden mittaus.

Valvomon fyysinen liittäminen logiikkaan tai automaatiojärjestelmään voidaan toteuttaa monella eri tapaa; SCADA – ohjelmistot tukevat useita eri sarjamuotoisia liityntöjä. Yleisimmin käytetyistä mainittakoon Profibus. Tulevaisuudessa kasvava ethernet ja Industrial Ethernet on myös erittäin käytetty liityntätapa.

2.3.5 Avoimet rajapinnat

SCADA – ohjelmistoissa voidaan käyttää apuna myös avoimen lähdekoodin omaavia rajapintoja ohjattaviin järjestelmiin liittyessä ja tiedonkeruulokeihin yhdistettäessä, seuraavassa on lueteltu muutamia erilaisia rajapintoja.

- OPC – rajapinta, datan lukemiseen ja kirjoittamiseen ohjattavan järjestelmän välillä.
- Open Data Base Connectivity (ODBC) jolla päästään käsiksi tiedonkeruulokeihin
- Ohjelmointikielet kuten C, C++ ja Visual Basic joiden avulla voidaan kirjoittaa ja lukea dataa servereiltä

Ohjelmistot sisältävät myös tuen useille Windows – ympäristössä käytössä oleville standardeille kuten DDE ja DLL joita käytetään esimerkiksi siirrettäessä prosessidataa Excel – työkirjaan. (Daneels & Salter, 340–341). Opinnäytetyössä on pureuduttu juuri ohjelmointikielten tehokkaampaan käyttöön valvomosuunnittelussa.

2.3.6 Ajonaikainen automaatio

SCADA – ohjelmistot sisältävät valmiudet moniin automaattisiin toimintoihin, kuten datankeruu, hälytysten tallennus ja raporttien generointi. Toimintoja voidaan suorittaa myös ohjelmistojen tukemilla ohjelmointikielillä. Kirjoittajan mielestä helpoin työkalu tähän tarkoitukseen on Visual Basic – ohjelmointikieli, joka opinnäytetyössä otettiin tehostettuun käyttöön. Avoimien ohjelmointikielten avulla päästään esimerkiksi käsiksi servereiden tietoihin eli voidaan lukea ja kirjoittaa logiikkaan. Ohjelmointikielillä voidaan tietoa kirjoittaa myös kolmansiin sovelluksiin kuten Exceliin, käyttäen ActiveX – komponentteja. (Daneels & Salter, 342)

Tässä raportissa esitellyn symbolikirjaston idea perustuu osaltaan ajonaikaiseen automaatioon, kun Visual Basic Scripting – ohjelmointikielen avulla linkitetään I/O – pisteitä eli tageja valvomon objekteihin. Tällöin osa suunnittelijan työstä säästyy ja se hoidetaan vasta loppukäyttäjän ajaessa laitosta valvomon avulla.

2.4 SIMATIC WinCC

SIMATIC WinCC on Siemensin valmistama SCADA – ohjelmisto. Se on pääasiassa suunniteltu toimimaan Siemensin ohjelmoitavien logiikoiden ja automaatiojärjestelmien kanssa. Ohjelmisto toimii käyttäjän ja prosessin välisenä rajapintana antaen käyttäjälle lähes reaaliaikasta tietoa prosessin tilasta ja mittausarvoista. Ohjelmiston kautta käyttäjä pystyy operoimaan prosessia, esimerkiksi käynnistämään moottoreita ja avaamaan venttiileitä, käyttäjä näkee myös prosessin hälytykset ja virhetilanteet operointipaneelilta. (Siemens HMI, 2010.)

SIMATIC WinCC on monipuolisesti laajennettavissa oleva ohjelmisto, johon voidaan liittää tehtaanlaajuinen tietokoneverkosto ja sen antamaa tietoa voidaan tarkkailla myös web – selaimella (Siemens AG, 2009). WinCC ohjelmisto käyttää tageja keskustellakseen prosessia ohjaavan logiikan tai järjestelmän kanssa. Jokainen tagi vastaa logiikan tiettyä tuloa tai lähtöä, oli arvo sitten binäärinen tai analoginen. Opinnäytetyössä pyritäänkin kehittämään WinCC –ohjelmointia edullisemmaksi sekä tilaajalle, että toimittajalle koska SIMATIC WinCC –lisenssit hinnoitellaan tagimäärien mukaan. Operointinäyttöjä suunniteltaessa sovellussuunnittelija lisää tageja näyttökuvakkeisiin saadakseen esimerkiksi symbolin taustavärin muuttumaan, kun tietty lähtö logiikassa vaihtaa tilaansa. WinCC ohjelmistossa tällaisia animointeja kutsutaan dynaamisiksi määrittelyiksi. Tehokas tapa luoda useita dynaamisia määrittelyjä kerralla, on käyttää VBS –ohjelmointikieltä, jolloin voidaan yhden tagin arvomuutoksella saada aikaan monia visuaalisia muutoksia operointinäytössä. (Siemens HMI, 2010.)

Ohjelmistossa on valmiudet prosesseista saatavan informaation tallennukseen ja analysointiin jälkeenpäin. WinCC –ohjelmistolla voidaan määritellä myös prosessista annettavat hälytykset ja varoitukset, sekä tarvittaessa tallentaa ne myöhempää tarkastelua varten SQL -tietokantoihin. WinCC ohjelmistosta voidaan siirtää prosessidataa esimerkiksi taulukkolaskentaohjelmiin OPC rajapinnan avulla tai suoraan Visual Basic scriptien kautta. Suunnittelutyötä nopeuttavia toimintoja ovat myös hälytyslistojen ja suurten tagimäärittelyjen massasiirto suoraan Excel – taulukoista .csv tiedostoina.
(Siemens HMI, 2010.)

2.5 Visual Basic

Visual Basic on Windows –ohjelmointikieli, jolla pystytään luomaan helposti omia ohjelmia Windows pohjaisiin käyttöjärjestelmiin. ”Visual” –sana viittaa menetelmään, jolla luodaan graafisia käyttöliittymiä (GUI). ”Basic” –sana taas viittaa BASIC –kieleen, joka on käytetyin ohjelmointikieli tietokonehistoriassa. VBScript –ohjelmointikieltä voidaan käyttää hyvin moneen erilaiseen tarkoitukseen, aina pienien Excel –makrojen kirjoittamisesta suurien kansainvälisten ohjelmakokonaisuuksien luomiseen. (Ohjelmoijan käsikirja 1999, 3.)

Visual Basic eroaa ohjelmointiympäristönä muista samaan tarkoitukseen kehitetyistä ohjelmistoista siinä, että se tarkkailee käyttäjän kirjoittamaa koodia reaaliajassa, ja ilmoittaa virheistä välittömästi, kun taas useat muut ohjelmistot ja ohjelmointikielet vaativat virheidentarkistukseen erillisen käännöksen, tällöin käyttäjä joutuu kirjoittamaan virheellisen koodin uudestaan ja tekemään käännöksen uudelleen.Visual Basic työskentely-ympäristöä kutsutaan integroiduksi kehitysympäristöksi, IDE:ksi (Integrated Developing Environment), koska ohjelmassa yhdistyy sekä suunnittelu, ohjelmankirjoitus, käännös ja virheidenetsintä. (Ohjelmoijan käsikirja 1999, 13.)

Visual Basic ohjelmasuoritus perustuu tapahtumaohjattuun malliin, jossa käyttäjä puuttuu ohjelman suoritukseen toimenpiteillään. Normaalisti esimerkiksi C-kielinen koodi ajetaan alusta loppuun lause kerrallaan kutsuen mahdollisia aliohjelmia ja lukien erilaisia ehtolauseita kun taas Visual Basic ohjelman ajossa käyttäjä määrää toiminnoillaan ohjelman ajojärjestyksen, napin painallus tai tietyn arvon muuttuminen vaikuttaa olennaisesti ohjelman ajoon. (Ohjelmoijan käsikirja. 1999, 12-13)

2.6 VBS ja VBA

VBS (Visual Basic Scripting) on Visual Basic – ohjelmointikielen alalaji jota käytetään WinCC – ohjelmistossa ajonaikaisten toimintojen suorittamiseen. VBS –kielellä voidaan suorittaa kaikki samat toiminnot kuin käsin asetelluin dynaamisin määrittelyin. VBS –kielen tehokkuus perustuu sen kykyyn hallita kaikkia operointinäytössä näkyviä objekteja. Mikäli yhden mittausarvon

muutos edellyttää useampia dynaamisia muutoksia operointinäytössä, voidaan kaikki määrittelyt kirjoittaa yhteen VBS – koodiin.
(Siemens product support, 2010.)

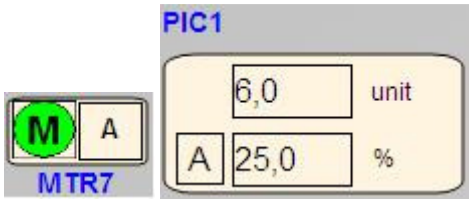
VBA (Visual Basic for Applications) – ohjelmointikieltä käytetään operointinäyttöjen konfiguroinninaikaisten toimintojen suorittamiseen. VBA – koodilla voidaan esimerkiksi piirtää massatuotantona objekteja näyttöihin ja lisätä ”tooltip” –tekstejä operointikuvakkeisiin. VBA – ohjelmointikieli toimii siis suunnittelijan avustajana operointinäyttöjä rakennettaessa lyhentäen jälleen suunnitteluaikaa ja siirtäen resursseja muihin suunnittelun vaiheisiin.
(Siemens product support, 2010.)

3 TOTEUTUS

3.1 Toteutusprosessi

Opinnäytetyön toteutus aloitettiin määrittelemällä symbolikirjastoon lisättävät toimilaitteet. Kirjastoon valittiin prosessiteollisuudessa yleisimmin käytössä olevat komponentit, kuten moottorit, venttiilit, säätimet ja mittaukset. Työn alkuvaiheessa aseteltiin myös rajat operointinäytöissä näytettävälle datalle. Moottoreista ja venttiileistä operaattorille näytetään toimilaitteen tila- ja lukitustiedot, mittauksista mittaus- ja raja-arvot. Säätimen data koostuu mittausarvosta, säätimen SetPoint- ja ohjausarvoista sekä hälytys- ja lukitusrajoista.

Tämän jälkeen toimilaitteista luotiin visuaalisesti hyväksyttävät objektit, jotta operointi olisi loppukäyttäjän kannalta mielekästä (ks. kuvio 1). Myös toimilaitteiden ohjausikkunat ja piirinäytöt muokattiin mahdollisimman informatiivisiksi ja selkeiksi lukea. Kun symbolikirjasto oli visuaalisesti toteutettu, suunniteltiin siirrettävän datan liikkuminen ohjelmointilaitteen ja WinCC –ohjelmiston välillä. Suunnittelussa tuli pyrkiä mahdollisimman vähäisen tagimäärän käyttöön. Kustannustehokkain ratkaisu oli siirtää dataa ohjelmointilaitteelta mahdollisimman suurissa paketeissa (maks. 64 bittiä) ja purkaa paketti bittimaskausten avulla VBS – koodeissa.



KUVIO 1. Moottoritoimilohkon ja säätimen näyttökuvakkeet

Moottorikuvakkeen väri kertoo moottorin tilan, vihreä: käynnissä, punainen: fault, harmaa: seis. Säädinkuvakkeessa näytetään mitattavan prosessisuureen arvo, säätimen ohjausarvo prosentteina sekä säätimen tila (manuaali/auto).

3.2 Bittimaskit

Bittien maskauksella pystytään lukemaan 32 bitin jonosta vain haluttu bittikombinaatio ja sen arvo. Esimerkiksi moottori- ja venttiililohkojen tila- ja häytystiedot tuotiin logiikasta yhtenäisenä, 32 bitin jonona jossa ensimmäinen (LSB) bitti vastaa toimilaitteen ajomoodia (manuaali/auto). Seuraavat kahdeksan bittiä vastaavat toimilaitteen tilaa (Fault, Stopped/Closed, Running/Open) ja bittinumerot 8-17 vastaavat toimilaitteen lukitustietoja. Kuviossa 2. nähdään esimerkki bittimaskauksen yleisestä toimintaperiaatteesta.

```
    10100011110111010001101101001110
AND                          11111111
  = 00000000000000000000000001001110
```

KUVIO 2. Maskaaminen kahdeksalla bitillä

Moottori- ja venttiiliohjauksissa päädyttiin myös tagimääriä säästävään ratkaisuun, jossa WinCC –ohjelmisto sisältää yhden ohjaustagin ja jokainen toimilaite indeksoidaan. PLC –laitteen sovellus lukee tämän indeksin ja osaa näin kirjoittaa ohjauskäskyn oikeassa toimilohkossa. Yrityksen aikaisemmissa projekteissa jokainen toimilaite on sisältänyt oman ohjaustagin, uudella menettelyllä säästetään merkittävästi tagimäärissä, erityisesti useampien kymmenien toimilaitteiden kokonaisuuksissa.

3.3 Toiminnallisuus

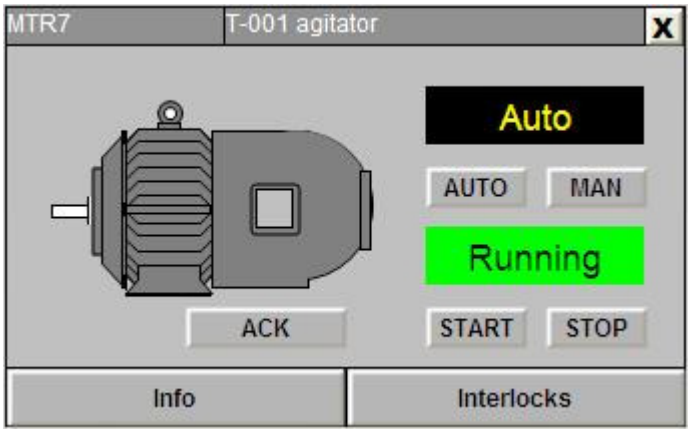
Kaikkien näyttöobjektien toiminnallisuus toteutettiin Visual Basic Scripting –ohjelmointikielellä. Toiminnallisuudessa pyrittiin siihen, ettei suunnitteluvaiheessa suunnittelijan tarvitse tehdä suuria muutoksia, tai määrittelyjä näyttöjä rakentaessa. Toteutuksessa pyrittiin ”drag&drop” –toiminnallisuuteen jossa suunnittelija voi lisätä objektit näyttöön toimilaitekirjastosta ja merkitä niille oikeat laite- ja instrumenttipositiot. Suunnittelijalle jää määriteltäviä parametreja toimilaitteiden lukituskuvaukset, positiointi ja indeksointi.

Hankalimmaksi osuudeksi toteuttamisen kannalta osoittautui toimilaitteiden tilatietojen lukeminen piirinäyttöihin, toteutusvaiheessa ei vielä tiedetty mahdollisten laitteiden positioita. Toteutuksessa päädyttiin ratkaisuun, jossa toimilaitteen klikkaus kirjoittaa oikeat tilatiedot piirinäyttöön, ohjausikkuna taas lukee aktiivisena olevan toimilaitteen positiokenttää, ja osaa vaihtaa tilatietoja laitteen ollessa valittuna. Liitteessä 1 on kuvattu toimilaitteen tila- ja lukitustietojen kirjoittaminen piirinäyttöön.

3.4 Moottoripiirit

Moottoritoimilohkon ohjausikkuna sisältää kaksi välilehteä, Info – välilehti sisältää moottorin tilatiedot (ks. kuvio 3), Interlocks – välilehti sisältää moottorin lukitustiedot (ks. kuvio 4). Lukitusikkunassa on lueteltuna kaikki mahdolliset ko. moottorin lukitukset, lukituksen aktivoituessa, kyseinen lukitus

korostetaan. Lukitusikkunassa on korostettuna myös viimeisin aktiivinen lukitus.

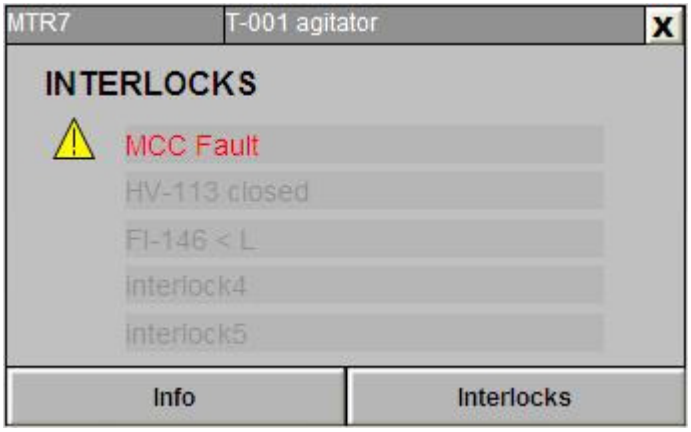


KUVIO 3. Moottorin ohjausvälilehti

Ohjausikkuna sisältää painikkeet moottorin ohjaukselle ja ajotilan vaihdolle, sekä moottorin tilatiedot. Interlocks –painikkeella voidaan siirtyä lukitussivulle. Piirinäytön ylälaidassa näkyy moottorin laitepositio sekä kuvaava nimi.

- AUTO; vaihtaa moottorin automaattiohjaukseen
- MAN; vaihtaa moottorin manuaaliohjaukseen
- START; käynnistää moottorin (manuaalitilassa)
- STOP; pysäyttää moottorin
- ACK; lukitustiedon kuittaus

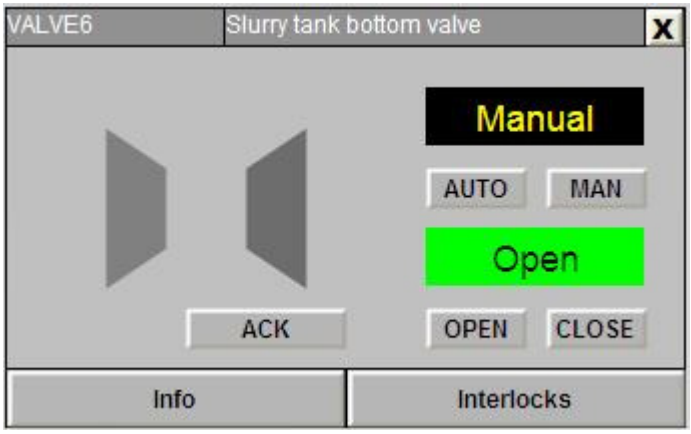
Lukitusikkuna sisältää moottorin mahdollisten lukitusten kuvaukset. Aktiivinen lukitus on merkitty punaisella värillä, viimeisen aktiivisen lukituksen vieressä on keltainen varoituskolmio.



KUVIO 4. Moottorin lukitusvälilehti

3.5 Venttiilipiirit

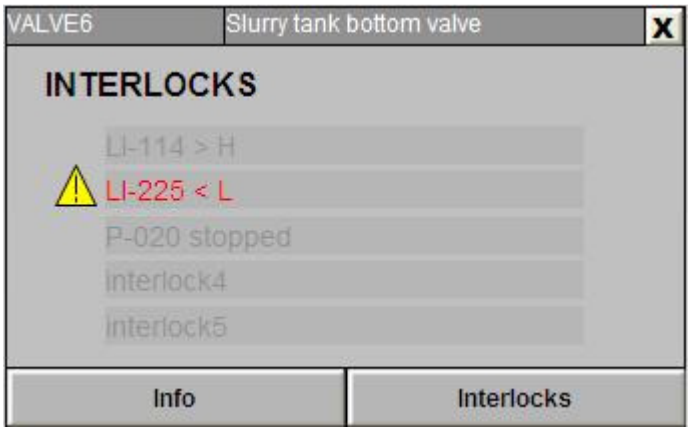
Venttiililohkon ohjausikkuna sisältää moottorilohkon tavoin kaksi välilehteä; venttiilin tila- ja ohjaussivun sekä lukitusvälilehden. (ks. kuvio 5) Ohjausikkuna sisältää painikkeet venttiilin ohjaukselle, sekä venttiilin tilatiedot.



KUVIO 5. Venttiilin ohjausvälilehti

- AUTO; vaihtaa venttiilin automaattiohjaukseen
- MAN; vaihtaa venttiilin manuaaliohjaukseen
- OPEN; avaa venttiilin (manuaalitilassa)
- CLOSE; sulkee venttiilin (manuaalitilassa)
- ACK; lukitustiedon kuittaus

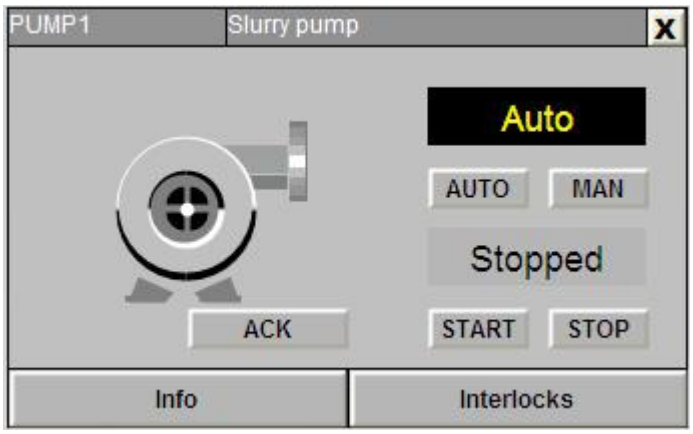
Lukitusikkuna sisältää moottorilohkon tavoin venttiilin aktiiviset ja passiiviset lukitukset, sekä viimeisen aktiivisen lukituksen. (ks. kuvio 6.)



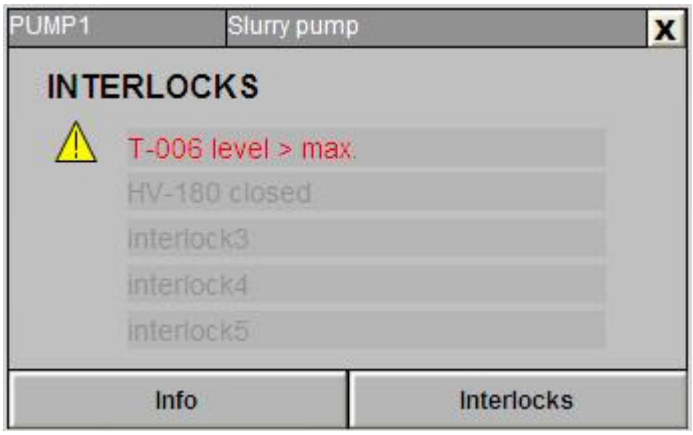
KUVIO 6. Venttiilin lukitusvälilehti

3.6 Pumppupiirit

Pumppupiirin ominaisuudet ovat identtiset moottoripiirien kanssa ,vain operointinäytön toimilaitekuvake ja ohjausikkunan kuvake poikkeavat moottoripiiristä. (ks. kuviot 7,8)



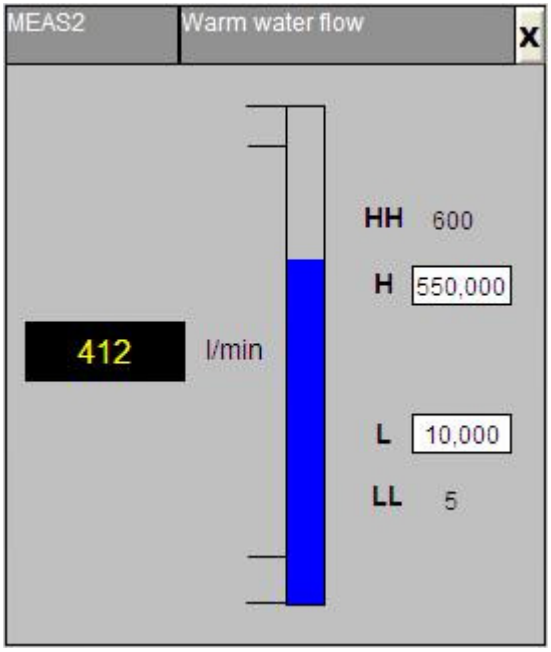
KUVIO 7. Pumpun ohjausvälilehti



KUVIO 8. Pumpun lukitusvälilehti

3.7 Mittaukset

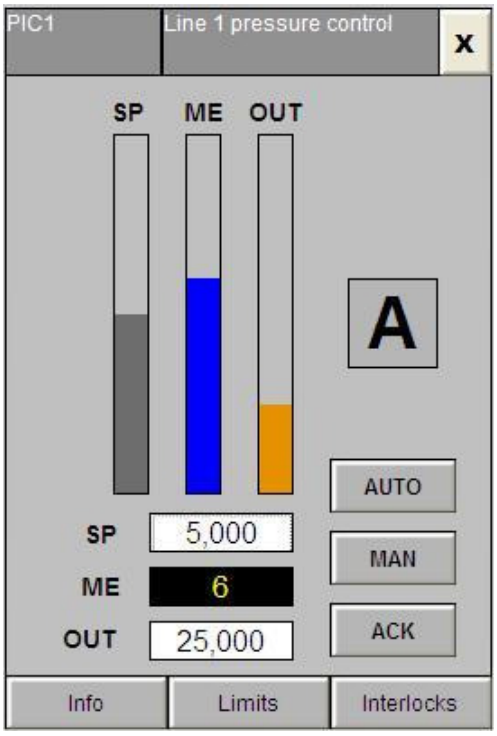
Mittauspiirin näyttökuvakkeessa näytetään mittauksen oloarvo ja mittauksen yksikkö. Klikkaamalla mittauskuvaketta, aukeaa mittauksen piirinäyttö, josta voidaan asetella ylä- ja alarajat, sekä voidaan tarkastella ylä-ylä ja ala-ala –rajoja. Mittauksen minimi- ja maksimiarvot tuodaan suoraan ohjelmoitavalta logiikalta. (ks. kuvio 9.)



KUVIO 9. Mittauksen piirinäyttö

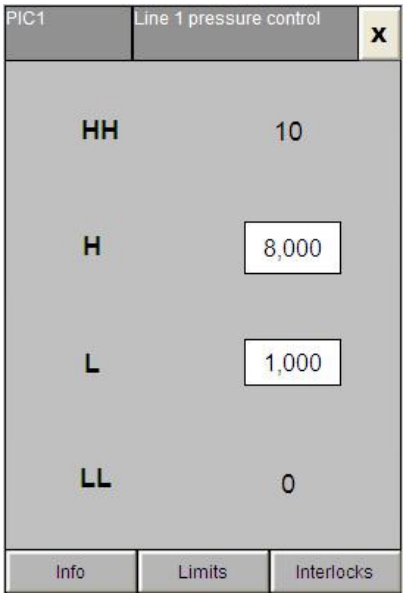
3.8 Säätimet

Säädinlohkoilla voidaan säätää haluttua prosessisuuretta, kuten virtausta, painetta tai pinnankorkeutta. Säätimen pääsivu sisältää säädinlohkon mittaus-, asetus-, ja ohjausarvot sekä MAN/AUTO - ja hälytyksen kuittaus –painikkeet. Säätimen ollessa manuaalitilassa voidaan pääsivulta antaa suora ohjausarvo säätävälle toimilaitteelle. Automaattitilassa voidaan muuttaa vain asetusarvoa. Pääsivun alarivin navigointipainikkeilla voidaan siirtyä välilehdeltä toiselle. (ks. kuvio 10.)



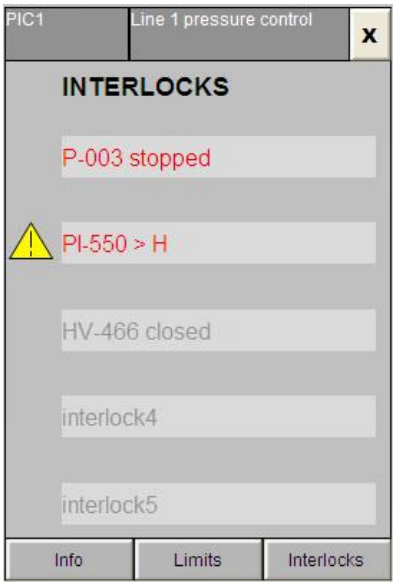
KUVIO 10. Säätimen pääsivu

Limits –välilehdellä voidaan mittauspiirin tavoin asetella säätimelle ylä- ja alarajat, sekä tarkastella lukitusrajoja. (ks. kuvio 11.)



KUVIO 11. Säätimen raja-arvosivu

Säädinlohkon Interlocks – välilehdellä on kuvattu edellisten toimilaitteiden tavoin säätimen aktiiviset ja passiiviset lukitukset. (ks. kuvio 12.)



KUVIO 12. Säätimen lukitusvälilehti

3.9 Testaus

Toimilaitekuvakkeita ja ohjausikkunoita testattiin toteutuksen aikana runsaasti oikean toiminnallisuuden saavuttamiseksi. Testaukseen käytettiin SIMATIC WinCC – ohjelmiston erityistä Tag Simulator – sovellusta josta voidaan syöttää eri arvoja käsiteltäville tageille. Sovelluksen toiminta vastaa ohjelmoitavan logiikan lähtöportteja. Yhdeksi haastavimmista osioista osoittautui lukitustietojen generointi ja oikea kuvaus, sekä viimeisen aktiivisen lukituksen kaappaus. VBS –koodit pyrittiin pitämään mahdollisimman suoraviivaisina ja yksinkertaisina, prosessoritehojen säästämiseksi.

Toimintakokonaisuuksista rakennettiin aluksi yksinkertainen runko VBS – editoriin, joka testattiin muutamilla arvomuutoksilla. Tällaisia toimintakokonaisuuksia olivat esimerkiksi toimilaitteen tilatietojen kirjoittaminen piirinäyttöön toimilaitetta klikattaessa, valittuna olevan toimilaitteen lukitustietojen generointi sekä mittausten hälytysrajojen asettelu. Runko-osan testauksessa valittu menetelmä todettiin toimivaksi, jonka jälkeen

koodiin voitiin lisätä toiminnallisuutta ja hienosäätöä. Jokainen asia voitiin toteuttaa Visual Basic Scripteillä monella eri tavalla, pitäen kuitenkin silmällä seuraavan sovellussuunnittelijan työmäärää ja loppukäyttäjän selkeää käyttöliittymää. Toisaalta scriptit tuli pitää selkeinä, jotta taloon uutena tuleva suunnittelija voi ottaa kirjaston käyttöön ongelmitta, ja jopa tehdä muutoksia ja lisäyksiä koodeihin.

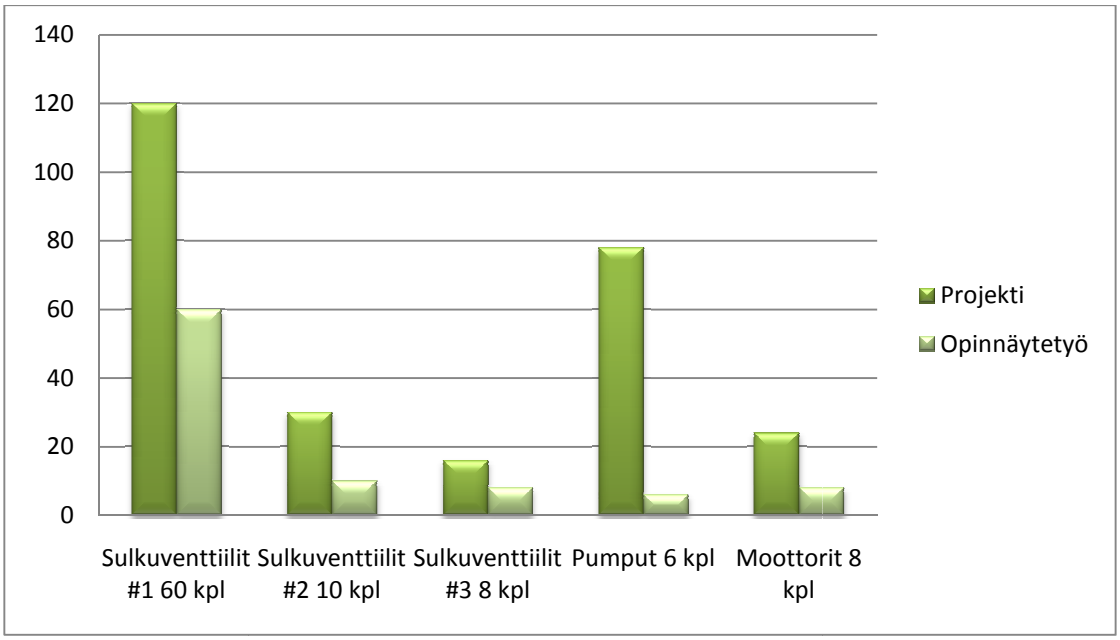
4 TULOKSET

Opinnäytetyön onnistumista voidaan mitata suunnittelutyön ajankäytön tehostamisella, ja kustannussäästöillä lisenssimaksuissa, sekä erityisesti toimivilla, testatuilla ja hyviksi todetuilla operointimenetelmillä joita voidaan jatkossa myydä asiakkaalle kuin asiakkaalle

Symbolikirjastoa käytetään suunnittelutyön apuna loppuvuonna 2010. Ensimmäinen kirjaston avulla luotu valvomo otetaan asiakasprojektissa käyttöön vuonna 2011.

4.1 Vertailu aiempaan projektiin

Jotta työn onnistumisesta saataisiin numeerisia tuloksia, tuli sitä verrata yrityksen aikaisempien projektien toteutukseen. Vertailukohdaksi otettiin erään kemikaalilaitoksen laitteisto- ja sovellussuunnittelu. Kyseessä on keskikokoinen, noin 90 toimilaitteen kokonaisuus. Kuviossa 13 on esitetty säästöt tagimäärissa laitteistokohtaisesti, mikäli projekti olisi toteutettu opinnäytetyön mukaisella menetelmällä. Arvoinnissa on laskettu edeltävän projektin tagimäärä toimilaitetta kohti, ja verrattu sitä symbolikirjaston avulla toteutettuun, vastaavaan toimilaiteohjaukseen.



KUVIO 13. Tagimäärien vertailu

4.2 Tulokset tulevaisuudessa

Lopulliset tulokset opinnäytetyön onnistumisesta saadaan, kun työtä päästään soveltamaan ensimmäiseen sopivaan projektiin, Tällöin operointinäytöt hyväksytetään asiakkaalla ja piiri- ja ohjausikkunat todetaan käytännöllisiksi. Symbolikirjastoa pyritään muokkaamaan ja parantamaan asiakkaiden vaatimusten mukaiseksi. Ohjausikkunoiden toimivuus saadaan lopullisesti testattua ohjelmoitavaan logiikkaan luodulla sopivalla sovelluksella jossa jokainen tulo ja lähtö voidaan testata erikseen.

5 KEHITYSKOHTEET

Avoimen Visual Basic –koodin ansiosta symbolikirjastoa voidaan muokata esimerkiksi projektikohtaisesti haluttuun suuntaan, tuoda lisää prosessidataa piirinäyttöihin, lisätä erikoistoimilaitteita ja luoda sekvensseille omat ohjausikkunat.

Potentiaalisia kehityskohteita olisi konfiguroinnin aikaisten scriptien (VBA) kirjoittaminen ja testaaminen, esimerkiksi toimilaitteiden lukitustiedot voitaisiin tuota suoraan import – toiminnolla .csv –tiedostoista. Suunnitteluprosessia tulisi muuttaa siten, että sovellussuunnittelijalla olisi käytössään listat toimilaitteista positiotietoineen, indekseineen ja lukituskuvauksineen. Nämä tiedot voitaisiin syöttää suoraan excel – taulukosta WinCC – ohjelmistoon.

Tulevaisuudessa työn pohjalta tullaan luomaan symbolikirjastot myös WinCC Flexible –ympäristöön, jota käytetään Siemensin valmistamien kenttäpaneeleiden konfigurointiin ja valvomosuunnitteluun. Myös S7-1200 logiikan ohjelmointisovellus Step 7 Basic:iin luodaan identtinen symbolikirjasto.

6 POHDINTA

Opinnäytetyö antoi hyvän tilaisuuden saada kehittää uuden yrityksen toimintaa entistä joustavampaan ja tehokkaampaan suuntaan, Yksi tärkeimpiä opinnäytetyön anteja oli ennestään tuntemattoman Visual Basic – ohjelmointikielen oppiminen, onhan Visual Basic Windows –ympäristön eniten käytetty ohjelmointikieli.

Työn tavoitteena oli luoda toimiva symbolikirjasto, joka tehostaisi yrityksen sovellussuunnittelua, ja vakinaistaisi projektien valvomosuunnittelussa käytetyt objektit valmiiksi testattuihin ja toimiviin elementteihin. Saaduista tuloksista voidaan päätellä, että työssä saavutettiin asetetut tavoitteet, eli kustannussäästöt tagimäärissä, suunnittelutyön tehokkuuden lisääminen ja valvomosuunnitteluajan lyhentäminen. Ennen kaikkea suunnittelutoimisto sai käyttöönsä helposti kehitettävissä olevan ja toimivaksi todetun symbolikirjaston teollisuuden yleisimmistä toimilaitteista ja mittauksista ohjauksineen ja piiri-ikkunoineen.

LÄHTEET

Daneels, A & Salter, W. 1999. What is SCADA? Viitattu 05.10.2010. http://www.elettra.trieste.it/ICALEPCS99/proceedings/papers/mc1i01.pdf

JEEC Oy. Kotisivut. 2010. Viitattu 11.8.2010. http://jeec.fi/.

Microsoft Visual Basic 6.0 Ohjelmoijan käsikirja. 1999, 3-4, 12-13. Toim. Microsoft Corporation. Jyväskylä: Gummerus Kirjapaino Oy.

Pekkanen, J. 2006. Automaatiojärjestelmien sovellussuunnittelu. Opinnäytetyö. IT-instituutti. Tekniikan ja liikenteen ala. Automaatiotekniikan koulutusohjelma.

PSK 4602. 1996. Automaation hankinta. Prosessinohjausjärjestelmä. Prosessiteollisuuden standardoimiskeskus r.y. Viitattu 26.9.2010. http://psk-standardisointi.fi, jäsenille.

Siemens AG. 2009. Human Machine Interface Systems ST 80, 4/2-4/3. Tuotekatalogi.

Siemens HMI – käyttäjäoppaat. SIMATIC WINCC käyttäjäopas. Siemens Finland kotisivu. Viitattu 11.8.2010. http://siemens.com, industry, teollisuus, tuotteet ja järjestelmät, automaatiotekniikka, käyttöliittymät.

SAS, 2001. Laatu automaatiossa. Parhaat käytännöt. 4, 5, 16, 17 ja 122. Helsinki: Suomen Automaatioseura ry.

Siemens product support. Viitattu 24.9.2010. http://support.automation.siemens.com Product information, Automation Technology, Operator control and monitoring, HMI Software, SCADA system SIMATIC WinCC, SIMATIC WinCC, Global Script

LIITTEET

Liite 1. Esimerkki toimilohkon klikkauksessa suoritettavasta VBS – koodista

```
Sub OnClick(ByVal Item)
HMIRuntime.ActiveScreen.ScreenItems("motor_ctrl_group").Visible = True
HMIRuntime.ActiveScreen.ScreenItems("ctrlbase").Visible = True
HMIRuntime.ActiveScreen.ScreenItems("valve_interlock_group").Visible = False
HMIRuntime.ActiveScreen.ScreenItems("motor_interlock_group").Visible = False
HMIRuntime.ActiveScreen.ScreenItems("pump_interlock_group").Visible = False
HMIRuntime.ActiveScreen.ScreenItems("valve_ctrl_group").Visible = False
HMIRuntime.ActiveScreen.ScreenItems("pump_ctrl_group").Visible = False

Dim pos_text
Dim strConc
Dim status
Dim modefield
Dim statusfield
Dim forcefield
Dim positio
Dim intlock1
Dim intlock2
Dim intlock3
Dim intlock4
Dim intlock5
Dim symbolicloopname

Set pos_text = HMIRuntime.ActiveScreen.ScreenItems("mtr7") 'MUUTTUJA!
Set forcefield = HMIRuntime.ActiveScreen.ScreenItems("motor_last_interlock")
Set symbolicloopname = HMIRun-
time.ActiveScreen.ScreenItems("symbolic_loopname_ctrl")
```

```
strConc = pos_text.Text & "_state"

Set status = HMIRuntime.Tags(strConc)
status.Read

Set positio = HMIRuntime.Tags("POSITIO")
positio.Write pos_text.Text

Set modefield = HMIRuntime.ActiveScreen.ScreenItems("motormode")
Set statusfield = HMIRuntime.ActiveScreen.ScreenItems("motorstate")

Set intlock1 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock1")
Set intlock2 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock2")
Set intlock3 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock3")
Set intlock4 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock4")
Set intlock5 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock5")

'--------------------------------------------------------------------------'
'--------------------------------------------------------------------------'
HMIRuntime.Tags("HMIActive").Write 10 'MERKITSE TÄHÄN KO. TOIMILAITTEEN
INDEKSI!
'MERKITSE TÄHÄN KO. TOIMILAITTEEN LUKITUKSET!                    '
intlock1.Text = "interlock1"                                     '
intlock2.Text = "interlock2"                                     '
intlock3.Text = "interlock3"                                     '
intlock4.Text = "interlock4"                                     '
intlock5.Text = "interlock5"                                     '
symbolicloopname.Text = "" 'MERKITSE TÄHÄN KO. TOIMILAITTEEN KUVAUS!
'
'--------------------------------------------------------------------------'
'--------------------------------------------------------------------------'
```

```
'luetaan moottorin status -tagia ja valitaan moottorikuvake ja M/A -arvo tagin arvon
mukaan

If (status.Read And 2^0) = 0 Then

modefield.Text = "Manual"

Elseif (status.Read And 2^0) = 1 Then

modefield.Text = "Auto"


End If


If (status.Read And 255) = 0 Then

With statusfield

.Text = "Fault"

.BackColor = RGB(255,0,0)

End With


Elseif (status.Read And 255) = 1 Then

With statusfield

.Text = "Fault"

.BackColor = RGB(255,0,0)

End With


Elseif (status.Read And 255) = 2 Then

With statusfield

.Text = "Stopped"

.BackColor = RGB(182,182,182)

End With


Elseif (status.Read And 255) = 3 Then

With statusfield

.Text = "Stopped"

.BackColor = RGB(182,182,182)
```

```
End With

Elseif (status.Read And 255) = 4 Then
With statusfield
.Text = "Running"
.BackColor = RGB(0,255,0)
End With

Elseif (status.Read And 255) = 5 Then
With statusfield
.Text = "Running"
.BackColor = RGB(0,255,0)
End With
Elseif (status.Read And 255) >= 6 Then
With statusfield
.Text = "Fault"
.BackColor = RGB(255,0,0)
End With
End If

If (status.Read And 2^8) = 256 Then
forcefield.Text = "FORCED"
pos_text.FlashForeColor = True
End If

If (status.Read And 2^10) = 1024 Then
forcefield.Text = "FORCED"
pos_text.FlashForeColor = True
End If

If (status.Read And 2^12) = 4096 Then
```

```
forcefield.Text = "FORCED"
pos_text.FlashForeColor = True
End If

If (status.Read And 2^14) = 16384 Then
forcefield.Text = "FORCED"
pos_text.FlashForeColor = True
End If

If (status.Read And 2^16) = 65536 Then
forcefield.Text = "FORCED"
pos_text.FlashForeColor = True
End If

If (status.Read And 2^8) = 256 Then
pos_text.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^10) = 1024 Then
pos_text.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^12) = 4096 Then
pos_text.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^14) = 16384 Then
pos_text.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^16) = 65536 Then
pos_text.FlashForeColor = True
forcefield.Text = "FORCED"
Else
pos_text.FlashForeColor = False
```

```
forcefield.Text = ""
End If
Set pos_text = Nothing
Set strConc = Nothing
Set status = Nothing
Set modefield = Nothing
Set statusfield = Nothing
Set forcefield = Nothing
Set positio = Nothing
Set intlock1 = Nothing
Set intlock2 = Nothing
Set intlock3 = Nothing
Set intlock4 = Nothing
Set intlock5 = Nothing
Set symbolicloopname = Nothing
End Sub
```

Liite 2. Esimerkki toimilaitteen tilamuutoksesta suoritettavasta VBS –koodista

```
Sub OutputValue_OnPropertyChanged(Byval Item, Byval value)
'alustetaan tarvittavat muuttujat

Dim positio
Dim mtr_pic
Dim strConc
Dim status
Dim mtrmode
Dim pos_text
Dim modefield
Dim statefield
Dim intlocktext1
Dim intlocktext2
Dim intlocktext3
Dim intlocktext4
Dim intlocktext5
Dim forcefield
Dim intwarn1
Dim intwarn2
Dim intwarn3
Dim intwarn4
Dim intwarn5

'merkitään positio -muuttujaan moottorin tekstikentässä oleva tunnus
Set positio = HMIRuntime.ActiveScreen.ScreenItems("mtr7")

'kirjoitetaan tagitunnukseen tekstikentän positio ja loppuosa
strConc = positio.Text & "_state"

'merkitään moottorikuvakkeen muuttujaksi kyseisen venttiilin kuva
```

```
Set mtr_pic = HMIRuntime.ActiveScreen.ScreenItems("mtr7pic")

'merkitään status -muuttujaan lopullinen luotu merkkijono viittaamaan
'kyseisen moottorin status -tagiin
Set status = HMIRuntime.Tags(strConc)
status.Read

'merkitään mtrmode -muuttujaan moottorin M/A tunnus
Set mtrmode = HMIRuntime.ActiveScreen.ScreenItems("mtr7mode")

'luetaan moottorin status -tagia ja valitaan moottorikuvake ja M/A tunnus tagin arvon mukaan
If (status.Read And 2^0) = 0 Then
mtrmode.Text "M"

Elseif (status.Read And 2^0) = 1 Then
mtrmode.Text "A"

End If

If (status.Read And 255) =  0 Then
mtr_pic.PictureName = "mtr_failure.bmp"

Elseif (status.Read And 255) = 1 Then
mtr_pic.PictureName = "mtr_failure.bmp"

Elseif (status.Read And 255) =2 Then
mtr_pic.PictureName = "mtr_stop.bmp"

Elseif (status.Read And 255) = 3 Then
mtr_pic.PictureName = "mtr_stop.bmp"

Elseif (status.Read And 255) = 4                Then
mtr_pic.PictureName = "mtr_run.bmp"
```

```
Elseif (status.Read And 255) = 5 Then
mtr_pic.PictureName = "mtr_run.bmp"

Elseif (status.Read And 255) >= 6 Then
mtr_pic.PictureName = "mtr_failure.bmp"



End If

'luetaan ko. moottorin statusta ja kirjoitetaan Manual/Auto sekä tilatekstit ohjausikkunaan
VAIN MIKÄLI SISÄINEN POSITIO
' -tagi sisältää ko. moottorin position!
Set pos_text = HMIRuntime.Tags("POSITIO")
pos_text.Read

If pos_text.Value = positio.Text Then

        Set modefield = HMIRuntime.ActiveScreen.ScreenItems("motormode")
        Set statefield = HMIRuntime.ActiveScreen.ScreenItems("motorstate")

        If (status.Read And 2^0) = 0 Then
        modefield.Text "Manual"

        Elseif (status.Read And 2^0) = 1 Then
        modefield.Text "Auto"

        End If

        If (status.Read And 255) = 0 Then
        With statefield
        .Text = "Fault"
        .BackColor = RGB(255,0,0)
```

```
End With

Elseif (status.Read And 255) = 1 Then
With statefield
.Text = "Fault"
.BackColor = RGB(255,0,0)
End With
Elseif (status.Read And 255) = 2 Then
With statefield
.Text = "Stopped"
.BackColor = RGB(182,182,182)
End With
Elseif (status.Read And 255) = 3 Then
With statefield
.Text = "Stopped"
.BackColor = RGB(182,182,182)
End With
Elseif (status.Read And 255) = 4 Then
With statefield
.Text = "Running"
.BackColor = RGB(0,255,0)
End With
Elseif (status.Read And 255) = 5 Then
With statefield
.Text = "Running"
.BackColor = RGB(0,255,0)
End With
Elseif (status.Read And 255) >= 6 Then
With statefield
.Text = "Fault"
.BackColor = RGB(255,0,0)
End With
End If
```

```
'lukitusten käsittely, kirjoitetaan ohjausikkunaan lukitustiedot mikäli ko.
toimilaite on valittuna

'luetaan statustagin bittejä ja merkitään lukitukset ja viimeisin lukitus näkyviin.


Set intlocktext1 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock1")
Set intlocktext2 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock2")
Set intlocktext3 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock3")
Set intlocktext4 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock4")
Set intlocktext5 = HMIRuntime.ActiveScreen.ScreenItems("motor_interlock5")
Set forcefield = HMIRuntime.ActiveScreen.ScreenItems("motor_last_interlock")
Set intwarn1 = HMIRuntime.ActiveScreen.ScreenItems("motor_intlock_warn1")
Set intwarn2 = HMIRuntime.ActiveScreen.ScreenItems("motor_intlock_warn2")
Set intwarn3 = HMIRuntime.ActiveScreen.ScreenItems("motor_intlock_warn3")
Set intwarn4 = HMIRuntime.ActiveScreen.ScreenItems("motor_intlock_warn4")
Set intwarn5 = HMIRuntime.ActiveScreen.ScreenItems("motor_intlock_warn5")

If (status.Read And 2^8) = 256 Then
intlocktext1.ForeColor = RGB(255,0,0)
forcefield.Text = "FORCED"
positio.FlashForeColor = True
Else
intlocktext1.ForeColor = RGB(148,148,148)
End If

If (status.Read And 2^10) = 1024 Then
intlocktext2.ForeColor = RGB(255,0,0)
forcefield.Text = "FORCED"
positio.FlashForeColor = True
Else
intlocktext2.ForeColor = RGB(148,148,148)
End If

If (status.Read And 2^12) = 4096 Then
```

```
intlocktext3.ForeColor = RGB(255,0,0)
forcefield.Text = "FORCED"
positio.FlashForeColor = True
Else
intlocktext3.ForeColor = RGB(148,148,148)
End If
If (status.Read And 2^14) = 16384 Then
intlocktext4.ForeColor = RGB(255,0,0)
forcefield.Text = "FORCED"
positio.FlashForeColor = True
Else
intlocktext4.ForeColor = RGB(148,148,148)
End If
If (status.Read And 2^16) = 65536 Then
intlocktext5.ForeColor = RGB(255,0,0)
forcefield.Text = "FORCED"
positio.FlashForeColor = True
Else
intlocktext5.ForeColor = RGB(148,148,148)
End If
If (status.Read And 2^8) = 256 Then
positio.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^10) = 1024 Then
positio.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^12) = 4096 Then
positio.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^14) = 16384 Then
positio.FlashForeColor = True
forcefield.Text = "FORCED"
Elseif (status.Read And 2^16) = 65536 Then
```

```
positio.FlashForeColor = True

forcefield.Text = "FORCED"

Else

positio.FlashForeColor = False

forcefield.Text = ""

End If

' "viimeisen hälytystiedon" käsittely, logiikka settaa molemmat hälytystä koskevat bitit, jolloin
generoidaan sekä hälytysteksti, että

' viimeisestä hälytyksestä kertova varoituskolmio, uuden hälytyksen generoindi resettaa kaikki
muut "viimeisen hälytyksen" bitit


If (status.Read And 2^9) = 512 Then

If HMIRun-
time.ActiveScreen.ScreenItems("motor_ctrl_group").Visible = False Then

If HMIRun-
time.ActiveScreen.ScreenItems("motor_interlock_group").Visible = True Then

intwarn1.Visible = True

Else

intwarn1.Visible = False

End If

Else

intwarn1.Visible = False

End If

Else

intwarn1.Visible = False

End If

If (status.Read And 2^11) = 2048 Then

If HMIRun-
time.ActiveScreen.ScreenItems("motor_ctrl_group").Visible = False Then

If HMIRun-
time.ActiveScreen.ScreenItems("motor_interlock_group").Visible = True Then

intwarn2.Visible = True

Else

intwarn2.Visible = False

End If
```

```
                    Else
                        intwarn2.Visible = False
                    End If
                Else
                    intwarn2.Visible = False
                End If
                If (status.Read And 2^13) = 8192 Then
                    If HMIRun-
time.ActiveScreen.ScreenItems("motor_ctrl_group").Visible = False Then
                        If HMIRun-
time.ActiveScreen.ScreenItems("motor_interlock_group").Visible = True Then
                            intwarn3.Visible = True
                        Else
                            intwarn3.Visible = False
                        End If
                    Else
                        intwarn3.Visible = False
                    End If
                Else
                    intwarn3.Visible = False
                End If

                If (status.Read And 2^15) = 32768 Then
                    If HMIRun-
time.ActiveScreen.ScreenItems("motor_ctrl_group").Visible = False Then
                        If HMIRun-
time.ActiveScreen.ScreenItems("motor_interlock_group").Visible = True Then
                            intwarn4.Visible = True
                        Else
                            intwarn4.Visible = False
                        End If
                    Else
                        intwarn4.Visible = False
                    End If
```

```
                Else
                        intwarn4.Visible = False
                End If
                If (status.Read And 2^17) = 131072 Then
                        If HMIRun-
time.ActiveScreen.ScreenItems("motor_ctrl_group").Visible = False Then
                                If HMIRun-
time.ActiveScreen.ScreenItems("motor_interlock_group").Visible = True Then
                                        intwarn5.Visible = True
                                Else
                                        intwarn5.Visible = False
                                End If
                        Else
                                intwarn5.Visible = False
                        End If
                Else
                        intwarn5.Visible = False
                End If
End If
'tyhjennetään muuttujat
Set positio = Nothing
Set mtr_pic = Nothing
Set strConc = Nothing
Set status = Nothing
Set mtrmode = Nothing
Set pos_text = Nothing
Set modefield = Nothing
Set statefield = Nothing
Set intlocktext1 = Nothing
Set intlocktext2 = Nothing
Set intlocktext3 = Nothing
Set intlocktext4 = Nothing
Set intlocktext5 = Nothing
```

```
Set forcefield = Nothing
Set intwarn1 = Nothing
Set intwarn2 = Nothing
Set intwarn3 = Nothing
Set intwarn4 = Nothing
Set intwarn5 = Nothing
End Sub
```