Tomi Välimäki

SIMULOIDUN SEKOITUSPROSESSIN OHJAUS

Sähkötekniikan koulutusohjelma
Automaatiotekniikan suuntautumisvaihtoehto
2010

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SIMULOIDUN SEKOITUSPROSESSIN OHJAUS

Välimäki, Tomi
Satakunnan ammattikorkeakoulu
Sähkötekniikan koulutusohjelma
Heinäkuu 2010
Ohjaaja: Asmala, Hannu
Sivumäärä: 73
Liitteitä:

Asiasanat: simulaatio, WinCC, 3DCreate, prosessin ohjaus

Tämän opinnäytetyön aiheena oli simuloidun sekoitusprosessin ohjaus. Opinnäytetyöhön liittyi projektina simulaatiodemo, joka esiteltiin Satakunnan Ammattikorkeakoulun esittelypisteessä Automaatio-09 messutapahtumassa 23.9-25.9.2009 Helsingin messukeskuksessa.

Simulaatiodemossa yhdistetään todellinen ohjausjärjestelmä simuloituun sekoitusprosessiin. Demo koostui kahdesta tietokoneesta, joista toisessa oli käyttöliittymä, ja toisessa 3D-mallinnettu sekoitusprosessi jota ohjattiin sille rakennetulla käyttöliittymällä. Järjestelmä toteutettiin täysin koulun toimesta Siemensin automaatiojärjestelmillä. Käyttöliittymä toteutettiin Siemensin WinCC valvomo-ohjelmistolla ja saman valmistajan Step 7 logiikkaohjelmalla ja S7-300-logiikalla. Ohjausjärjestelmä koostui edellä mainitusta kokoonpanosta. Siemensin logiikka liitettiin Profibus DP-väylän kautta toiseen tietokoneeseen, jossa varsinainen prosessi oli mallinnettuna.

Opinnäytetyön tekstiosuudessa kerrotaan teoriaa simulaatiosta ja kuvataan demon toteuttamiseen vaadittuja laitteita ja ohjelmistoja, sekä opastetaan niiden käyttöä. Opinnäytetyön lopussa esitetään rakennettu simulaatiodemo ja arvioidaan toteutettua järjestelmää.

CONTROL OF A SIMULATED MIXING PROCESS

Välimäki, Tomi
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Electrical Engineering
July 2010
Supervisor: Asmala, Hannu
Number of pages: 73
Appendices:

Key words: simulation, WinCC, 3DCreate, process control

The subject of this thesis was the control of a simulated blending process. The thesis included as a project a simulation demo that was presented at Satakunta University of Applied Sciences's convention stand at the Automaatio-09 exhibition in the Helsinki Exhibition and Convention Centre on 23rd-25th of September 2009.

In the simulation demo a real control system was combined into a simulated mixing process. The demo consisted of two computers, in which the first had an operating system on it, and the other had the 3D-modeled process that was controlled by the operating system. The system was entirely made at our school with Siemens's automation systems. The operating system was made with Siemens WinCC-software, STEP 7-plc programmer and a S7-300-PLC which were also from Siemens. The control system consisted of the previously mentioned software and hardware. A Siemens's PLC was connected via a Profibus DP-fieldbus to the other computer, in which the process was modelled.

The written part of the thesis consists of theory about simulation, descriptions of the devices and programs that were required in making the demo, and guidance in using the programs. In the final section of the thesis the simulation demo is presented, and the system is evaluated.

TERMILUETTELO

PLC = Programmable Logic Controller (ohjelmoitava logiikka)

OPC = OLE for Process Control (avoimen tiedonsiirron standardi, jota käytetään teollisuuden automaatiosovelluksissa)

OLE = Object Linking and Embedding (Microsoft:in kehittämä tekniikka, joka mahdollistaa sisällyttämisen ja linkittämisen dokumentteihin ja muihin objekteihin)

COM = Component Object Model (Microsoft:in kehittämä binäärirajapintastandardi ohjelmistojen komponenteille)

DCOM = Distributed Component Object Model (Microsoft:in kehittämä tekniikka ohjelmien väliseen kommunikointiin verkon yli)

SCADA = Supervisory Control And Data Acquisition (Tietokoneohjelmisto, joka voidaan käsittää valvomona tai valvomo-ohjelmistona)

HMI = Human Machine Interface (ihmisen ja koneen välinen vuorovaikutus)

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli toteuttaa Satakunnan Ammattikorkeakoululle simuloidun sekoitusprosessin ohjaus. Opinnäytetyöhön kuului työn valvojan Hannu Asmalan tekemään 3D-mallinnettuun prosessiin käyttöliittymän rakentaminen. Käyttöliittymä toteutettiin Siemensin WinCC valvomo-ohjelmiston ja Siemensin S7-300 ohjelmoitavan logiikan avulla. Työ aloitettiin opinnoissa jo tutuksi tulleen Siemens STEP 7-ohjelman parissa. Tämän ohjelman avulla tehtiin prosessiin liitettyyn logiikkaan käyttöliittymän perustana toiminut prosessiohjaus. Tähän logiikkaohjelmaan liityttiin WinCC:stä, jonka avulla prosessia lopulta ohjattiin tietokoneen näytöltä. Projektin valmistuttua se esiteltiin Satakunnan Ammattikorkeakoulun esittelypisteessä Automaatio-09 messutapahtumassa Helsingissä.

Opinnäytetyön tekstiosuudessa käsitellään aluksi teoriaa simulaatiosta, järjestelmässä käytetyistä ohjelmista, laitteista ja rajapinnoista. Teoriaosuuden jälkeen kuvataan tarkemmin projektissa käytettyjen ohjelmien käyttöä. Opinnäytetyö keskittyy suurimmilta osin WinCC:llä työskentelyyn ja sen ominaisuuksiin.

2 SIMULAATIO

Asioiden oppiminen on nykymaailmassa jokapäiväinen asia. Kirjoja lukemalla pääsee toki pitkälle, mutta käytännön kokemusta parempaa opinahjoa ei tähän mennessä ole keksitty. Nykyajan insinööreiltä vaaditaan yhä enemmän ja enemmän koulutusta ja osaamista valmistumisen jälkeen, mutta tosiasia on se, ettei kaikkea voi tietää koulupenkiltä päästyään. Ajatellaan tilannetta, jossa insinööri saa työpaikan valmistuttuaan ydinvoimalan käyttöhenkilönä. Kuinka moni insinööri osaisi käyttää kyseisenlaista järjestelmää suoraan valmistuttuaan? Sanoisin ettei yksikään. Näin suurista järjestelmistä puhuttaessa simulaatiosta on muodostunut korvaamaton apu nykyajan työelämän koulutuksessa. Kokemattomat ja uudet työntekijät pääsevät turvallisesti tutustumaan ja opettelemaan heille täysin uusia asioita ilman mittavia vahinkoja itselleen, puhumattakaan muista ihmisistä.

Tälle vuosituhannelle mentäessä simulaatio on tietokoneiden kehittymisen ja virtuaalimaailman tuomien mahdollisuuksien myötä kasvanut entisestään. Tänä päivänä melkein mitä tahansa tapahtumaa tai asiaa pystytään simuloimaan hyvin tarkasti tietokoneiden avulla. Ongelmana tietokoneiden käyttöön perustuvassa simulaatiossa on kuitenkin se, ettei asioita aina voida tehdä täysin todenmukaiseksi. Siksi usein simulaatiossa pyritään keskittymään vain tarpeellisiin asioihin, eikä aina lähdetä hakemaan 100%:sta tarkkuutta sen toteutuksessa.

Simulaation hyötyinä voidaan ajatella, ettei todellisia järjestelmiä tarvitse välttämättä kuormittaa tai töissä olevia henkilöjä asettaa vaaraan testauksen aikana. He voivat jatkaa normaaleja työtehtäviään ja testaushenkilöstö pääsee koko ajan testaamaan tuotantolaitoksen laitteita. Tästä ei yrityksille seuraa mittavia tappioitakaan sillä prosesseja ei tarvitse ajaa alas tutkimusten ajaksi.

2.1 Simulaation luokittelu

Simulaatio on käsitteenä varsin laaja, sillä esim. edellä mainittu koulutussimulaatio on vain yksi simulaation tyypeistä. Koulutussimulaatio itsessään voidaan jakaa osiin riippuen minkä tyylistä koulutusta käydään, esim. virtuaalinen ja todellinen koulutus. Virtuaalista koulutusta voisi olla tehtaan käyttö simuloidussa 3d-ympäristössä, kun taas todellista koulutusta voisi olla esim. poliisien harjoittelu rikollisten taltuttamiseen simuloidussa tilanteessa.

Simulaatio voidaan jakaa kolmeen pääryhmään. Jako tapahtuu simulaation tavoitteiden avulla siten, että mitä simulaatiolla pyritään saamaan aikaan ja miten se toteutetaan. Näitä ryhmiä ovat opetus ja koulutus-, tutkimus ja kehitys-, ja hankinta ja hyväksyntäsimulaatio. /1/

2.1.1 Opetus ja koulutus

Opetus- ja koulutussimulaatioilla on monia eri tavoitteita, joista ne voidaan sekä yhdistää toisiinsa, että erottaa toisistaan. Näillä on kuitenkin yksi selkeä päämäärä: saada aikaan tietoutta tietystä prosessista, välineestä tai metodista yksilölle tai yksilöille

tavoitteena parantaa yksilön suorituskykyä simuloitua asiaa kohtaan. Opetussimulaatioita käytetään edesauttamaan laitteen tai järjestelmän ymmärtämistä. /1/ Esimerkkinä voisi ajatella vaikkapa tuotantolinjastoa, jossa käyttöhenkilöstölle pyritään selvittämään kuinka heidän tulisi järjestelmän eri tilanteissa toimia, ja miten eri tilanteet ilmenevät järjestelmässä. Tämänkaltainen simulaatio voi myös olla käytössä pelkästään tutustuttamaan käyttöhenkilöstö järjestelmän toimintaan ja sen käyttämiseen.

2.1.2 Tutkimus ja kehitys

Tämän luokan simulaatioiden tarkoitus on arvioida prosesseja tai prosessien osia, sekä parantaa niiden ominaisuuksia. Tutkimus ja kehitys-luokka on suurin simulaation ala, ja samalla monikäsitteisin, sillä tähän luokkaan luokitellut simulaatiot voidaan käsittää myös muina luokkina. Esimerkkinä tämänkaltaisista simulaatioista voi mainita Thomas Schulzen (1993) tutkimuksen ”Simulation of streetcar and bus traffic”, jossa hän kuvaa järjestelmän rakentamista, jonka avulla pystytään tutkimaan henkilöautojen ja bussiliikenteen aikatauluja, matkustajien vaihtoaikoja eri henkilöautokaistojen välillä, henkilöautoliikenteen vaikutusta busseihin, bussien vaihtoaikoja riippuen päivänajasta, sekä uusien reittien mahdollista parantavaa vaikutusta. Tämänkaltaisissa simulaatioissa yhtenäisenä tekijänä on se, että niillä kaikilla pyritään tutkimaan jotain simuloitua tilannetta, jotta suunniteltuja muutoksia voidaan tarkastella niiden toimivuuden kannalta. /1/

2.1.3 Hankinta ja hyväksyntä

Kuten edellä Opetus- ja koulutusluokan simulaatioissa, tässä luokassa voidaan havaita selkeät erot hankinnan ja hyväksynnän tavoitteiden välillä, ja samalla todeta niiden oleva samankaltaiset. Tämän luokan simulaatiot ovat tarkoitettu arvioimaan prosesseja tai tuotteita, soveltuvatko nämä joihinkin vaatimuksiin, tai todentamaan että laitteet täyttävät ennalta määrätyt vaatimukset. Esimerkki hankintasimulaatiosta voisi olla, että ostoa suunnitteleva yritys haluaa saada tuotetta myyvältä yritykseltä simulaatiomallin järjestelmästä ennen hankintapäätöksen tekemistä. Hyväksyntäsimulaatiossa taas pyritään tutkimaan täyttyvätkö jollekin laitteelle sille ennalta määrätyt asetukset. /1/

2.2 Simulaation tavoitteet

Kaikilla simulaatioluokilla on yhteisenä tavoitteena luoda tarkkuutta ongelma-alueen simuloinnissa. Tämä ilmenee eri alojen miljoonasopimuksista, jotka saattavat olla riippuvaisia hankinta- ja hyväksyntä simulaatioiden tuloksista. Mallien tarkkuus on monilla aloilla, kuten sotateollisuus ja lääkintäteollisuus todella tärkeää, koska esim. näillä aloilla käsitellään elämää ja kuolemaa koskevia asioita. On sanomattakin selvää kuinka tärkeää simuloinnin tarkkuus ja todenmukaisuus näissä on. Simulaation todenmukaisuudessa on tärkeää ajatella saatua tulosta myös sen vaatimiin kustannuksiin. Yritysten halutessa tarkempia malleja on huomioitava samanaikainen simulaatiomallin kustannusten kasvu. Tästä johtuen jo simulaation suunnitteluvaiheessa on kiinnitettävä huomiota siihen mikä on tarpeellista ja mikä ei. Muita yhtenäisyyksiä eri luokilla on kyky luoda mahdollisuuksia käyttäjille tehdä eri valintoja asioiden suhteen ennen niiden hyödyntämistä tosielämään. Tätä ominaisuutta voidaan ajatella yhtenä simulaation parhaista vahvuuksista. /1/

2.2.1 Opetus ja koulutus

Eräs opetus- ja koulutussimulaation tavoite on antaa käyttäjälle ymmärrystä järjestelmästä, laitteesta tai tilanteesta. Usein nämä simulaatiot tarjoavat harjoitusta ja kokemusta harjoittelijalle auttaakseen häntä päämäärässään. Selkeässä opetusmallissa looginen idea simulointiympäristössä on parantaa henkilön suorituskykyä ja arvioida hänen kehitystään. Jos tällaisessa tilanteessa yksilön tai ryhmän taitoja pystytään mittaamaan ennen ja jälkeen harjoituksen toteuttamista, tavoitteena voidaan pitää henkilön tai ryhmän taitojen kehitystä. Tässä luokassa voidaan myös huomata että simulaatiomallin tarkkuutta voidaan vähentää ilman, että tavoiteltu lopputulos kärsii huomattavasti. Yleensä tämän luokan simulaatioissa pyritään karsimaan mallin todenmukaisuutta, jotta simulaation kustannuksissa saadaan säästettyä. Tällaisissa tilanteissa simulaatioon osallistuvien tulee huomioida simulaatiomallin puutteet ja toimia niiden mukaisesti. Käytännössä tämä on hyvä keino säästää kustannuksissa ja ylläpitää tehokasta harjoitusta. /1/

2.2.2 Tutkimus ja kehitys

Tämän luokan päätavoitteena on arvioida prosessin suorituskykyä. Tässä luokassa prosessi on laaja käsite, joka voi olla vaikkapa ohjelmiston osa, tuotantolaitoksen osa, pikaruokalan tuottavuus tai samaisen ruokalan tehokkuus henkilöstön määrän ja myynnin suhteen yms. Yleisesti tässä luokassa pyritään tekemään vertailukelpoista tutkimusta jostain prosessista, jotta tutkimustuloksia pystyttäisiin käyttämään muissa samankaltaisissa tilanteissa vertailuna tai mahdollisena parannuskeinona. /1/

2.2.3 Hankinta ja hyväksyntä

Näitä simulaatiomalleja käytetään lähes ainoastaan valitsemaan jokin tuote rajatuista vaihtoehdoista. Simulaatiota saatetaan tarvita esimerkiksi tilanteessa, jossa halutaan tietää miten jokin laitteiston osa käyttäytyy tietyssä tilanteessa ja onko sen käyttäytyminen hyväksyttävää. Tarvittavat mittaukset voidaan täten tehdä ilman laitteiston varsinaista testaamista fyysisesti vastaavissa olosuhteissa, sekä olosuhteita voidaan tarvittaessa toistaa tai muuttaa, jotta saadaan luotua luotettavaa tietoa laitteiston toiminnasta. /1/

Yksi hankintasimulaation tavoite on arvioida prosessin tai tuotteen suorituskykyä. Tämä tavoite on samankaltainen Tutkimus- ja kehitysluokan kanssa, mutta erona on se, että järjestelmää arvioidaan sen mukaan halutaanko sellainen hankkia vai ei. /1/

Hyväksyntä- ja hankintasimulaation suurin ero on ajoituksessa. Usein hyväksyntäsimulaatiossa hyödynnetään ennalta määrättyjä kriteerejä, jotka saattavat olla peräisin hankintasimulaation tuloksista. Päätavoite tällaisella simulaatiolla on vahvistaa tilatun prosessin tai tuotteen suorituskyky. Hyväksyntäpäätöksen tulee perustua simulaation osan toimivuuteen kokonaisessa järjestelmässä. /1/

2.3 Simulaation toteutustavat

Simulaatiota voidaan toteuttaa monin eri tavoin ja yleispäteviä apuohjelmia on laajalti saatavilla yrityksiltä jotka ovat keskittyneet tähän alaan. Hankaluutena näissä ohjelmissa on se, etteivät ne ole kovinkaan keskitettyjä mihinkään tietyntyyppisen simulaation mallintamiseen. /1/

Opetus- ja koulutuskäyttöön ei yleisesti tarvita suurta tarkkuutta laskelmissa. Toisaalta mallintaminen saattaa vaatia erikoislaitteistoja ja ohjelmia riippuen halutuista yksityiskohdista ja realismista jota kyseisessä simulaatiossa vaaditaan. Todenmukaisen simulaation toteuttamiseen saatetaan vaatia visualisointia, ääntä, 3D-mallinnusta, puhetta, sekä laitteistoja ja ohjelmia. /1/

Tutkimus- ja kehityssimulaatiossa käytetään usein toteutuskeinoja, jotka eivät vaadi simulaatiomallilta kovin tarkkaa toteutusta (esim. 3D-mallin ei tarvitse olla täysin realistinen). Vain mallin ulosannin tulee olla tarkkaa hyvä tutkimusdatan aikaansaamiseksi. Yleisiä keinoja on simulaatiomallista mitattujen arvojen taulukoiminen tai arkistointi, ja niihin kerätyn datan ja tilastojen tutkiminen. Tilastoinnin avulla pystytään havainnoimaan järjestelmässä harvoin tapahtuvia vikatilanteita ja saadaan kartoitettua niiden ilmenemisen mahdollisuuksia. Pitkäaikaisten mittausten ja niiden tilastoinnin avulla saadaan määriteltyä kuinka luotettavasta laitteistosta on kyse. /1/
Hankinta- ja hyväksyntäsimulaatiot vaativat usein korkeatasoista tarkkuutta. Tästä johtuen järjestelmissä vaaditaan laitteistojen hyvää suorituskykyä. Lisäksi usein tarvitaan monia lisälaitteita, kuten servoja, signaalien luontia tai mittauslaitteita. /1/

2.4 Emulaatio

Simulaatio on tärkeä työkalu teollisuudessa uusien järjestelmien suunnittelussa. Monesti simulaation avulla yritykset säästävät paljon aikaa ja rahaa, koska simulaatiosta saatua tutkimusdataa päästään hyödyntämään todellisten järjestelmien suunnittelussa. Ajatellaan tilannetta, jossa yritys tekee simuloidun prosessin, jonka avulla käyttäjät pääsevät tutkimaan ohjausjärjestelmän toimivuutta keinotekoisessa ympäristössä jos-

ta saatua tietoa päästään hyödyntämään todelliseen järjestelmään. Tutkimustiedon hyödyntämisessä ilmenee kuitenkin usein ongelmia. On erittäin vaikeaa toteuttaa simulaatiomallia, jota voidaan verrata suoraan todelliseen ympäristöön. Tästä johtuen käyttäjät joutuvat tekemään suunnittelutyön kahteen kertaan sekä simulaatiomallissa, että todellisessa järjestelmässä. Tämän takia käyttäjät pyrkivät hyödyntämään keinoja jotka pyrkivät vähentämään tarpeetonta työtä, epäilyjä järjestelmän toimivuudesta, sekä karsimaan ylimääräisiä kustannuksia todellisen järjestelmän toteuttamisessa. /2/

Edellä mainitussa tilanteessa voidaan ajatella että on olemassa kaksi vaihtoehtoa: luoda ohjausjärjestelmä, tai käyttää todellista ohjausjärjestelmää, jolla määritellään tarvittavat ohjaustoiminnot simulaatiomallissa. Ongelmana molemmissa vaihtoehdoissa on se, että simulaatiomalli ja ohjausjärjestelmä eroavat toisistaan kahdella melko tärkeällä yleistoiminnallisella tavalla. Simulaatiotuotteiden tulee olla mahdollisimman joustavia, jonka ansiosta mallit ovat yleensä helposti muokattavissa ja rakennettavissa. Simulaatiomallien tavoitteena on luoda kokeita ja testitilanteita mahdollisimman paljon nopeassa ajassa, jotta kyseistä tilannetta voidaan toistaa mahdollisimman useaan kertaan nopeassa ajassa. Ohjausjärjestelmiä taas ei rakenneta tällaiseen toimintaan, vaan ne ovat usein suunniteltu toimimaan reaaliajassa mahdollisimman tehokkaasti ja luotettavasti. Ongelmaksi tässä tulee se, että jos reaaliaikaista ohjausjärjestelmää yritetään hyödyntää nopeassa simulaatiomallissa, ei tutkimusdataan voida luottaa, kun taas soveltamalla simulaatiomalli reaaliaikaiseksi vastaamaan ohjausjärjestelmän toiminta-aikaa, koko järjestelmästä tulee hidas, eikä sitä voida tehokkaasti tutkia. /2/

Tällaisia ohjausjärjestelmien testaustilanteita varten on olemassa kolmas vaihtoehto, jonka avulla simulaatiomallin ja ohjausjärjestelmän ristiriidat saadaan ratkaistua. Tätä vaihtoehtoa kutsutaan emulaatioksi. Simulaatiomalli on järjestelmä, jota voidaan esim. kuormittaa, ja samalla analysoida kuormittamisen vaikutusta toimivuuteen. Jos malli on tarpeeksi todenmukainen, siitä saatuja mittaustuloksia voidaan tarvittaessa hyödyntää todellisessa järjestelmässä. Simulaatio auttaa käyttäjiä tutkimaan laitteita tai niiden osia, jonka ansiosta käyttäjät pystyvät tekemään perusteltuja päätöksiä ilman todellisen järjestelmän rakentamista. Sen avulla voidaan määrittää asioita joita olisi todellisuudessa vaikeaa ilmentää.

Emulaatiomallien avulla pyritään usein paljon yksinkertaisempiin tavoitteisiin. Niiden tarkoituksena on antaa vastetietoa ohjausjärjestelmälle. Emulaatiomallin voidaan ajatella olevan ohjausjärjestelmää varten rakennettu malli jostakin prosessista, jota ohjausjärjestelmällä ohjataan. Malli luo ympäristön jossa on ennalta määriteltyjä toimintoja ja ominaisuuksia. Malli tuo sisältönsä ulos jonkin standardoidun rajapinnan kautta, jota ohjausjärjestelmä pääsee hyödyntämään. Mikäli emulaatiomallin I/O-maailma toteutetaan todellisessa järjestelmässä olevan rajapintamäärittelyn avulla, ohjausjärjestelmä ei tiedä, että sille tuotu I/O-maailma ei ole todellista. Tämän ansiosta ohjausjärjestelmää päästään rakentamaan suoraan käytettävään laitokseen ilman todellisen järjestelmän kuormittamista.

Ajatellaan esimerkkinä teollista prosessia, johon yritys suunnittelee tekevänsä uuden käyttöliittymän. Kyseisestä prosessista ja sen toiminnoista rakennetaan 3D-malli johon tekeillä oleva ohjausjärjestelmä liitetään testausta varten. Kun järjestelmää on testattu tarpeeksi, voidaan ohjausjärjestelmä liittää todelliseen prosessiin. Emulaatiomalleissa on erittäin epätodennäköistä, että niihin liitetään mitään satunnaisia ilmiöitä mitä prosessissa saattaa mahdollisesti tapahtua, tällaisten ilmiöiden mallintaminen kuuluu pitkälti tutkimus- ja kehitysluokan simulaatioon. Emulaatio keskittyy perustoimintoihin ja niiden mallintamiseen. /2/

2.4.1 Emulaation hyödyt

Emulaatiota hyödynnetään lähinnä peruskäyttäjiin keskittyvissä yrityksissä, sekä järjestelmien yhdistämisessä, esim. uuden käyttöjärjestelmän liittämisessä prosessiin. Emulaatio antaa mahdollisuuden testata uusia käyttöjärjestelmiä ja niiden toimivuutta eri olosuhteissa. Ohjausjärjestelmien testauksessa on tärkeää tutkia miten se käyttäytyy tietyissä erikoistilanteissa, esim. hälytyksissä tai prosessin kuormittamisessa. /2/

Emulaatio tarjoaa turvallisen tutkimisympäristön tällaisissa tilanteissa. Emulaatio on myös erinomainen työkalu järjestelmien käyttökoulutuksessa, sillä se voidaan liittää paitsi käyttöliittymiin, myös HMI:hin (Human Machine Interface). Tämän ansiosta käyttöhenkilöstö saa prosessista koulutusta, jota tosielämässä saattaa olla mahdotonta

saada. 3D-mallinnettua prosessia päästään vapaasti tutkimaan eri kuvakulmista, jolloin käyttäjä saa paremman käsityksen siitä, mitä prosessissa oikeasti tapahtuu. 3D-mallinnuksen avulla voidaan esim. esittää käyttäjälle mitä prosessissa tapahtuu kun valvomosta annetaan komentoja tms. Tällaista kokemusta ja havainnollistamista voi olla tosielämässä hyvin vaikeaa saada, sillä valvomotyöntekijät eivät yleensä näe todellista prosessia valvomossa työskennellessään. /2/

Emulaatiosta on suuri apu myös yrityksille, jotka tekevät tilauksesta uusia käyttö- ja ohjausjärjestelmiä teollisuuden yrityksille. Tilaajalla voi olla todellisesta laitoksesta tehty 3D-mallinnettu prosessi johon yritys tekee käyttöliittymän jota testataan virtuaalisessa ympäristössä. Tämä vähentää todellisessa prosessissa tarvittavaa testaustyötä sekä asennusriskejä, ja ihanteellisessa tilanteessa käyttöliittymä voidaan asentaa suoraan todelliseen ympäristöönsä. Emulaation käytöstä on hyötynä se, että käyttöliittymää päästään todenmukaisessa ympäristössä testaamaan turvallisesti. Uuden käyttöliittymän käyttöönotossa ei välttämättä ole muita vaihtoehtoja kuin että se toimii, sillä pitkät seisokit tehtaissa saattavat tulla yrityksille kalliiksi. Käyttöliittymän toimivuus on vielä tärkeämpää tilanteissa, jossa käyttöliittymä käyttöönotetaan jatkuvasti käynnissä olevaan prosessiin lennosta. Virheet ohjelmoinnissa tai suunnittelussa saattavat aiheuttaa suuria ongelmia näissä tapauksissa. Tästä syystä emulaation hyödyntäminen on kannattavaa, koska edellä mainittuja tilanteita päästään emulaatiomallin avulla testaamaan.

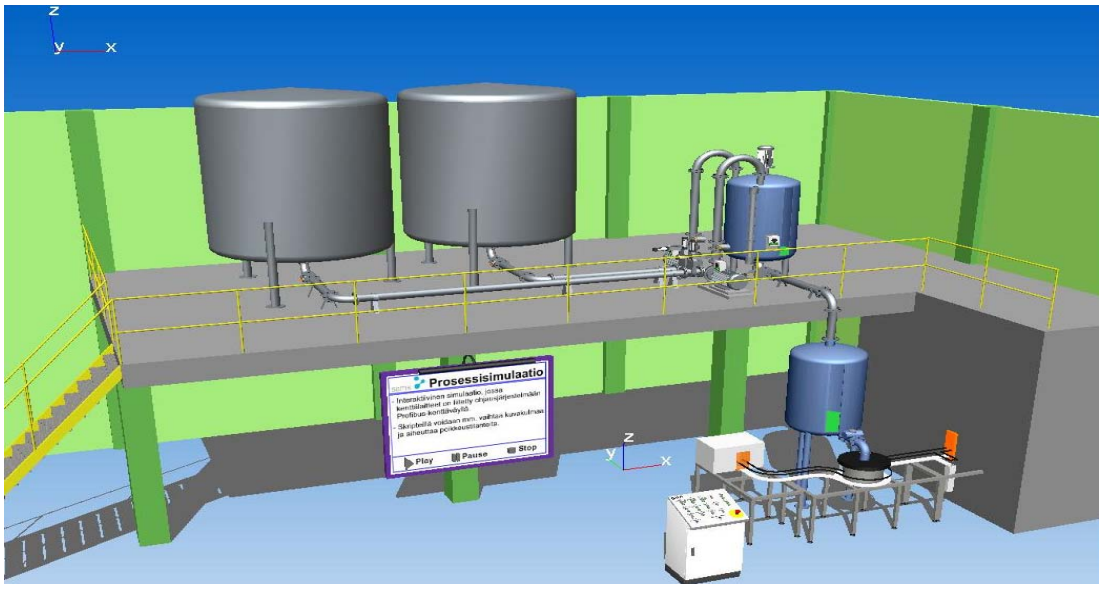
Tällaisilla yrityksillä on usein tiukka aikataulu ja töiden valmistuminen ei aina ole itse yrityksestä itsestään kiinni. Emulaatio mahdollistaa komponenttien ja prosessien testauksen ennen todellisten osien tai laitteistojen valmistumista. Tämän ansiosta monia järjestelmän osia päästään testaamaan edellä aikataulusta, minkä ansiosta tilaajayritykselle voidaan antaa takuita laitteiden toimivuudesta. Emulaatio mahdollistaa myös järjestelmien korjausta jälkeenpäin, mikäli käyttöönoton jälkeen niissä on ilmennyt ongelmia. Emulaatiomallia päästään testaamaan uudelleen ja selvittämään vikaa turvallisesti ilman todellisen prosessin rasittamista. /2/

2.4.2 Emulaatiossa käytettävä tekniikka

Tavallinen tutkittava simulaatiomalli pitää sisällään havainnollistetun fyysisen järjestelmän ja loogiset toiminnot jotka ohjaavat mallia. Emulaatio korvaa ainakin osan näistä toiminnoista todellisella ohjausjärjestelmällä. Jotta emulaatiomallia pystytään ohjaamaan todellisella ohjausjärjestelmällä, mallin ja käyttöliittymän tulee pystyä kommunikoimaan keskenään, sekä mallin tulee todenmukaisesti totella ohjausjärjestelmästä annettavia käskyjä. Jossain tapauksissa on mahdollista sisällyttää ohjauskeinoja mallin sisään toimimaan itsenäisesti. Usein kuitenkin ohjausjärjestelmä on ainoa keino ohjata mallin tapahtumia, ja näiden välille on rakennetta jokin keskusteluyhteys /2/. Opinnäytetyöhön liittynyt simulaatiodemo on juurikin tämän kaltainen järjestelmä ja seuraavissa kappaleissa kerrotaan esimerkki kuinka tällainen järjestelmä voidaan toteuttaa.

3 SIMULOIDUN SEKOITUSPROSESSIN KUVAUS

Opinnäytetyössä toteutettiin 3D-mallinnetun sekoitusprosessin käyttöliittymä, joka esiteltiin Satakunnan ammattikorkeakoulun esittelypisteessä Automaatio-09 messutapahtumassa Helsingissä. Laitteistoltaan opinnäytetyö koostui kahdesta tietokoneesta ja ohjelmoitavasta logiikasta. Toinen tietokone toimi valvomona, josta ohjattiin toisessa tietokoneessa ollutta 3D-mallinnettua sekoitusprosessia. Käyttöliittymänä käytettiin Siemensin WinCC- ja Step7 Simatic Manager ohjelmistoja, joiden avulla ohjattiin Visual Components:in 3DCreate-ohjelmassa mallinnettua prosessia Siemens S7-300 logiikalla Profibus-kenttäväylän ja OPC-rajapinnan kautta. Työssä pyrittiin yhdistämään todellista käyttöliittymää ja ohjausjärjestelmää virtuaaliseen prosessiin. Tässä kappaleessa kuvataan projektin rakennetta ja siinä käytettyjä ohjelmia ja laitteita.

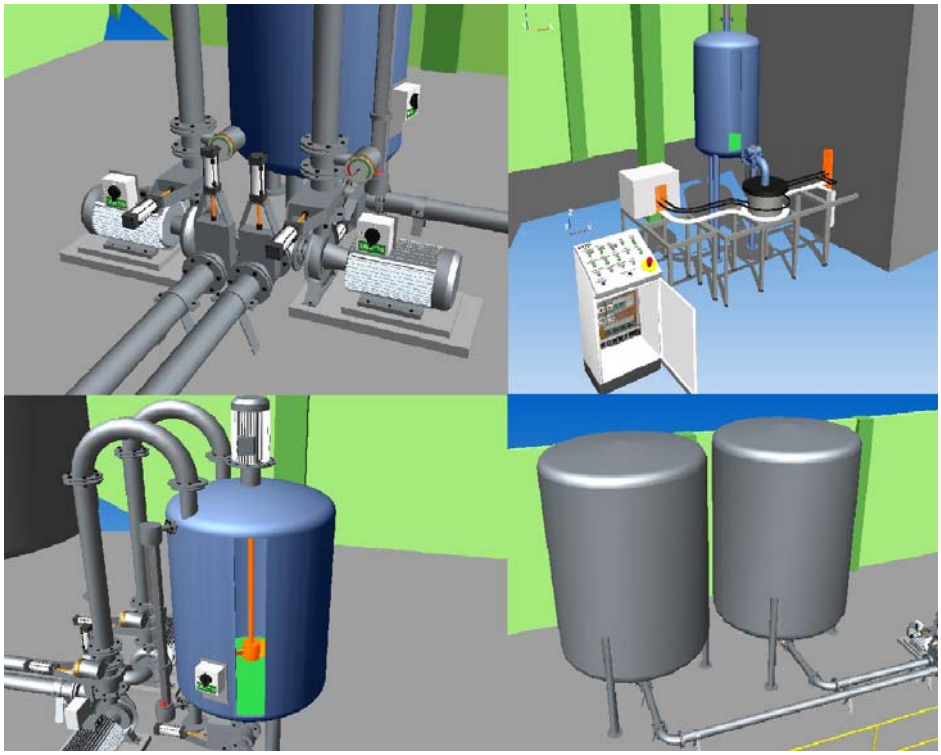


Kuva 3.1 Simuloitu sekoitusprosessi

3.1 Prosessi

Simuloitu sekoitusprosessi koostui neljästä tankista, kahdesta pumpusta, sekoittimesta sekä pullotuslinjastosta. Kaksi isoa tankkia olivat raaka-ainetankkeja, ja kaksi pienempää olivat sekoitustankki ja pullotustankki. Raaka-aineiden A ja B tankeille oli omat pumput, joiden moottorit toimivat ennalta määrätyillä teholla ja nopeudella. Sekoittimen moottori toimi myös samalla tavalla. Moottorien pyörimisnopeutta ei tässä prosessissa voinut ohjata, vaan ne toimivat niille määritellyllä nopeudella ON/OFF-tyyppisesti. Valvomosta voitiin näiden pumppujen toimintoja ohjata. Mikäli jompikumpi pumpuista asetettiin päälle ja pumppujen molemmin puolin olleet venttiilit olivat auki, ainetta alkoi virrata raaka-ainetankeista sekoitustankkiin. Sekoitustankissa aineita voitiin halutessa sekoittaa, ja samaisesta tankista aineiden yhdiste siirrettiin tyhjennysventtiilin kautta pullotustankkiin. Kun pullotustankkiin oli tullut tarpeeksi valmista yhdistettä, prosessi pullotti tankin sisällön ja siirsi pullot eteenpäin (pois prosessista). Pullotus ei ollut valvomosta ohjattavissa, vaan sen toiminnallisuus rakennettiin prosessimallin sisään siten, että pullotus alkoi kun pullotustankkiin tuli ainetta. Simulaatiomallissa oli myös oma ohjauspulpetti, josta prosessin toimintoja voitiin ohjata. Tämän pulpetin sisältä löytyivät moottorien lämpöreleet, jotka halutessa katkaisivat moottorien toiminnan siksi aikaa kunnes lämpöreleet käytiin kuittaa-

massa. Lämpöreleiden lisäksi jokaisen moottorin vierestä löytyi niiden omat turvakytkimet, jotka toimivat samalla tavalla kuin lämpöreleetkin.



Kuva 3.2 Prosessin laitteita

Kuvassa 3.2 oik. ylhäällä näkyy pullotuslinja ja ohjauspulpetti, oik. alhaalla raaka-ainetankit. Vas. ylhäällä on pumput, venttiilit ja turvakytkimet, ja vas. alhaalla sekoitustankki ja sekoitin.

3.2 Valvomo

Edellä esitellyn prosessin ohjaamiseen rakennettiin WinCC:llä graafinen käyttöliittymä. Tämän valvomon avulla prosessin eri toimintoja pystyttiin ohjaamaan. Valvomoon rakennettiin kolme eri käyttö-moodia: Käsiohjaus, Reseptiohjaus ja Pulpettiohjaus. Käyttäjälle muodostuva näkymä prosessista ja sen ohjausvaihtoehdot vaihtelivat valitun moodin mukaan. Yhtenäistä kaikille moodeille oli prosessi-ikkuna. Tästä ikkunasta johon prosessin oli kuvattuna, käyttäjä näki valvomokoneelta sen mitä prosessissa tapahtuu (pumput päällä, tyhjennys päälle, sekoitus päällä jne). Muita valvomossa olevia valvontaikkunoita oli hälytysikkuna (tähän listattiin prosessissa ta-

pahtuneet virhetilat tai hälytykset esim. hätäseis tai lämpöreleen laukeaminen) ja reaaliaikainen trendi, joka piirsi käyrää sekoitustankin pinnankorkeudesta. Enimmäkseen ohjaus-moodit erosivat toisistaan niiden ohjausvalikkojen koostumuksista.

3.2.1 Käsiohjaus

Käsiohjauksessa prosessin toimintoja ohjattiin valvomosta käsin manuaalisesti. Valvomossa olevien painikkeiden avulla voitiin asettaa pumput päälle, sekoitus päälle ja tyhjennys päälle. Painikkeet toimivat prosessille asetettujen sääntöjen mukaan, ja jossain tapauksessa tiettyjen painikkeiden ohjaaminen oli estetty (esim sekoitus päällä → ei voi tyhjentää). Tämän moodin tarkoitus oli tutustuttaa käyttäjää prosessin ohjaamiseen, jotta hän saisi kokonaiskuvan prosessin toiminnasta.

3.2.2 Reseptiohjaus

Reseptiohjauksessa prosessia ohjattiin automaattisesti sitä varten valvomoon rakennetun ”Resepti-Wizardin” avulla. Tämä WinCC:llä rakennettu apuohjelma oli rakenteeltaan samankaltainen kuin esim. Windowsin asennusohjelmat, eli Wizard-ikkunaan tuli valikko johon arvoja sai asettaa, tämän jälkeen painettiin ”Seuraava”-painiketta, josta Wizard siirtyi seuraavaan ikkunaan jne. Aluksi Wizard pyysi käyttäjää asettamaan aineiden A ja B syötettävät ainemäärät ja sekoitusajan. Tämän jälkeen käyttäjältä pyydettiin vahvistusta määrätyistä arvoista, jonka jälkeen prosessi käynnistyi ja Wizardin ikkunaan tuli teksti ”Prosessi käynnissä…”. Tämä teksti pysyi näytöllä niin kauan kuin prosessi oli käynnissä. Kun prosessi oli käyty loppuun, eli pullotus oli saatettu loppuun, Wizard-ikkunaan tuli loppukatsaus pumpatusta ainemäärästä, sekoitusajasta ja pullotettujen pullojen lukumäärästä. Tästä ikkunasta painamalla ”Valmis” päästiin takaisin Reseptiohjauksen alkuvalikkoon josta Wizard voitiin aloittaa uudestaan.

3.2.3 Pulpettiohjaus

Pulpettiohjauksessa prosessia ohjattiin sen omasta pulpetista. Kun pulpettiohjaus valittiin, valvomo kysyi lupaa käynnistää paikallisohjaus. Kun tämä lupa annettiin, prosessin kamera siirtyi ohjauspulpetin lähelle, ja siitä käytiin paikallisohjauskytkin aktivoimassa. Prosessin oman ohjauspulpetin käyttäminen oli messutapahtumaa varten toteutettu valvomosta käsin, sillä pulpetin käyttäminen itse prosessista olisi ollut melko vaikeaa. Pulpettiohjauksessa tämä toiminnallisuus toteutettiin valvomosta käsin siten, että kyseinen pulpetti ja sen painikkeet oli mallinnettu valvomoon, ja kun valvomosta painettiin jotakin painiketta, simulaatiomalli toimi tämän käskyn mukaisesti. Simulaatiomalliin rakennettujen Scriptien avulla prosessin kamera siirtyi pulpettiin, ja prosessi kävi painamassa kyseistä painiketta, jonka jälkeen haluttu toiminto lähti päälle (esim. pumppu A päälle). Pulpettiohjaukseen oli painikkeiden painamisen lisäksi lisätty häiriötilanteiden mallinnus. Valvomosta käsin voitiin määrätä että esim. lämpörele laukeaa, eli kyseistä kytkintä käytiin vääntämässä toiseen asentoon scriptien avulla samoin kuin muiden pulpetin painikkeiden kohdalla. Lisäksi kun kytkin oli käännettynä, prosessin kaikki muut toiminnot oli estettynä, kunnes se käytiin kuittaamassa. Tämä vikatilanne näkyi hälytysikkunassa sekä prosessi-ikkunassa.

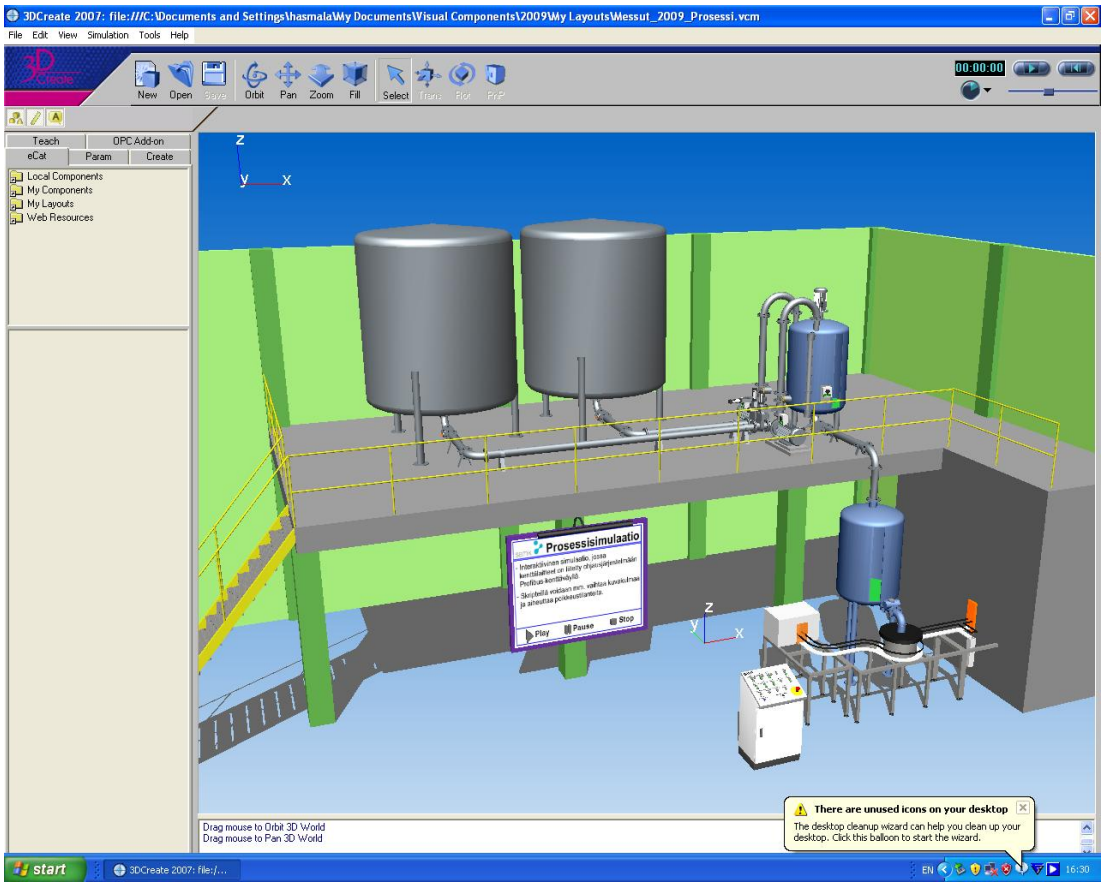
3.3 Prosessitietokoneen ohjelmistot ja laitteet

Prosessitietokoneessa käytettyjä ohjelmia olivat Visual Components Oy:n 3DCreate sekä samaan ohjelmaan tarkoitettu lisäosa PLC Add-On. Tietokoneeseen oli lisäksi liitettynä Molex:in BradCommunications STT Profibus DP kenttäväyläkortti. Prosessi oli mallinnettuna 3DCreate:ssä, ja PLC Add-on ja kenttäväyläkortti mahdollistivat prosessin I/O-maailman siirtämisen logiikalle.

3.3.1 3DCreate

3DCreate on Visual Components Oy:n tekemä ohjelma, jonka avulla käyttäjä voi mallintaa jo olemassa olevia prosesseja sekä tehtaita, tai suunnitella uusia järjestelmiä ilman fyysisen prosessin rakentamista. Ohjelma käyttää olemassa olevia CAD-tietomuotoja joiden avulla prosessin osia voidaan mallintaa ja liittää toisiinsa.

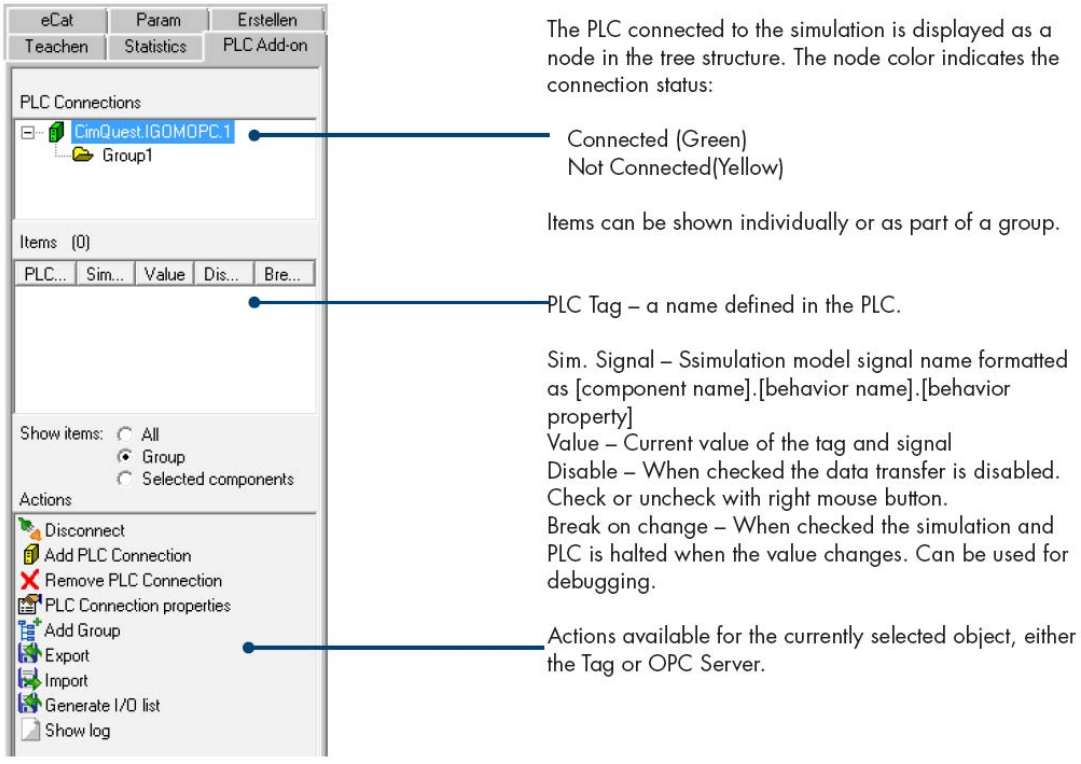
Mallinnetut kohteet voivat olla roboteista ja kuljettimista aina laajoihin tehdaskokonaisuuksiin asti. 3DCreate:n työkalut mahdollistavat sovelluskohtaisten lisäosien asentamisen itse ohjelmaan riippuen prosessista. Näiden lisäosien ansiosta toisiin ohjelmiin on mahdollista liittää 3DCreate-ikkuna josta mallinnettua prosessia voidaan tarkastella esim. valvontatarkoituksissa. Esitystarkoituksissa yritykset voivat tuoda vaikka koko tuotevalikoimansa 3D-mallinnettuna yhdelle näytölle josta asiakkaille voidaan helposti esittää laitteiden toimivuutta ilman todellisten järjestelmien esille tuomista. Ohjelma sisältää COM-rajapinnan, jonka ansiosta sitä voidaan hyödyntää monissa eri kehitysympäristöissä, sekä se voidaan liittää osaksi SCADA, MES ja ERP ympäristöihin PLC Addon-lisäohjelman avulla.



Kuva 3.3 3DCreate

3.3.2 PLC add-on

PLC add-on on 3DCreate:en asennettava lisäohjelma, joka mahdollistaa 3DCreate:n yhdistämisen ohjelmoitaviin logiikoihin. Lisäosa mahdollistaa simulaatiojärjestelmien ohjaamisen reaaliaikaisesti käyttäen OPC-standardiin perustuvaa rajapintamäärittelyä tai PLC-toimittajien laitteistokohtaisia rajapintoja. PLC add-on yhdistää OPC servereissä tai logiikoissa määritettyjä tageja simulaatiomallissa määritettyihin tageihin. Tämä kommunikointi on kaksisuuntaista. Tällä hetkellä Beckhoffin Twincat toimii suorakytkennällä, muut logiikat vaativat OPC-rajapinnan kommunikointiin (ohjelmaan on tulossa liityntä Siemens:in PLCSim:iin). Toiminnaltaan PLC Add-on näkyy 3DCreate ohjelmassa uutena välilehtenä josta lisäosan ominaisuuksia voidaan muokata. /3/



Kuva 3.4 PCL Add-on /3/

3.4 OPC-rajapintamäärittely

Teollisen ohjaus- ja automaatioteknologian kehittyminen kasvaa koko ajan. Koneiden ja laitteiden vaatimukset niiden yhteensopivuudesta, nopeudesta ja yleisestä toimivuudesta kasvavat koko ajan. Samalla on huomattu, että tietokoneiden, internetin ja monien avoimien standardien käyttö osana automaatioteknologiaa tuovat yhä enemmän hyötyä valmistajille. Tietokoneita käytetään kasvavassa määrin asioiden mallintamiseen, tiedon hankkimiseen, prosessien ohjaamiseen ja näiden kohteiden soveltamiseen automaatioalalla. Tietokoneet täydentävät tai jopa korvaavat perinteiset automaatiologiikat ja operointipaneelit. Yksi syy tähän on jatkuvasti massatuotettujen tietokoneiden hintojen lasku, niiden suorituskyvyn jatkuva kasvaminen, tehokkaampien ohjelmistojen ja komponenttien kehitys ja mahdollisuus yhdistää järjestelmiä yleisesti käytössä oleviin järjestelmiin esim. Microsoft Office tuotteisiin. /4/

Maailmalla suuren suosion ja levinneisyyden saavuttaneet Microsoft Windows käyttöjärjestelmät ja niiden Win32-API ominaisuuden ansiosta alettiin kehittää keinoja, joilla ohjelmistot pystyivät kommunikoimaan toistensa kanssa standardisoitujen rajapintojen avulla. Ensimmäinen virstanpylväs näistä oli DDE (Dynamic Data Exchange), jota täydennettiin myöhemmin paremmaksi OLE-järjestelmäksi (Object Linking and Embedding). Kun ensimmäiset tietokoneympäristöön perustuneet HMI ja SCADA (Supervisory Control And Data Acquisition) ohjelmat esiteltiin 1989 ja 1991, DDE:tä käytettiin ensimmäistä kertaa ohjelma-ajurien liitäntänä prosessin rajapintaan. Kuitenkin DDE:ssä oli muutamia epäkohtia erityisesti suorituskykyä tarkasteltaessa. Tästä johtuen HMI-valmistajat alkoivat määritellä DDE:hen tiettyjä laajennuksia tehokkuuden parantamiseksi. Tämän seurauksena syntyi valmistajakohtaisia DDE-sovelluksia, jotka kehitettiin ohjelmakohtaisiksi, eikä niinkään yleisesti yhteensopiviksi muihin ohjelmistoihin. Windows NT:n kehittämisen aikoihin DCOM (Distributed Component Object Model) kehitettiin OLE:n jatkeeksi ja Windows NT otettiin käyttöön laajalti teollisuudessa. Erityisesti nopeasti laajentuneet HMI, SCADA ja DCS (Digital Control System) järjestelmät tukivat NT:tä. /4/

3.4.1 Oliot, komponentit ja COM

Oliopohjaisen ohjelmistokehityksen tuottamista eduista on keskusteltu jo kauan. Ohjelmistotekniikan saralla se on saavuttanut merkittävän aseman ja parantanut ohjelmistojen kehitystä monin eri tavoin. Oliopohjainen kehitys antaa eväät monimutkaisten ohjelmistojen suunnitteluun ja toteuttamiseen, mutta edelleen on paljon ratkaisemattomia kysymyksiä. Yleinen yhteisymmärrys on yhä hukassa siitä, miten olioiden tulisi kommunikoida prosesseille asetettujen rajojen ja verkon ylitse, mikä on edellytys binäärimuotoisten olioiden hajauttamiseen. Tämä standardi on kuitenkin olemassa komponenteille. Toiseksi oliot on toteutettava samalla olio-ohjelmointikielellä jotta ne toimisivat keskenään. Komponentit taas kapseloivat koko toteutuksensa, toteutuskieli mukaan lukien, paljastaessaan rajapintansa. Komponenttitekniikoistaan tunnetuimpia ovat OMG:n (Object Management Group) CORBA (Common Object Request Broker Architecture) ja Microsoftin kehittämä COM (Component Object Model). Kolmas voimakkaasti kehittyvä komponenttitekniikka on Sunin Enterprise JavaBeans, joka tämänhetkisessä tilassaan sopii lähinnä käyttöliittymäkomponenttien rakentamiseen Java-ohjelmointikielisistä komponenteista. /4/

COM on systeemiarkkitehtuuri riippumattomasti tuotetuille ja uudelleenkäytettäville, binäärimuotoisille komponenteille. Sen yleisyyttä on lisännyt Microsoft Windows-käyttöjärjestelmien suosio. Windows-käyttöliittymät sisältävät COM-arkkitehtuuriin tarvittavat systeemipalvelut. Samalla tavalla kuten muissakin komponenttipalveluissa, COM:ssa komponentit julkaisevat toimintonsa rajapinnoilla. COM määrittelee rajapinnan binäärimuodon, joten komponentin toteutus voidaan ohjelmoida millä tahansa kielellä, joka pystyy muodostamaan oikean binäärirakenteen. DCOM (Distributed COM) määrittelee verkkokäytön eri koneilla sijaitsevien komponenttien välille esim. tietokoneesta verkossa sijaitsevalle printterille. /4/

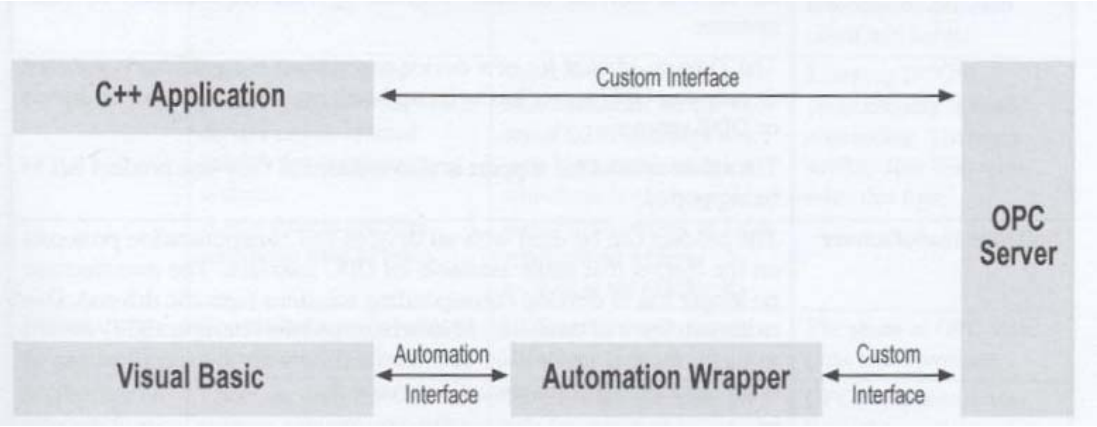
3.4.2 OPC perusteet

OPC on ohjausjärjestelmien ja kentällä sijaitsevien laitteiden väliseen kommunikointiin kehitetty käytännöllinen ja tehokas järjestelmä, joka on suunniteltu lähinnä teollisuuden tarpeisiin. Se tarjoaa mahdollisuuden Office-tuotteiden ja yritysten tietopal-

veluiden yhdistämiseen. Prosessidataa kentältä pystytään esittämään esim. Excel taulukko-ohjelmistossa. OPC:n avulla tilatietoja ja tuotantodataa voidaan arkistoida ilman minkäänlaisia ongelmia, ja tätä dataa voidaan myös jatkaen soveltaa tuotannon suunnittelujärjestelmissä (esim. tuotantoajan määrittämiseen). Tämän hetkinen OPC perustuu Microsoftin DCOM:iin. Tästä johtuen kaikki määritelmät mitkä löytyvät DCOM:istä (objektit, rajapinnat, menetelmät) pätevät myös OPC:hen. /4/

OPC:n käyttämisen etuna on se, että OPC servereiden ja clienttien valmistajat eivät ole riippuvaisia itse komponenttien omista monimutkaisuuksista, toiminnallisuuksista tai niiden toteutustavoista. Niin kauan kun valmistajat noudattavat oikeaoppisia määrittelyjä komponenttien rajapinnoista he voivat luottaa siihen, että heidän komponenttinsa ovat yhteensopivia muiden valmistajien järjestelmiin ja laitteisiin. Mitään lisäohjelmointia ei tarvita komponenttien toisiinsa liittämisessä. /4/

OPC määrittelyt ovat vapaasti käytettävissä olevia teknisiä määrittelyjä jotka määrittelevät yleiset rajapinnat eri alojen sovelluksien käytölle automaatioteknologiassa. Nämä rajapinnat tarjoavat mahdollisuuden tehokkaalle tiedonsiirrolle eri valmistajien ohjelmistojen välillä. OPC serverit ovat hyödynnettävissä monilla eri ohjelmointikielillä ohjelmoiduille clienteille (esim. C++, Visual Basic ja scripti-kielet). OPC erottaa kaksi eri rajapintaa: *OPC Custom Interface* ja *OPC Automation Interface*. Riippuen millä ohjelmointikielellä clientit on toteutettu, saattaa olla, että joudutaan käyttämään erillistä ohjelmaa jonka avulla clientit saadaan kommunikoimaan OPC serverin kanssa (OPC Automation Interface).



Kuva 3.5 OPC Serverin rajapinnat /4/

OPC voidaan toiminnaltaan kuvata seuraavanlaisesti. OPC client on käyttöliittymä, ohjelma tms. johon mittauskohteet, esim. anturit, ovat yhteydessä. OPC serveri on taas ohjelma, joka on yhteydessä OPC clienttiin (serverit voivat olla yhteydessä useampaan clienttiin, ja päinvastoin clientti useampiin servereihin). OPC serveri tuo clientilta esiin standardisoidun rajapinnan jota pystytään esim. logiikan kautta hyödyntämään (lämpötilan arvo tms.). /4/

OPC:ssa on kolme eri määrittelyä, jotka selvittävät miten rajapintamäärittely järjestelee dataa ja aikaansaa järjestelmien välisen kommunikoinnin. Nämä määrittelyt ovat jo sinällään todella laajoja ja vaikeaymmärteisiä perehtymättömälle lukijalle, joten niiden osuus opinnäytetyössä pidetään varsin yleisellä tasolla. /4/

Opinnäytetyössä käytettiin OPC Data Access-määrittelyä rajapintana kenttäväyläkortin ja PLC Addon:in välillä.

3.4.3 OPC Data Access määrittely

Data Access-määrittely on OPC-määrittelyistä vanhin. Se määrittelee clientin ja serverin välisen rajapinnan siitä, miten ohjelmat pääsevät käsiksi prosessidataan. Data Access serverit tuovat esiin yhden tai useampia Data Access clientteja joilla on transparentti pääsy eri tietolähteisiin (esim. lämpötila-anturit) ja tietokantoihin (esim. ohjaimet). Nämä tietolähteet – ja pankit voivat sijaita I/O-korteissa jotka on suoraan liitetty tietokoneisiin; ne voivat myös sijaita laitteissa, kuten ohjaimissa ja I/O-moduleissa, jotka ovat yhdistetty toisiinsa useiden linkkien tai väylien avulla. Data Access Clientit voivat myös olla yhteydessä useampiin Data Access Servereihin samanaikaisesti. /4/

Data Access Clientit voivat olla yksinkertaisia (esim. Excel taulukko) tai monimutkaisia (esim. Visual Basic malli). Ne voivat myös olla osana jotain isompaa järjestelmää (esim. HMI tai SCADA järjestelmät). /4/

Data Access Serverit voivat olla yksinkertaisia ohjelmia jotka esim. aikaansaavat yhteyden jonkin logiikan rekistereihin sarjaporttiyhteyden avulla. Ne voivat olla moni-

mutkaisempia ohjelmia jotka on yhdistetty suureen määrään muuttujia laajassa laitteistoympäristössä joka sijaitsee jossain suuressa kenttäväylässä. Data Access Serverit voivat myös olla osana jossain isoissa ohjelmistoissa (esim. tietokonepohjainen ohjausjärjestelmä), jotka tekevät ohjelmien tiedon muille saataviksi. Määrittelyissä ei ole mitään yleistä sääntöä tai rajoja näille toteutuksille. /4/

3.4.4 OPC Alarms and Events määrittely

Tämä määrittely käsittää client- ja serveriohjelmien välisen rajapinnan tapahtumien ja hälytysten valvomiseen ja tiedostamiseen. Alarms and Events Server voi tallentaa ja arvioida arvoja eri tietolähteistä ja laitteista joko jotain on tapahtunut tai ei. Nämä tietolähteet voivat olla suoraan tietokoneeseen liitettävässä I/O-kortissa. Nämä voivat kuitenkin olla myös laitteissa, kuten säätimissä ja I/O-moduleissa jotka on yhdistetty toisiinsa sarjalinkeillä tai kenttäväylillä. /4/

Alarm and Events servereillä ja Data Access servereillä voi olla yhteisiä tietolähteitä. Näiden eroina on se, että Data Access serverit käsittelevät suhteellisen jatkuvaa dataa. Tämän datan automaattinen tiedonsiirto tapahtuu yleensä kun jokin arvo muuttuu. Alarms and Events serveri ei lähetä varsinaisia arvoja clienteille, vaan lähettää tietoa siitä, että jotain on tapahtunut (esim. lämpötila on ylittänyt sallitun arvon). Alarms and Events serverillä on tälle arvolle ennalta asetettu raja-arvo. Tämä määrittely on tehtävä ennen kuin serveriä sovelletaan varsinaisessa sovelluksessa jossa muuttujia ja laitteita tarkastellaan. /4/

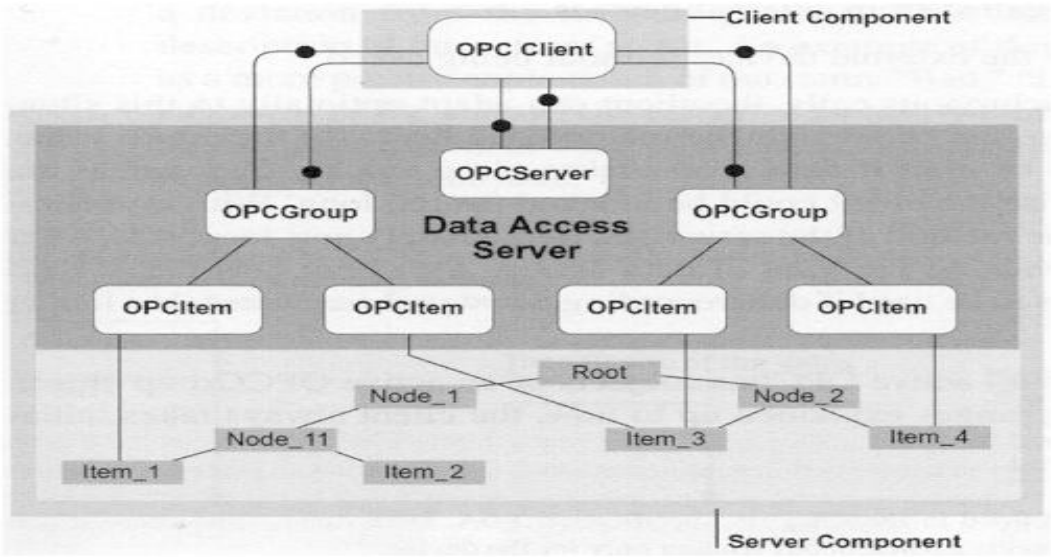
OPC Alarms and Events määrittelyyn ei kuulu se, miten edellä mainitut raja-arvot on määritetty, tällaiset arvot pystytään määrittämään mille tahansa muuttujalle (analoginen, binäärinen, yms.), sekä yhdistetyille muuttujille (esim. lämpötila ja aika). Alarms and Events serverit voidaan toteuttaa yksittäisinä komponentteina jotka hankkivat tietoa suoraan laitteilta ja sovelluksilta, tai Data Access servereiltä. Jälkimmäisessä niillä tulee olla osia Data Access clientin sisällä. Alarms and Events serveri voidaan myös toteuttaa komponenttina yhdessä Data Access serverin kanssa. /4/

3.4.5 OPC Historical Data Access määrittely

Historical Data Access serverit mahdollistavat clienteille pääsyn historiadataan. Tätä dataa on olemassa kahta eri tyyppiä, raw data, ja aggregated data (dataa jota on otettu raa'asta datasta). Näistä jälkimmäistä luodaan ainoastaan jos client sitä pyytää. Kirjoitus- tai muuttamisoikeuksilla client voi päällekirjoittaa olemassa olevaa historiadataa, kommentoida sitä, tai liittää uutta dataa. Tämä määrittely kattaa tallennettavan datan liikkumisen OPC-rajapinnassa (esim. trendi moottorin lämpötilasta, joka tallennetaan johonkin rekisteriin). /4/

3.4.6 OPC määrittelyjen rakenne

Data Access määrittely pitää sisällään kaksi eri käsitettä joita Data Access Server voi toteuttaa ja joita Data Access Client voi käyttää: Nimiavaruus ja OPC objekti-hierarkia. Nimiavaruus pitää sisällään kaikki tietolähteet ja datapankit jotka serveri tuo esiin. Nimiavaruus voidaan käsittää puurakenteiseksi (hierarkinen nimikeavaruus). Se voi myös olla tasainen, jossa kaikki lehdet (seuraavissa kohdissa puhutaan oksista ja lehdistä viitaten nimiavaruuden puunkaltaiseen rakenteeseen) ovat samanarvoisia ja samalla tasolla. Hierarkisessa nimiavaruudessa olevilla solmukohdilla voidaan jakaa esim. laitteet osiin, joiden avulla saadaan kartoitettua sen, tai sen osien tietolähteitä tai datapankkeja. /4/



Kuva 3.6 OPC nimiavaruus ja objektihierarkia /4/

Määrittelyissä on huomioitava OPCServerin ja OPC serverin ero niiden merkityksessä. OPCServer tarkoittaa esim. Data Access Serverin ylintä objektia. OPC serveri tarkoittaa taas kaikkia servereitä, joita voidaan luoda perustuen OPC määrittelyihin. /4/

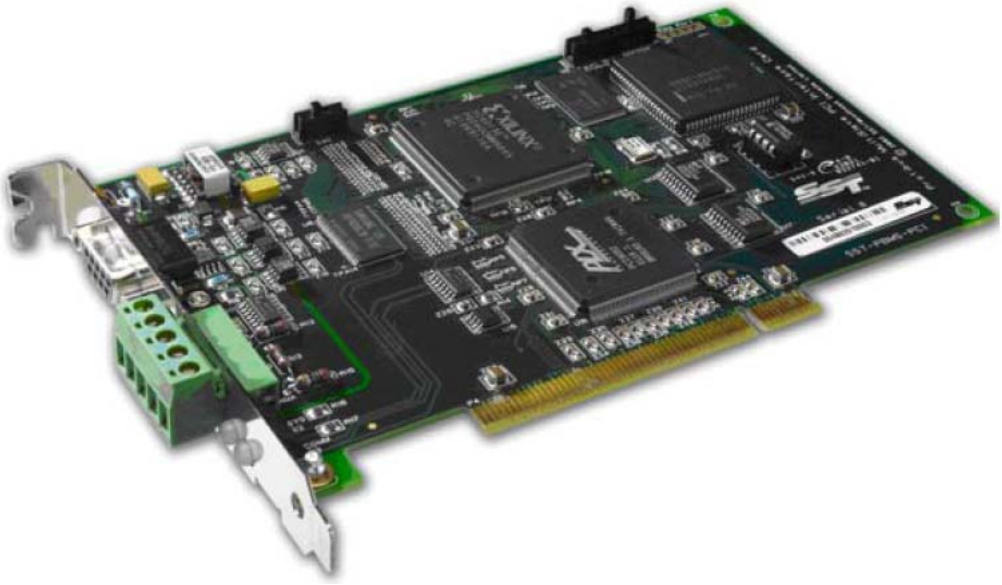
Data Access Client voi luoda useita OPC objekteja Data Access Serverissä määritelläkseen sen näkymän prosessista. OPCServer objekti sijaitsee puurakenteen yläpäässä. OPCServerin alapuolella on OPCGroup objektit. OPCGroup objekti voidaan käsittää lehtenä tai lehden ja solmukohdan ominaisuutena nimikeavaruudessa. OPCGroup objekteja käytetään järjestämään OPCItem objekteja tarpeiden mukaan (esim. järjestämään jonkin prosessin osan muuttujat samaan paikkaan, tai järjestämään samankaltaiset muuttujat samaan paikkaan). OPCItem:it voivat pitää sisällään esim. mittaustuloksia kenttäantureilta. OPCServerissä voi olla useita tämänkaltaisia rakenteita riippuen miten Data Access Clientit ovat yhteydessä siihen. Jokaisella clientilla on ominainen hierarkia-määrittely joka ilmentää sen omia vaatimuksia. /4/
Datapankit ja tietolähteet ovat sinällään sovellusten tärkeitä osia. Näiden sisältö on kuitenkin kenties vielä tärkeämpää. Jos tätä sisältöä kopioidaan sovelluksessa turhaan, se aiheuttaa tarpeetonta nimiavaruuden laajenemista. Siksi solmukohtiin (OPCGroup) ja lehtiin (OPCItem) sisällytetään niiden ominaisuudet (Properties). Näitä ominaisuuksia voi olla esim. laitteen valmistaja tai laitteen tyyppi. /4/

Ominaisuuksien lisäksi OPC Itemeille voidaan määritellä tiedonsiirtoreitit (Accesspaths), joilla määritellään mitä kautta Data Access Server kommunikoi laitteen kanssa (esim. COM1, 9600kBit/s). Accesspaths:in määritteleminen ei ole pakollista ja voidaan jättää pois haluttaessa. OPCItemeillä ei ole muokattavia rajapintoja. Tämä johtuu siitä, että todellisissa sovelluksissa dataa kirjoitetaan ja luetaan koko ajan. Tästä johtuen on paljon tehokkaampaa päästä useaan OPCItem objektiin käsiksi yhdellä kutsulla. Tämä toteutetaan OPCGroup objektien rajapintojen avulla, ja tämän ansiosta OPCItem objekteilla ei ole muokattavia rajapintoja ja ne voidaan pitää melko yksinkertaisina. OPCItemit voivat olla ohjearvoja tai mitattuja muuttujia laitteessa. /4/

Tässä kappaleessa annettiin esimerkkinä Data Access määrittelyn kuvaus nimiavaruudesta ja objektihierarkiasta. Periaatteeltaan muut määrittelyt rakentavat serverinsä samalla tapaa, mutta niissä voi olla hieman eri ominaisuuksia ja komponentteja riippuen datan valvomismuodosta (esim. Alarms and Events määrittely voi nähdä nimikeavaruuden luettelona, jossa eventit on kategorisoituina eri luokkiin).

3.4.7 Profibus-kortti

Prosessitietokoneeseen oli liitettynä Molex:in BradCommunications STT Profibus DP kenttäväyläkortti. Tämä kortti mahdollistaa PCI-väyläiseen tietokoneeseen liitettynä 1:stä 125:een Profibus DP slave laitteen emuloimista tai valvomista vain yhdellä fyysisellä yhteydellä. Tätä korttia voidaan käyttää tietokoneen liittämiseen HMI:hin tai rajapintaohjelmistoihin Profibus DP:ssä, I/O-emulaatioon, verkon monitorointiin, sekä testaamaan täysin kuormitettua Profibus DP Marter-verkkoa ja muita Profibus tuotteita.



Kuva 3.7 Molex BradCommunications STT Profibus DP kenttäväyläkortti /5/

Etuja kortin käytöstä on:

-Vähäinen tai ei ollenkaan hidastumista ohjelmankiertoaikaan: kortti kuuntelee passiivisesti verkkoa, ja tämän ansiosta HMI järjestelmä voi valvoa verkkoon kytkettyä I/O-maailmaa ohjelmankiertoaikaan vaikuttamatta.

-Vaihtoehto Ethernetille: HMI käyttäjät voivat siirtää suuria määriä dataa ilman vaikutusta verkkoon tai PLC:n ohjelmankiertoaikaan, tai muita haittoja Ethernet-yhteyden laatuun.
-Laitteet ja johdotus eivät ole välttämättömiä: Käyttäjä voi luoda laajan verkon Profibus slaveja yhdessä kortissa nopeassa ajassa. Simulaation avulla voidaan etsiä mahdollisia virheitä ja ongelmakohtia mallintamalla todellisen prosessin ja I/O:n käyttäytymistä ennen kuin todellisessa tehtaassa tapahtuu mahdollisia kustannuksia laitteiden rikkoutumisesta. Lisäksi on mahdollista tehdä nopeita vertailuja eri kaapelointien, laitteiden tai yhteysongelmien välillä mallintamalla todellista tutkimusdataa emulaation avulla. /5/

3.5 Valvomotietokoneen ohjelmistot

Valvomotietokoneesssa käytettiin Siemens Automation:in valmistamia WinCC ja Simatic Manager ohjelmistoja, joiden avulla pystyttiin luomaan prosessin ohjausta varten tarvittava ohjausjärjestelmä. WinCC:llä toteutettiin varsinainen visuaalinen käyttöliittymä. Simatic Managerilla toteutettiin käyttöliittymän ”alle” logiikkaohjelma, jossa määrättiin prosessin toiminnallisuus ja käsiteltiin prosessista tulevaa dataa. Nämä kaksi ohjelmistoa toimivat perustana opinnäytetyössä toteutetun prosessin ohjaukselle.

3.5.1 WinCC

WinCC (Windows Control Centre) on tietokoneeseen pohjautuva prosessien visualisointiohjelmisto. Ohjelmisto koostuu WinCC:stä, WinCC:n asetuksista, sekä WinCC:n lisäosista. Yhdessä nämä muodostavat laajasti yksilöitävän kokonaisuuden käyttäjien sovelluksiin. Kokonaisuutena WinCC on tehokas ja laajalti sovellettava ohjelmisto, joka tarjoaa kaikki nykyaikaisen HMI ohjelmiston vaatimukset. WinCC CS komponentti sisältää editorit, joiden avulla suunnittelijat voivat helposti tehdä tehokkaita ja helppokäyttöisiä käyttöpaneeleja. Kirjastot ja wizardit tekevät projektien luomisesta nopeaa ja helppoa, sekä vähentävät tekijän mahdollisten virheiden syntymistä. Ohjelmana WinCC suoriutuu erinomaisesti monimutkaisista HMI tehtävistä, koska se pystyy käsittelemään kattavia projekteja ja suuria määriä dataa. /6/

Perusjärjestelmä on kaikella tapaa skaalattavissa käyttämällä WinCC asetuksia. Perusohjelmistoa voidaan itsessään soveltaa laajeneviin rakenteisiin (esim. koko ajan kasvava prosessi), ja tämänkaltaisia rakenteita voidaan helposti päivittää prosessitagien kasvavan määrän suhteen. Käyttämällä WinCC/Server:iä WinCC:tä voidaan käyttää joko client-, tai server-sovelluksena. WinCC/Redundanry:ä käytetään rakentamaan helposti saatavilla olevia SCADA järjestelmiä. WinCC/CAS antaa mahdollisuuden pohjustaa skaalattavaa keskustyyppistä tiedon arkistointia. WinCC/Web Navigator antaa käyttäjille mahdollisuuden visualisoida ja käyttää tehtaita ja prosesseja internetin välityksellä lähes samoilla toiminnallisuuksilla kuin WinCC client:kin. /6/

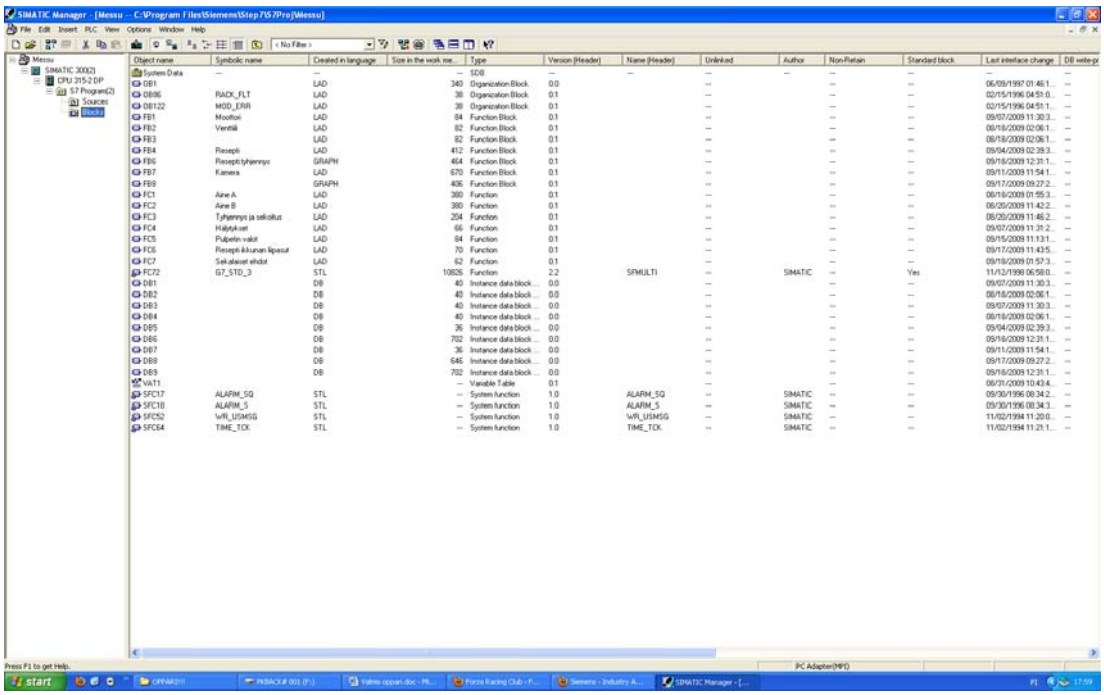
SIMATIC WinCC:llä tavoitellaan yhteneviä ja mahdollisimman avoimia konstruktioita, sekä mahdollisuutta yhdistyä jo olemassa oleviin sovelluksiin käyttäen standardoituja menetelmiä ja ohjelmistotyökaluja. Ohjelma pyrkii yhdistämään perustekniikat, käyttöjärjestelmät, kommunikointimenetelmät tai kyvyn yhdistää scriptejä tavalla joka ei ole riippuvainen laitteiden erilaisuudesta tai ominaisuuksista. /6/

Perusjärjestelmä on suunniteltu teknologia- ja osa-alueriippumattomaksi. Siitä huolimatta se täyttää monien alojen asettamat erikoisvaatimukset. SCADA (Supervisory Control and Data Aquisition) toimii älykkäiden tehdasratkaisujen perustana. WinCC pystyy hankkimaan ja arkistoimaan dataa, jota pystytään keräämään, analysoimaan ja lähettämään MES-järjestelmille (Manifacturing Execution Systems) jatkokäsittelyä varten käyttämällä WinCC Plant Intelligence asetuksia. /6/

Opinnäytetyöhön liittyneessä projektissa WinCC:llä ohjattiin Step 7 Simatic Managerissa ohjelmoituun logiikkaohjelmaan luotuja muuttujia.

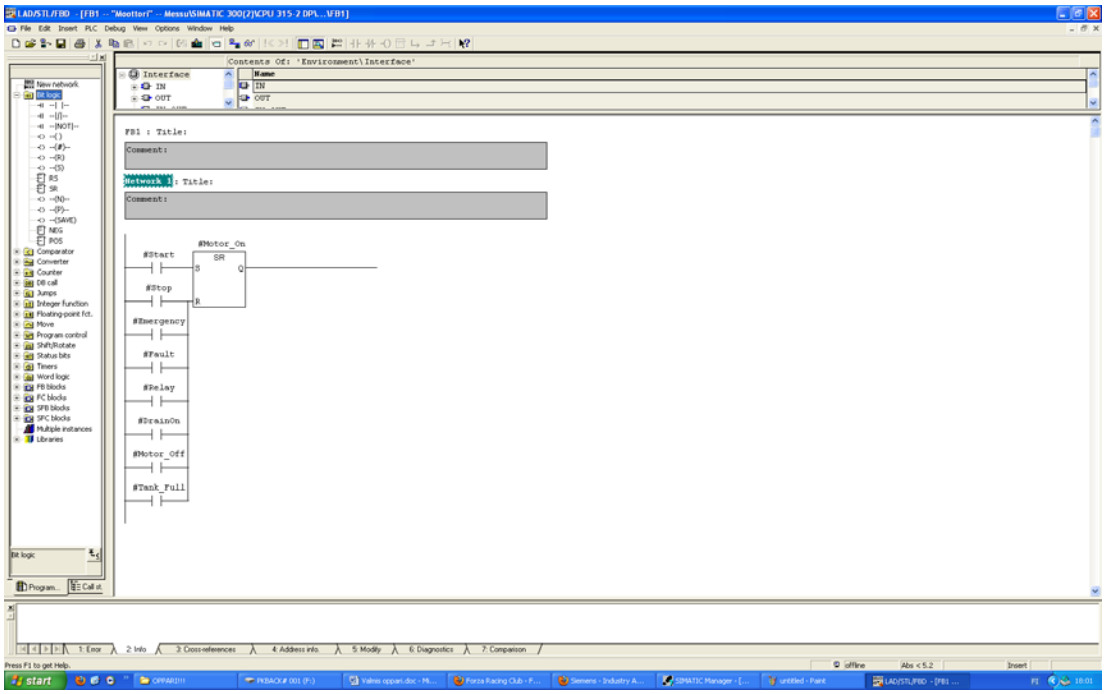
3.5.2 Step 7 Simatic Manager

Siemensin Step 7Simatic Manager on kaikille SIMATIC-järjestelmille tarkoitettu ohjelmointiympäristö. Sen avulla käyttäjät pystyvät ohjelmoimaan hankkimiaan komponentteja toimimaan haluamallaan tavalla valmiiden ohjelmointikomponenttien ja logiikkakaavioiden mukaisesti.



Kuva 3.8 Simatic Manager perusnäkymä

Kuvassa 3.8 näkyy Simatic Managerissa luotu projekti, johon oikealla on listattuna erilaisia ohjelmalohkoja, joiden sisään on ohjelmoitu järjestelmän erilaisia toiminnallisuuksia. Tästä näkymästä käyttäjä voi luoda aliohjelmia, ohjelmakutsuja, muuttujataulukoita jne. järjestelmän tarpeiden mukaan. Vasemmalla näkyvästä pienemmästä kentästä ohjelmoijat pääsevät käsiksi laitteiston konfiguroimiseen sekä projektien hallintaan.



Kuva 3.9

Kuvassa 3.9 näkyy Simatic Managerin yleinen ohjelmointiympäristö. Vasemmalla olevasta työkalupalkista käyttäjä voi lisätä erilaisia ohjelmointipalikoita oikealla olevaan ohjelmakenttään (esim. AND-piiri, jossa määrätään lähdön Q1.0 menevän päälle kun tulot I1.0 ja I1.1 ovat päällä). Tässä ympäristössä toteutettiin simulaatiodemon logiikkaohjelma.

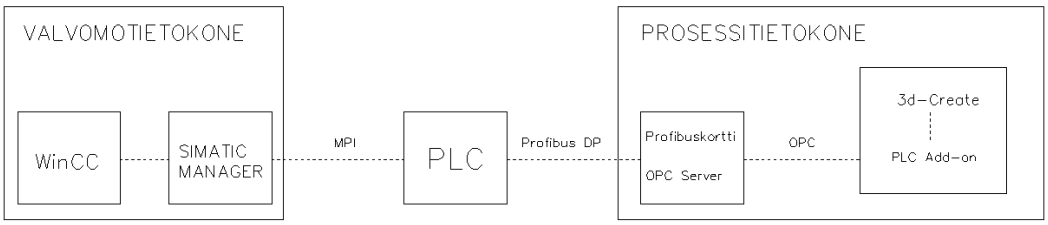
Yleisesti ohjelmaa käytetään Siemensin ohjelmoitavien logiikoiden ohjelmoimiseen ja se toimii peruspilarina monissa Siemensin automaatiojärjestelmissä.

4 SIMULOIDUN SEKOITUSPROSESSIN TOTEUTUS

Tässä osiossa käsitellään opinnäytetyönä tehdyn prosessin rakentamista ja toteuttamista keskittyen enemmin WinCC:llä työskentelyyn ja tietokoneiden välisiin yhteyksiin. Yhteydet-osiossa kerrotaan tietokoneiden välisistä ja sisäisistä kommunikointiväylistä ja rajapinnoista, Prosessi-osiossa kuvataan prosessin käyttämiseen liittyviä asioita ja asetuksia, sekä Valvomo-osiossa opastetaan WinCC:n käyttämistä sekä graafisten käyttöliittymien tekoa.

4.1 Yhteydet

Järjestelmän kommunikointi tapahtui usean eri raja-pinnan kautta. Valvomotietokoneen sisäinen kommunikointi tapahtui WinCC:n ja Simatic Managerin välisellä suorakommunikoinnilla. Logiikasta lähdettiin Profibus DP-väylän avulla prosessitietokoneen Profibus-kortille. Kyseiselle kortille ohjelmoitiin laitteisto mitä se mallinsi. Tämän määrittelyn jälkeen profibus-kortti välitti tietoa PLC-addonin kautta 3dCreate:en mallinnetusta prosessista.



Kuva 4.1 Järjestelmän kokoonpano

4.1.1 WinCC ja Simatic Manager

WinCC:n ja Simatic Managerin välinen kommunikointi tapahtui tietokoneen sarjaportin kautta. WinCC:ssä tageja luotaessa valitaan yhteyskanava, jota kautta ohjelma hakee I/O:ta ja ohjasi luotuja tageja. Tässä tapauksessa kommunikointi tapahtui MPI-väylän kautta (tietokoneen sarjaportti) johon S7-300 logiikan CPU oli kytkettynä. WinCC siis haki luodut tagit (esim. I0.0) S7-300 logiikalta.

4.1.2 Profibus-väylä

Simuloidun sekoitusprosessin tietokoneiden välinen kommunikointi tapahtui Profibus DP-väylän kautta. Siemensin S7-300-logiikka yhdistettiin prosessitietokoneen Profibus-korttiin logiikassa olevan DP-portin kautta Profibus DP-kaapelilla. Tätä väylää pitkin kuljetettiin kaikki I/O-tieto prosessilta, sekä ohjauskäskyt prosessille.

4.1.3 Profibus-kortti ja OPC-rajapinta

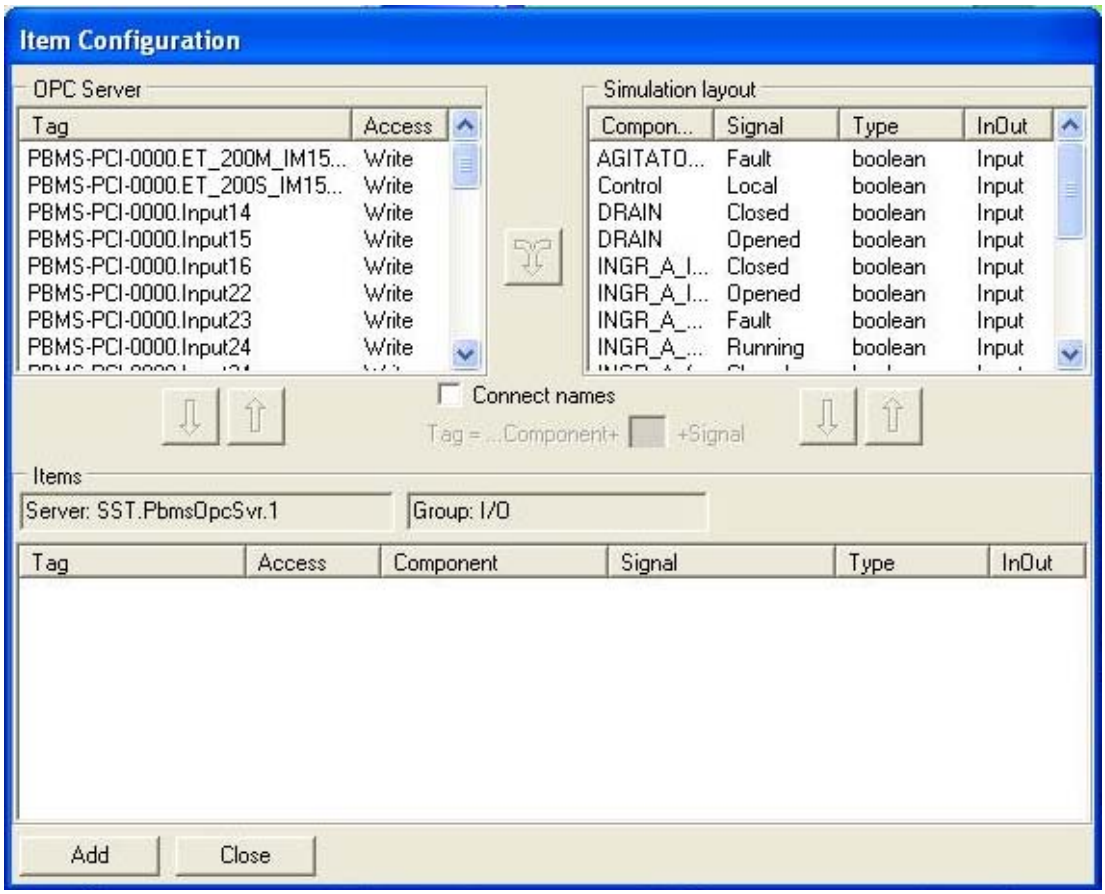
Profibus-kortin asentamisen ohessa prosessitietokoneeseen asennettiin kortin valmistajan tekemä konfigurointiohjelma. Kortti konfiguroitiin ohjelmiston ”Direct-Link Configuration”-ohjelmalla. Tässä ohjelmassa määritettiin OPC-serveri, johon konfiguroitiin profibus-kortin simuloimat kenttäasemat. OPC-serveri kommunikoi PLC Add-on:in kanssa, joka toimi OPC Clienttina.

Profibus-kortille määritettiin Siemensin ET200M ja ET200S kenttäasemat. ET200M:n alla oli 16-tuloinen DI-kortti (digital input), 16-lähtöinen DO-kortti (digital output), 8-tuloinen AI-kortti (analogue input) sekä 4-lähtöinen AO-kortti (analogue output). ET200S-asemassa oli 2-tuloinen AI-kortti.

Kun tämä konfigurointi oli tehty, pystyttiin simuloidun prosessin I/O-maailmaa ja profibus-korttiin simuloidun kenttäaseman tuloja ja lähtöjä linkittää toisiinsa PLC Add-on:issa.

4.1.4 PLC Add-on ja OPC-rajapinta

Kun edellä mainittu kortin konfigurointi oli tehty, päästiin prosessin I/O:ta liittämään kenttäasemien tulokortteihin. Tämä toimi periaatteeltaan siten, että kenttäaseman tulokortissa oli esim. tulo I0.0, tämä kyseinen tulo voitiin PLC add-on:issa linkittää prosessissa oleviin komponentteihin, esim. hätäseispainike ”Emergency”. Ohjelma loi kaksi listaa, joista toisessa oli profibus-kortille ohjelmoidut logiikkakortit ja niiden tulot ja lähdöt, ja toisessa listassa oli taas prosessin laitteet ja niiden sisältämät signaalit.

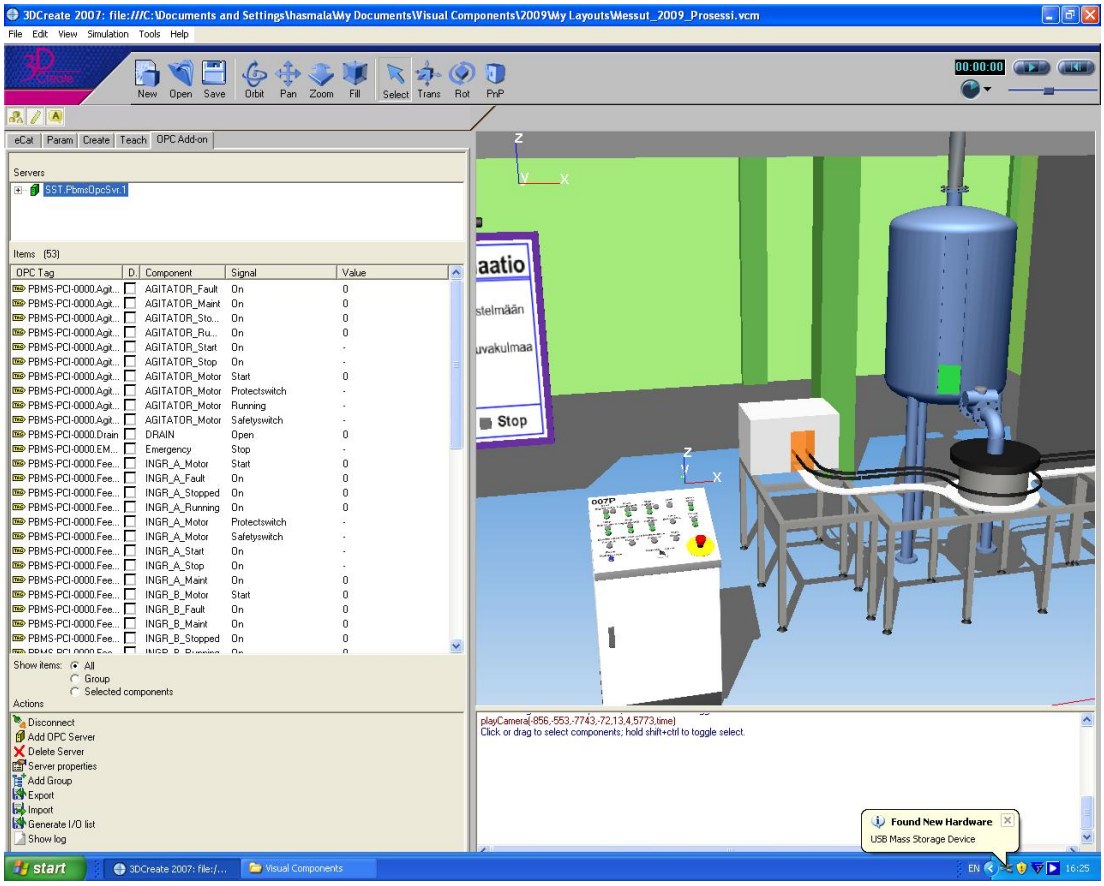


Kuva 4.2 Tulojen ja lähtöjen linkittäminen PLC Add-on:issa

Linkittäminen toimi siten, että kummastakin listasta valittiin yksi komponentti ja tulo, jonka jälkeen painettiin ”Add”, ja ohjelma lisäsi ne PLC Add-on:in OPC Tag-listaan (Kuva 4.2).

Tämän opinnäytetyön tapauksessa komponenttien ja tulojen liittäminen oli melko hankalaa, koska tulot eivät näkyneet kovinkaan yksinkertaisessa muodossa, sillä projektiin valittu OPC Serveri ilmoitti siihen ohjelmoidut tulot ja lähdöt muodossa ”Input 1, 2 jne.”, ja täten oli todella vaikeaa saada selvyyttä minkä niminen tulo se todellisuudessa oli.

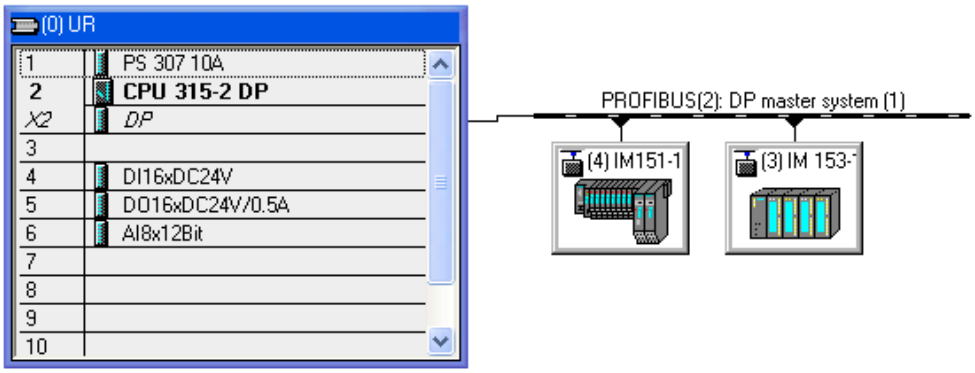
Tulojen ja komponenttien toisiinsa linkittäminen toteutettiin hieman helpommin Excel-ohjelman avulla. Excel nimesi OPC Server:in tagit ohjelmoitavan logiikan symbolilistan mukaisiksi, joka nimesi OPC Serverin tulo- ja lähtölistan järkevän muotoiseksi ja täten helpotti niiden linkittämistä. Kun linkitys oli tehty, PLC Add-on:in valikosta valittii ”Connect”, jonka jälkeen simuloidulla kentällä oleva laitteisto oli toiminnassa ja valmiina käsiteltäväksi valvomotietokoneen puolella.



Kuva 4.3 PLC Add-on:in käynnistäminen

4.1.5 Laitteiston määrittely Simatic Managerissa

Kun kaikki edellä mainitut toimenpiteet oli tehty, voitiin laitteiden järjestelmän toimivuutta testata Simatic Managerin HW-config:issa. Laitteiston määrittelyssä luotiin S7-300 logiikan CPU:n DP portista Profibus DP-väylä, ja tähän väylään lisättiin profibuskortille määritellyt kenttäasemat ja niiden I/O-kortit. Tämän jälkeen Simatic Manager näki simuloidut kenttäasemat samalla tapaa kuin todellisetkin asemat näkyisivät.

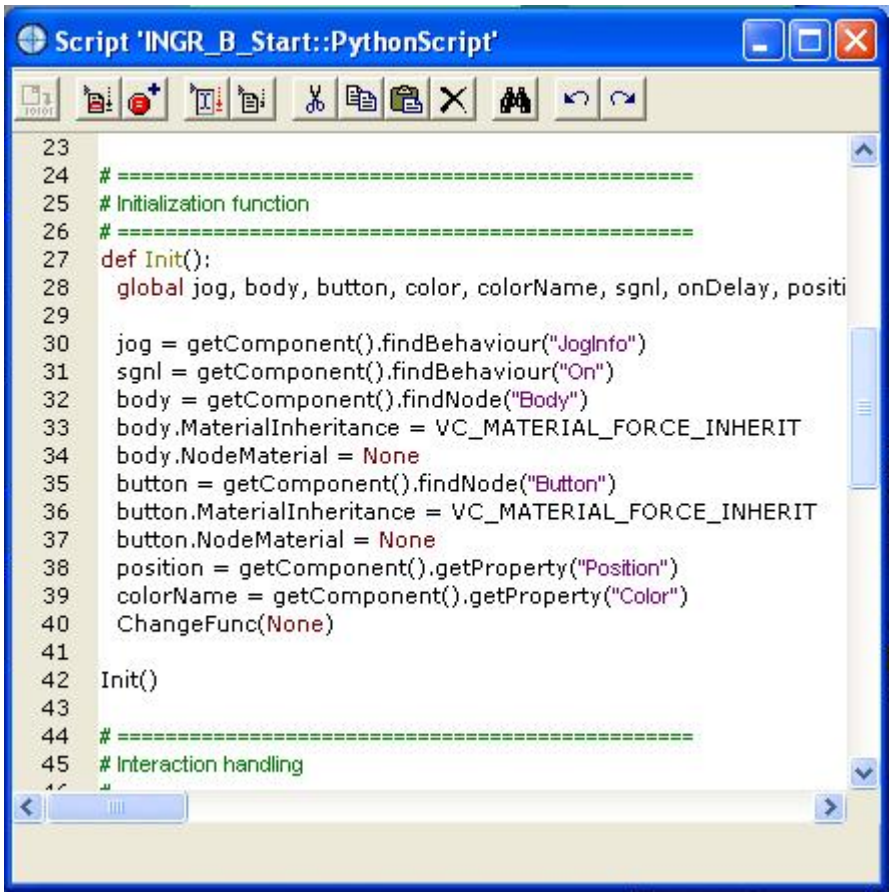


Kuva 4.4 Laitteiston määrittely

Kun laitteet oli määritetty HW-configissa oikein, ja samoin prosessitietokoneen puolella, määrittely ajettiin logiikkaan ilman virheitä ja logiikassa paloi vain vihreät valot. SF-valo (serial fault) sekä BF-valo (bus failure) paloivat vain jos määrittelyssä oli tehty jokin virhe, tai väylä oli mennyt poikki (esim. tilanteessa jossa DP-kaapeli on huonosti kiinni tai poikki). Väylän katketessa normaalisti logiikka täytyi asettaa Stop-tilan kautta takaisin Run-moodiin käsin. Opinnäytetyössä logiikkaohjelmaan lisättiin OB122- ja OB86-blockit joidenka ansiosta edellä mainittua toimenpidettä ei tarvinnut suorittaa, vaan logiikka palautui normaaliin tilaansa heti väylän palattua toimintaan.

4.2 3d-Create:lla mallinnettu prosessi

Prosessi koostui SolidWorks:sillä ja 3d-Create:n omilla työkaluilla tehdyistä 3d-kappaleista. Kappaleiden toiminnallisuus toteutettiin Python-kielellä tehdyllä koodilla, ja tämä kappalekohtainen koodi määräsi komponentin fyysisen toiminnan. Komponenteille luotiin 3d-Create:ssä signaaleja, jotka toimivat simuloituina tuloina ja lähtöinä joita OPC Add-on välitti edelleen OPC Serverille. Se miten signaali vaihtoi tilaansa, esim. painike antoi tiedon että sitä painetaan, määritettiin kappaleen Python-koodissa.



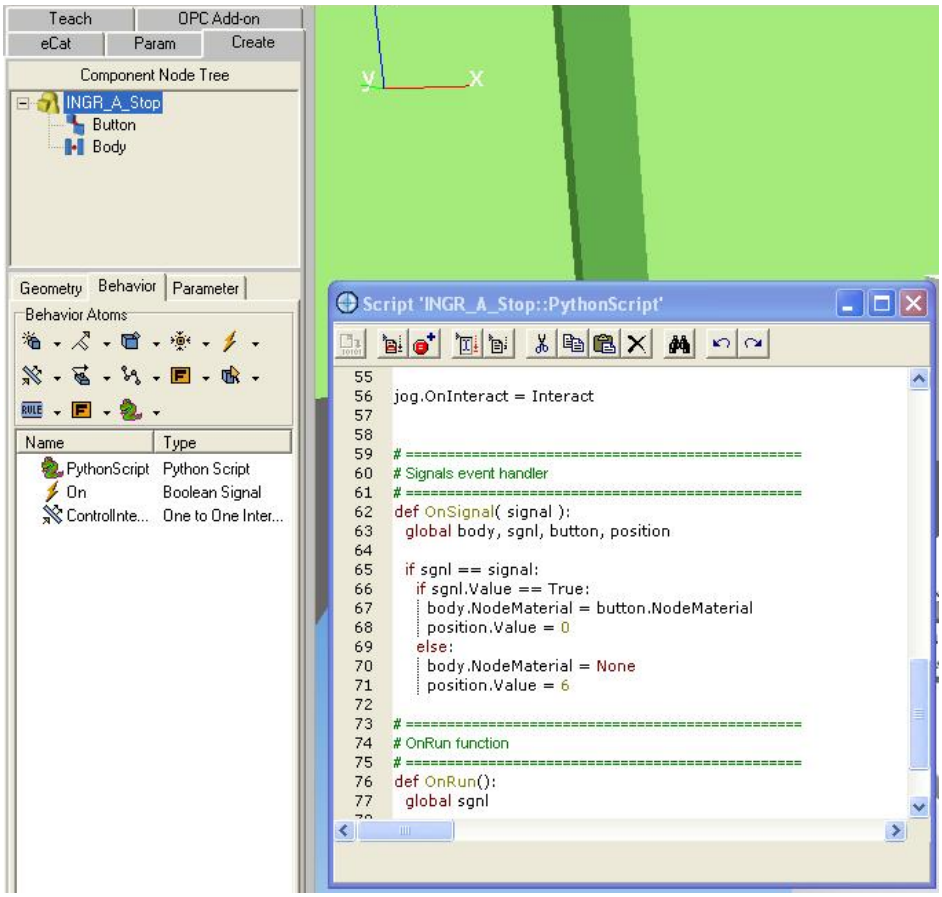
Kuva 4.5 Esimerkki painonapin PythonScript:istä

Painikkeen toiminta ja geometria rakennettiin siten, että vihreän sylinterin ympärille rakennettiin kehys, jotta se näyttäisi normaalilta nappulalta. Tämä kaksiosaisuus oli kuitenkin tärkeää sen realistisen toimivuuden kannalta. Python-koodissa määritettiin, että kun painikkeen vihreän osan kohdalla klikattiin hiiren vasenta painiketta, painikkeen vihreä sylinteriosa liikkui kehyksen sisään vaihtamalla koordinaattiaan. Täten hiirellä painaminen sai aikaa realistisen näköisen painikkeen painamisen.

Samalla tapaa toteutettiin esim. sekoitusprosessin tankkien pinnan nousu ja lasku. Tankkien sisälle oli tehty vihreä sylinteri. Python-koodissa tämä sylinteri oli liitettynä tankin pinnankorkeuteen siten, että kun tankin pinnankorkeuden arvo nousi, niin tankin sisällä olleen vihreä sylinterin korkeus kasvoi samassa suhteessa. Samalla tapaa toteutettiin myös muita mallintamisia prosessissa. Kappaleiden koodeissa määritettiin kappaleisiin liitettyjen signaalien tilan vaihtumisista johtuvia värinvaihdoksia, esim. pumppu päällä → pumpun väri vaihtuu vihreäksi.

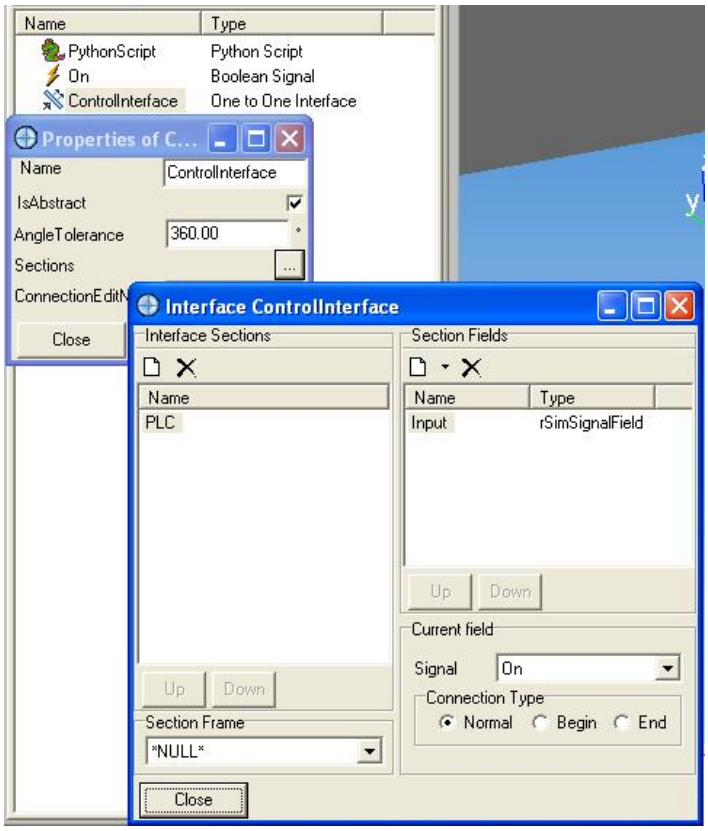
4.2.1 Prosessin I/O-maailma

Jokaiselle kappaleelle määritettiin 3DCreatessa ominainen käyttäytyminen. Tässä osiossa voitiin kappaleille määritellä signaaleja ja toimintoja. Kuvassa 4.6 vasemmassa reunassa näkyy 3DCreate:n ”Create”-valikko, jossa voitiin kappaleille määritellä sen geometria, käyttäytyminen ja parametrit.



Kuva 4.6 Kappaleiden ominaisuudet

Create-valikon alla olevassa listassa näkyy ”PythonScript”, ”On” sekä ”ControlInterface”. PythonScript:iin kirjoitettiin Python-koodilla kappaleiden toimivuus esim. geometrian muuttuminen. Samassa kyseinen geometrian muuttuminen voitiin liittää muihin toimintoihin, esim. ”On”-signaalin tilan vaihtoon. Behavior-valikossa voitiin luoda kappaleille signaaleja. Signaaleja oli monia eri tyyppejä, painikkeen tapauksessa valittiin signaalityypiksi ”Boolean” eli 1 tai 0. Signaali liitettiin PythonScript:iin, jossa signaalin muuttumista edellyttävät tapahtumat määritettiin, esim. painikkeen painaminen hiirellä. ControlInterface:ssä määritettiin kappaleen I/O-rajapinta, josta sen signaalit saatiin vietyä eteenpäin logiikalle.



Kuva 4.7 ControlInterface rajapinnan määrittely

Kuvassa 4.7 näkyy ControlInterface:sta aukeavat valikot. Interface Sections:iin oli erittäin tärkeää määrittää rajapinnan nimeksi PLC, sillä 3DCreate:n OPC Add-On haki juurikin tällä nimellä signaaleja prosessista. Kuvan oikeassa reunassa näkyvässä Section Fields:iin määritettiin minkä muotoinen signaali se on, eli tulo vai lähtö. Täten jatkossa signaaleja eteenpäin vietäessä signaaleja ei tarvinnut erikseen jaotella, vaan OPC Add-On oli tämän tehtävän hoitanut jo signaaleja prosessista hakiessaan.

4.2.2 Scriptit

3DCreate:ssä pystyi ohjelmoimaan ohjelman sisäisiä scriptejä, joiden avulla pystyttiin ohjaamaan prosessissa tapahtuvia asioita (esim. kameran liikkuminen tai napin painallus). Nämä scriptit olivat linkitettynä profibus-korttiin konfiguroitudun kenttäaseman kortin analogiseen lähtösanaan. Scripteihin pääsi käsiksi valitsemalla prosessissa olevan taulun, jonka sisään scripti-editori oli rakennettu. Tämän jälkeen valittiin vasemmalta ”Tabbed panel:ista” ”Param”, jonka alta ”UserScripts” ja oike-alta ”…”. Näyttöön avautui teksti-ikkuna, johon scriptejä pystyi kirjoittamaan.

Esimerkki scriptistä jossa siirretään prosessin kameraa:

```
global_2
def_2():
playCamera(2352,-3987,-7934,-62,20,10,5422,1)
```

Yllä olevassa esimerkissä ”global” ja ”def” perässä oleva luku viittaa scripteihin linkitetyn analogisen lähtösanan arvoa. Kun kyseisen sanan arvo on 2, 3DCreate siirtää kameran kyseiseen koordinaattiin. ”PlayCamera”-kohdan perässä oleva numerosarja tarkoittaa kameran koordinaattia. Näitä koordinaatteja saatiin esiin kun valittiin scriptien muokkaamiseen rakennettu taulu, ja sen ”Param” alavalikosta ”General”, josta painettiin ”Print Camera Coord”. Tämän jälkeen kameran sen hetkiset koordinaatit ilmestyivät 3DCreaten alareunassa olevaan ”Message Panel”:iin. Kamerakoordinaattien viimeinen luku 1 tarkoittaa aikaa, jonka ohjelma käyttää kameran siirtämiseen haluttuihin koordinaatteihin.

Opinnäytetyössä käytettiin scriptejä myös prosessissa olevien painikkeiden painamiseen. Prosessin ohjauksessa oli vaihtoehtona käyttää prosessia sen omasta käyttöpulpetista painamalla sen painikkeita. Kyseisen pulpetin käyttäminen messutilanteessa olisi ollut vaikeaa, joten tämä ohjaus toteutettiin valvomosta käsin ”Pulpettiohjaus”-moodissa.

Pulpetin painikkeiden painaminen valvomosta käsin toteutettiin scriptien avulla siten, että kun valvomoon rakennetusta pulpetista painettiin nappia, prosessi ohjasi kameran pulpettiin, painoi nappia, ja kamera palasi perusasemaan. Ohjaus tapahtui muistipaikkojen avulla logiikkaohjelmassa. Kun valvomosta painettiin painiketta, se laittoi muistipaikan päälle. Tämä muistipaikka antoi analogiselle lähtösanalle tietyn arvon, jonka arvon mukaista scriptiä 3DCreate lähti toteuttamaan.

Esimerkki scriptistä, jossa käydään painamassa painiketta:

```
global_10
def_10():
playCamera(518,870,-4052,-57,0,871,2)
setValue(’INGR_B_Start’,On’,1)
delay(1)
setValue(’ ’INGR_B_Start’,On’,0)
```

Yllä olevassa scriptissä kamera ajetaan pulpetille, painetaan painiketta ”INGR_B_Start”, pidetään painiketta sekunti pohjassa, ja sen jälkeen vapautetaan painike. Edellä mainitusta tapahtumasta johtuen logiikkaohjelma reagoi painikkeeseen linkitetyn tulon aktivoitumiseen, ja käynnisti prosessissa halutun toiminnon.
Scriptejä hyödynnettiin opinnäytetyössä edellä kuvattujen tapahtumien toteuttamiseen eri tilanteissa, esim. Käsiohjauksessa kamerakuvakulmien muuttamiseen, Reseptiohjauksessa kameran automaattiseen liikkumiseen, sekä Pulpettiohjauksessa painikkeiden painamiseen ja kameran liikuttamiseen. Opinnäytetyössä oli kaiken kaikkiaan 33 erilaista scriptiä jotka aikaansaivat erilaisia tapahtumia, esim. napin painallus tai kameran liikkuminen. Scriptejä ei suoraan ohjelmasta löydy, vaan tämä ominaisuus oli SAMK:issa toteutettu lisäominaisuus ohjelmaan.

4.3 Valvomo

Prosessien ja laitosten ohjausjärjestelmien tekemisessä on tärkeää suunnitella valvomon eri osia jo ennen niiden tekemistä. Opinnäytetyön projektin aloittamisvaiheessa oli jo selvä kuva prosessista ja sen toiminnasta. Tämän ansiosta pystyin tekemään valmiin listan prosessin toiminnallisuuksista ja sen jälkeen aloittamaan valvomon suunnittelun. Valvomoa tehtiin logiikkaohjelman kanssa samanaikaisesti, koska tavallaan kummatkin olivat osiltaan riippuvaisia toisistaan (esim. valvomosta ei voinut ohjata juuri mitään ilman logiikkaohjelmaa, eikä taas logiikkaohjelmaa voinut ohjelmoida tietyn pisteen ohi ennen kuin valvomoa oli rakennettu eteenpäin). Työskentely oli varsin jouheaa kun logiikkaohjelma oli saatu toiminnaltaan kuntoon, eikä sii-

hen tarvinnut tehdä muuta kuin lisäillä ohjattavia muistipaikkoja. Tässä osiossa kuvataan projektiin liittynyttä logiikkaohjelmointia, sekä kuvataan miten WinCC:llä sai tehtyä valvomon eri osia ja toimintoja.

4.3.1 Logiikkaohjelma

Logiikkaohjelma toteutettiin Siemens:in Simatic Managerilla. Siinä hyödynnettiin tavallista LADDER-ohjelmointia, sekä Siemens:in GRAPH-ohjelmointiympäristöä. Ohjelma koostui seitsemästä aliohjelmasta (FB) ja kuudesta ohjelmakutsusta (FC). Ohjelma rakennettiin hierarkisesti siten, että OB1:n alla oli ohjelmakutsuja joiden sisään oli rakennettu aliohjelmia ja muita ohjelmakutsuja riippuen mitä prosessin laitetta haluttiin ohjata. Hierarkinen rakenne tehtiin logiikkaohjelman selventämiseksi. Lisäksi logiikkaohjelmaan lisättiin operation block:it (OB) OB86 ja OB122. OB86 liitettiin siksi, että jos yhteys logiikkaan katkeaa, se ei aiheuttanut vikatilaa ohjausjärjestelmässä. OB122 lisättiin siksi, ettei logiikkaa tarvinnut resetoida siinä tapauksessa, että väylä sattui katkeamaan esim. DP-kaapeli irroitetaan.

Logiikkaohjelman toiminnot jaettiin osiin eri FB:eihin ja FC:eihin. Etuina aliohjelmien käytöstä oli se, että kun prosessissa oli identtisiä osia kuten pumput, niille ei tarvinnut erikseen tehdä omia ohjelmia vaan niille tehdyt aliohjelmat voitiin liittää omiin toimintalohkoihinsa joihin liitettiin pumppujen omat tulot ja lähdöt. Valmis logiikkaohjelma on opinnäytetyön LIITTEET-osiossa.

4.3.2 WinCC:llä työskentely

WinCC:n käynnistämisen jälkeen näytölle ilmestyy ohjelman perusikkuna. Tästä ikkunasta käyttäjä pääsee rakentamaan valvomoa, määrittämään tageja, hälytyksiä, datan arkistointia yms. Käyttäjä voi luoda uuden projektin, tai ladata koneelle aiemmin tehtyjä projekteja. On tärkeää huomioida, ettei WinCC:ssä tehdyt projektit toimi suoraan muilla tietokoneilla kuin sillä jossa projekti on luotu. Tämän voi toki jälkeenpäin muuttaa oikea-klikkaamalla ”Computer”-ikonia ja valitsemalla Properties.

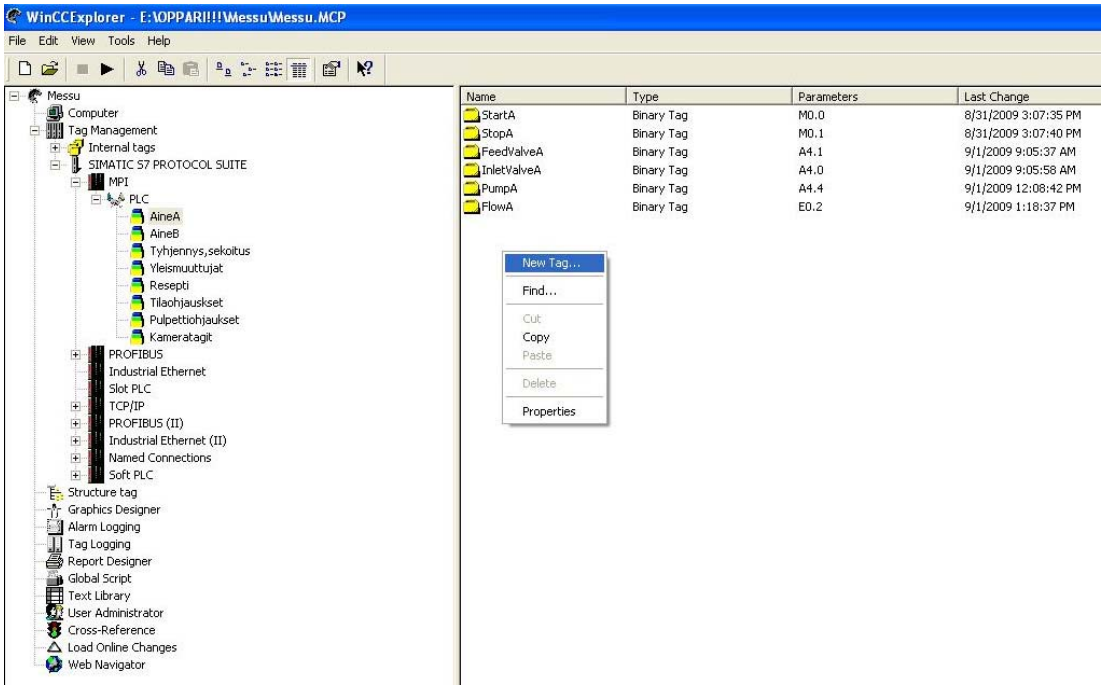
Computer:in properties-valikosta käyttäjälle esitetään lista projektiin liitetyistä tietokoneista. Valitsemalla haluttu kone ja alhaalta ”Properties”, käyttäjä pääsee muokkaamaan projektin käynnistämiseen liittyviä asetuksia (esim. minkä ikkuna tulee valvomoa käynnistettäessä ensimmäisenä esiin), ja yleisiä valvomon käyttämiseen liittyviä asetuksia (esim. miten valvomon saa sammutettua).

WinCC:n perusvalikon yläreunassa olevista ”Play”- ja ”Stop”-painikkeista käyttäjä pääsee käynnistämään luodun valvomon. Lisäksi tupla-klikkaamalla vasemmassa valikossa olevia ikoneja, WinCC käynnistää projektin muokkausohjelmia kuten Graphics Designer (prosessikuvien muokkausohjelma) tai Alarm Logging (prosessin hälytysten muokkausohjelma). Opinnäytetyössä käytettiin kahden edellä mainitun ohjelman lisäksi Tag Logging-ohjelmaa, jolla voitiin rekisteröidä prosessidataa arkistoon.

4.3.3 Tagien luominen

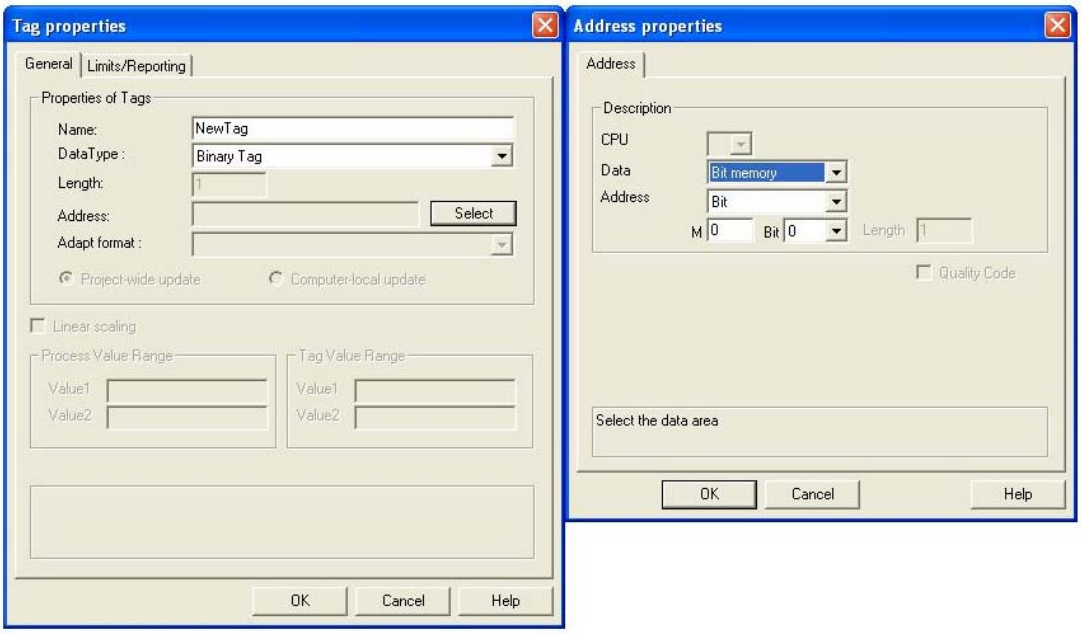
WinCC:llä voidaan ohjata prosessin tapahtumia tagien avulla. Tämä ominaisuus on WinCC:ssä järjestelmien ohjattavuuden kannalta kenties tärkein ominaisuus. WinCC:hen listataan tagit, jotka ovat prosessissa olevia muuttujia tai logiikkaohjelman muistipaikkoja. Näitä voidaan tagin tyypin mukaan monitoroida tai ohjata. Tässä projektissa Simatic Managerissa ohjelmoituihin muuttujiin liitettyjä tageja pystyttiin ohjaamaan suoraan logiikan MPI-väylän kautta. Ohjelma pääsi käsiksi kyseisen väylän kautta logiikan I/O-kortteihin ja niiden tuloihin ja lähtöihin, ja samaa kautta pystyttiin ohjaamaan Simatic Managerissa luotuja muistipaikkoja (HUOM! WinCC:llä ei voi ohjata olemassa olevaa fyysistä I/O:ta esim. lähtöä Q4.0, vaan ohjaukset toteutetaan aina muistipaikkojen kautta. Fyysistä I/O:ta voidaan vain monitoroida).

Esimerkki tagien luomisesta:



Kuva 4.8 Tag management

WinCC:n alkunäkymästä valitaan ”Tag Management”-valikko. Tämän alle ilmestyy ”Internal tags” ja ”SIMATIC S7 PROTOCOL SUITE”. ”Internal tags:issa” voi luoda WinCC:n sisäisiä tageja ja ”SIMATIC S7 PROTOCOL SUITE:ssa tageja, joita voi ohjata/monitoroida ohjelman ulkopuolelta. Näistä valitaan jälkimmäinen jonka alle avautuvasta valikosta valitaan haluttu väylä jonka kautta tageja käsitellään, tässä tapauksessa ”MPI”. Sen alle luodaan ”New Driver Connection”, joka on tässä tapauksessa nimettynä ”PLC”. ”PLC”:n sisälle voidaan varsinaisia tageja luoda. Aluksi tageille täytyy määritellä tagiryhmä (ryhmittely helpottaa niiden käyttöä jatkossa). Ryhmän sisään muodostetaan tageja painamalla hiiren oikeaa näppäintä WinCC:n oikeanpuoleisessa ikkunassa. Valitaan ”New Tag”, jonka jälkeen aukeaa valikko, josta määritellään tagin ominaisuudet.



Kuva 4.9 Tag Properties

Ilmestyneeseen valikkoon voidaan nimetä ja valita tagin tyyppi. Address kohdasta valitsemalla voidaan määritellä tagin osoite:

Name: Tagin nimi
Datatype: Tagin tyyppi (esim. Binary Tag)
Address: Tagin osoite (esim. Bit memory M0.0)

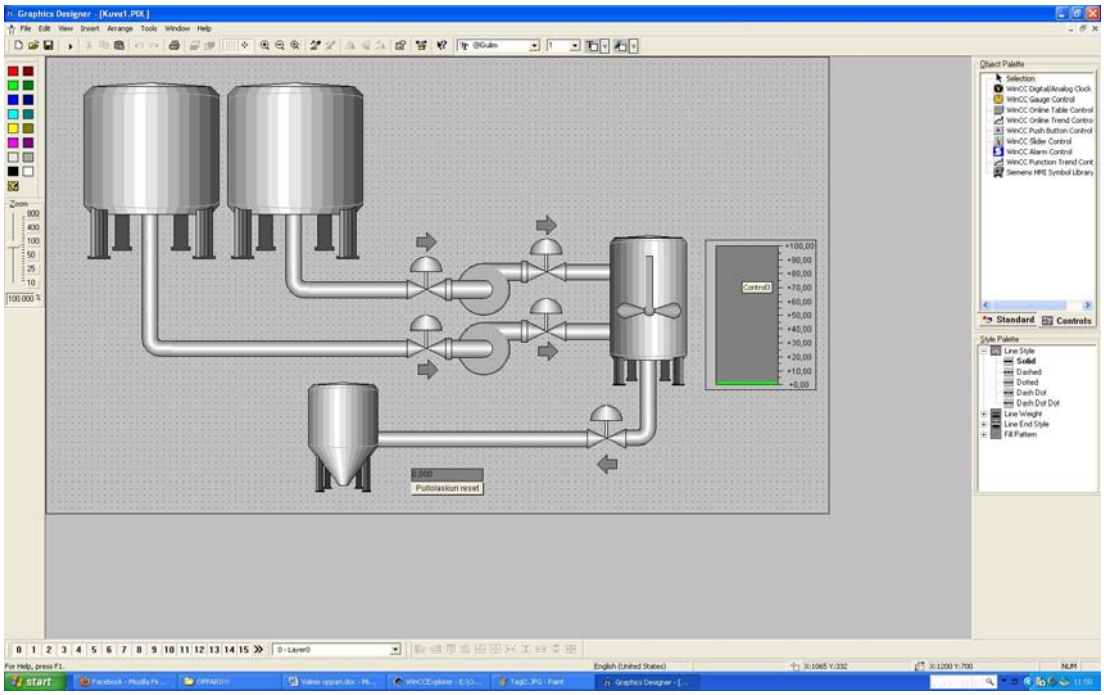
Luotuja tageja voidaan jatkossa käyttää prosessi-ikkunoissa valvonta- tai ohjaustarkoituksiin. Kyseinen tagi ”M0.0” voidaan esim. liittää valvomon painikkeeseen siten, että kun painiketta painetaan, muuttuja M0.0 saa arvon ”1”.

4.3.4 Prosessi-ikkunoiden rakentaminen

Tässä kappaleessa opastetaan prosessi-ikkunoiden ja valvomon rakentamista WinCC:llä. Prosessi-ikkunoiden luominen ja niiden todenmukaistaminen on suunnittelijalle haastavaa työtä. Ikkunoiden tulisi aina antaa käyttäjälle selvä käsitys käytettävän prosessin rakenteesta, sekä antaa tietoa siitä, mitä prosessissa tapahtuu. Valvomoista käsin ohjataan usein monimutkaisia prosesseja, joissa tapahtuu samanaikaisesti useita eri asioita ja tästä johtuen myös valvomot ovat usein monimutkaisia.

Kuitenkin valvomoiden tulisi olla helposti ohjattavissa, jotta käyttäjät pystyvät nopeasti reagoimaan mahdollisiin prosessissa tapahtuviin muutoksiin tai vikatilanteisiin.

Hyvän käyttöliittymän suunnittelussa kannattaa lähteä liikkeelle prosessin mallintamisesta kuvaksi näytölle. WinCC:ssä tämä tapahtuu ”Graphics Designer:in” kautta. Tietokoneen ruudulle muodostuva näkymä valvomosta on yleensä täysin luotu Graphics Designerilla. Tämä WinCC:n osa muistuttaa monella tapaa tavallista kuvankäsittelyohjelmaa. Suunnittelija voi piirtää itse kuvia tai kappaleita, tai valita valmiiksi piirrettyjä kappaleita objektikirjastosta.



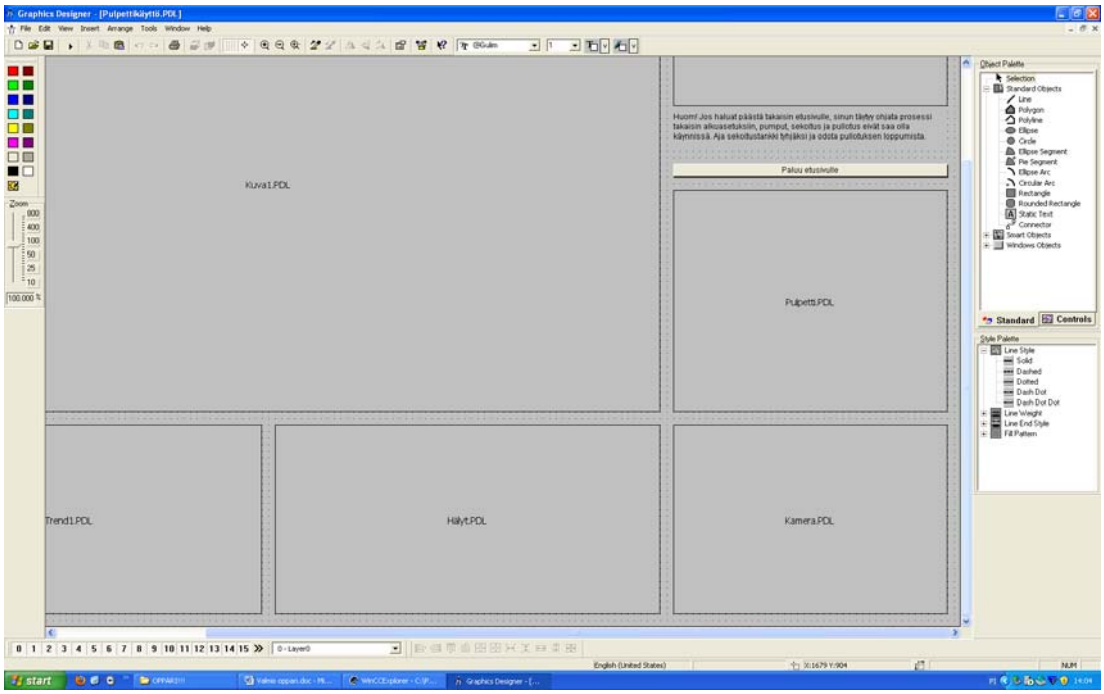
Kuva 4.10 Graphic Designer:illa mallinnettu prosessi

Yllä olevassa kuvassa näkyy Graphics Designer:in perusnäkymä. Kyseistä prosessikuvaa käytettiin opinnäytetyössä valvomon näkymänä prosessista. Kuva kasattiin valmiiksi tehdyistä osista, jotka haettiin objektikirjastosta ja liitettiin kuvaan. Jokaista kappaletta voitiin vapaasti liikutella ja asettaa ne haluttuihin paikkoihin. Objektien ”Layereitä” voitiin vaihdella, jos haluttiin jonkin kappaleen oleva toisen päällä esim. kuvassa 4.10 oleva potkuri oikealla olevan tankin päällä.

Valvomoa rakennettaessa Graphics Designer:illa tehtiin useita erilaisia kuvia joista edelle mainittu prosessi-ikkuna oli vain yksi esimerkki. Valvomon näkymä toteutettiin ”picture window”:ien avulla.

Picture Window:t olivat prosessikuviin liitettäviä kenttiä, joiden sisään sai liitettyjä muita Graphics Designer:illa tehtyjä kuvia. Tehdyille kuville määritettiin niiden koko, esim. 1000*800, ja ohjausmoodien pääkuvissa kuviin liitettiin samankokoinen picture window, johon haluttu kuva liitettiin. Esim. Käsiohjaus-moodin ohjauspainikkeet olivat erillinen kuva, joka liitettiin kyseisen moodin valvomokuvan picture window:iin. Käsiohjaus-moodin valvomonäkymä siis koostui kuvasta, jossa oli useita picture window:eja, joihin oli liitettynä halutut kuvat. Tällä tyylillä tehdyt valvomonäkymät helpottivat ohjausvaihtoehtojen liittämistä toisiinsa, esim. kameraohjaus, joka oli sekä käsiohjaus- että pulpettiohjausmoodissa. Täten kameraohjauksen painikkeita ei tarvinnut erikseen tehdä kahteen kuvaan.

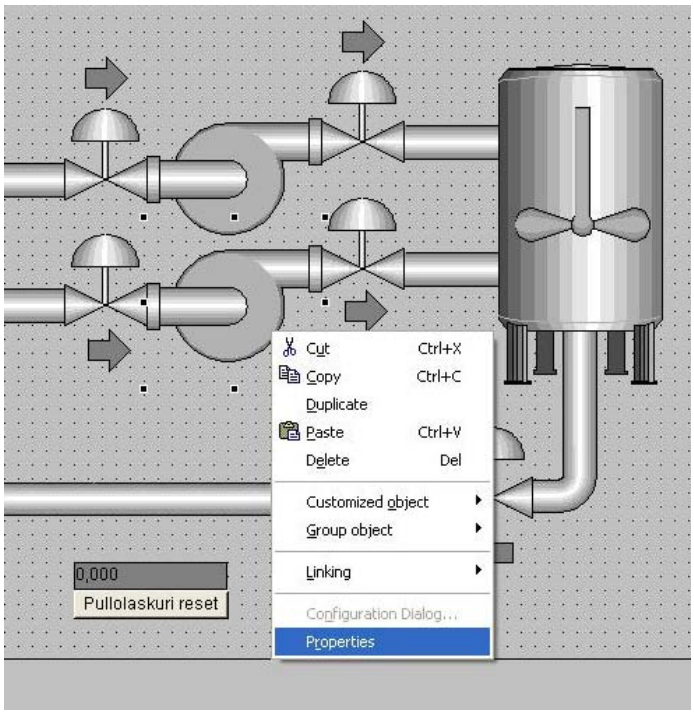
Valvomossa oli edellä mainittuja ns. pääkuvia neljä. Jokaiselle käyttömoodille oli rakennettu omat kuvat sekä valvomon etusivu. Kun valvomon käynnisti, käyttöliittymä alkoi etusivulta (kuva 5.1). Tässä kuvassa olevilla painikkeilla pääsi navigoimaan moodien välillä. WinCC:llä pystyi liittämään painikkeisiin ominaisuuden, jolla se vaihtoi ohjelmaan tehtyjä kuvia. Kuvien välillä navigointi toimi siten, että etusivulta valittiin haluttu ohjausmoodi, jonka painiketta painamalla valvomossa siirryttiin moodin kuvaan. Jokaisessa moodissa oli painike, jonka avulla päästiin takaisin etusivulle. Tämän ominaisuuden sai liitettyä painikkeeseen sen properties:in kautta.



Kuva 4.11 Pulpettiohjaus-moodin näkymä Graphics Designerissa.

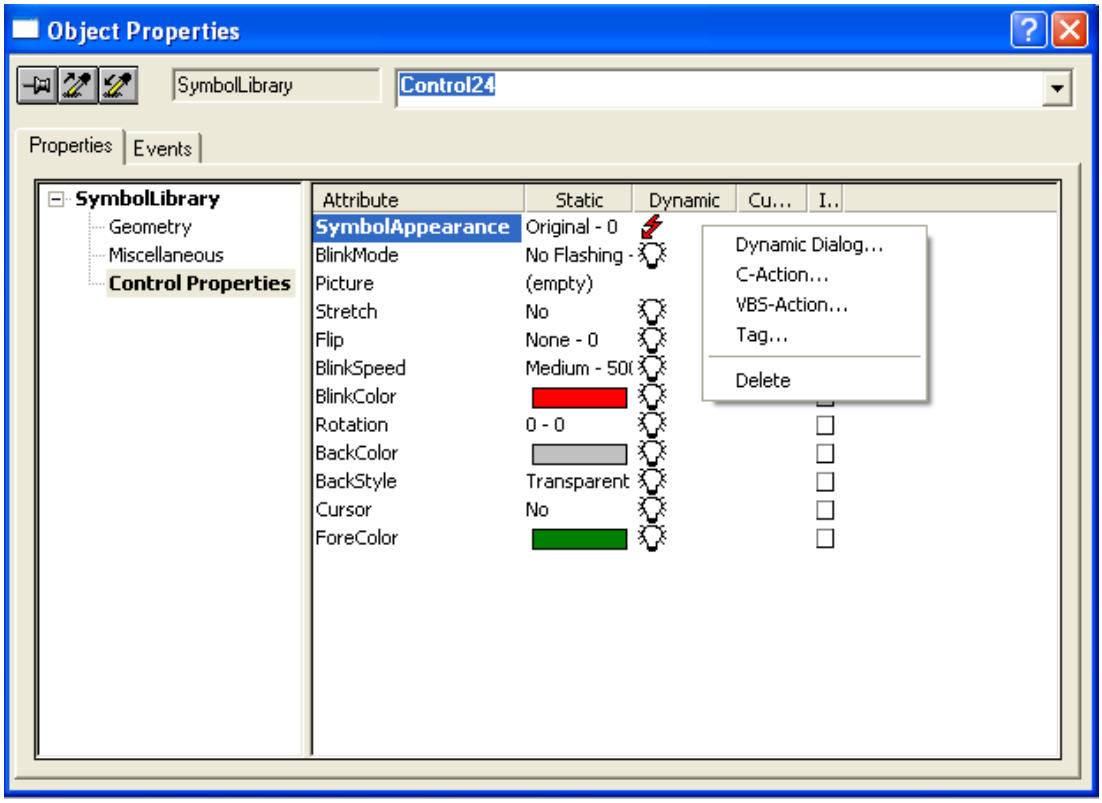
Yllä olevasta kuvasta näkyy miltä valvomossa näkyvä ikkuna näytti Graphics Designerissa (vertaa kuva 5.4). Jokainen käyttömoodin ”pääkuva” oli rakennettu tällä tavoin. Lisänä picture window:eihin pystyi liittämään myös tietokoneella olevia valokuvia, esim. valvomon etusivulla olevat kuvat prosessista (Screenshotteja prosessista) ja SAMK-logo (netistä ladattu jpg-kuva).

Prosessinäkymän todenmukaisuus visuaalisesti on sinällään tärkeä ominaisuus hyvässä käyttöliittymässä, mutta tämä ei aina riitä. Käyttäjän tulee jollakin tapaa saada tietoa prosessin tapahtumista. Tämä toteutetaan usein värien tai ilmoitustekstien muodossa. Kun valmis kuva prosessista on tehty, voidaan alkaa miettiä miten käyttäjille saadaan ilmoitettua prosessissa tapahtuvat asiat. Opinnäytetyössä prosessitapahtumien indikoiminen toteutettiin muuttamalla kuvaan liitettyjen kappaleiden värejä tilanteen mukaan. Otetaan esimerkkinä aineen A pumppu. Kun ainetta A pumpataan sekoitustankkiin, kyseisen pumpun väri näytöllä muuttuu vihreäksi. Tämä toteutettiin seuraavanlaisesti:



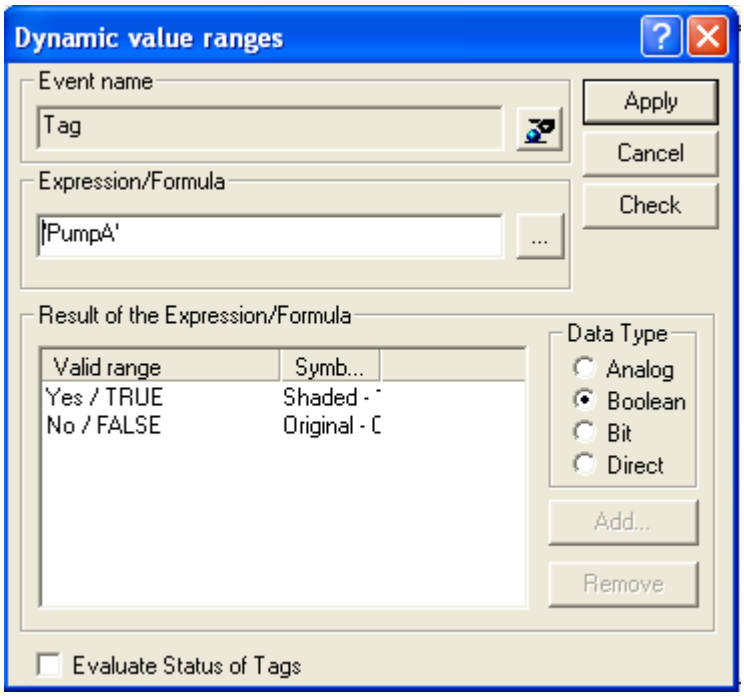
Kuva 4.12 Pumppu A

Aluksi valitaan pumppu A, ja painamalla hiiren oikeaa painiketta valitaan ”properties”. Tämän jälkeen aukeaa ”object properties”-valikko, josta kyseisen kappaleen asetuksia voidaan muuttaa.



Kuva 4.13 Object Properties

Valitaan Properties ja sen alta Control Properties. Valikosta oikealla oikea-klikkaamalla Dynamic-kohtaa ilmestyvästä valikosta valitaan Dynamic Dialog. Tästä aukeaa valikko josta voidaan kyseiseen ominaisuuteen, eli SymbolAppearanceen, liittää tagi. Tageja voi myös liittää useampiin ominaisuuksiin ikonin properties:issa. Ikonin voi halutessaan esim. laittaa vilkkumaan BlinkMode-attribuutista liittämällä sen Dynamic Dialog:iin tagin.



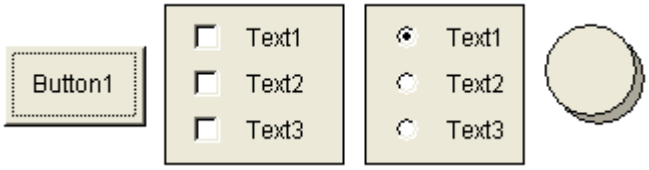
Kuva 4.14 Dynamic value ranges

Yllä olevassa kuvassa olevasta valikosta painetaan ”…”, ja valitaan Tag. Seuraavaan valikkoon ilmestyy lista ”Tag Management”-valikosta, josta haluttu tagi noudetaan. Kun tagi, tässä tapauksessa PumpA: Q4.4 on valittu, valitaan oikealta Boolean, sen jälkeen asetetaan Yes/TRUE ja No/FALSE arvojen vierestä oikea-klikkaamalla kappaleen ulkonäkö halutuksi. Kun nämä toimenpiteet on tehty, pumpun A kuvake palaa vihreänä kun lähtö Q4.4 (pumpun lähtö) on päällä, ja kun se ei ole päällä se näkyy valvomossa harmaana. Samalla tapaa voidaan kustomoida muita valvomon objekteja ja niiden näkyvyyttä eri tilanteissa, esim. pumppu A vilkkuu punaisena kun lämpörele on lauennut. Object Properties valikossa on useita eri vaihtoehtoja joiden avulla kappaleiden toimintaa voidaan muokata monella eri tapaa.

Värinvaihdon olisi voinut myös tehdä esim. siten, että pumpun ikonin kohdalle olisi laitettu useampi kuva päällekkäin, ja niiden properties:eista määritetty tagi pumpun näkyvyydelle. Jokainen pumpun kuva olisi väritetty tilanteen mukaan eri väreillä esim. pumppu päälle → vihreä, pumppu ei päällä → harmaa jne. Sen jälkeen olisi määritetty että kun pumppu A on päällä, niin vihreä pumppu on näkyvillä, ja taas kun pumppu A ei ole päällä, niin vain harmaa ikoni näkyisi. Visualisointia voi tehdä WinCC:llä monella eri tapaa, se riippuu täysin suunnittelijasta ja siitä miten hän haluaa projektia toteuttaa.

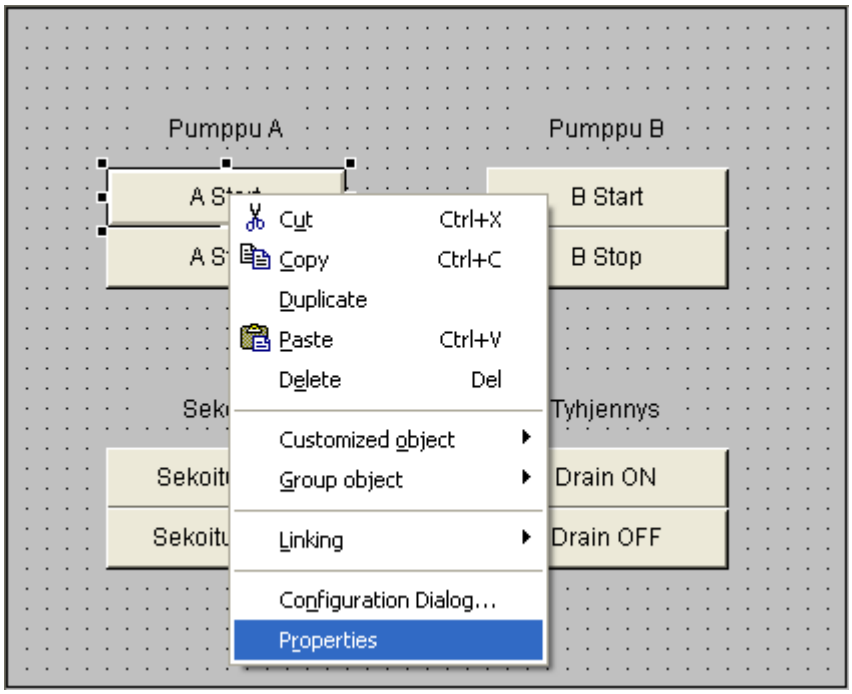
4.3.5 Painikkeet

WinCC:n Graphics Designerin avulla valvomoihin voi tehdä ohjauspaneeleja, joista painikkeiden avulla prosessin toimintoja voidaan ohjata. Graphics Designer tarjoaa useita valmiiksi tehtyjä painikkeita moniin eri sovelluksiin ja käyttötarkoituksiin esim. ON/OFF-kytkin, peruspainike, pyöreä painike, Check-painike sekä muita Windows-ympäristöstä tuttuja painikkeita.

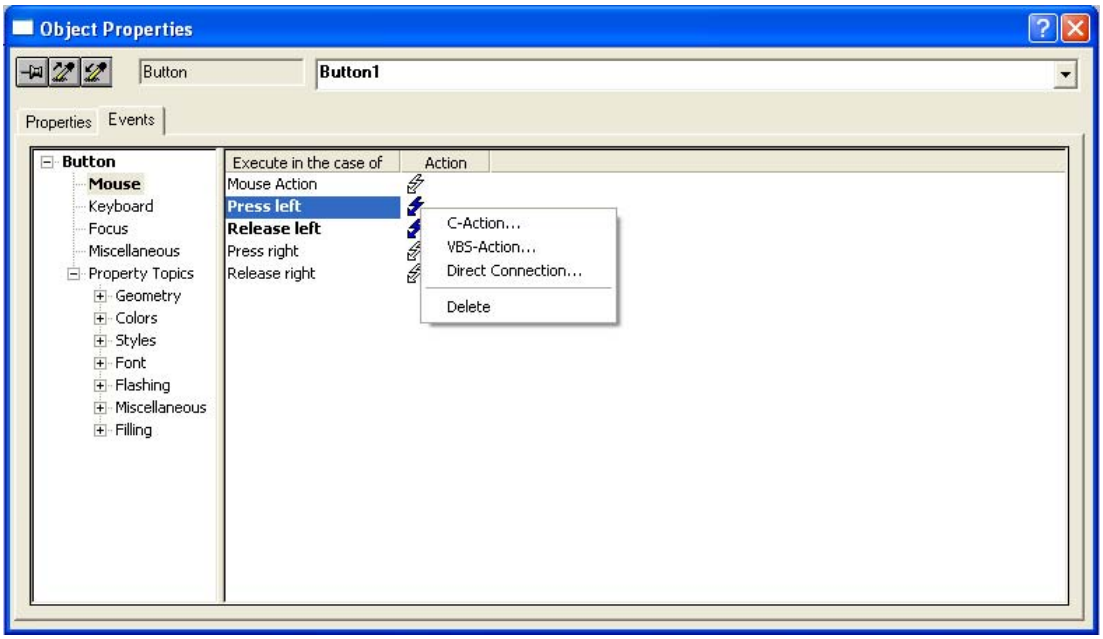


Kuva 4.15 Esimerkkejä WinCC:n painikevaihtoehdoista

Painikkeita voi lisätä prosessikuviin WinCC:n Graphics Designerissa oikealla olevasta valikosta avaamalla ”Windows Objects”-valikon. Valittu painike siirretään kuvaan, jonka jälkeen sen kokoa, muotoa ja värejä voidaan muokata. Painikkeiden toimintaa voidaan muuttaa oikea-klikkaamalla painiketta ja valitsemalla ”object properties”.

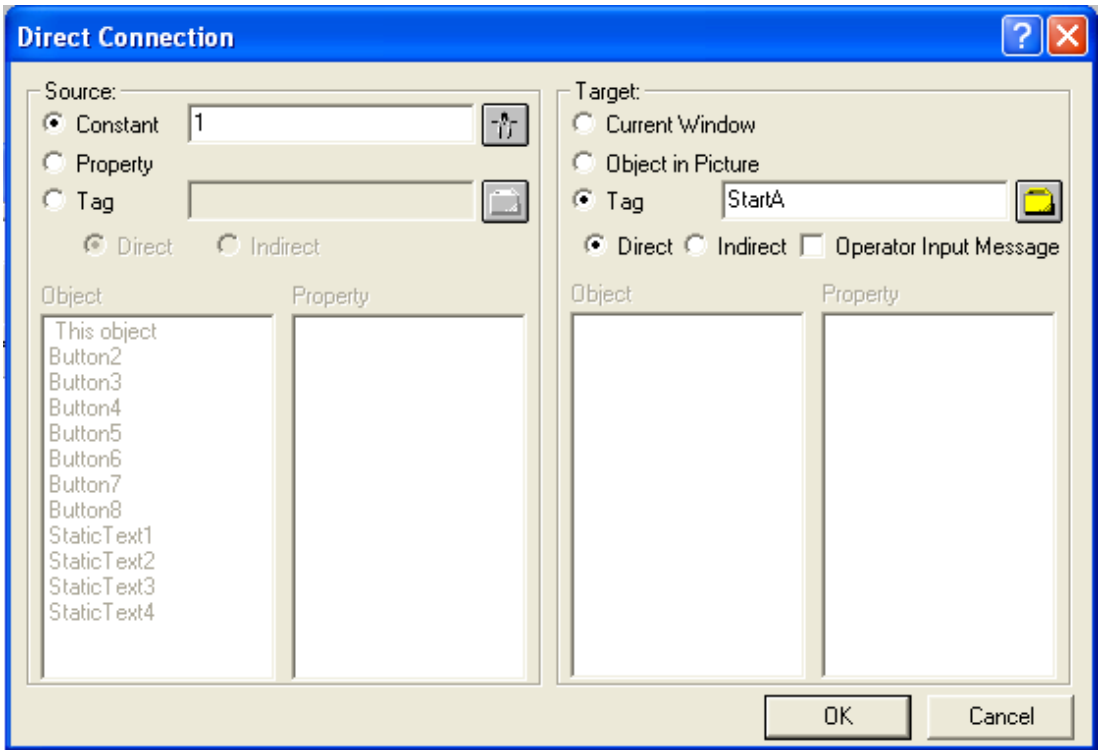


Kuva 4.16 Tagien liittäminen painikkeisiin 1



Kuva 4.17 Tagien liittäminen painikkeisiin 2

Avautuneesta valikosta valitaan Events ja sen jälkeen valitaan ”Mouse”. Oikealle ilmestyvät tietokonehiiren toiminnot esim. hiiren vasemman painikkeen painaminen (Press left) tai hiiren oikean painikkeen irrottaminen (Release right). Näihin toimintoihin voidaan liittää tageja, joita kyseisen painikkeen painalluksella ohjataan. Eri toimintoihin voi liittä vain yhden tagin, eli samaan hiiren toimintaan, esim. hiiren painamiseen ei voida liittää kuin yksi muuttuja. Oikea-klikataan halutun toiminnon vierestä ”Action”-valikon alla olevaa salamaa ja valitaan ”Direct Connection”. Tästä valikosta voidaan määritellä painikkeelle tagi.



Kuva 4.18 Direct Connection valikko

Vasemmasta yläreunasta valitaan ”Constant” ja kirjoitetaan tähän kohtaa ”1”. Oikealta puolelta valitaan ”Tag”-kohta, ja sen oikealta puolelta haetaan hakemistosta haluttu tagi. Tehty painike toimii nyt siten, että kun sitä painetaan hiiren vasemmalla painikkeella, se asettaa tagin ”StartA” ykköseksi. Mikäli ”Constant”-kohdan muuttaa arvoksi ”0”, tagi saa painettaessa arvon nolla.

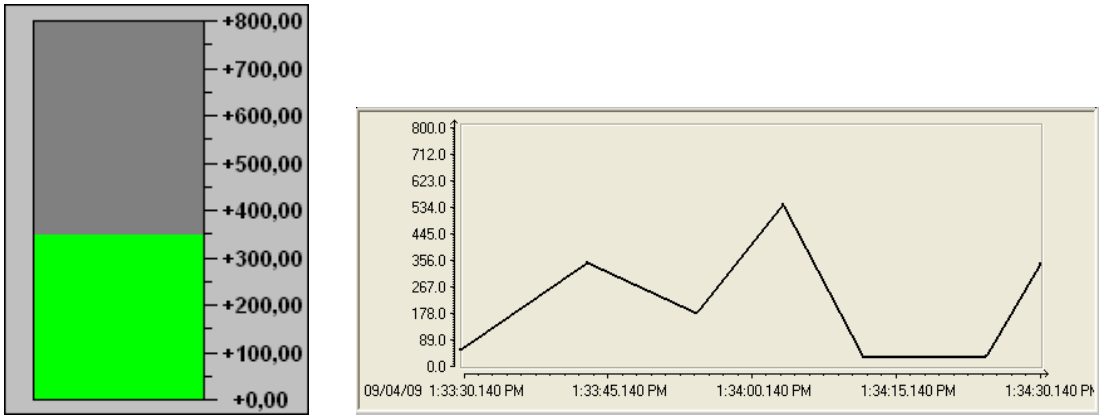
Edellä kuvattu toiminta painikkeelle tarkoittaa sitä, että painikkeen painaminen aiheuttaa tagin asettumisen johonkin tiettyyn arvoon. Ajatellaan tilannetta, jossa valvomosta halutaan ohjata painikkeella jotain muistipaikkaa logiikassaohjelmassa. Tällä tavalla tehtynä muistipaikka jää kyseiseen tilaan, eikä se muutu, ellei sitä erillisistä painikkeesta aseteta 0:ksi, eli esim. M1.0 jää ykköseksi, ja jotta se haluttaisiin muuttaa takaisin nollaksi, se tarvitsisi toisen painikkeen. Sinällään kahden painikkeen käyttäminen on toimiva, mutta logiikkaohjelmoinnin kannalta se saattaa aiheuttaa ongelmia.

Jos halutaan rakentaa painike siten, että muistipaikka käy ykkösenä, sen toiminta pitää rakentaa painikkeeseen. Tämä toimii siten, että ykköseksi asettaminen tehdään ”Press left”-toimintoon, ja nollaksi asettaminen ”Release left”-toimintoon.

Näin tehtynä painike toimii siten, että kun hiiren vasenta painiketta painetaan, muistipaikka saa arvon ”1”, ja kun irrotetaan hiiren vasemmasta painikkeesta, muistipaikka saa arvon ”0”. Kun painikkeet rakennetaan edellä mainitulla tavalla, niiden ohjattavia muistipaikkoja on helpompi liittää jälkeenpäin logiikkaohjelmaan, jos esim. moottorin ohjaus on tehty aliohjelmalla ja sen käynnistys- ja pysäytysohjaukset on tuotu aliohjelman ulkopuolelle. Täten ohjausmuistipaikkoja voidaan lisäillä toistensa rinnalle eikä tarvitse huolehtia siitä jääkö jokin muistipaikka päälle.

4.3.6 Trendit ja palkit

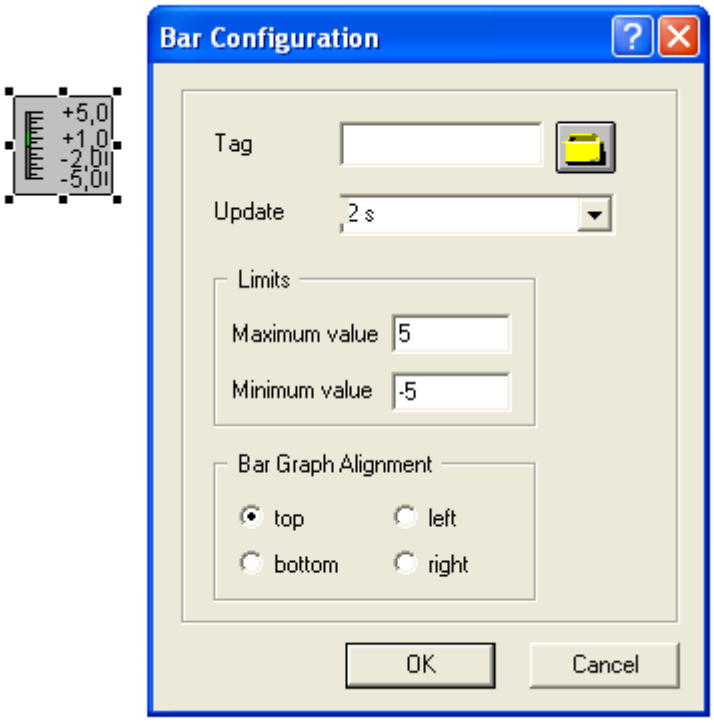
Valvomoissa käytetään trendejä ja palkkeja ilmaisemaan mitattavia asioita graafisesti. Graafinen esittäminen antaa käyttäjälle pelkistetymmän käsityksen mittauskohteiden arvoista. Ajatellaan tilannetta jossa valvomosta pystytään tarkastelemaan kohteen lämpötilan arvoa. Tällaisessa tapauksessa on toki suotavaa että näkyvillä on lämpötilan numeerinen arvo esim. 90 ºC. Kuitenkin pelkkä numeroarvo ei aina anna valvomon käyttäjälle tarvittavaa tietoa siitä miten kyseiseen lämpötilaan tulisi suhtautua. Tässä tapauksessa jos kohteen lämpötilan maksimiraja-arvona olisi 100 ºC, eikä käyttäjä olisi tästä tietoinen, kohde saattaisi kärsiä tai jopa rikkoutua. Palkkien avulla voidaan antaa käyttäjälle suuntaa-antavaa tietoa lämpötila-arvon laadusta. Trendejä voidaan valvomoissa käyttää piirtämään käyrää jonkin asian kehittymisestä ajan mukaan, esim. lämpötila ja sen laskut ja nousut ajan suhteen.



Kuva 4.19 Palkki ja Trendi

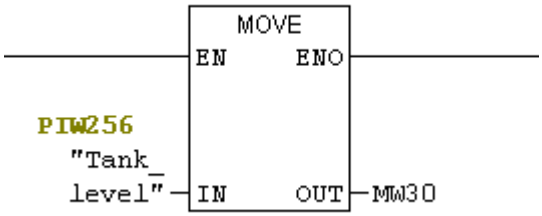
WinCC:llä tehtyihin palkkeihin voidaan asettaa maksimi- ja minimiarvot, sekä palkkien värejä voidaan muuttaa riippuen siitä, kuinka lähellä maksimia ja minimiä mitattavat arvot ovat. Värien avulla valvomon käyttäjä saa heti tietoa siitä, minkä laatuinen mitattavan kohteen arvo on. Tällaisesta voisi hyvänä esimerkkinä olla tilanne, jossa lämpötilaa esitetään palkkina. Palkin reunalla ovat lämpötila-arvot numeroina 0-100 ºC. Palkki lähtee kasvamaan sen mukaan, miten lämpötila nousee. Värejä voidaan tässä käyttää siten, että ensimmäinen kolmannes palkista olisi vihreän värinen, toinen kolmannes keltainen, ja viimeinen punainen. Mikäli tässä tapauksessa palkki vaihtaisi väriään punaiseksi, jo pelkästään tämän värin näkeminen antaisi käyttäjälle tiedon siitä, että jokin on menossa rikki tai prosessissa on jotain vialla.

Opinnäytetyössä toteutetussa valvomossa palkkeja käytettiin tankkien pinnankorkeuden ilmentämiseen, ja trendejä käytettiin sekoitustankin pinnankorkeuden esittämiseen käyrämuodossa. Palkkien luominen WinCC:llä tapahtuu WinCC Graphics Designerilla. Oikealla olevasta valikosta avataan ”Smart Objects” ja valitaan ”Bar”, joka viedään kuvaan. Kuvaan liittämisen jälkeen aukeaa valikko, josta määritellään palkin asetukset.



Kuva 4.20 Palkkien asetukset

Avautuneesta valikosta määritellään tagi kohdasta ”Tag”, päivitysnopeus ”Update” (aikaväli jolla tagin arvo päivittyy näytölle), sekä raja-arvot ”Limits”. Tagin liittämisessä on huomioitava, ettei WinCC:llä voida valvoa suoraan logiikassa olevaa sanaa, vaan kyseinen sana täytyy Simatic Managerissa siirtää muistisanaan, jota voidaan käyttää WinCC:ssä.

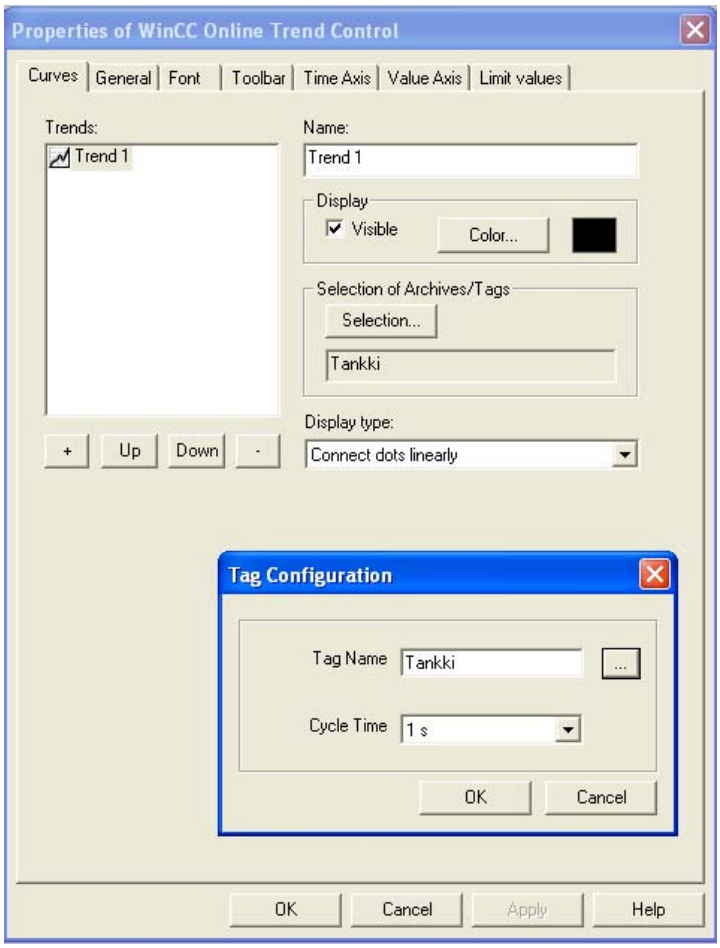


Kuva 4.21 Pinnankorkeussanan muuttaminen muistisanaksi

Kun muuttaminen on tehty, kyseinen muistisana voidaan luoda tagiksi WinCC:hen. Tämä tagi MW30 haetaan palkin tag-valikosta, päivitysnopeudeksi ”Upon Change” (palkki päivittyy aina kun muistisanan arvo muuttuu), ja määritellään halutut ylärajat. Määritellyt rajat tulevat näkyviin palkin viereen maksimi-, ja minimiarvoina esim. Maximum value 800 ja Minimum value 0. Näin toteutettu palkki näyttää tankin korkeuden välillä 0-800 litraa.

Trendejä luotiin WinCC:ssä Graphics Designerin avulla. Trendi-objekti löytyi Designer:in oikealla olevasta valikosta valitsemalla ”Controls” ja sen jälkeen ”WinCC Function Trend Control”. Kappale liitetään kuvaan, jonka jälkeen sen ominaisuuksia pääsee muokkaamaan.

Tuplaklikkaamalla Trendi-ikkunaa, aukeaa valikko josta trendille voi määrätä arkiston tai tagin jota se alkaa mitata.



Kuva 4.22 Trendi-ikkunan asetukset

Asetuksista määritellään trendille tagi jonka arvoa aletaan piirtää käyränä valitsemalla ”Selection of Archives/Tags”-kohdasta ”Selection”. Tämän jälkeen aukeaa Tag Configuration-valikko, ja siitä valitsemalla ”…” aukeaa ”Tag Management”-valikko, josta haluttu tankin pinnankorkeuden tagi ”Tankki:MW30” haetaan. Tämän jälkeen poistutaan asetuksista ja trendi alkaa mitata tagia valvomossa. Trendi-ikkunaa oikeaklikkaamalla pääsee käsiksi itse ikkunan asetuksiin ja muokkaamaan esim. sen ulkonäköä, värejä, raja-arvoja yms.

4.3.7 Hälytykset

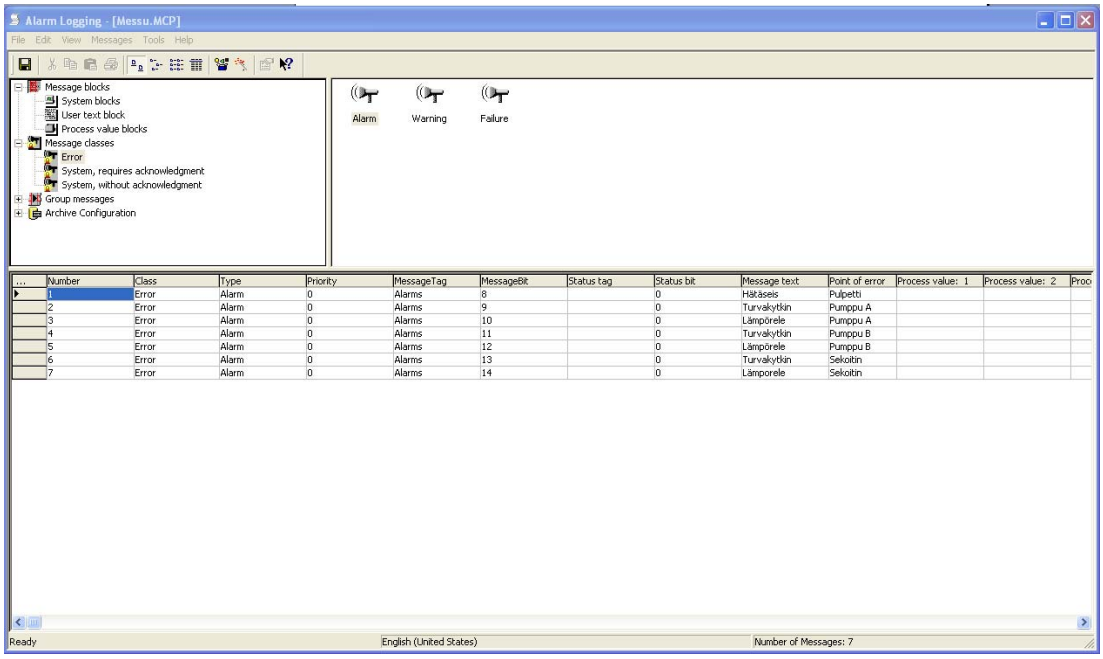
Sekoitusprosessia käyttäessä saattoi ilmetä tiettyjä vikatiloja, joista seurasi hälytys. Vikatiloja oli hätäseis, moottorien turvakytkimet ja suojakontaktorien laukeamiset sekä virtauksen loppuminen. Kytkimillä, kontaktoreilla ja hätäseispainikkeella oli

prosessissa omat signaalit jotka oli linkitetty tulokortteihin, kun taas virtauksen loppuminen tapahtui syöttötankkien pohjassa olevan käsiventtiilin sulkemisella.

Näitä hälytystiloja saatiin aikaiseksi valvomossa olevasta pulpettiohjaus-moodista, tai käsin tehtynä prosessista. Messutapahtumassa vikatiloja sai aikaiseksi vain valvomosta käsin.

Prosessissa sattuneesta vikatilasta tuli aina ilmoitus valvomoon. WinCC:ssä on hälytyksiä varten ”Alarm Logging”-ohjelma, josta hälytyksiä voitiin luoda ja niiden ominaisuuksia muokata. Ohjelmassa hälytykset voitiin luokitella kolmeen eri luokkaan: Alarms, Warnings ja Failure. Opinnäytetyön projektissa kaikki hälytykset luokiteltiin Alarm:eiksi.

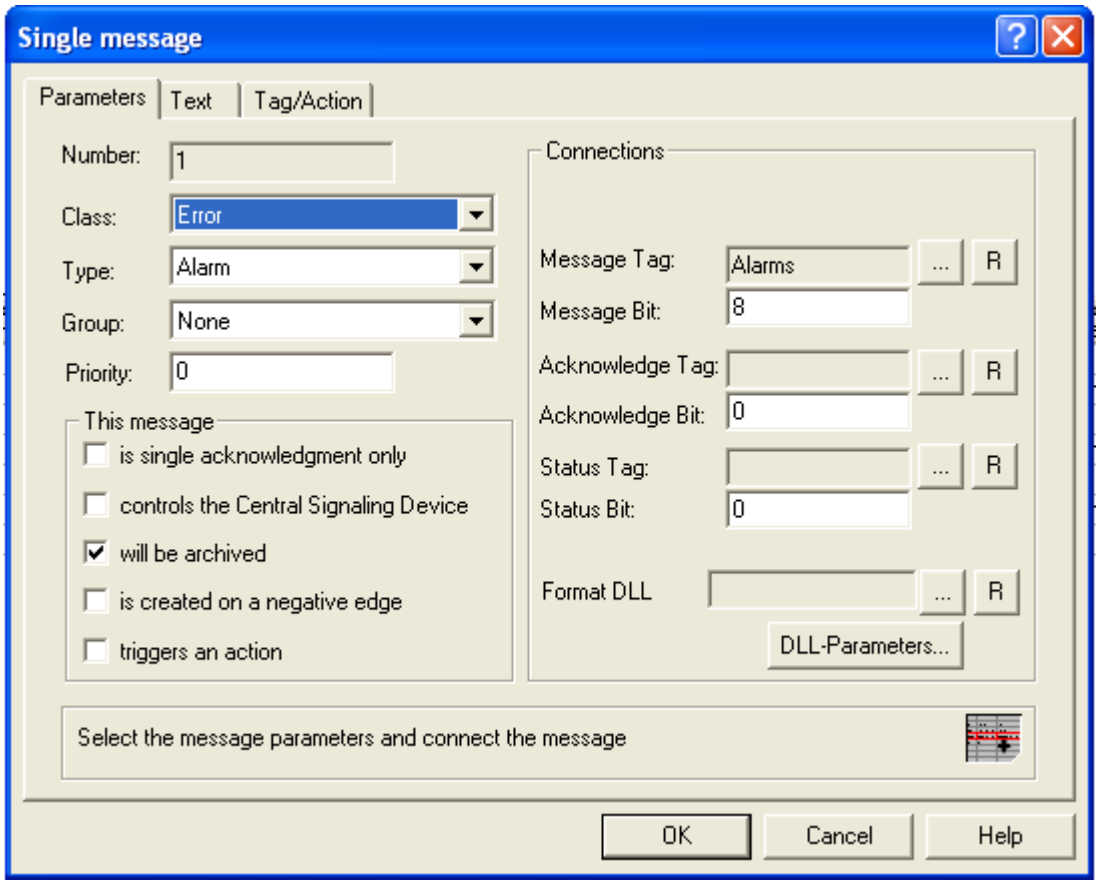
Ensimmäiseksi hälytyksiä tehtäessä niille luotiin WinCC:hen oma tagi. Tagin tuli olla 16-bittinen muistisana (MW), tämän muistisanan muistipaikkoihin liitettiin logiikkaohjelmoidut hälytykset.



Kuva 4.23 Alarm Logging

Kuvassa 4.23 näkyy Alarm Logging-ohjelman perusnäkymä. Ruudun alla olevaan kenttää käyttäjä sai lisätä hälytyksiä ja määritellä niiden ominaisuuksia esim. nimi, tyyppi, hälytysteksti yms. Oikea-klikkaamalla kenttää esiin tulevasta valikosta valit-

tiin ”Append New Line” ja luotua hälytystä oikea-klikkaamalla pääsi määrittelemään sen asetukset ”Properties”.



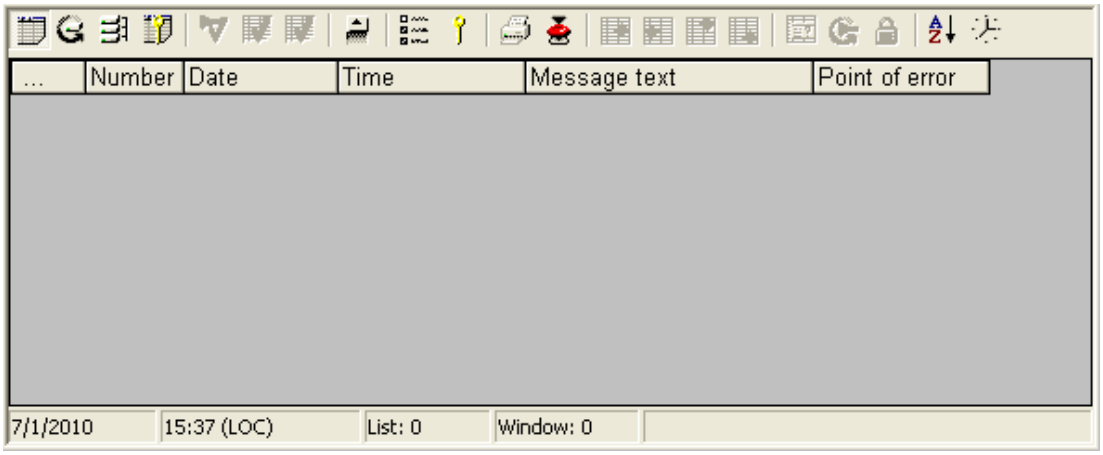
Kuva 4.24 Alarm Properties

Ilmestyneestä valikosta määritettiin hälytyksille luotu tagi ja kuinka mones bitti hälytystagista on kyseessä. Tässä kohtaa oli tärkeää huomata kuinka monenteen bittiin hälytystä viittasi, sillä muistisana käyttää kaksi tavua (MW20 = MB20 + MB21). Esim. MW20-muistisanan bitti M20.0, on vasta sanan 8. bitti. Muistisanan bittijärjestys lähtee sanan perältä, eli vasemmalta lukien sanan bitit koostuisivat M20.7, M20.6, M20.5, … M21.1, M21.0, jossa viimeinen muistibitti M21.0 on sanan ensimmäinen bitti, jonka järjestysnumero Alarm Logging:issa on 1. Opinnäytetyön hälytykset liitettiin logiikka ohjelmassa muistibitteihin M20.0 – M20.7, joten tässä tapauksessa ensimmäinen bitti oli järjestysnumeroltaan 8.

Kuva 4.25 Hälytykset logiikkaohjelmassa

Kuvassa 4.25 on esimerkki tulon liittämisestä muistibittiin Simatic Managerissa. Kun hätäseispainikkeen (NC) painiketta painetaan, tulo saa tilan ”0” ja muistisana M20.0 tilan ”1”, joka taas edelleen antaa tiedon Alarm Logging-ohjelmalle, että kyseistä painiketta on painettu.

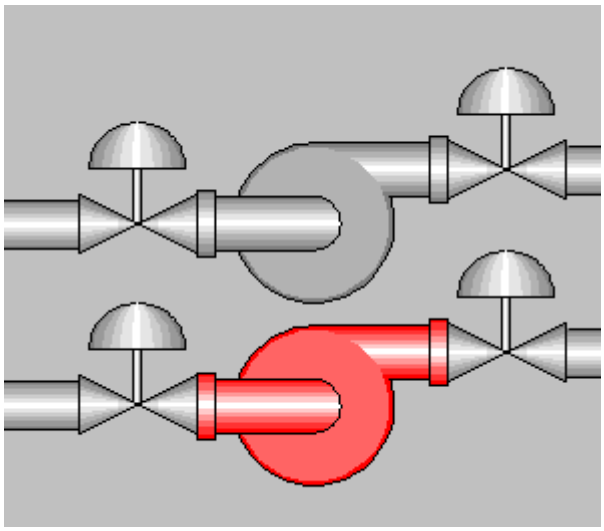
Hälytysten visualisointi valvomossa tehtiin siihen sijoitetulla hälytysikkunalla.



Kuva 4.26 Hälytysikkuna

Mikäli prosessissa tapahtui jokin vikatila, hälytysikkunaan ilmestyi asiasta ilmoitus. Hälytykset säilyivät siinä niin kauan, kunnes ne kuitattiin ikkunassa olevasta työkalupalkin kuittauspainikkeesta.

Lisävisualisointia hälytyksistä tehtiin valvomon prosessi-ikkunan kappaleisiin esim. jos pumpun moottorin turvakytkin väännettiin 0-asentoon, kyseisen pumpun ikoni alkoi vilkkua punaisella. Tästä käyttäjä sai heti tiedon siitä, että prosessissa on joku vialla.



Kuva 4.27 Viallinen pumppu

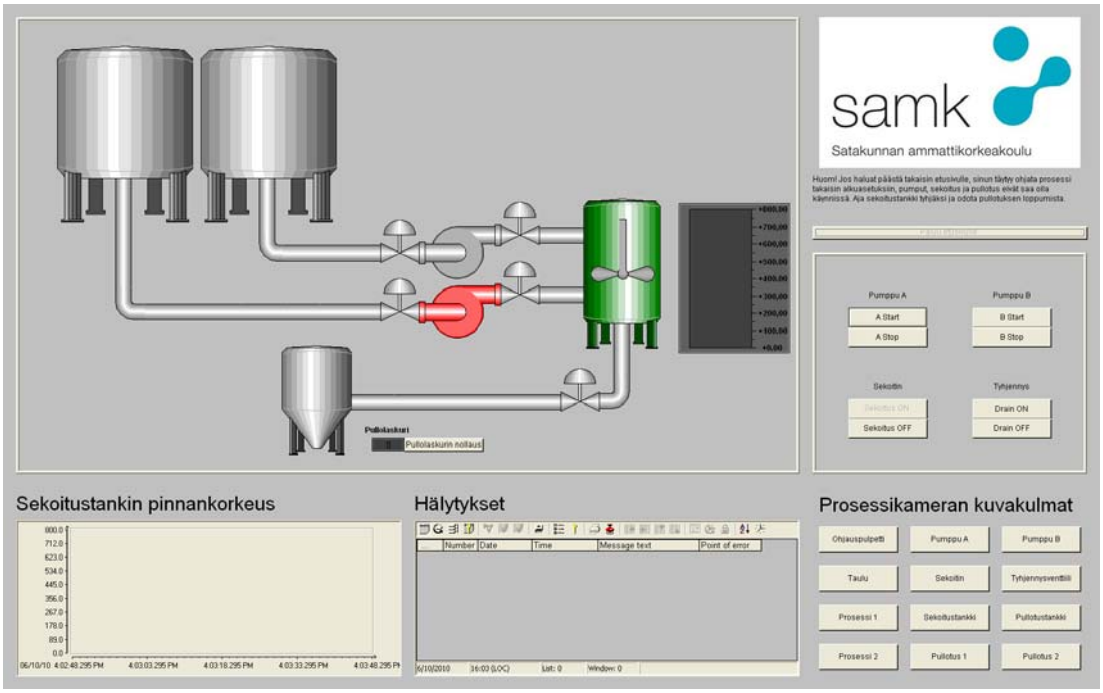
Automaatiomessuilla tehdyssä sekoitusprosessin demossa hälytyksien synty oli tehty scripteillä siten, että ohjelma kävi painamassa esim. hätäseis-painiketta ja valvomo reagoi siihen. Turvakytkimiä ja suojakontaktoreja ei päässyt vapaasti käyttämään messutapahtumassa, koska prosessin painikkeiden käyttäminen messutilanteessa oli hankalaa.

5 YHTEENVETO

Kuva 5.1

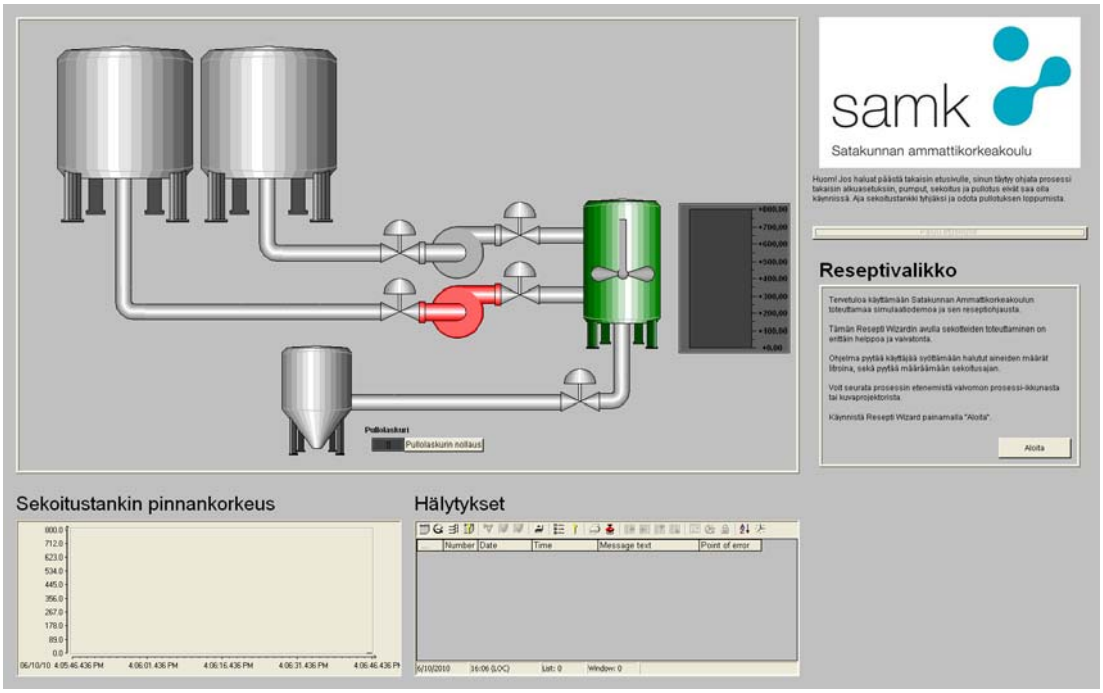
Kuvassa 5.1 näkyy valvomon aloitussivu. Sivun vasemmassa reunassa olevassa tekstikentässä annettiin lyhyt kuvaus järjestelmästä sekä sen ominaisuuksista. Lisäksi käyttäjälle annettiin lyhyt opastus valvomon eri ohjausmoodeista, jotka olivat valittavissa sivun keskellä olevista painikkeista. Painikkeita painamalla valvomon näkymä muuttui valitun moodin käyttöikkunaan.

Todellisessa valvomossa ei tämäntyylistä etusivua välttämättä olisi, mutta messutilanteen takia oli kannattavaa tehdä siitä näyttävän näköinen laittamalla siihen kuvia. SAMK-logo oli totta kai tärkeä koulun mainostamisen takia.



Kuva 5.2 Käsiohjaus-moodi

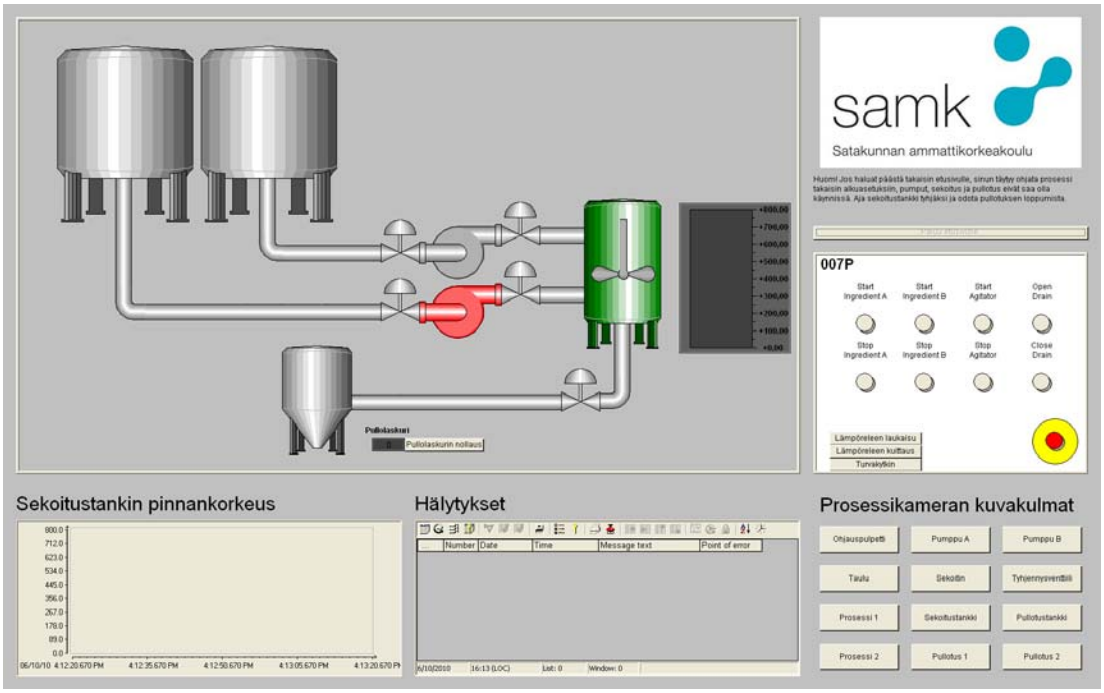
Käsiohjausvalikossa käyttäjä pystyi ohjaamaan prosessia manuaalisesti oikealla olevista painonapeista (esim. laittaa pumppuja päälle tai sekoitusta päälle vapaasti oman mielensä mukaan). Osa käyttövaihtoehdoista oli estetty riippuen prosessin tilasta. Käsiohjausmoodissa pääsi myös ohjaaman prosessin kameraa. Oikeassa alareunassa olevien painikkeiden avulla ohjattiin prosessiin rakennettuja scriptejä, jotka liikuttivat kameraa tiettyihin ennalta määrättyihin pisteisiin prosessissa esim. sekoitustankki tai pumppu A.



Kuva 5.3 Resepriohjaus-moodi

Reseptiohjauksessa käyttäjä ei voinut vapaasti ohjata prosessia, vaan valvomoon ja logiikkaohjelmaan rakennettu ”Resepti Wizard” hoiti automaattisesti prosessin ohjaamisen. Käyttäjältä kysyttiin syötettäviä tietoja (ainemäärät ja sekoitusaika), jonka jälkeen hän pystyi käynnistämään prosessin. Logiikkaohjelma käynnisti automaattisen käytön ja prosessi tuotti käyttäjän asettamien arvojen mukaisen tuotoksen. Lopuksi valvomoon tuli lukema kuinka paljon kumpaakin ainetta pumpattiin, yhdisteen sekoitusaika, sekä kuinka monta pulloa tehdystä sekoitteesta pullotettiin.

Reseptiohjausmoodissa kameraa ei päässyt ohjaamaan itse, vaan sen toiminta tapahtui automaattisesti logiikkaohjelman käskyjen mukaan. Reseptin ollessa käynnissä prosessin kameraa ohjattiin riippuen siitä, missä vaiheessa resepti oli. Tämän avulla messuilla annettiin kävijöille lisää infoa siitä mitä prosessissa tapahtuu. Kamera liikkui esim. pullotustankille kun pullotus oli käynnissä.



Kuva 5.4 Pulpettiohjaus-moodi

Tässä moodissa käyttäjä pääsi ohjaamaan prosessia prosessin omasta käyttöpulpetista. Kuten edellä mainittuna itse 3d-mallinnetun prosessin painikkeiden painaminen oli messutilanteessa hankalaa, joten pulpetin ohjaus toteutettiin valvomosta käsin. Kun ikkunan oikeassa reunassa olevasta paneelista painettiin painiketta, prosessiin rakennetut scriptit lähtivät käyntiin, ja prosessista käytiin painamassa kyseistä painiketta.

Lisänä tähän ohjausmoodiin tehtiin vikatilaesitykset. Pulpetin vasemmassa alanurkassa olevista painikkeista ohjattiin scriptejä, joissa prosessista käytiin asettamassa turvakytkimiä ja suojakontaktoreja 0-asentoon. Nämä toimenpiteen aiheuttivat hälytyksiä, joista tuli ilmoitukset valvomoon. Mikäli katkaisijoiden asentoa oli käyty muuttamassa, valvomosta ei päässyt ohjaamaan mitään muuta kuin kyseisiä katkaisijoita takaisin normaaliin tilaansa. Tämä tehtiin siksi, ettei messutilanteessa muihin ohjausmoodeihin olisi tullut komplikaatioita.

Pulpettiohjausmoodissa prosessin kameraa ohjattiin samalla tapaa kuin Käsiohjausmoodissa.

Kuva 5.5 Automaatio-09 messutapahtuma

Opinnäytetyöhön liittynyt projekti esitettiin Satakunnan Ammattikorkeakoulun esittelypisteessä Automaatio-09 messutapahtumassa Helsingissä.

Simuloidun sekoitusprosessin ohjaus oli työnä varsin mielenkiintoinen, sillä siinä yhdistettiin todellista ohjausjärjestelmää virtuaaliseen laitokseen. Tämänkaltaisia projekteja on alettu nykyään tehdä enenevissä määrin. Esimerkiksi Olkiluotoon valmistuvaa ydinvoimalaa varten on tällainen simulaatiomalli jo olemassa, jonka avulla tulevan voimalaitoksen käyttöhenkilöstöä koulutetaan jo ennen fyysisen laitoksen valmistumista. Mikäli simulaatiomallin tekijät ovat onnistuneet työssään hyvin, käyttöhenkilöstön ei tarvitse Olkiluodon tapauksessa tehdä muuta kuin vaihtaa työmaataan.

Tämän projektin arvioiminen simulaation onnistumisen kannalta on tässä tapauksessa melko hankalaa. Ongelmana on se, ettei 3d-mallinnettua prosessia ole fyysisesti olemassa ja tästä johtuen todellinen prosessin ja simulaatiomallin vertaaminen keskenään on mahdotonta. Mikäli tällainen prosessi olisi jossakin olemassa, tämä saattaisi soveltua parhaiten koulutuskäyttöön käyttöhenkilöstölle. Tutkimus- ja kehityskäytössä tällä voisi kehittää esim. logiikkaohjelmaa ajatellen pullojen tuotantoaikaa

tai verrata pitkässä juoksussa kuinka paljon sekoitusajan pidentäminen vaikuttaisi vuosittaiseen pullojen tuotantoon.

Prosessia voisi pikemminkin kuvailla käyttöliittymän suunnittelussa käytetyksi emulaatiomalliksi, jolla pyrittiin mallintamaan virtuaalisesti valvomosta annettavia käskyjä. Järjestelmä oli rakenteeltaan hyvin samanlainen ja tosielämässä myös sopisi samaan käyttöön kuin kappaleessa 2.4 kuvattu emulaatiomalli, joka rakennettiin käyttöliittymän testausta varten.

Projektia voidaan pitää onnistuneena, sillä järjestelmä toimi sille annetuissa puitteissa moitteitta ja Automaatio-09 messutapahtumassa järjestelmää kävi monet kiinnostuneet katsomassa, projektista tehtiin jopa pienimuotoinen lehtiartikkeli Prosessori-lehteen. Työnä projekti oli todella mielenkiintoinen ja opettava, sillä en ollut aiemmin käyttänyt Siemensin WinCC-ohjelmistoa.

Lopuksi haluaisin erityisesti kiittää opinnäytetyöni valvojaa Hannu Asmalaa, joka teki kovan työn 3D-prosessin rakentamisessa ja oli korvaamaton apu WinCC:llä työskentelyssä ja sen käyttämisen oppimisessa.

LÄHTEET

1. E. Yucesan, C.-H Chen, J. L. Snowdon, J.M Chames, eds, Generalizing: Is it Possible to create all-purpose simulations?, Proceedings of the 2002 Winter Simulation Conference

2. Ian McGregor, Richard A. Walters, Emulation: Emerging on the material handling frontier, Control Solutions International, April, 2003

3. Visual Components oy, PLC Add-on Manual and Tutorial, [Viitattu 9.3.2010] http://download.visualcomponents.net/www/Documents/Manuals/PLCAdd-on%20Manual.pdf

4. Frank Iwanitz, Jürgen Lange, OPC: Fundamentals, Implementation and Application, Hüthig GmbH & Co. KG Heidelberg, 2. painos, 2002

5. Visual Components oy, Profibus DP Multislave Interface for Emulating or Monitoring Profibus DP Devices, [Viitattu 11.3.2010] http://www.industrialnetworking.com/Manufacturers/SST-Network-Interface-Cards/SST-PBMS-PCI

6. Siemens Automation oy, Simatic WinCC – Basic Software, [Viitattu 9.3.2010] http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/simatic-wincc/Pages/Default.aspx

LIITE1

1. Siemens Step7 Simatic Managerilla tehty logiikkaohjelma.

Logiikkaohjelmaa ei löydy kansitetusta versiosta liitteenä, vaan ohjelma on tulostettuna kokonaisuudessaan SAMK TekPo:n automaatiolaboratoriossa opinnäytetyöstä tehdyssä mapitetussa versiossa.