Joni Rantala

Cromi S -sarjan ohjelmointisovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

4.5.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Joni Rantala Cromi S -sarjan ohjelmointisovellus 23 sivua + 2 liitettä 4.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	järjestelmäarkkitehti Kimmo Malmi lehtori Markku Inkinen

Työn tarkoituksena oli suunnitella THT Control Oy:n Cromi S -sarjan ohjelmoitavien logiikoiden ohjelmointisovellus. Ohjelmointisovellus vähentäisi ohjelmointityön välivaiheita ja keräisi olemassa olevat ohjelmapohjat ohjelmakirjastoksi.

Sovellus suunniteltiin mahdollisimman helppokäyttöiseksi, laitteistoriippumattomaksi ja monipuoliseksi. Suunnittelun alussa selvitettiin sovelluksen tarve ja tutkittiin nykyistä ohjelmointitapahtumaa. Todettiin sovelluksen voivan vähentää ohjelmointityöstä jopa seitsemän välivaihetta.

Sovelluksen ohjelmointikieleksi valittiin Java-kieli ja kehitysympäristöksi NetBeans IDE ja JavaFX Scene Builder. Toiminnot päätettiin toteuttaa sovellukseen lisättävillä aliohjelmilla, joita kutsutaan ajureiksi. Ajureiden tehtäviin kuului muuttujien, sovellusten ja ohjelmakoodin luominen ja muokkaus.

Sovelluksesta toteutettiin proof of concept -tyylinen demo, jolla varmistettiin sovelluksen toteutuksen olevan mahdollista. Sovelluksen toimintojen toteutustavasta saatiin selkeä suunnitelma, joka mahdollistaa sovelluksen joustavan jatkokehityksen.

Avainsanat	Java, PLC-ohjelmointi

Author(s) Title Number of Pages Date	Joni Rantala Programming Application for Cromi S -Series 23 pages + 2 appendices 4 May 2017
Degree	Bachelor of Engineering
Degree Programme	Automation Technology
Specialisation option	
Instructor(s)	Kimmo Malmi, System Architect Markku Inkinen, Senior Lecturer

The purpose of this study was to design a programming application for THT Control's Cromi S -series programmable logic controllers. The Programming application would decrease steps from programming work and gather existing program samples to program library.

Application was designed to be as easy to use as possible, hardware independent and versatile. Necessity of the application was determined in the beginning of design and current programming events were examined. It was noted that application could reduce as much as seven steps from programming work.

Programming language for the application was determined to be Java language and to use NetBeans IDE environment with Java FX Scene Builder application. Functions of the application were decided to be implemented as subroutines which are called drivers. Tasks of the drivers included code, variables and applications generation and editing.

As a result of the study, proof of concept demo was developed, which proved that implementation of the application is possible. A clear plan was made for implementing the applications functions, which enables flexible further development of the application.

Keywords	Java, PLC programming

Sisällys

Lyhenteet

IDE	*Integrated Development Environment.* Ohjelmointiympäristö.
I/O	*Inputs/Outputs.* Systeemin tulot ja lähdöt.
JAVA	Sun Microsystemsin kehittämä teknologiaperhe ja ohjelmistoalusta.
PC	*Personal Computer.* Yleisluontoinen tietokone.
PLC	*Programmable Logic Controller*. Ohjelmoitava logiikka.
RTU	*Remote Terminal Unit.* Etäterminaaliyksikkö.
SCADA	*Supervisory Control And Data Acquisition.* Valvomo-ohjelmisto.
ST	*Structured Text.* IEC61131-3:n mukainen ohjelmointikieli.
TCP	*Transmission Control Protocol.* Tietoliikenneprotokolla.
WLAN	*Wireless Local Area Network.* Langaton lähiverkko.
XML	*Extensible Markup Language.* Merkintäkielien yläkäsite tai standardi.

1 Johdanto

1.1 Yleistä

Insinöörityössä suunnitellaan ohjelmointisovellus Cromi S -sarjan ohjelmoitavan logiikan ohjelmointiin. Työ tehdään THT Control Oy:lle. Työn alkuperäinen tarkoitus oli toteuttaa suunniteltava sovellus, mutta sovelluksen tarpeesta, toiminnoista ja ulkoasusta ei ollut suunnitelmaa. Tästä johtuen työ vaihdettiin toteutustyöstä selvitys- ja suunnittelutyöksi. Ohjelmointisovelluksen tarkoituksena on vähentää ohjelmoinnin välivaiheita, nopeuttaa ohjelmointia ja koota olemassa olevat ohjelmat kirjastoksi. Cromi S -sarjan logiikoiden ohjelmointi tapahtuu nykyisin laitteessa olevalla selainkäyttöliittymällä, jossa toiminnot on sijoitettu usealle eri sivulle. Sivujen välillä siirtyminen lisää työskentelyn välivaiheita ja hidastaa ohjelmointityötä. Ohjelmointisovelluksen tarkoitus on koota ohjelmoinnissa tarvittavat toiminnot yhteen ikkunaan, josta ne työskennellessä on helppo löytää ja käyttää.

1.2 THT Control Oy

THT Control Oy suunnittelee ja toimittaa räätälöityjä automaatioratkaisuja teollisuuteen ja yksityisille toimijoille. THT Control Oy toimii ympäri Suomea, mutta päätoimipaikat sijaitsevat Tampereella ja Oulussa. [1.]

Asiakkaiden inspiroimana yritys on kehittänyt modulaarisen Cromi -tuoteperheen, johon kuuluvat S -sarjan ohjelmoitavat logiikat S1000 ja S500 sekä SCADA-valvomo [1].

2 Määrittely

Työn alussa selvitetään käytettävät laitteet ja tarve suunniteltavalle sovellukselle. Sovelluksen tarvetta arvioidaan tutkimalla nykyistä ohjelmointitapaa, selvittämällä sen työvaiheet ja vertaamalla sitä tavoitteeseen.

2.1 Laitteet

Cromi S -sarjan logiikat ja niiden lisäkortit valmistaa kanadalainen yritys nimeltään Inico Technologies Ltd.

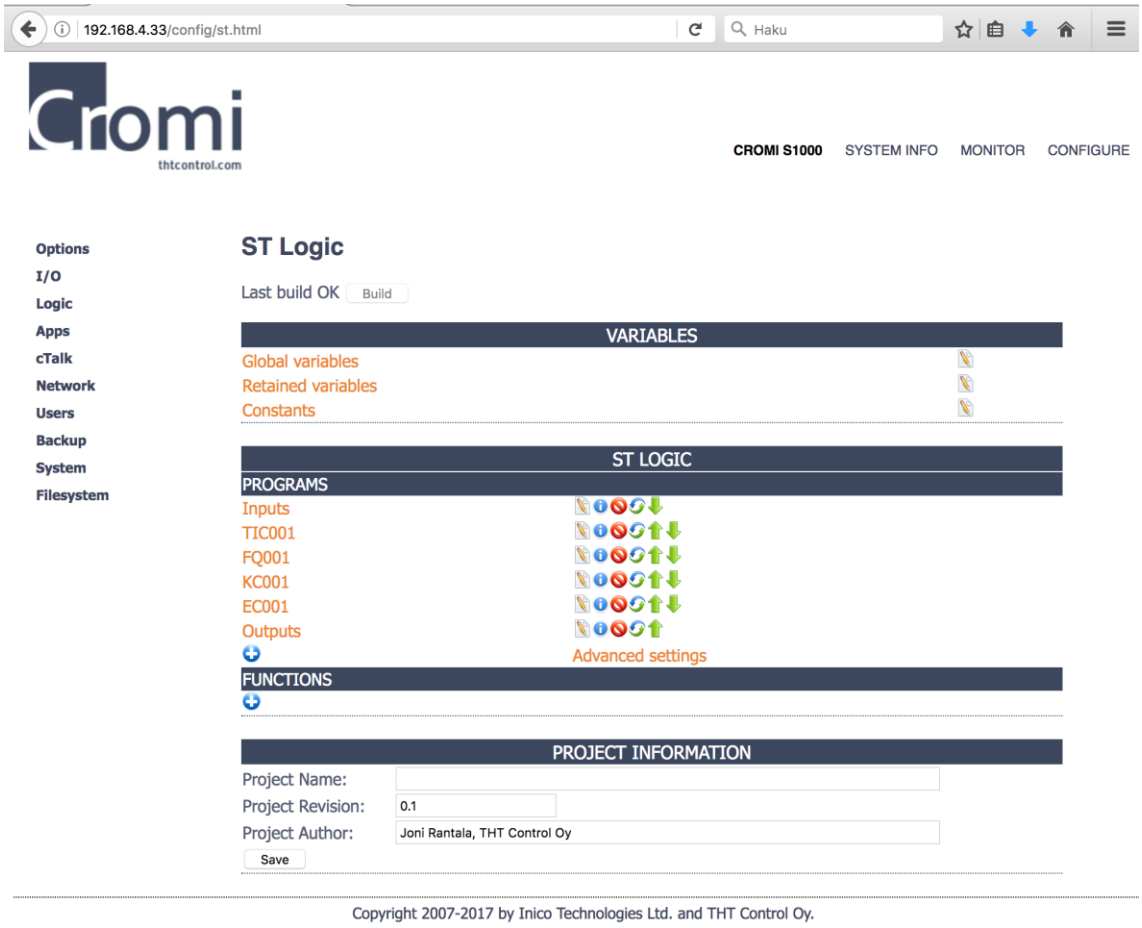
Cromi S500 -logiikka on tarkoitettu hajautettuihin järjestelmiin eikä siinä ole omia tuloja tai lähtöjä. Cromi S500 -logiikka on varustettu kahdella Ethernet-liitännällä, jotka voidaan määrittää kahdeksi eri verkoksi. Laitteessa on RS-485-sarjaliikenneväylä, joka tukee Modbus/RTU-protokollaa. Logiikan tulot ja lähdöt liitetään Ethernet- tai RS-485-väylällä ja kommunikointiprotokollana Modbus/TCP tai Modbus/RTU.

Cromi S1000 -logiikassa on omia tuloja ja lähtöjä: neljä analogista tuloa, kahdeksan digitaalista tuloa, kahdeksan digitaalista lähtöä, RS-232-sarjaportti ja USB1.1 huoltoportti. Logiikassa on yksi Ethernet-liitäntä ja WLAN on saatavilla lisäkorttina. Etäkäyttöä ja hälytysten lähetystä varten on saatavilla GSM-modeemi, joka tukee 2G- ja 3G-yhteyksiä. [3.]

Cromi S1000 -logiikkaan voidaan liittää neljä laajennuskorttia, yleisimpiä korttityyppejä ovat DIO-, IOX- ja Comm-RS485 -kortit.

DIO-kortissa on 16 kappaletta digitaalisia tuloja ja 16 kappaletta digitaalisia lähtöjä. IOX-kortissa on kahdeksan analogista tuloa, neljä analogista lähtöä, kahdeksan digitaalista tuloa ja kahdeksan digitaalista lähtöä. Comm-RS485-kortissa on RS-485-sarjaliikenneväylä, joka tukee Modbus/RTU-protokollaa.

2.2 Lähtötilanne



Kuva 1. Cromi S1000 -logiikan selainkäyttöliittymä.

Cromi S -sarjan logiikoiden asetukset ja ohjelmointi tehdään laitteessa olevan selainkäyttöliittymän avulla, joka on esitetty kuvassa 1. Laitteita ohjelmoitaessa selainkäyttöliittymällä joudutaan usein siirtymään sivulta toiselle, joka hidastaa työskentelyä ja voi aiheuttaa sekaannusta.

I/O-sivulla määritetään liitetyt lisäkortit, analogiatulojen skaalaukset ja mahdolliset väyläliitynnät.

Logic-sivulla tehdään logiikkaohjelmointi, globaalien ja kestomuistiin talletettavien muuttujien määritys. Logiikkaohjelma koostuu useasta tiedostosta, joiden suoritusjärjestys määräytyy listan mukaan. Ohjelmia voidaan ottaa käyttöön ja poistaa käytöstä sekä niille voidaan antaa yksilölliset suoritusvälit ja suoritusvälin poikkeamat. Muutosten jälkeen ohjelma tulee kääntää konekielelle painamalla Build-painiketta.

Apps-sivulla on laiteohjelmistoon kuuluvat sovellukset, joita ovat mm. PID-säädin, muuttujien arvojen seuranta ja ohjaus.

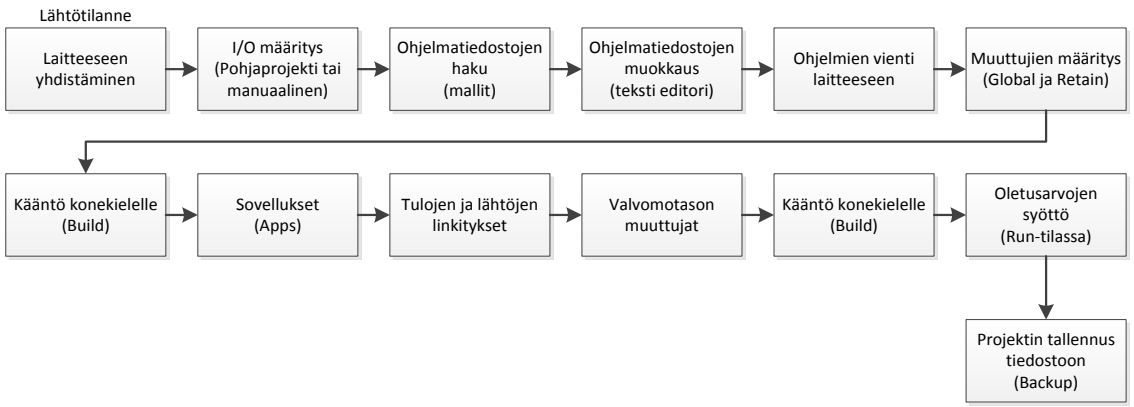
cTalk-sivulla määritellään valvomotason muuttujat ja Cromi S -sarjan logiikoiden välinen kommunikointi.

cTalk on ei-julkinen, binäärimuotoinen ja tapahtumapohjainen tiedonsiirtomuoto, joka on kehitetty erityisesti pienten ja yksinkertaisten laitteiden tiedonsiirtoon. Skaalautuu tarvittaessa massiivisten datamäärien siirtoon. Yhden viestin datamäärä on kuitenkin rajoitettu; eli siirtää vain mittaustietoja ja alle 255 merkin pituisia tekstejä. Tukee myös historian tallennusta, mikäli yhteys palvelimeen on poikki. cTalk tukee laite–laite ja laite–palvelin-tyyppistä tiedonsiirtoa. [2.]

Backup-sivulla voidaan tallentaa projekti PC:lle, palauttaa projekti logiikalle tai poistaa tietoja logiikalta.

Filesystem-sivulla voidaan selata, lisätä, poistaa tai muokata logiikassa olevia tiedostoja ja kansioita.

2.3 Ohjelmointitapahtuma



Kuva 2. Ohjelmointitapahtuma lähtötilanteessa.

Työssä käytetään ohjelmoinnin esimerkkinä lämpötilamittausta ja PID-säädintä. Ohjelmointitapahtuma lähtötilanteessa on esitetty kuvassa 2.

Ohjelmointi aloitetaan yhdistämällä PC ohjelmoitavaan laitteeseen Ethernet-kaapelilla ja avaamalla internet-selaimella laitteen selainkäyttöliittymä.

Selaimella siirrytään I/O-sivulle ja määritetään käytettävä I/O manuaalisesti tai siirrytään Backup-sivulle ja ladataan pohjaprojekti laitteeseen, joka sisältää tarvittavan I/O-määrityksen. Pohjaprojektin käytön etuna on, että se sisältää tulojen ja lähtöjen muuttujat ja logiikkaohjelmat tuloille ja lähdöille.

I/O-määrityksen jälkeen logiikkaohjelmointi aloitetaan siirtymällä Logic-sivulle. Tarvittavat ohjelmatiedostot lisätään ohjelmalistaan, asetetaan ohjelmien parametrit ja järjestetään ohjelmat niin, että tuloja käsittelevä ohjelma on ensimmäisenä ja lähtöjä käsittelevä ohjelma viimeisenä. Yrityksellä on kattava valikoima testattuja malliohjelmia, joita muokkaamalla saadaan toteutettua monimutkaisiakin kokonaisuuksia. Malliohjelmat muokataan tarpeeseen sopiviksi selainkäyttöliittymän ST-editorilla tai PC:n tekstieditorilla. Ohjelmasta vaihdetaan muuttujien nimet vastaamaan ohjelmoitavan piirin nimeä, esimerkiksi lämpötilamittauksen muuttujat alkavat tekstillä TIC001. Ohjelma kopioidaan tekstieditorista ohjelmatiedostoon ja tallennetaan tehdyt muutokset.

Ohjelmien luomisen ja muokkauksen jälkeen määritellään käytettävät muuttujat globaaleihin (Global variables) ja muistaviin (Retained variables) muuttujalistoihin. Muistavien muuttujien arvot säilyvät sähkökatkossa ja logiikan uudelleen käynnistyksessä, toisin kuin globaalien muuttujien arvot.

Muuttujien määrityksen jälkeen ohjelma käännetään konekielelle. Mikäli ohjelmassa on virheitä kääntäjä ilmoittaa niistä. Kääntäjä julkaisee muuttujat logiikan sisäiseen käyttöön, jonka jälkeen niitä voidaan käyttää sovelluksissa.

Laiteohjelmistoon kuuluvia sovelluksia lisätään Apps-sivulla. Melkein jokainen ohjelma tarvitsee arvojen seurantaan ja asetukseen sovelluksen. Yhdellä sovelluksella voidaan seurata tai ohjata useamman ohjelman muuttujia, eli ne eivät ole sidoksissa logiikkaohjelmiin. PID-säädin on sovellus johon tuodaan säätimen muuttujat. Sovellukset määritellään logiikan ollessa ohjelmointi-tilassa, ja muuttujien arvoja voidaan muuttaa suoritustilassa.

Sovellusten lisäämisen jälkeen ohjelmien ja sovellusten käyttämät tulot ja lähdöt lisätään tulojen ja lähtöjen logiikkaohjelmiin.

Valvomotason muuttujat voidaan lisätä cTalk-sivulla yksitellen tai muodostamalla muuttujat laskentataulukossa ja kopioimalla muodostettu XML-koodi cTalk.xml-tiedostoon Filesystem-sivun kautta.

Ohjelma käännetään konekielelle ja logiikka käynnistetään uudelleen suoritustilaan.

Projektin asetusarvot syötetään sovellusten kautta logiikan suorittaessa ohjelmaa. Projektin tallennus PC:lle logiikan ollessa suoritustilassa tallentaa myös asetusarvot muistiin.

2.4 Tavoite

Tavoitteena on vähentää välivaiheita, helpottaa ja nopeuttaa ohjelmointia sekä koota olemassa olevat ohjelmat kirjastoksi. Tavoitteeseen pääsemiseksi ohjelmointisovelluksen toiminnot on suunniteltava huolella ja käyttöliittymän selkeyteen kiinnitettävä erityistä huomiota. Ohjelmointitapahtuman tavoite on esitetty kuvassa 3.

Ohjelmointisovelluksen avulla projekti voidaan ohjelmoida valmiiksi ilman laitetta ja tallentaa se PC:lle käyttöönottoa varten. Käyttöönotossa valmis tiedostopaketti ladataan logiikkaan, ohjelmat käännetään konekielelle ja logiikka käynnistetään uudelleen suoritustilaan.

Ohjelmointi aloitetaan luomalla uusi tai avaamalla aiemmin luotu projekti.

Ensimmäisenä määritetään projektin laiteasetukset. Laiteasetuksista valitaan käytettävä logiikka ja siihen liitetty I/O. Määrityksen hyväksyttyä sovellus luo projektin kansiorakenteen, laiteasetuksiin liittyvät tiedostot ja sovellus palautuu päänäyttöön.

Logiikkaohjelmointi tapahtuu sovelluksen päänäytössä. Yläriviltä pudotusvalikosta valitaan tarvittava ohjelmapohja ja painetaan Lisää ohjelma -painiketta. Sovellus avaa ohjelman asettelu-ikkunan, jossa annetaan uudelle ohjelmalle nimi ja tarvittavat parametrit. Kaikille ohjelmille yhteisiä parametreja ovat ohjelman suoritusväli, suoritusvälin poikkeama ja ohjelman käynnistyksen tyyppi. Ohjelman tulot ja lähdöt määräytyvät lisättävän ohjelman mukaan, jotka määritellään saman ikkunan välilehdillä. Muuttujien

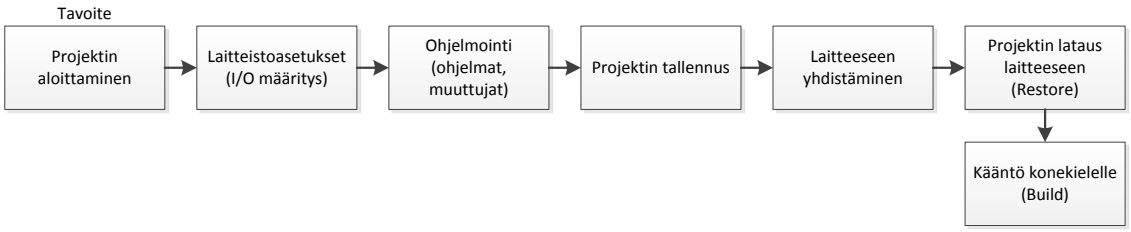
oletusarvot syötetään parametrien asetuksen yhteydessä. Hyväksynnän jälkeen sovellus luo ohjelman tarvitsemat kansiot, tiedostot, muuttujat ja sovellukset.

Logiikkaohjelmoinnin jälkeen projekti tallennetaan PC:lle logiikan tukemaan tiedostomuotoon.

Yhdistetään PC ohjelmoitavaan laitteeseen Ethernet-kaapelilla ja avataan internetselaimella laitteen selainkäyttöliittymä.

Selaimella siirrytään Backup-sivulle ja ladataan luotu projektitiedosto logiikkaan Restore-toiminnon avulla.

Projektin palauttamisen jälkeen käännetään ohjelmat konekielelle ja käynnistetään logiikka uudelleen suoritustilaan.



Kuva 3. Ohjelmointitapahtuman tavoite.

2.5 Tarve

Lähtötilanteen ja tavoitteen perusteella sovellukselle on selkeä tarve.

Ohjelmointisovelluksen käyttö vähentää välivaiheiden määrän 13:sta seitsemään, eli noin 40 % välivaiheista jää pois. Kaikkea kirjoittamiseen ja toimintojen välillä siirtymiseen kuluvaa aikaa ei voida poistaa, joten ohjelmoinnin oletetaan nopeutuvan 20–30 %. Lisäksi yrityksessä on vain muutama työntekijä joilla on riittävä osaaminen logiikkaohjelmointiin. Sovelluksen avulla logiikkaohjelmointi helpottuu ja yksinkertaisemmat ohjelmointityöt voi hoitaa vähemmälläkin osaamisella.

Ohjelmakirjastosta on ollut yrityksessä aiemmin keskustelua. Silloin päädyttiin tallentamaan malliohjelmat verkkolevylle, josta niitä haetaan tarpeen mukaan. Tekstitiedos-

tona olevia malliohjelmia ei voi suoraan viedä logiikkaan, koska tiedostoon on tallennettu kaikki ohjelman käyttämät muuttujat ja logiikkaohjelma. Malliohjelman siirto pitää tehdä monessa osassa ja moneen tiedostoon.

Ennen päätöstä sovelluksen toteuttamisesta vaihtoehtona oli tehdä malliohjelmat laiteohjelman sovelluksilla. Ongelmaksi muodostui ohjelmien muokattavuus ja hallinta, ohjelman muutos tai lisäys vaatisi aina uuden laiteohjelman ohjelmoinnin.

3 Ohjelmointisovelluksen suunnittelu

Sovelluksen suunnittelukriteereinä olivat helppokäyttöisyys, laitteistoriippumattomuus ja monipuolisuus. Sovelluksen käyttö ilman aiempaa ohjelmointikokemusta ja välttävillä tietoteknisillä taidoilla tulee onnistua ilman käyttöohjetta. Työssä tehty käyttöliittymä toimii pohjana toimintojen suunnittelulle. Toimintojen toteutuksen jälkeen sovellusta testautetaan eri käyttäjäryhmillä, joiden palautteen perusteella käyttöliittymää muokataan.

Laitteistoriippumattomuuden vuoksi päädyttiin käyttämään Java-ohjelmointikieltä. Javan valintaa tuki myös mahdollisuus toteuttaa sovellus yrityksen palvelimelle, josta se ladataan ja käynnistetään selaimen avulla. Palvelimelta ladattavan ja suoritettavan sovelluksen etuna on se, että ohjelmakirjastot ovat yhdessä paikassa, eikä levitettynä useaan paikkaan, jolloin sovelluksen ylläpito ja päivitys helpottuu.

Ohjelmointisovelluksen suunnittelu aloitettiin käyttöliittymän suunnittelulla, koska käyttöliittymän käytettävyys määrää toimintojen toimintatavan. Käyttöliittymällä on suuri rooli sovelluksessa: jos käyttäjä ei löydä tarvitsemaansa helposti, ei sovellusta silloin käytetä.

3.1 Kehitysympäristö

Ohjelmointikieleksi valittiin Java, kehitysympäristöksi NetBeans IDE 8.2 ja JavaFX Scene Builder 2.0 käyttöliittymän ohjelmointiin. THT:llä käytetään Java-kieltä laajasti valvomo-ohjelmoinnissa ja työntekijöillä on kokemusta kyseisistä ohjelmistoista. Sovel-

luksen kehitykseen pystyy osallistumaan useampi yrityksen työntekijä ilman erillistä koulutusta.

3.1.1 Java

Java on saanut nimensä Java-kielen kehittäjien nauttiman kahvilajin mukaan, joka on peräisin Jaavan saarelta Indonesiasta. Ohjelmointikielen kehityksen aloittivat vuonna 1991 James Gosling, Mike Sheridan ja Patrick Naughton Sun Microsystemsillä. Ensimmäinen Java, versio 1.0 julkaistiin syksyllä 1995. [7.]

Java-kielen tarkoituksena on olla laitteistoriippumaton, tarkoittaen että käännetty Java-koodi voidaan suorittaa millä tahansa alustalla, joka tukee Javaa. Java-sovellukset käännetään tavukoodiksi, joka voidaan suorittaa missä tahansa Java-virtuaalikoneessa. [7.]

Vuonna 2007 Sun Microsystems uudelleen lisensoi suurimman osan Java-tekniikoista GNU General Public License -lisenssillä, vapauttaen Java-tekniikat vapaaseen jakeluun [7].

Vuonna 2010 Oracle Corporation hankki Sun Microsystemsin, jolloin Java, NetBeans ja JavaFX siirtyivät Oraclen omistukseen [8].

3.1.2 NetBeans IDE

NetBeans IDE on Java-kielellä kirjoitettu ohjelmakehitysalusta, joka koostuu moduuleiksi kutsutuista sovelluskomponenteista. NetBeans vaatii toimiakseen Java-ajoympäristön ja Java-virtuaalikoneen (Java Virtual Machine), mutta on siksi alustasta riippumaton ja toimii kaikissa Javaa tukevissa käyttöjärjestelmissä.

Vuonna 1996 Kaarlen yliopistossa Prahassa käynnistyi NetBeans-projekti, joka kehittyi kaupalliseksi tuotteeksi. Javan kehittänyt Sun Microsystems osti NetBeansin kehittämiseen keskittyneen yrityksen.

Vuoden 2000 kesäkuussa Sun Microsystems julkaisi avoimen lähdekoodin NetBeans-projektin.

3.1.3 JavaFX Scene Builder

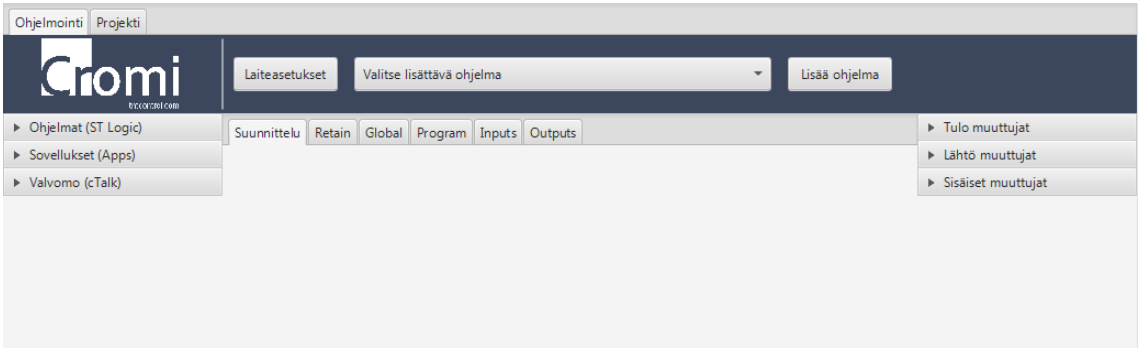
JavaFX Scene Builder on käyttöliittymien suunnitteluun tarkoitettu työkalu, jonka käyttämiseen ei tarvita koodausta. Käyttäjä voi vetää ja pudottaa käyttöliittymän komponentteja työskentelyalueelle ja muokata niiden ominaisuuksia. Työkalu tuottaa ulkoasun FXML-koodin automaattisesti. [5.]

JavaFX Scene Builder -työkalua voidaan käyttää minkä tahansa Java-kehitystyökalun kanssa, mutta NetBeans IDE:hen se integroituu parhaiten. [5.]

Vuonna 2007 Sun Microsystems ilmoitti ensimmäisen kerran JavaFX:stä JavaOne-nimisessä maailmanlaajuisessa Java-kehittäjien konferenssissa. JavaFX 1.0.2 julkaistiin vuoden 2008 joulukuussa. [6.]

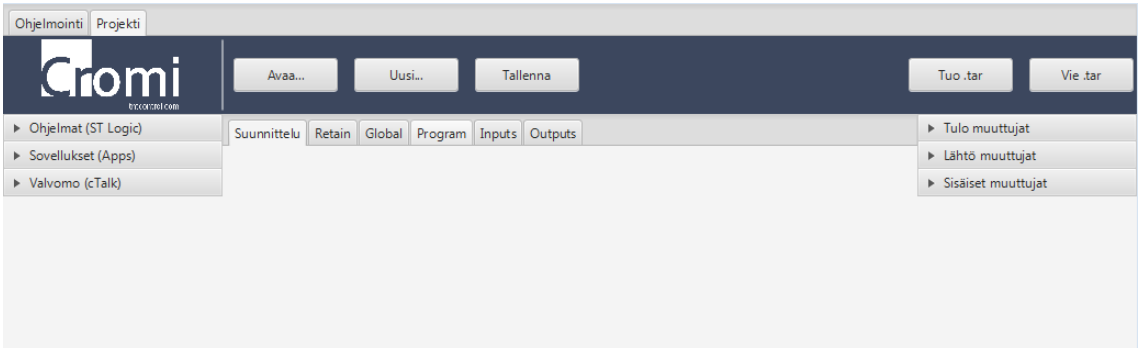
3.2 Käyttöliittymä

Käyttöliittymä koostuu yhdestä ikkunasta ja kahdesta välilehdestä. Ohjelmointiin tarvittavat toiminnot ovat ensimmäisellä välilehdellä, ja projektin hallintaan liittyvät toiminnot ovat toisella välilehdellä.



Kuva 4. Logiikkaohjelmointi.

Kuvassa 4 on esitetty sovelluksen käyttöliittymän Ohjelmointi-välilehti.

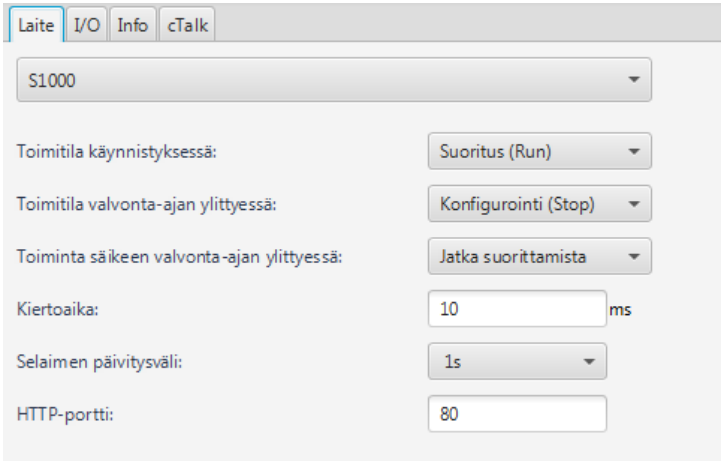


Kuva 5. Projektin hallinta.

Kuvassa 5 on esitetty sovelluksen käyttöliittymän Projekti-välilehti, josta löytyy projektin hallintaan tarvittavat toiminnot.

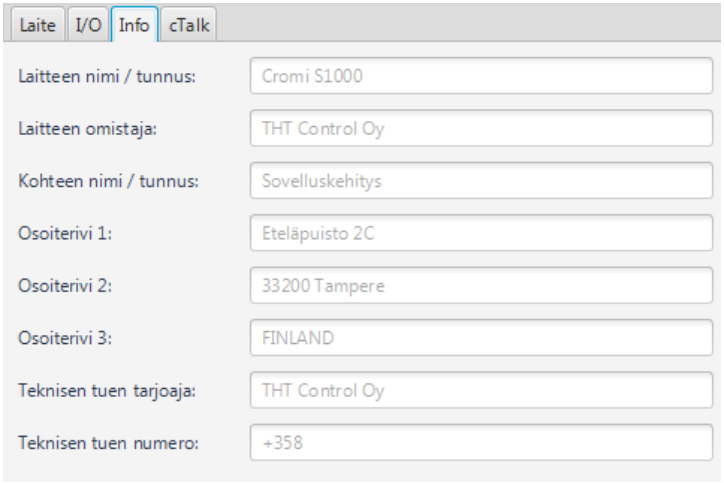
3.2.1 Laiteasetukset

Laiteasetuksiin siirrytään Ohjelmointi-välilehdeltä painamalla Laiteasetukset -painiketta, joka avaa kuvan 6 mukaisen ikkunan.



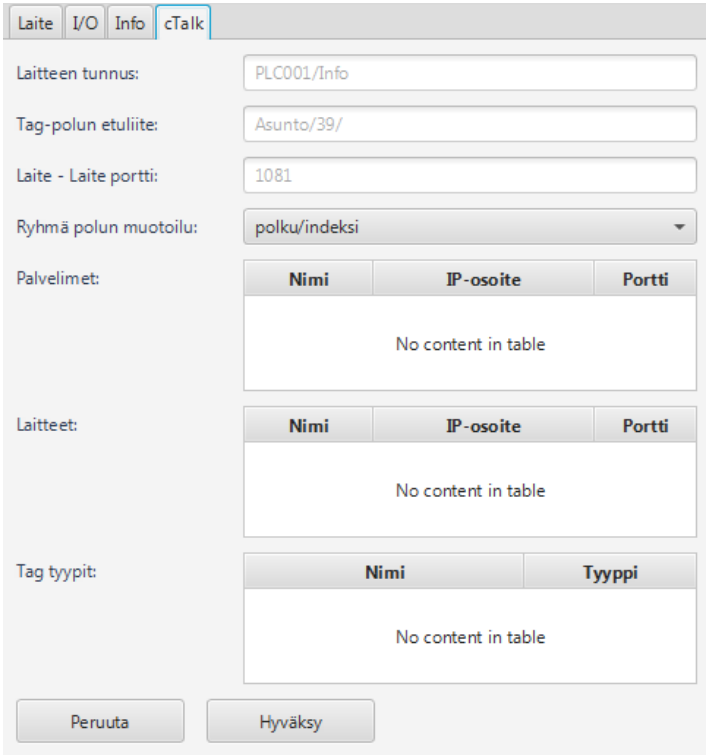
Kuva 6. Laitteen valinta ja asetukset.

Laiteasetusten ensimmäisellä välilehdellä valitaan käytettävä laite alasvetovalikosta. Logiikan toimitilat eri tilanteissa ja selainkäyttöliittymän päivitysväli valitaan alasvetovalikoilla. Kiertoaika-asetuksella määritellään, missä ajassa logiikka suorittaa ohjelmansa. Selainkäyttöliittymän portti voidaan vaihtaa muuttamalla HTTP-portti-asetusta.



Kuva 7. Laitteen lisätiedot.

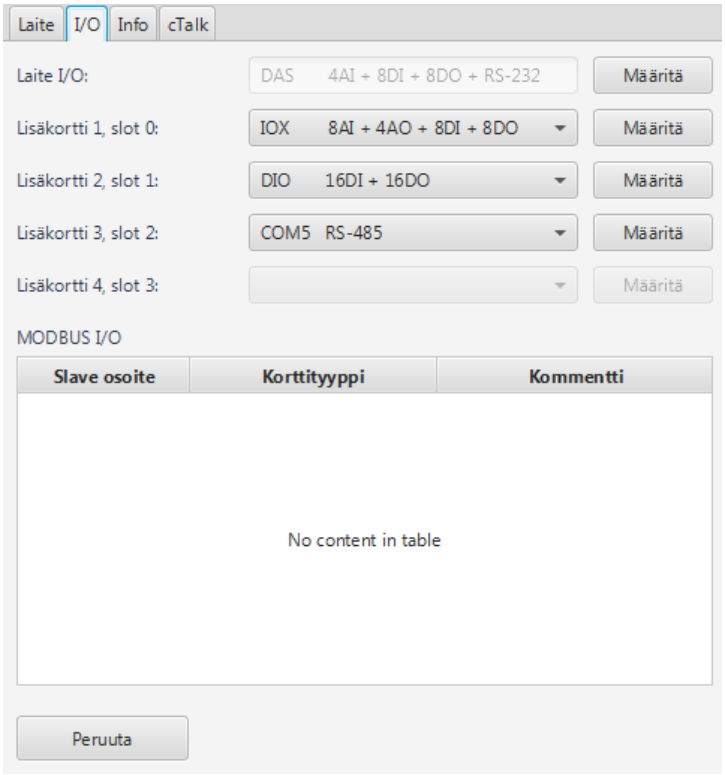
Lisätietoihin annetaan logiikan täydentävät tiedot, kuten kuvassa 7.



Kuva 8. cTalk kommunikoinnin asetukset.

cTalk-välilehdellä asetetaan laitteen kommunikointiasetukset, jotka esitetty kuvassa 8. Logiikalle annetaan yksilöllinen tunniste ja polun etuliite kuvaa minne logiikka on asennettu. Laite–Laite porttiasetuksella määritetään logiikoiden välisen tiedonsiirron käyttämä portti.

3.2.2 I/O-määritys



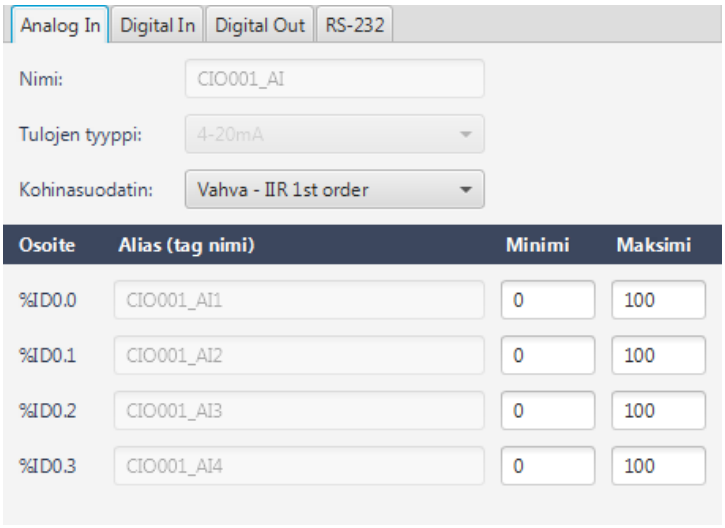
Kuva 9. Käytettävän I/O:n valinta.

I/O-välilehdellä määritetään logiikkaan kytketyt tulot, lähdöt ja väylälaitteet.

Mikäli logiikaksi on valittu Cromi S1000 kuten kuvassa 9, voidaan alasvetovalikoilla valita lisäkortit ja suorittaa niiden määritys. Cromi S500 -logiikassa voidaan käyttää ainoastaan Modbus-protokollalla liitettyä I/O:ta, jota voidaan käyttää myös Cromi S1000 -logiikassa. Taulukkoon lisätään käytettävät kortit, Slave-osoite voi olla IP-osoite tai sarjaväylää käytettäessä kortin Slave-numero. Kortin tyyppi valitaan alasvetovalikolla. Kommenttiin voidaan kuvata tarkemmin laitetta tai muuta tietoa.

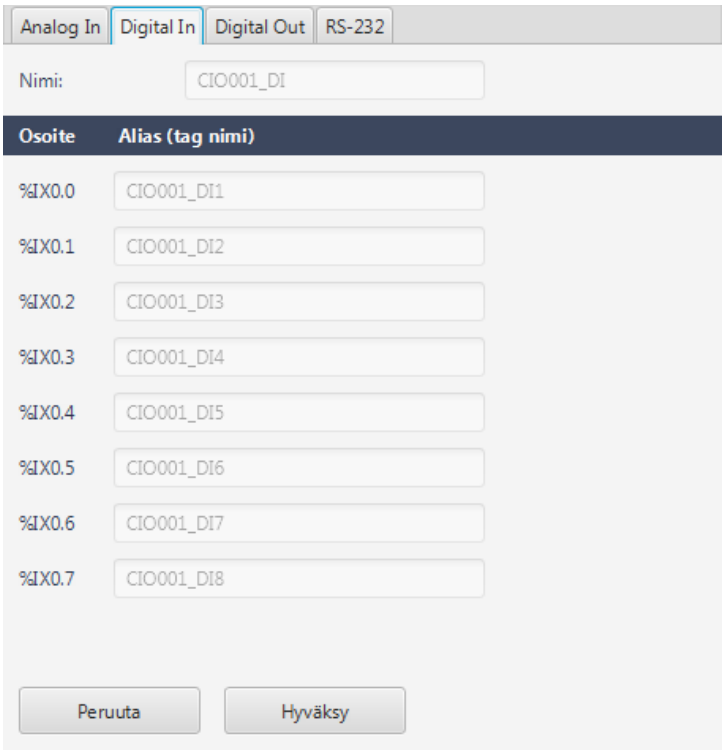
Cromi S1000 -logiikan lisäkorttien asetuksiin siirrytään painamalla Määritä-painiketta.

I/O-korttien nimiä tai kanavan nimimerkkiä ei voi muokata, sillä se noudattaa THT:n määrittelemää nimeämiskäytäntöä, joka on esitetty liitteessä 1.

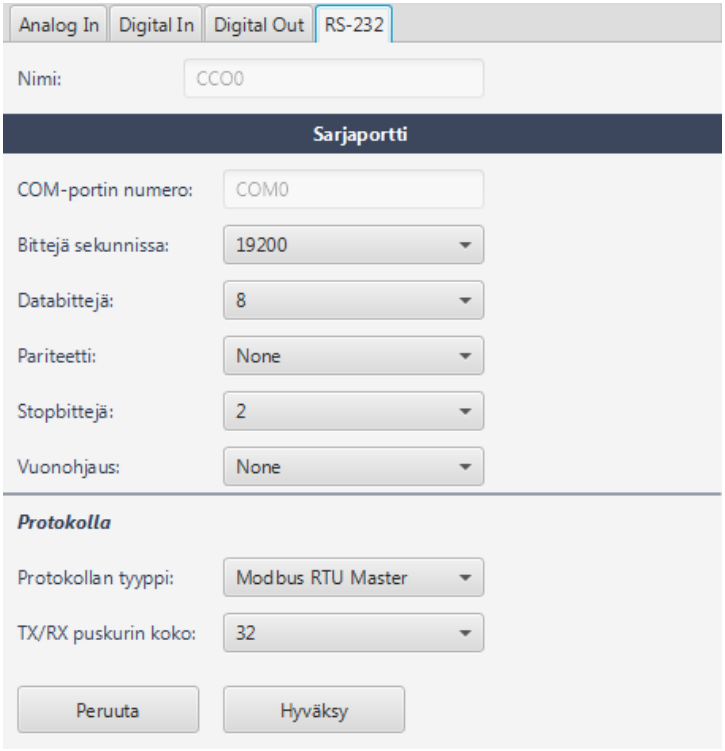


Kuva 10. Cromi S1000 -logiikan I/O-kortin määritys, analogiatulot.

Analogiatulot ovat 0/4–20 mA:n virtatuloja, jännitetuloilla varustettuja kortteja valmistetaan tilauksesta. Analogiatuloille voidaan määrittää kohinasuodatin alasvetovalikolla ja mittausalueet määritellään minimi- ja maksimi-asetuksilla kuvan 10 mukaisesti.

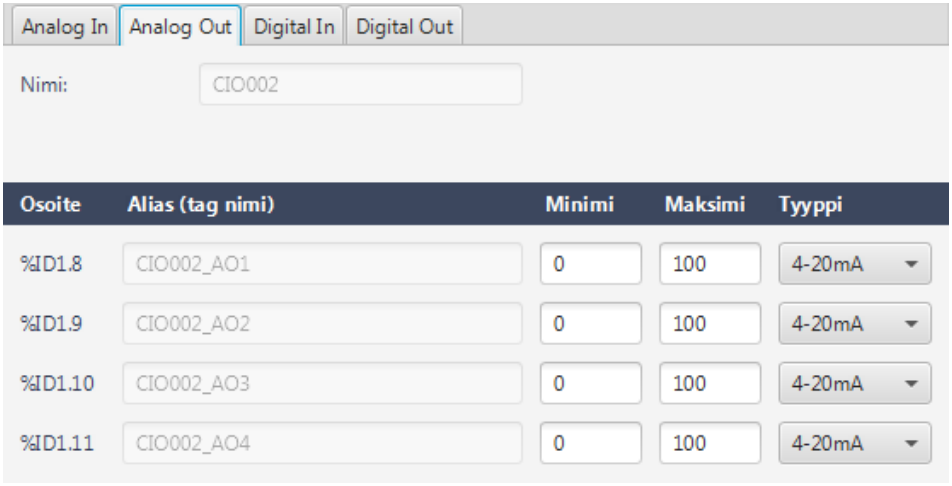


Kuva 11. Digitaalisille tuloille ja lähdöille ei ole muutettavia asetuksia.



Kuva 12. RS-232- ja RS-485-porteilla on samat asetukset.

Kuvassa 12 on sarjaportin asetukset, jotka kaikki valitaan alasvetovalikoilla.

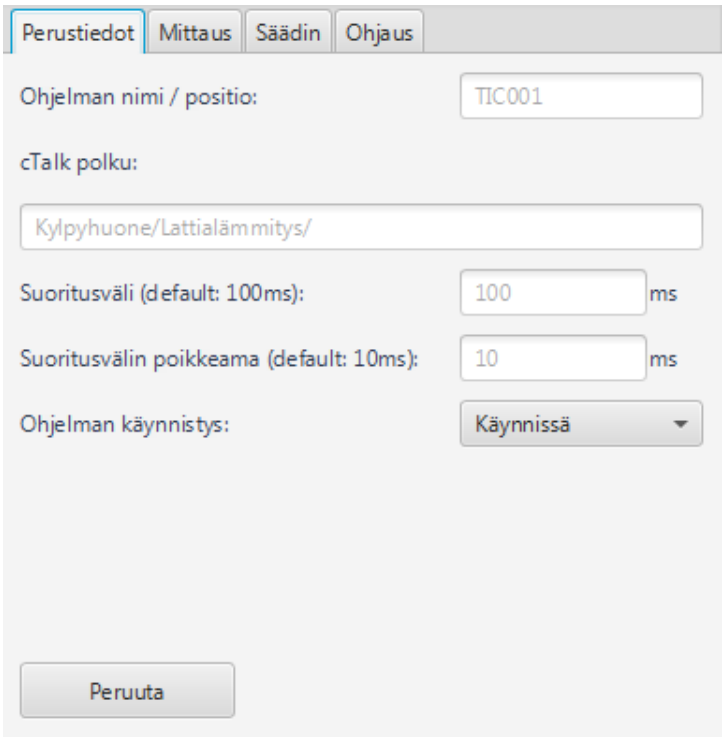


Kuva 13. IOX-kortin analogialähtöjen asetukset.

Analogialähtöjen alue määritellään minimi- ja maksimi-asetuksilla kuvan 13 mukaisesti. Analogialähdöt ovat 0/4–20 mA virtalähtöjä.

3.2.3 Ohjelmointi

Ohjelmointi tapahtuu Ohjelmointi-välilehdellä, lisättävä ohjelma valitaan alasvetovalikosta ja painetaan Lisää ohjelma -painiketta. Sovellus avaa kuvan 14 mukaisen ohjelman asetteluikkunan, jossa asetetaan ohjelman parametrit ja oletusarvot.



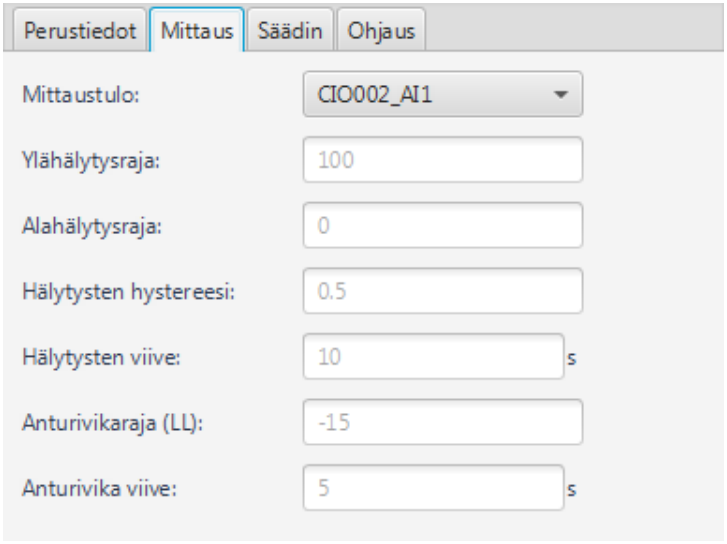
Kuva 14. Lisättävän logiikkaohjelman perustiedot.

Perustiedoissa ohjelmalle annetaan nimi, joka noudattaa liitteen 2 nimeämiskäytäntöä.

cTalk-polussa kuvataan logiikkaohjelma tarkemmin. Polku kuvaa, mihin kyseinen ohjelma vaikuttaa ja mitä sillä tehdään.

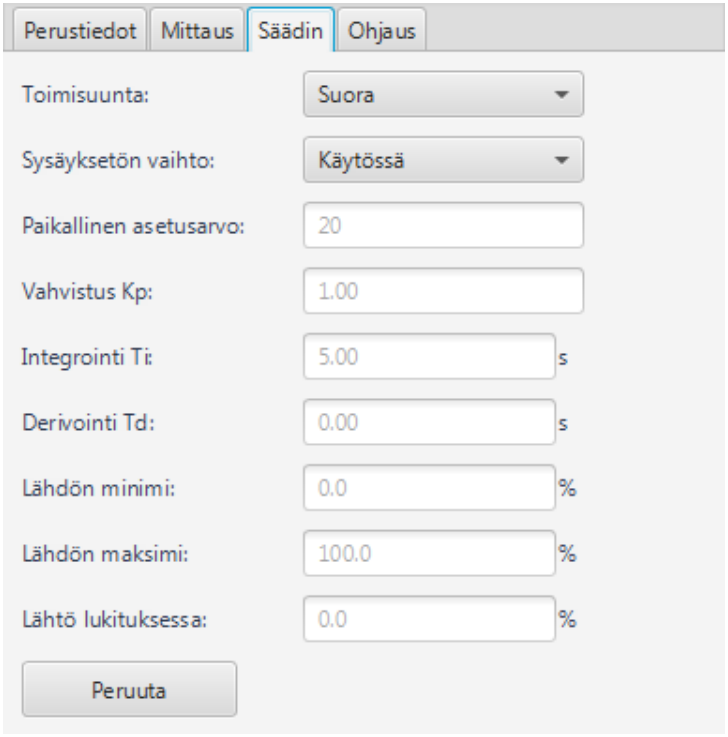
Suoritusvälillä määritetään kuinka usein ohjelmaa suoritetaan. Poikkeaman tarkoituksena on siirtää sovelluksen suoritusta logiikan kierrossa, laajemmissa ohjelmakokonaisuuksissa poikkeutusten porrastus nopeuttaa logiikan suoritusta.

Logiikkaohjelma voidaan poistaa logiikan suorituskierrosta valitsemalla alasvetovalikosta Pysäytetty.



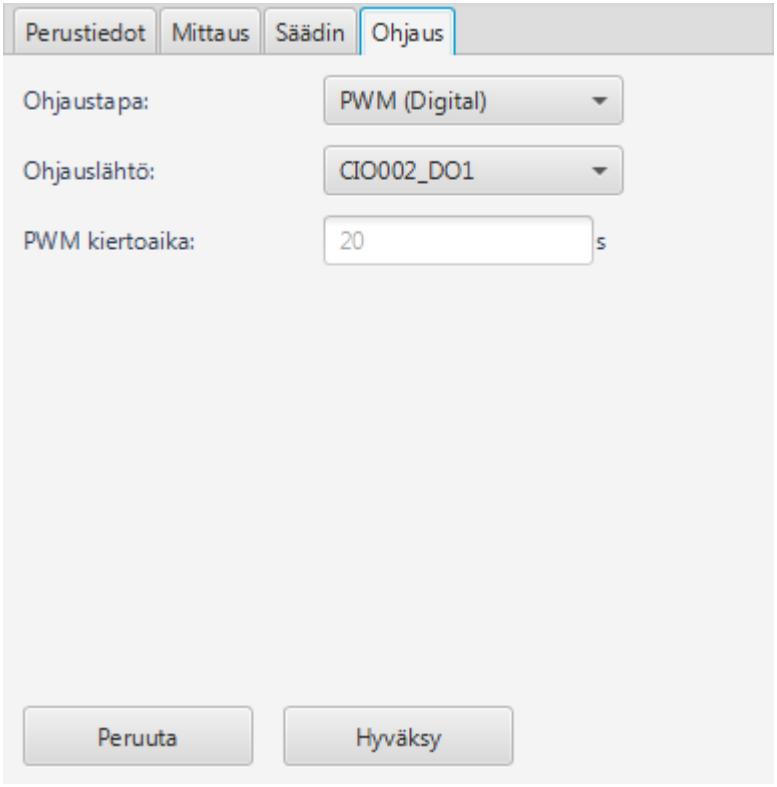
Kuva 15. Lisättävän logiikkaohjelman lisäasetukset 1.

Kuvassa 15 on Mittaus-välilehti, jossa valitaan käytettävä mittaustulo ja asetetaan mittauksen oletusarvot.



Kuva 16. Lisättävän logiikkaohjelman lisäasetukset 2.

Säädin-välilehdellä valitaan säätimen toimisuunta ja toiminta siirryttäessä käsiajosta automaatille. Asetusarvojen oletusarvot syötetään kuvassa 16 esitettyihin kenttiin.



Kuva 17. Lisättävän logiikkaohjelman lisäasetukset 3.

Ohjaus-välilehdellä valitaan säätimen ohjaustapa kuvan 17 mukaisesti. Säätimellä voidaan ohjata digitaalista tai analogista lähtöä. Ohjaustavan valinnasta riippuen Ohjauslähtö-valikkoon haetaan joko digitaalisia tai analogisia muuttujia.

Logiikkaohjelma muodostetaan painamalla Hyväksy-painiketta ja asetteluikkuna sulkeutuu.

Kuva 18. Lisätyn logiikkaohjelman esitys sovelluksessa.

Sovellus muodostaa asettelun perusteella logiikassa suoritettavan logiikkaohjelman, muuttujat ja sovellukset. Vasemmasta valikosta valitaan tarkasteltava ohjelma ja oikeasta valikosta nähdään käytetyt muuttujat. Logiikkaohjelmia voidaan muokata Program-ikkunassa, kuten kuvassa 18.

3.3 Sovelluksen toiminnot

Sovelluksen toiminnot toteutetaan aliohjelmilla, joita kutsutaan ajureiksi. Ajurien käyttö vähentää sovelluksen ohjelmointia ja mahdollistaa sovelluksen joustavan kehityksen.

Ajurit kootaan omaan kansioon, josta sovellus hakee käytettävissä olevat ajurit ja suorittaa niiden sisältämät Java-ohjelmat kutsuttaessa. Ajurit muokkaavat, lisäävät ja poistavat projektin tiedostoja ajuriin ohjelmoidulla tavalla.

3.3.1 Laiteasetukset ja I/O-määritys

Laiteasetusten ajurit sisältävät ohjelmat kansiorakenteiden ja tiedostojen muodostamiseen. Ajurit koostuvat perusajureista ja I/O-ajureista.

Perusajureiden tehtävä on muodostaa logiikan käyttämä kansiorakenne ja luoda perusasetusten tiedostot. Cromi S1000- ja S500 -logiikoille tulee omat perusajurit, sillä niiden kansiorakenne eroaa toisistaan.

I/O-ajureiden tehtävänä on luoda käyttäjän määrityksen mukaiset I/O-tiedostot ja muodostaa tuloja ja lähtöjä käsittelevät logiikkaohjelmat. Jokaiselle I/O-korttityypille tehdään oma ajuri, joka sisältää tiedostojen ja muuttujien muodostuksessa tarvittavat tiedot. Sovellus määrittää automaattisesti I/O-korttien ja kanavien nimet nimeämiskäytännön mukaisesti.

Laiteasetusajureilla korvataan pohjaprojektin lataus ja manuaaliset järjestelmäasetukset.

3.3.2 Ohjelmointi

Ohjelmointiin käytetään ohjelma-ajureita. Ajurit sisältävät ST-kielisen logiikkaohjelman lisäksi muuttujien, sovellusten ja arvojen muodostamiseen tarvittavat tiedot. Ajuri sisältää myös ohjelman poisto ominaisuuden, jolla voidaan poistaa ajurilla muodostettu ohjelma. Ajuri muokkaa tulojen ja lähtöjen logiikkaohjelmia, mutta ei voi poistaa niitä.

Ohjelma-ajurilla korvataan useita välivaiheita ohjelmointityössä. Välivaiheiden määrä riippuu toteutettavan ohjelmakokonaisuuden laajuudesta, yksittäisen logiikkaohjelman osalta ajurin käyttö vähentää seitsemän välivaihetta.

Ohjelma-ajureiden muodostus aloitetaan tekemällä ajuriin sijoitettava ohjelmakokonaisuus logiikkaan. Ajureihin tulevat logiikkaohjelmat testataan logiikassa mahdollisimman kattavasti mahdollisten virheiden poistamiseksi. Olemassa olevia ohjelmia pyritään muokkaamaan mahdollisimman vähän, koska ne ovat toimiviksi todettuja ja muokkaus voisi muuttaa niiden toimintaa.

Ajuriin sijoitettava logiikkaohjelma muunnetaan yleiskäyttöiseksi koodiksi, eli ohjelman nimi ja muuttujien etuliite korvataan ajuriin syötettävällä parametrilla. Käyttöliittymässä parametri annetaan ohjelmaa lisättäessä kenttään Ohjelman nimi / positio. Sovellukset nimetään samoilla nimillä kuin lisättävät ohjelmat. Mikäli sovelluksia tulee useampi, lisätään nimen loppuun sovellusta kuvaava tunnus.

Muuttujat nimetään liitteessä 2 olevan nimeämiskäytännön mukaisesti. Nimen etuliite on sama kuin niitä käyttävän logiikkaohjelman nimi. cTalk-polku kuvaa tarkemmin ohjelman tarkoitusta ja määrittää valvomotasonmuuttujien kansiorakenteen. Ajuri lisää muuttujat globaaleihin, muistaviin ja cTalk-tiedostoihin. Globaalit ja muistavat muuttujat määritellään ST-kielellä ja cTalk-muuttujat määritellään XML-kielellä.

4 Lopputulos

Ohjelmointisovelluksen perustoimintoja suunnitellessa selvisi tarve sovellukselle. Sovellus nopeuttaisi ja yksinkertaistaisi logiikkaohjelmointia. Sovellus voisi keventää yrityksen logiikkaohjelmoijien työtaakkaa, sillä perusohjelmien ohjelmointi sovelluksen avulla olisi helpompaa kuin nyt käytössä olevalla ohjelmointitavalla.

Sovelluksen toteutus Java-kielellä tekee sovelluksesta laitteistoriippumattoman. Java-kieli on myös tehokasta ja monipuolista. Lisäksi internetistä löytyy paljon ohjeita ja esimerkkejä ohjelmointiin.

Sovelluksen toimintojen toteutustavasta saatiin selkeä suunnitelma, joka mahdollistaa sovelluksen joustavan jatkokehityksen. Sovelluksen toiminnot toteutetaan määritetyillä aliohjelmilla joita kutsutaan ajureiksi. Ajureiden päivitys ja lisäys eivät vaadi sovelluksen päivitystä, eli uusia logiikkaohjelmia ja I/O-määrityksiä voidaan lisätä ja muuttaa tarvittaessa. Logiikkaohjelmien muuntaminen sovelluksen käyttöön pyritään tekemään mahdollisimman helpoksi ajurien avulla.

Käyttöliittymästä toteutettiin työn aikana vain pohjakuvat, joissa ei ole toimintoja, mutta joita voidaan käyttää sovelluksen kehityksessä. Kaikki ohjelmointiin liittyvät toiminnot sijoitettiin omalle välilehdelleen, jolloin ohjelmoijan ei tarvitse siirtyä sivujen välillä työskennellessään. Laiteasetusten ja I/O-määritysten tekeminen tapahtuu yhdestä ikkunasta, joka yhdistää toiminnot yhdeksi kokonaisuudeksi käyttäjälle. Alasvetovalikoita pyrittiin käyttämään mahdollisimman paljon, sillä se estää virheellisten arvojen syöttämisen sovellukseen. Projektin hallintaan liittyvät toiminnot, kuten tallennus ja projektin avaus sijoitettiin toiselle välilehdelle.

Ajureista ja osasta käyttöliittymän toimintaa tehtiin proof of concept -demot, joilla voitiin varmistaa, että sovellukseen suunnitellut toiminnot on mahdollista toteuttaa.

5 Yhteenveto

Ohjelmointisovelluksen tarve selvisi suunnittelun myötä ja sovelluksen ulkoasusta ja toiminnoista saatiin riittävän kattava suunnitelma.

Ohjelmointisovelluksen toimintojen toteutustavasta saatiin työn aikana selkeämpi kuva. Toimintojen toteutus ajureilla antaa sovelluksen kehitykseen joustavuutta ja mahdollistaa kehityksen hajautuksen yrityksen eri henkilöille.

Käyttöliittymää suunniteltaessa pyrittiin siitä tekemään mahdollisimman helppokäyttöinen, selkeä ja monipuolinen. Käyttöliittymän pohjakuvien perusteella sovellus on riittävän selkeä ja helppokäyttöinen.

Ohjelmointisovellus toteutetaan ensin demo-vaiheeseen ja sitä testataan ensin yrityksen sisällä. Yrityksen sisäisen testauksen ja kehityksen jälkeen sovellusta testautetaan ulkopuolisilla, joiden käyttäjäpalautteen perusteella sovellusta parannetaan ja kehitetään. Riittävän testauksen ja kehityksen jälkeen sovellus voidaan julkaista levitykseen.

Tulevaisuudessa sovellukseen lisätään logiikkaohjelmien graafinen esitys ja muokkaus. Seuraavat jatkokehityskohteet ovat

- suoritettavan ohjelman online-seuranta (Debugger)
- logiikkaohjelmien kääntäjä, jolla ST-kieliset logiikkaohjelmat käännetään konekielelle
- logiikkasimulaattori, jolla voidaan simuloida logiikan toimintaa.

Lähteet

1 THT Control Oy. Verkkodokumentti.
http://thtcontrol.com. Luettu 27.3.2017.

2 Malmi, Kimmo. 2017. Järjestelmäarkkitehti, THT Control Oy, Tampere.
Keskustelu 28.3.2017.

3 S1000 Technical specification. Verkkodokumentti.
http://thtcontrol.com/s1000/. Luettu 27.3.2017.

4 NetBeans IDE. Verkkodokumentti.
https://netbeans.org/features/index.html. Luettu 27.3.2017.

5 JavaFx Scene Builder. Verkkodokumentti.
http://docs.oracle.com/javase/8/scene-builder-2/get-started-tutorial/overview.htm#JSBGS164. Luettu 27.3.2017.

6 Java (Programming language). Verkkodokumentti.
https://en.wikipedia.org/wiki/Java_(programming_language). Luettu 28.3.2017,

7 JavaFX. Verkkodokumentti.
https://en.wikipedia.org/wiki/JavaFX. Luettu 28.3.2017.

8 Oracle and Sun Microsystems. Verkkodokumentti.
https://www.oracle.com/sun/index.html. Luettu 28.3.2017.

9 Kuutti, Anna. 2015. Cromi S1000 User Manual Liite 1. Käyttöohje.
THT Control Oy.

Cromi S1000 IO-korttien nimet

Liitteessä kuvataan Cromi S1000 -logiikan I/O-korttien nimeämiskäytäntö. [9.]

IO-korttien nimet

1. Laitteen IO -kortit (S1000.DAS)

IO -kortin nimi	Kommentti
CIO001_AI	Analogiset tulot
CIO001_AO	Analogiset lähdöt
CIO001_DI	Digitaaliset tulot
CIO001_DO	Digitaaliset lähdöt
CCO0	Sarjaportti

2. Laajennus IO -kortit (IOX tai DIO)

IO -kortin nimi	Kommentti
CIO002_AI	Analogiset tulot
CIO002_AO	Analogiset lähdöt
CIO002_DI	Digitaaliset tulot
CIO002 _DO	Digitaaliset lähdöt

Huom! Seuraavat IO -kortit ovat CIO003_XX, CIO004_XX .... CIOnnn_XX

IO -kortin nimi	Kommentti
CCO1-9	Sarjaportit
CMO001-9	Modeemi
CEM001-9	eMeter
MSR001-9	Modbus serverit
MHR001-9	Modbus holding rekisterit
MIR001-9	Modbus tulorekisterit
MCO001-9	Modbus coils
MDI001-9	Modbus digitaaliset

IO -kanavien nimet

Ensimmäinen osio nimestä periytyy IO -kortin nimestä, johon liitetään kanavalle annettu juokseva numero 001-999.

IO -kanavan nimi	Kommentti
CIO001_AI1 - nn	Analogiset tulot
CIO001_AO1 - nn	Analogiset lähdöt
CIO001_DI1 - nn	Digitaaliset tulot
ICIO001 _DO1 – nn	Digitaaliset lähdöt
MIR001_AI1 - nn	Modbus analogiset tulot
MHR001_AO1 - nn	Modbus analogiset lähdöt
MDI001_DI1 - nn	Modbus digitaaliset tulot
MCO001_DO1 - nn	Modbus digitaaliset lähdöt

Cromi muuttujien nimet

Liitteessä kuvataan Cromissa käytettävien piirien ja muuttujien nimeämiskäytäntö. [9.]

THT Control Oy
Eteläpuisto 2 C
33200 Tampere
thtcontrol.com

1. Piiritunnus (AAA)

Mittausmuuttujan nimen ensimmäinen kirjain määräytyy prosessisuureen mukaan (P, T, F..). Kirjainyhdistelmä IA (= indicator/alarm) liitetään ensimmäisen kirjaimen perään esim. PIA.

Prosessisuureet	Kommentti
D	Tiheys
E	Sähkösuureet
F	Virtaus
G	Pituus, asento
H	Käsiohjaus (ei käytetä koskaan)
K	Aika tai aikaohjelma
L	Pinta
M	Kosteus
N	Vapaasti valittavissa
O	Vapaasti valittavissa
P	Paine
Q	Laatu esim. happi, pH
R	Säteily
S	Nopeus, taajuus
T	Lämpötila
U	Monimuuttuja
V	Viskositeetti
W	Paino, voima
X	Määrittelemättömät
Y	Vapaasti valittavissa

Säädinmuuttujan nimen ensimmäinen kirjain määräytyy myös prosessisuureen mukaan (P, T, F..), mutta kirjainyhdistelmä IC (= indicator controller) liitetään ensimmäisen kirjaimen perään esim. PIC.

Moottorimuuttujille käytetään oletuksena kirjainyhdistelmää MTR kaikille moottorityypeille. Esim. PPU tai PUH voidaan käyttää, jos asiakkaalla on sellaiset jo käytössä.

Kytkinmuuttujille käytetään oletuksena kirjainta S esim. PSA.

Venttiilimuuttujille käytetään oletuksena kirjainyhdistelmää VAL kaikille venttiilityypeille.

2. Piirinumero (001-999)

Kaikkiin muuttujiin lisätään piirinnumero, joka on kolminumeroinen 001-999 tai jos asiakkaalla on olemassa oleva numerointi sitä voi hyödyntää esim. PIC123.

3. Tyyppi (_Aa)

Muuttujan loppuosan muodostaa lyhenne muuttujan tarkoituksesta. Nämä lyhenteet ovat kaikille yhteisiä.

Lyhenne	Määritelmä	Tyyppi
_Pv	mittaus	REAL
_Filter	suodatus	REAL
_Delay	viive	
_Fail	virhe	BOOL

_LLa	ala-alaraja hälytys	BOOL
_LLset	ala-alaraja asetus	REAL
_LLhyst	ala-alaraja hystereesi	REAL
_La	alaraja hälytys	BOOL
_Lset	alaraja asetus	REAL
_Lhyst	alaraja hystereesi	REAL
_Ha	yläraja hälytys	BOOL
_Hset	yläraja asetus	REAL
_Hhyst	yläraja hystereesi	REAL
_HHa	ylä-yläraja hälytys	BOOL
_HHset	ylä-yläraja asetus	REAL
_Hyst	hysteresi	REAL
_Set1	oletus arvo	REAL
_Am	auto man	BOOL
_Lr	local/remote	BOOL
_Ctrl	ohjaus	BOOL
_Run	käyntitieto	BOOL
_Alm	hälytys	BOOL
_Fw	eteen esim. RunFw tai CtrlFw	
_Rw	taakse	
_Ind	indikointi	
_Status	tila	
_Lsp	paikallisasetus	REAL
_Rsp	etä-asetusarvo	REAL
_P	vahvistus	REAL
_I	integrointi	
_D	derivointi	
_Out	lähtö	REAL
_Fault	laitevika	BOOL
_Warn	varoitus	BOOL
_Pwm	pulssileveys	
_Sw	turvakytkin	BOOL
_Intl	lukitus	BOOL
_W	sana esim. CtrlW	INT
_Spd	nopeus	
_Ref	referenssi/ohje esim. SpdRef	INT
_Ack	kuittaus	BOOL
_Calc	laskenta	REAL
_Reset	resetointi/nollaus	BOOL
_Bias	ennakointi	REAL
_Act	todellinen esim. ActSpd	