Tampereen ammattikorkeakoulu

Tehokasta rakennusautomaation ohjelmointia

Janne Mäkelä

OPINNÄYTETYÖ
Joulukuu 2020

Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Sähköinen talotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Sähköinen talotekniikka

MÄKELÄ, JANNE:
Tehokasta rakennusautomaation ohjelmointia

Opinnäytetyö 47 sivua, joista liitteitä 0 sivua
Joulukuu 2020

Rakennusautomaatiourakoinnissa kilpailu voi olla kovaa, ja kilpailussa ei kukaan voita, jos hintoja poljetaan. Tässä työssä haetaan keinoja tehostaa automaatioprojektinhoitajan ja -ohjelmoijan ajankäyttöä automatisoimalla ohjelmointivaiheita ja luomalla kiinteitä rutiineja ohjelmointiin. Säästetty aika voidaan käyttää projektien määrän tai projektinhoidon laadun lisäämiseen. Sekä laatu että ajansäästö tuovat projekteihin toivottua kilpailuetua.

Menetelmien ja ohjelmien kehitys tuli tarpeelliseksi, kun juuri perustettu rakennusautomaatiourakointiin keskittyvä Teratek Automaatio Oy alkoi urakoida uudella automaatiojärjestelmällä, jolle ei ollut vielä olemassa laajoja ohjelmakirjastoja. Tavoitteena oli hyödyntää yrityksessä olemassa olevaa osaamista muista automaatiojärjestelmistä ja välttää niiden tiedossa olevat hitaat työvaiheet automaatiourakan ohjelmoinnissa.

Automaatiourakassa ohjelmoitavana ovat prosessiohjelmat ja käyttöliittymägrafiikka. Näistä varsinkin käyttöliittymägrafiikan tekoon liittyy paljon käsin tehtävää työtä, joka vie runsaasti aikaa. Työhön liittyvä ohjelmointityö soveltuu sellaisenaan vain Actiweb-automaatiojärjestelmälle, eikä vastaavaa toteutusta voi välttämättä muilla järjestelmillä edes tehdä. Tehdyt ohjelmat jäävät Teratek Automaation Oy:n käyttöön ja opinnäytetyö sisältää vain pieniä näytteitä valmiista ohjelmasta.

Ensimmäisten kokeiluiden perusteella menetelmät helpottavat ohjelmoijan työtä ja nopeuttavat yleisimpien rakennusautomaatioprojektien ohjelmointia merkittävästi. Jatkokehitettynä menetelmistä saadaan tehokas työkalu rakennusautomaatiourakointiin.

Asiasanat: rakennusautomaatio, tehostaminen, ohjelmointi

ABSTRACT

Tampereen Ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Building Services Engineering, Electrical Systems
Electrical Building Systems Engineering

MÄKELÄ JANNE:
Efficient Building Automation programming

Bachelor's thesis 47 pages, appendices 0 pages
December 2020

Competition in building automation contracting can be fierce and prices may sometimes plummet in bidding situations. This thesis is about finding methods to increase the efficiency of the working hours of the project manager/programmer by automating various programming-related tasks. Saved time may then be used for more projects or to increase the overall quality of project management. Less time used on projects and increased quality also gives an edge over the competition.

This work became necessary when the newly founded building automation contractor Teratek Automaatio Oy needed new methods and program libraries to start using a new building automation system, which did not yet have extensive program libraries. The goal was to make use of the experience on other building automation systems within the company to avoid known, time consuming work stages in building automation programming.

Programming work in building automation projects includes designing the graphical user interface and programming the actual program that controls the automation processes. Designing the graphical user interface often involves a lot of time-consuming manual labor.

Programs made while working on this thesis are only applicable to the Actiweb building automation system. Similar methods may not even be possible on other systems. This thesis contains small samples of the source codes used and the work is entirely the property of Teratek Automaatio Oy.

According to the first test cases, the methods reduce the amount of programming work involved in most typical building automation projects. Some more development and these methods will be a powerful tool for building automation contracting.

Key words: building automation, efficiency, programming

SISÄLLYS

LYHENTEET JA TERMIT

AI	Analog Input. Analoginen I/O-piste, eli jonkin suureen mittauspiste.
BACnet	Building Automation and Control network. Automaatiojärjestelmän tiedonsiirtoprotokolla.
css	Cascading Style Sheets. Web-sivulla käytettävä sivutyyliohje.
csv	Comma Separated Values. Tiedostomuoto, jossa taulukossa olevat tiedot erotellaan välimerkillä, esimerkiksi pilkku tai sarkain.
DDC	Direct Digital Control, Suora digitaalinen säätö.
FBD	Function Block Diagram, toimilohkokaavio. Logiikkaohjelmointikieli, jossa ohjelman toimilohkot yhdistetään toisiinsa piirtämällä niiden loogiset yhteydet viivoilla.
HTML5	Hyper Text Markup Language, versio 5. Verkkosivujen tekemiseen käytetty merkintäkieli.
I/O-kanava	I/O-moduulin liityntäkanava, johon laitteen johdot kytketään.
I/O-moduuli	Input/Output-moduuli. Säätimen fyysiset liitännät mittauksille, ohjauksille ja säädöille.
I/O-piste	Fyysinen liityntäpiste automaatiojärjestelmään. Esimerkiksi mittaus.
IEC 61131-3	PLC-säätimien ohjelmoinnin standardi.
JavaScript	Yleisesti web-ohjelmoinnissa käytetty komentokokoelmakieli.
json	JavaScript Object Notation. tapa tallentaa ja vaihtaa tietoa palvelimen ja web-sivun välillä.
Ladder	Logiikkaohjelmointikieli, jossa ohjelma muistuttaa tikapuumaista piirikaaviota
Linux	Tietokoneen käyttöjärjestelmäperhe.
Lua	Ohjelmointikieli.
LVI(S)	Lämpö, vesi, ilma (ja sähkö).
Modbus	Yleisesti rakennusautomaatiojärjestelmissä käytössä oleva sarjaliikenneprotokolla.

ohjelmarutiini	Tiettyä toimintoa suorittava ohjelman osa.
PLC	Programmable Logic Controller, ohjelmoitava logiikkasäädin.
pneumatiikka	Paineilmalla toteutettu logiikkaohjausjärjestelmä.
RS-485	Sarjaliikenneväylä.
slcengine	Actiweb automaatiojärjestelmän pääohjelma, joka yhdistää tietokannan ja prosessiohjelman toisiinsa.
ST-kieli	Structured text. Logiikkaohjelmointikieli, jossa ohjelma on nimen mukaisesti tekstiä.
symboli	Rakennusautomaatiojärjestelmän graafisen käyttöliittymän osa, joita yhdistelmällä voidaan luoda kokonaisuuksia.
verkkotopologia	Toisiinsa liitettyjen laitteiden kaapeloinnin fyysinen rakenne.
väylä	Automaatiojärjestelmän laitteiden toisiinsa liittävä kaksisuuntainen tiedonsiirtoratkaisu.
XML	eXtensible Markup Language. Tiedostomuoto, jossa data voidaan esittää helposti luettavassa muodossa sekä ihmiselle, että tietokoneohjelmalle.

1 JOHDANTO

Rakennusautomaatiouran työvaiheet voidaan jakaa kolmeen osaan: projektinhoitoon, ohjelmointityöhön ja asennustyöhön. Suuri osa rakennusautomaatiota urakoivista yrityksistä ostaa asennustyön aliurakkana ja keskittyy itse ohjelmointiin ja projektinhoitoon. Tampereen alueella automaatiourakoinnissa asennustyöt päätyvät usein yhden suuremman yrityksen aliurakoitavaksi varsinaisesta rakennusautomaatiourakoitsijasta riippumatta ja näin hintakilpailu asennustöiden osalta muuttuukin hyvin hankalaksi. Kilpailuetu täytyy siis hakea ohjelmointityön kustannuksista ja työn yleisestä laadusta.

Tämän opinnäytetyön tavoitteena on kehittää ohjelmointimenetelmät ja ohjelmakirjastot automaatiourakan ohjelmointityön ajankäytön vähentämiseksi. Ohjelmakirjaston avulla voidaan automatisoida usein toistuvia ohjelmointivaiheita, ja samanaikaisesti parantaa työn laatua sekä ohjelmointivaiheessa, että käyttöönottovaiheessa. Ohjelmoijan vastuulla on usein myös urakan projektinhoito, joten säästetty ohjelmointiaika saadaan siirrettyä projektinhoitoon, jolloin senkin laatu voi parantua. Ohjelmointimenetelmien omaksumisen kannalta on tärkeää, että menetelmät ovat yksinkertaisia ja perustuvat helposti toistettaviin rutiineihin.

Opinnäytetyössä kehitettävät menetelmät ja ohjelmat tulevat Teratek Automaatio Oy:n käyttöön. Teratek Automaatio Oy:n on perustettu vuonna 2019 ja sen päätoimiala on rakennusautomaatiourakointi. Automaatiojärjestelmä, jolla Teratek Automaatio urakoi, on melko uusi, eikä sille vielä ole olemassa laajaa ohjelmakirjastoa. Tämän vuoksi on luontevaa tehdä ohjelmakirjastot alusta alkaen hyvin suunnitellun mallin mukaisesti.

2 RAKENNUSAUTOMAATIO

Rakennusautomaatiota on jossain muodossa urakoitu 1950-luvulta asti. Aluksi rakennusautomaatiota on toteutettu analogisesti termostaattiohjauksilla, myöhemmin pneumaattisilla ja lopulta ohjelmoitavilla DDC-pohjaisilla, eli suoraan digitaaliseen säätöön perustuvilla järjestelmillä, joita nykyisetkin järjestelmät ovat (Rakennusautomaatiojärjestelmät 2018, 13).

Suomessa on käytössä useita eri rakennusautomaatiojärjestelmiä, joista kotimaisia ovat ainakin Fidelix, Ouman ja Actiweb. Ulkomaisia järjestelmiä edustavat esimerkiksi Siemens, Schneider Electric, Trend, DEOS ja Sauter.

Rakennusautomaation tehtävä rakennuksissa on ylläpitää rakennuksessa toivottuja olosuhteita erilaisten mitta- ja säätölaitteiden avulla. Rakennusautomaatiojärjestelmän avulla ylläpitotoiminnot pystytään toteuttamaan energiatehokkaasti, ja rakennuksen energiatehokkuutta voidaan myös seurata (Rakennusautomaatiojärjestelmät 2018, 21). Koska rakennusautomaation rooli LVIS-järjestelmien toiminnassa on niin suuri, voi huonosti toteutettu automaatioprojekti rampauttaa koko rakennuksen järjestelmät (Rakennusautomaatiojärjestelmät 2018, 193).

2.1 Rakennusautomaatioprojekti

Urakoinnissa rakennusautomaatioprojektiin yleisimmin sisältyy rakennuksen ilmanvaihdon, lämmityksen ja valaistuksen ohjaus. Rakennusautomaatiourakka on usein alistettuna putki- tai ilmanvaihtourakkaan, jolloin rakennusautomaatiourakoitsija toimii aliurakoitsijana kyseisessä urakassa. Suunnilleen yhtä usein rakennusautomaatiourakka on kokonaan oma urakkansa, jolloin sopimussuhde on suoraan rakennuttajaan tai pääurakoitsijaan. Käytännössä suurin ero on siinä, kuka laskut maksaa, ja kenelle urakasta raportoidaan (Rakennusautomaatiojärjestelmät 2018, 190).

Rakennusautomaatioprojektissa varsinainen työ alkaa siitä, että projektiin nimetty projektinhoitaja käy läpi urakan suunnitelma-asiakirjat, sekä muut tarjoukseen sisältyneet materiaalit. Projektinhoitajalle hankitaan käyttöoikeus mahdolliseen projektipankkiin, johon tallennetaan kaikki projektin suunnitelmat ja muut

asiakirjat, jotta projektinhoitajan on helpompi pysyä ajan tasalla muun muassa suunnitelmamuutoksista. Projektille laaditaan oma aikataulu, jonka mukaan projektinhoitaja voi tahdistaa oman työnsä vaiheet. Aikataulusta selviää tärkeinä vaiheina esimerkiksi vastaanottoon liittyvät vaiheet ja erilaiset välitavoitteet. Joskus esimerkiksi käyttöliittymägrafiikka täytyy hyväksyttää tilaajalla tiettyyn päivämäärään mennessä ja ohjelmointityön valmistumisella saattaa olla aikataulussa takaraja (Rakennusautomaatiojärjestelmät 2018, 199).

Uudisrakennuksissa rakennusautomaatiourakka työmaalla alkaa yleensä vasta kokonaisurakan loppuvaiheella, ja loppuu viimeisten joukossa. Tämä johtuu siitä, että automaatiourakka on riippuvainen lähes kaikista muista urakoista, varsinkin lämpö-, vesi-, ilma- ja sähköurakoista, joiden täytyy olla lähes valmiina ennen kuin automaatiourakka voi valmistua (Rakennusautomaatiojärjestelmät 2018, 272). Joskus automaatiojärjestelmään halutaan liittää vaikkapa äänentoistojärjestelmää tai kulunvalvontaa, jolloin riippuvuudet muihin urakoitsijoihin vain lisääntyvät. Isoissakin automaatioprojekteissa suuri osa työstä tapahtuu työmaan ulkopuolella, eikä useimmat työmaat vaadi jatkuvaa läsnäoloa. Tällaista työtä ovat toteutuksen suunnittelu, ohjelmointityö sekä urakan dokumentointi. Useimmissa tapauksissa yksi projektinhoitaja hoitaa nämä vaiheet. Automaatiourakassa urakan projektinhoitaja on usein myös ohjelmoija. Kaikki ohjelmoijan tehtävät eivät kuitenkaan ole projektinhoitoa, vaikka niissä onkin paljon toisiinsa limittyviä vaiheita. Projektinhoitajalla muita tehtäviä ovat esimerkiksi työmaakokouksissa käyminen ja työmaan resurssien hoitaminen.

2.2 Ohjelmoijan tehtävät

Rakennusautomaatioprojektissa ohjelmoijan tehtävä on saada järjestelmä toimimaan suunnitellulla tavalla. Ohjelmointityötä helpottaa paljon, jos ohjelmoija on itse tehnyt myös urakan kytkentä- ja kaapelointisuunnitelmat, sillä I/O-pisteiden liitännät ovat oleellista tietoa ohjelman suunnittelussa ja toteutuksessa. Erillinen projektinhoitaja, joka ei itse ohjelmoi, ei välttämättä osaisi ottaa huomioon erilaisia ohjelmointia helpottavia asioita kytkentäsuunnitelmia tehdessään. Ohjelmoijan täytyy myös käydä suunnitelmat perusteellisesti läpi, jotta kaikki suunnitellut

toiminnot tulevat myös ohjelmoitua. Tämän vuoksi ohjelmoijalla on yleensä erittäin hyvä käsitys koko projektista. Siksi on luontevaa, että urakan toteutussuunnittelun tekeekin juuri ohjelmoija.

2.3 Toteutussuunnittelu

Urakan toteutuksessa ensimmäisiä toteutussuunnittelun tehtäviä on suunnitella kaapelointiluettelo (Kuva 1) sähköurakoitsijaa varten (Rakennusautomaatiojärjestelmät 2018, 273), sekä valita lämmönjaon säätöventtiilit kaukolämpövaihtimen valmistajalle. Venttiilit on usein suunnitelmiin valittuna valmiiksi, tai niistä on annettu vähintään mitoitustiedot, joiden avulla venttiilit voidaan valita. Kaukolämpöventtiilit täytyy hyväksyttää kaukolämpölaitoksella ennen tilausta. (Rakennusautomaatiojärjestelmät 2018, 65)

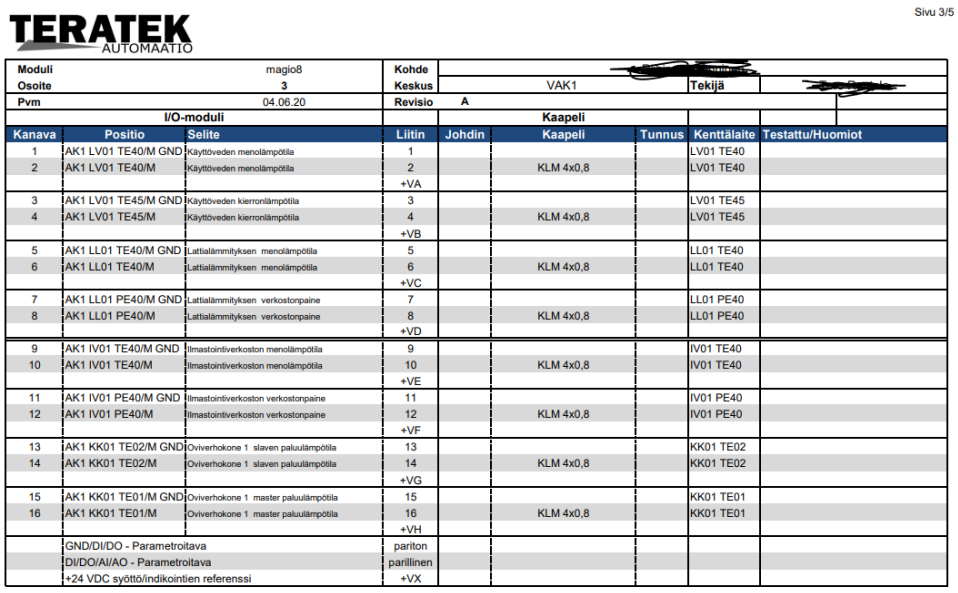
Kohde	[redacted]		Pvm.	14.10.2020
Keskus	VAK1		Tekijä	[redacted]
Tunnus		Selite	Kaapeli	Toimilaite
IV01 PE40		Ilmastointiverkoston verkostonpaine	KLM 4x0,8	
IV01 TE40		Ilmastointiverkoston menolämpötila	KLM 4x0,8	
IV01 TV40		Ilmastointiverkoston säätöventtiili	KLM 4x0,8	
KK01 1		Oviverhokone 1 master	Nomak 4x2x0,5+0,5	
KK01 2		Oviverhokone 1 slave ohjaus	Nomak 4x2x0,5+0,5	
KK01 TE01		Oviverhokone 1 master paluulämpötila	KLM 4x0,8	
KK01 TE02		Oviverhokone 1 slaven paluulämpötila	KLM 4x0,8	
KK02		Oviverhokone 2	Nomak 4x2x0,5+0,5	
KK02 TE01		Oviverhokone 2 paluulämpötila	KLM 4x0,8	
KK03		Oviverhokone 3	Nomak 4x2x0,5+0,5	
KK03 TE01		Oviverhokone 3 paluulämpötila	KLM 4x0,8	
KL01 LM01		Kaukolämmön energiamittari	KLM 4x0,8	
KV01 VM01		Kylmävesimittari	KLM 4x0,8	
LL01 PE40		Lattialämmityksen verkostonpaine	KLM 4x0,8	
LL01 TE40		Lattialämmityksen menolämpötila	KLM 4x0,8	
LL01 TV40		Lattialämmityksen pumppu	KLM 4x0,8	
LV01 PU45		Käyttöveden pumppu	Nomak 8x2x0,5+0,5	
LV01 TE40		Käyttöveden menolämpötila	KLM 4x0,8	
LV01 TE45		Käyttöveden kierronlämpötila	KLM 4x0,8	
LV01 TV40		Käyttöveden säätöventtiili	KLM 4x0,8	
PK01		PK01 radon	Nomak 4x2x0,5+0,5	
PK02		PK02 LJH	Nomak 4x2x0,5+0,5	
PK02 TE10		LJH huonelämpötila	KLM 4x0,8	
TE00		Ulkolämpötila	KLM 4x0,8	
PK/JK		Varaus erillistilatiedoille	NOMAK ??	
PK/JK		Varaus erillisohjauksille	MMO ??	
TK01		Ilmanvaihtokone "modbusväylä"	Nomak 4x2x0,5+0,5	

KUVA 1. Esimerkki kaapeliluettelosta

Kaapeliluetteloa ei kuitenkaan aina kannata tehdä ensimmäisenä, sillä useissa järjestelmissä ohjelmointi alkaa kytkentäsuunnitelmista, joista kaapeliluettelo saadaan muodostettua. Joillakin järjestelmillä kytkentä- ja kaapeliluettelo voidaan

luoda automaattisesti, kun ohjelmoinnissa on ensin määritetty kaikki fyysisesti kytkettävät pisteet. Tällaisessa tapauksessa ohjelmointi voi olla jo todella pitkällä ennen, kuin kaapeliluetteloon asti päästään. Esimerkiksi Sauter-järjestelmän ohjelmointityökalu on tällainen.

Kytkentäluettelot (Kuva 2) suunnitellaan käymällä läpi automaatiosuunnitelmista kaikki kaapelia tarvitsevat laitteet. Yleensä jokaiselle automaatioon liitettävälle laitteelle tarvitaan oma kaapeli laitteelta alakeskukseen (Rakennusautomaatiojärjestelmät 2018, 265), mutta esimerkiksi väyläpohjaiset laitteet voidaan kaapeloida yhdellä kaapelilla, joka kiertää laitteelta laitteelle. Väylälaitteissa tulee kuitenkin muistaa huomioida syöttöjännitteen alenema, jos laite saa sähkönsä alakeskukselta. Tämä rajoittaa kaapelin pituutta, mikä taas tarkoittaa väylän jakamista useampaan kaapeliin. Kaapelin valintaan vaikuttaa tarvittavien johtimien määrä ja häiriösuojauksen tarve. Lämpötila-anturit ovat yleensä passiivisia komponentteja ja tarvitsevat vain kaksi johdinta, ja lähetintyyppiset laitteet tarvitsevat sähkönsyötön ja jokaista mittausviestiä varten oman johdon. Joskus anturit liitetään väylällä, jolloin tarvitaan kaksi johdinta syöttöä varten ja väylälle kaksi. Väylälaitteissa täytyy kaapelin olla kierretyllä parilla, jotta vältytään häiriöiltä (Kiinteistöjen tiedonsiirtoväylät 2017, 79).



TERATEK AUTOMAATIO

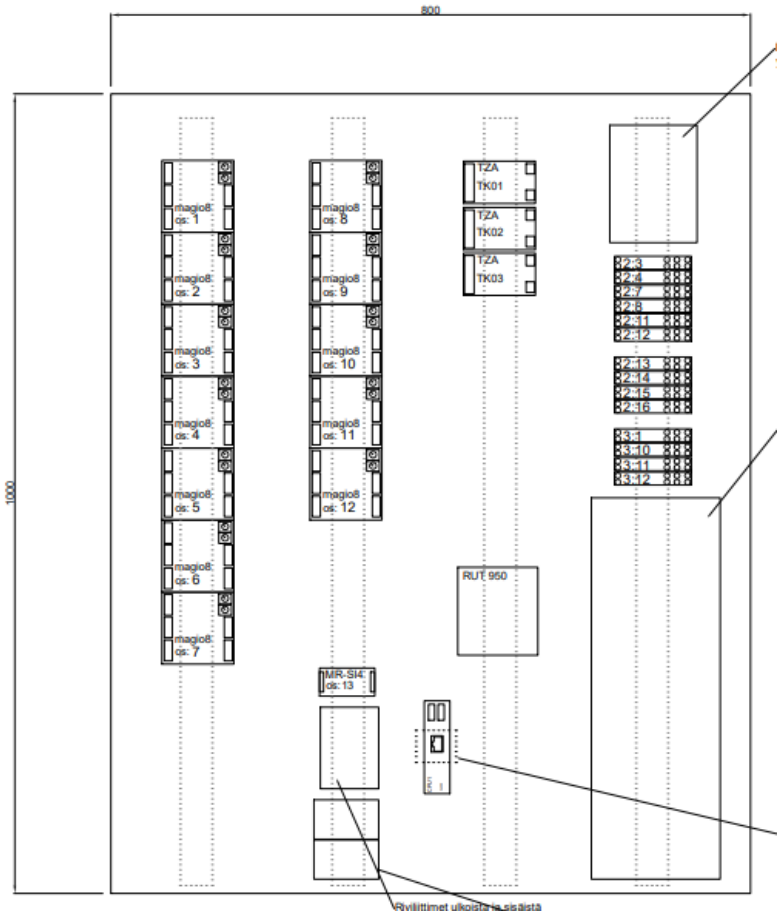
Sivu 3/5

Moduli		magio8	Kohde					
Osoite		3	Keskus	VAK1			Tekijä	
Pvm		04.06.20	Revisio	A				
I/O-moduli				Kaapeli				
Kanava	Positio	Selite	Liitin	Johdin	Kaapeli	Tunnus	Kenttälaite	Testattu/Huomiot
1	AK1 LV01 TE40/M GND	Käyttöveden menolämpötila	1				LV01 TE40	
2	AK1 LV01 TE40/M	Käyttöveden menolämpötila	2		KLM 4x0,8		LV01 TE40	
			+VA					
3	AK1 LV01 TE45/M GND	Käyttöveden kierronlämpötila	3				LV01 TE45	
4	AK1 LV01 TE45/M	Käyttöveden kierronlämpötila	4		KLM 4x0,8		LV01 TE45	
			+VB					
5	AK1 LL01 TE40/M GND	Lattialämmityksen menolämpötila	5				LL01 TE40	
6	AK1 LL01 TE40/M	Lattialämmityksen menolämpötila	6		KLM 4x0,8		LL01 TE40	
			+VC					
7	AK1 LL01 PE40/M GND	Lattialämmityksen verkostonpaine	7				LL01 PE40	
8	AK1 LL01 PE40/M	Lattialämmityksen verkostonpaine	8		KLM 4x0,8		LL01 PE40	
			+VD					
9	AK1 IV01 TE40/M GND	Ilmastointiverkoston menolämpötila	9				IV01 TE40	
10	AK1 IV01 TE40/M	Ilmastointiverkoston menolämpötila	10		KLM 4x0,8		IV01 TE40	
			+VE					
11	AK1 IV01 PE40/M GND	Ilmastointiverkoston verkostonpaine	11				IV01 PE40	
12	AK1 IV01 PE40/M	Ilmastointiverkoston verkostonpaine	12		KLM 4x0,8		IV01 PE40	
			+VF					
13	AK1 KK01 TE02/M GND	Oviverhokone 1 slaven paluulämpötila	13				KK01 TE02	
14	AK1 KK01 TE02/M	Oviverhokone 1 slaven paluulämpötila	14		KLM 4x0,8		KK01 TE02	
			+VG					
15	AK1 KK01 TE01/M GND	Oviverhokone 1 master paluulämpötila	15				KK01 TE01	
16	AK1 KK01 TE01/M	Oviverhokone 1 master paluulämpötila	16		KLM 4x0,8		KK01 TE01	
			+VH					
	GND/DI/DO - Parametroitava		pariton					
	DI/DO/AI/AO - Parametroitava		parillinen					
	+24 VDC syöttö/indikointien referenssi		+VX					

KUVA 2. Esimerkki kytkentäluettelosta

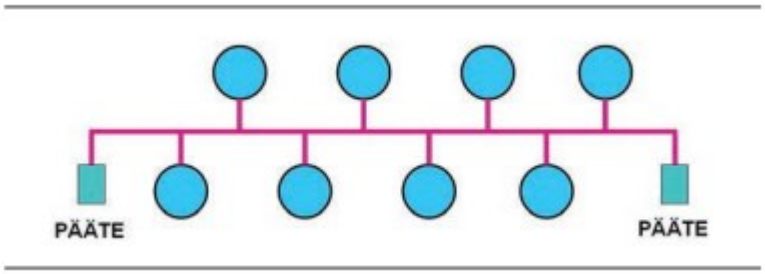
Kytkentäluetteloa tehdessä tarkentuu myös tarvittavien I/O-moduulien lukumäärä. Usein urakoissa vaaditaan esimerkiksi 15 %:n laajennusvara, minkä seurauksena alakeskukseen voi tulla myös kokonaan, tai lähes tyhjiä moduuleita. On myös olemassa niin kutsuttuja universaalimoduuleita, joissa kanavaan voi kytkeä eri tyypin I/O-pisteitä. Tällaisia laitteita käyttämällä voidaan minimoida turhiksi jäävien I/O-kanavien määrä.

Kun kytkentäluettelot on tehty, suunnitellaan niiden pohjalta alakeskuskaappiin I/O-moduulien sijoittelu (Kuva 3). Pienemmissä keskuksissa ei aina tarvitse piirtää erillistä sijoittelukuvaa, sillä keskusvalmistajilla on yleensä hyvä käsitys automaatiokaappien rakenteesta. Suuremmissa keskuskaapeissa sijoituskuva kannattaa piirtää, jotta kytkettävien laitteiden kaapeleiden sijoittelu tulee kaapissa loogiseksi. Hyvin suunnitellussa alakeskuskaapissa voidaan esimerkiksi sähkökeskukselta tulevat tilatietokaapelit kytkeä helposti vierekkäisille kanaville, eikä kaapelia tarvitse haaroittaa eri puolille kaappia.



KUVA 3. Yksinkertainen alakeskuksen laitesijoittelukuva keskusvalmistajalle

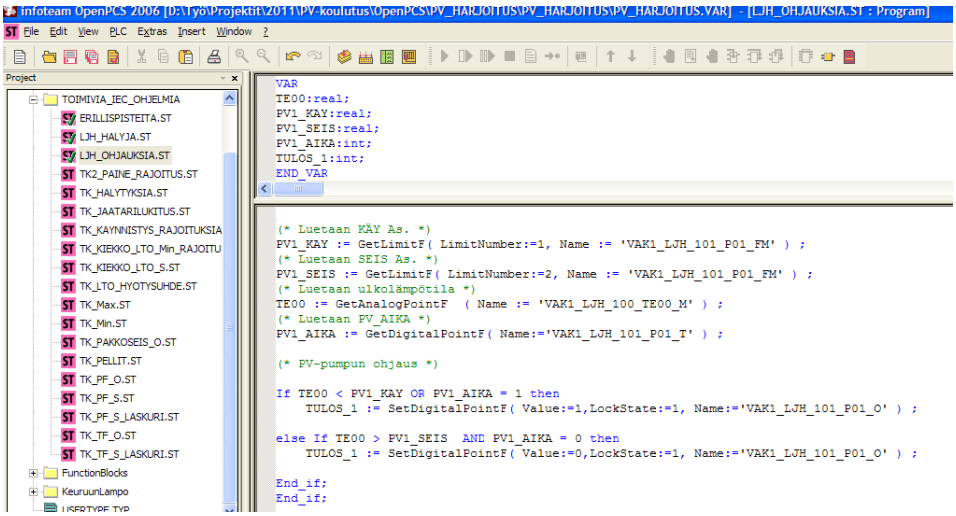
Mikäli urakassa on paljon väylälaitteita esimerkiksi huonesäätöjä varten, on usein myös tarpeen suunnitella sähköurakoitsijan tueksi väylän kaapelointireitit. Kun väylän kaapelointireitin on itse suunnitellut, niin voi varmistua siitä, että väylälaitteet saa ketjutettua oikein käytetyn väylän edellyttämällä tavalla. Yleisimmissä rakennusautomaatioväylissä, eli Modbus ja BACnet, käytetään RS-485 pohjaista sarjaväylää. Suosituksena RS-485 pohjaisessa väylässä on, että laitteet kaapeloidaan laitteelta laitteelle katkeamattomana ketjuna (Kuva 4), niin kutsuttuna väylätopologiana, englanniksi *daisy-chain* tai *bus topology* (The RS-485 Design Guide 2016, 1). Verkon topologialla tarkoitetaan väylän rakennetta. Käytännössä väylä toimii vapaammassakin topologiassa, mutta silloin väyläkaapelin kokonaispituus ei välttämättä voi olla yhtä suuri ja väylä on alttiimpi heijastumisille, jotka aiheuttavat virheitä väylän sanomiin. (Kiinteistöjen tiedonsiirtoväylät 2017, 79). Väärin kaapeloitua väylää on käytännössä mahdotonta saada toimimaan hyvin, jos ollenkaan.



KUVA 4. Väylätopologian rakenne (Kiinteistöjen tiedonsiirtoväylät)

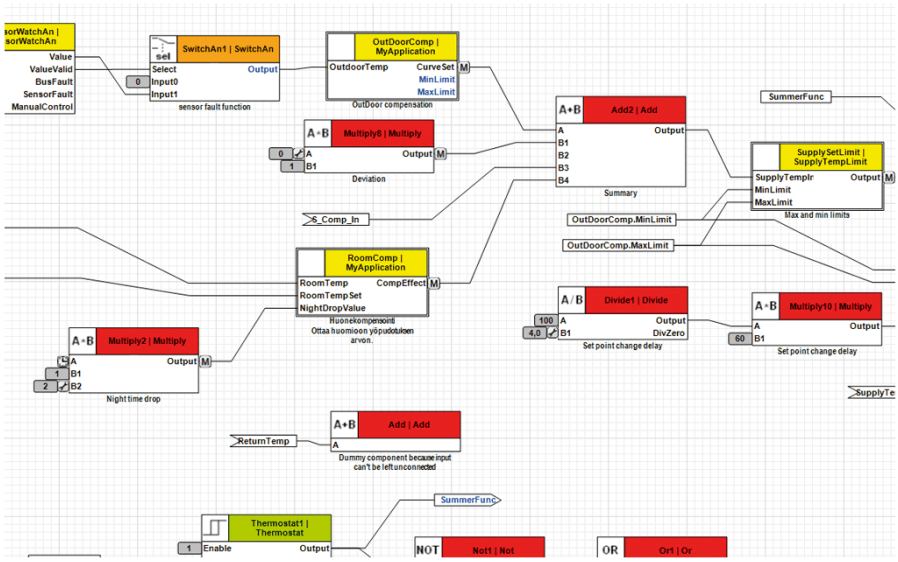
2.4 Ohjelmointityö

Ohjelmointityö on oleellinen, ja usein myös vaikein vaihe automaatiourakan toteutuksessa Ilman ohjelmointia ei ole myöskään automaatiota. Automaatiourakassa ohjelmoinnilla tarkoitetaan kaikkea sitä työtä, jota järjestelmässä tarvitaan, jotta säädöt, hälytykset ja muut ohjelmalliset toiminnot saadaan toimimaan (Rakennusautomaatiojärjestelmät 2018, 274).



KUVA 5. Tekstipohjaista ST-kielellä ohjelmointia

Itse ohjelmointityö on hyvin merkkiriippuvaista. Jotkin järjestelmät ohjelmoidaan tekstipohjaisilla työkaluilla (Kuva 5) ja toiset esimerkiksi Ladder- tai toimilohkokaaviolla (ST 711.15 2018, 5). Kaikissa tavoissa on omat hyvät ja huonot puolensa, mutta yleensä valinnan tekee automaatiojärjestelmän valmistaja, eikä ohjelmoija. Graafisessa ohjelmointikielessä, kuten toimilohkokaaviossa (Kuva 6) etuna on sen havainnollisuus (Rakennusautomaatiojärjestelmät 2018, 76). Yhteistä kaikille ohjelmointitavoille on se, että ohjelmointi on suurelta osin vanhojen ohjelmakirjastojen kopiointia projektista toiseen. Aloittelevien automaatio-ohjelmoijien kannattaa yleensä turvautua aiemmin tehtyihin toimiviin ohjelmiin, kun kokeneemmat tekevät uusia ohjelmia kirjastoon (Rakennusautomaatiojärjestelmät 2018, 274).



KUVA 6. Toimilohkokaaviolla ohjelmointia Ouman Ouflex järjestelmässä

2.4.1 Prosessiohjelman dokumentointi

Uusia prosessiohjelmia ei aina tarvitse tehdä, sillä yleisimmille toiminnoille löytyy lähes kaikista järjestelmistä jo valmiit kirjastot, joita hyödyntämällä pääsee ohjelmoinnissa pitkälle. Kun ohjelmaa kuitenkin tarvitsee tehdä, on sen hyvä dokumentointi myös tärkeää, jotta ohjelmaa on helpompi huoltaa ja laajentaa vielä vuosienkin päästä (Rakennusautomaatiojärjestelmät 2018, 274). Talotekniikassa ohjelman dokumentointi ei tarkoita pelkästään oman ohjelmointityön dokumentointia, vaan siihen sisältyy myös rakennusautomaatiosuunnitelmien toimintaselostukset ja muu suunnitelmissa esitetty toiminnan kuvaus. Dokumentointi onkin tarkoitettu järjestelmän loppukäyttäjälle, jotta huolto- ja käyttöhenkilöstö voi perehtyä järjestelmän toimintaan paremmin (Ohjelmistojen dokumentointi 2009, 5). Toimintaselostukset ja ohjelmaluettelot, jotka ovat osa dokumentaatiota, ovat yleensä valmiina suunnitelmissa, mutta dokumentoitavaa jää myös projektinhoitajalle. Projektinhoitajan laatimia dokumentteja urakoissa ovat esimerkiksi järjestelmän käyttöönottopöytäkirjat ja erilaiset I/O-piste- sekä parametriluettelot. Luettelot voi yleensä tulostaa suoraan järjestelmästä, eikä sitä tarvitse erikseen tehdä.

Hyvän dokumentaation ei tarvitse tarkoittaa pelkkää kirjoitettua opasta ohjelman ymmärtämiseen, vaan dokumentoinnin tukena voi olla ohjelmaan itseensä tehty

dokumentointi. Hyvin nimetyt I/O-tunnukset, jotka noudattavat kiinteää logiikkaa, ovat käyttäjälle helposti omaksuttavissa. Selkeästi jäsennetty ja hyvin kommentoitu ohjelmakoodi helpottaa ohjelman ymmärtämistä. Ohjelmasta saadaan monilla tavoilla tehtyä itsensä dokumentoivaa (Ohjelmistojen dokumentointi 2009, 7).

Kommentointi on ohjelmoijan paras keino parantaa ohjelman luettavuutta, mutta ohjelman ylläpidon helpottamiseen on olemassa muitakin hyviä käytäntöjä. Laajojen ohjelmakokonaisuuksien jako pienempiin osiin ja aliohjelmiin, joiden tarkoitus on kerrottu kommenteilla, helpottaa ohjelman luettavuutta. Kun aliohjelmat ovat hyvin suunniteltuja ja toimivaksi todettuja, voidaan niitä käyttää useammassa paikassa ja ylläpito helpottuu entisestään. Aliohjelman esittelykommentteihin voi kirjoittaa paljonkin tekstiä, sillä kommentit eivät vaikuta ohjelman suoritusaikaan millään tavalla. Kommentointi ennen ohjelmointia auttaa yleensä myös ohjelmoijaa hahmottamaan minkälaista ohjelmaa on kirjoittamassa. Ohjelmassa muuttujat kannattaa myös nimetä kuvaavasti ja kommentoida niiden tarkoitus, jos mahdollista. Näin ohjelmasta voidaan lukea sen toimintaa myös rivi kerrallaan suoraan ohjelmakoodista (Ohjelmistojen dokumentointi 2009, 11). Kuvassa 7 on aliohjelma, jossa lasketaan lämmön talteenoton hyötysuhdetta. Kommentit helpottavat ymmärtämään mitä eri vaiheissa tehdään, ja muuttujat ovat nimettynä kuvaavasti.

```
function f_ltoHysu ()
    --Lasketaan LTO-hyötysuhde
    local sPoisto = Data.get(IVKone.Poistoilma..idSuffix.AI..".pv") or -999
    local sLtoJalk = Data.get(IVKone.LTOJalk..idSuffix.AI..".pv") or -999
    --Käytetään raitisilmakanavan mittausta hyötysuhdelaskennassa, jos se on olemassa. Muuten ulkoanturia...
    local sRaitisilma   = (Data.get(IVKone.Raitisilma..idSuffix.AI..".pv") or Data.get(IVKone.Ulkoilma..idSuffix.AI..".pv") or -999)



    --Käytetään jäteilman mittausta hyötysuhdelaskennassa, jos se on olemassa.  Muuten ulkoanturia...
    --local sJateilma   = (Data.get(IVKone.Jateilma..idSuffix.AI..".pv") or Data.get(IVKone.Ulkoilma..idSuffix.AI..".pv") or -999)
    -- Käytetään ulkolämpötilaa jäteilman sijaan..
    local sJateilma = sRaitisilma
    --Lasketaan ilmamäärien suhde
    local sTimaara = Data.get(IVKone.TImaara..idSuffix.AI..".pv") or 0
    local sPimaara = Data.get(IVKone.PImaara..idSuffix.AI..".pv") or 0
    local suhde = 0
    if sPimaara > 0 then
        suhde = sTimaara/sPimaara
    else
        suhde = 1
    end
    --Laskenta tässä
    local hyotysuhde = 0
    -- ei lasketa hyötysuhdetta, eikä luoda pisteitä, jos mikä tahansa pisteistä puuttuu tässä vaiheessa


    if sLtoJalk ~= -999 and  sRaitisilma ~= -999 and  sPoisto ~= -999 and sJateilma ~= -999 then
        if (sLtoJalk-sRaitisilma) < 0.3 or (sPoisto - sJateilma) < 0.3 or Data.get(IVKone.Tulopuh..idSuffix.DI..".pv") == 0 then
            hyotysuhde = 0
        else
            hyotysuhde=((sLtoJalk-sRaitisilma)/(sPoisto-sJateilma))*suhde*100
            hyotysuhde = hvac_limit(hyotysuhde,100,0)
        end
        --Lasketaan LTO-hyötysuhde ja luodaan sille samalla piste
        Data.create (IVKone.LTOhysu..idSuffix.AI,"AV",{pointId="LTO", description="Lämmön talteenoton hyötysuhde",dispUnit="%"})
        Data.set(IVKone.LTOhysu..idSuffix.AI..".pv",hyotysuhde)
    end
end
```

KUVA 7. Esimerkki kommentoidusta ohjelmasta

Projektinhoitajan laatiman dokumentoinnin ja automaatiojärjestelmän valmistajan toimittaman käyttöohjeen lisäksi loppukäyttäjän tukena järjestelmän käyttöön on ainoastaan projektinhoitajan antama käyttökoulutus. Hyvin laadittu ja opastusta tukeva dokumentaatio vähentää myöhempiä neuvontapuheluita.

2.4.2 Graafisen käyttöliittymän suunnittelu

Nykyaikaisissa automaatiojärjestelmissä on lähes aina graafinen käyttöliittymä ja sitä varten täytyy ohjelmoijan tehdä myös käyttöliittymägrafiikka prosessikuvineen. Käyttöliittymägrafiikka voi olla alakeskuksessa omassa näytössä, valvomossa, tai molemmissa. Valvomoon voi tehdä hieman kattavamman käyttöliittymän sillä valvomoon voidaan helposti laittaa suurempi näyttö, eikä valvomolaitteiston suoritustehoa tarvitse uhrata automaatioprosessien pyörittämiseen. Yleisimmin käytössä olevissa automaatiojärjestelmissä on lähes poikkeuksetta www-palvelin, jonka kautta käyttöliittymää käytetään selaimella, ja käyttöliittymä voi olla täysin sama paikallisella näytöllä ja vaikka internetin yli etäkäytettynä. Käyttöliittymägrafiikan tehtävä on havainnollistaa järjestelmän käyttäjille ja huoltohenkilöstölle prosessien toimintaa (ST-kortti 721.01 2016, 1)

Käyttöliittymägrafiikan ohjelmointi on samankaltaista kaikilla järjestelmillä, mutta työkalut ovat hyvin vaihtelevia. Yhteistä kuitenkin on, että esitettävään järjestelmään tehdään taustakuva, jonka päälle liitetään järjestelmän pisteet erilaisiin dynaamisiin elementteihin, joista havaitaan prosessiin liittyvien pisteiden arvojen muutokset. Taustakuvan voi tehdä kokonaan erillisellä piirtotyökalulla, tai sen voi rakentaa staattisista symboleista, eli valmiiksi piirretyistä taustakuvan elementeistä. Symboleita käytettäessä taustakuvaan on hieman helpompi tehdä pieniä muutoksia, kun koko taustakuvaa ei tarvitse avata piirtotyökalussa. (Rakennusautomaatiojärjestelmät 2018, 63)

Käyttöliittymä täytyy kuitenkin suunnitella hyvin ennen ohjelmointia. Grafiikan on syytä olla looginen ja helppokäyttöinen, jotta järjestelmää voidaan hallita tehokkaammin. Huonosti suunniteltu käyttöliittymä voi johtaa väärinkäyttöön, jolloin automaatiojärjestelmän hyödyt esimerkiksi energiansäästössä voi jäädä saavuttamatta (Rakennusautomaatiojärjestelmät 2018, 123).

Hyvän käyttöliittymän voi tunnistaa siitä, että kaikki esitettävä tieto on havainnollista ja yksiselitteistä, eikä siihen jää tulkinnanvaraa. Käyttöliittymän käytön on oltava helposti opittavissa ja käyttöliittymän pitää ohjata käyttäjää vuorovaikutteisesti. Lisäksi järjestelmän on hyvä olla visuaalisesti selkeä ja väritykseltään huolella mietitty, jotta käyttäjän huomio kiinnittyy oikeisiin paikkoihin. Käyttöliittymä tulee kuitenkin lopulta asiakkaan käyttöön, eli myös asiakkaan toiveet tulee huomioida. (Kiinteistöjen valvontajärjestelmät 2017, 23)

Rakennusautomaatiojärjestelmässä käyttöliittymässä yleensä esitettäviä asioita ovat ilmanvaihtokoneiden ja lämmönvaihtimien prosessien kaaviot. Prosessikaaviot on hyvä esittää grafiikalla mahdollisimman yksinkertaisena mallina, jotta ne ovat selkeitä. ST-kortissa on 721.01 on omat suositukset käyttöliittymägrafiikan dynaamisien ja staattisten osien väreille (Kuva 8). Staattiset osat ovat taustakuvaa, eli prosessikaaviota tai muuta vastaavaa, ja dynaamiset osat esittävät mittauksia, tilatietoja ja muita järjestelmän liikkuvia osia.

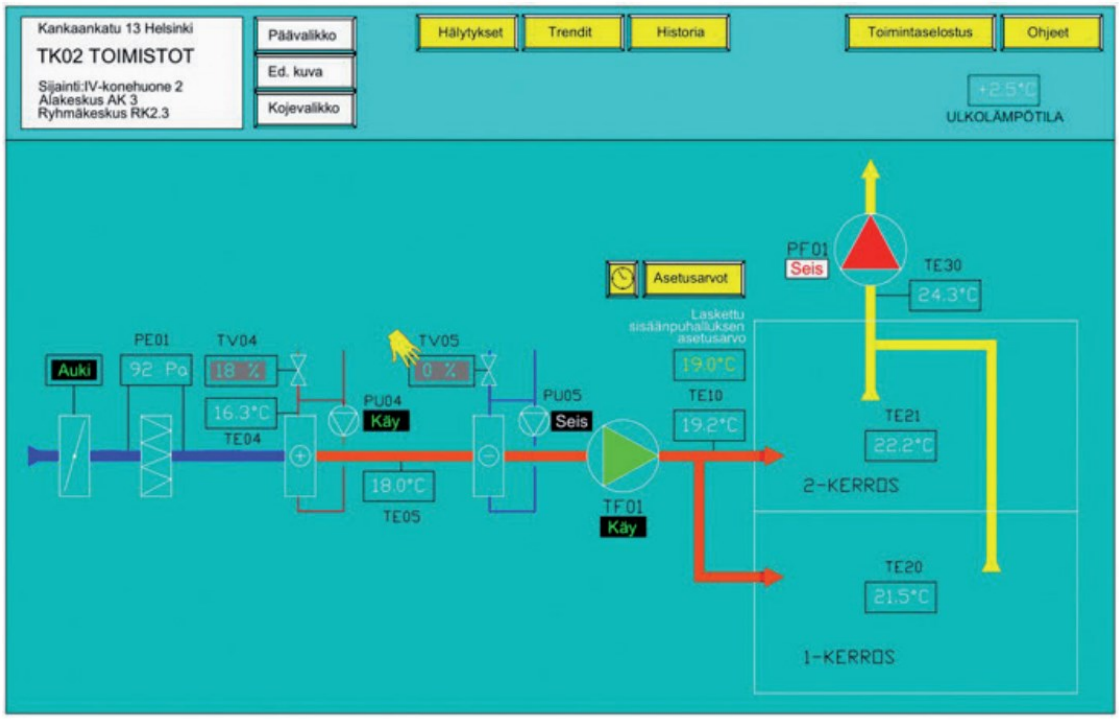
Kaavionäyttö (dynaaminen)		
Symboli/tieto	**Väri**	**Huomautus**
Hälytys	Punaiset sävyt	Eri hälytysluokille eri sävyt
Käy-tila	Vihreä	Puolinopeus esim. turkoosi
Seis-tila	Valkoinen	Seis luvatta: punainen
Asetusarvo	Keltainen	
Säätöohjaus	Turkoosi	
Mittaus	Valkoinen Vaihtoehtoisesti: vihreä	Yli sallitun arvon on punainen Alle sallitun arvon on sininen
Lepotila	Valkoinen Vaihtoehtoisesti: harmaa	
Siirtymäpiste	Keltainen Vaihtoehtoisesti: valkoinen	

Kuvapohja (staattinen)		
Symboli/tieto	**Väri**	**Huomautus**
Perusviivat	Valkoinen	vaaleat sävyt, mikäli värejä tarvitaan
Alueet	Valkoinen, harmaa	esim. ilmastoinnin vaikutusalue
Tekstit	Valkoinen, harmaa	
Symbolit	Valkoinen, harmaa	Dynaaminen symboli, ks. edellinen taulukko
Prosessin osat	Mahdollisimman vaaleat sävyt esim. tuloilma = punainen poistoilma = keltainen likainen poistoilma = ruskea ulkoilmakanava = sininen lämmin vesi = punainen kylmä vesi = sininen	

KUVA 2. ST-kortin 721.01 värisuositukset

Prosessikaaviossa esitetään kaikki prosessiin liittyvät mittaukset ja tilatiedot, mutta suuremmissa prosesseissa käyttöliittymägrafiikka on hyvä jakaa kahteen tai useampaan sivuun käytössä oleva näyttötila huomioiden. Jakamalla kuva useampaan sivuun saadaan grafiikasta helppolukuisempi, kun dynaamisilla osilla on

enemmän tilaa ympärillään (Kiinteistöjen valvontajärjestelmät 2017, 35). Prosessikaavioissa tulee ST-kortin 721.01 mukaan käyttää Suomen rakentamismääräyskokoelman mukaisia piirrosmerkkejä, eli samoja piirrosmerkkejä, joita automaatiosuunnitelmien säätökaavioissakin käytetään. Kuvassa 9 on ST-kortin mallin mukainen esitys ilmanvaihtokoneen grafiikkakuvasta. Ilmanvaihtokoneen asetusarvoja ja hälytysrajoja varten on syytä tehdä erillinen asetussivu, jotta kaikkea ei tarvitse mahduttaa prosessikuvan kanssa samalle sivulle.



KUVA 3. ST-kortin esimerkki: Ilmanvaihtokoneen prosessikuva

Prosessikuvien lisäksi myös navigoinnin sivujen välillä täytyy olla loogista ja selkeää. Puumainen rakenne, jossa eri järjestelmät ovat hajautettuna omiin oksiinsa helpottaa järjestelmän käyttöä. Suuremmissa kokonaisuuksissa samalle prosessisivulle täytyy päästä useampaa reittiä, esimerkiksi kerrostasokuvan kautta. Tärkeintä kuitenkin on, että käyttäjälle on selvää, miten järjestelmän valikoissa liikutaan. Navigointia voi helpottaa käyttäjäkohtaisesti määrittyvillä valikoilla, joissa alemman käyttäjätason tunnuksilla osa valikoista voidaan piilottaa ja aloitussivu voidaan vaihtaa loppukäyttäjälle sopivaksi. (ST-kortti 721.01, 9)

2.5 Testaukset, toimintakokeet ja säätöjen viritys

Kun järjestelmä on ohjelmoitu valmiiksi ja työmaalla asennukset alkavat olla valmiina, alkaa ohjelmoijalla tärkeä vaihe työmaalla, eli järjestelmän testaus. I/O-pisteiden kytkentöjen testaus voidaan sopia kuuluvaksi kytkennät tekevälle aliurakoitsijalle, mutta lopulta automaatiourakoitsija on kuitenkin itse vastuussa siitä, että järjestelmä toimii niin kuin pitää. Järjestelmän pistetestauksista, sekä prosessikohtaisten toimintojen tarkastuksista tehdään itselleluovutusdokumentointi. Itselleluovutusdokumentteja ovat muun muassa pistetestausluettelot ja muut oman työn tarkastuksesta syntyneet pöytäkirjat. Dokumentoinnin avulla urakoitsija voi itse pitää kirjaa testatuista ja testaamattomista asioista, ja dokumentti annetaan tilaajalle osoituksena urakan valmiudesta vastaanottomenettelyiden aloittamiseen (Rakennusautomaatioprojektin hallinta 2013, 8).

Kun muut urakkaan liittyvät urakoitsijat ovat ilmoittaneet valmiutensa vastaanottomenettelyiden aloitukseen, on urakassa seuraavana vaiheena toimintakokeet, jos urakassa sellaista vaaditaan. Toimintakokeissa käydään yleensä pistokoemaisesti läpi ainakin ilmanvaihtokoneiden lukitustoiminnot, kuten jäätymissuojat ja ilmastoinnin hätäpysäytys, sekä mittausten raja-arvohälytysten toiminta. Toimintakokeissa simuloidaan vikatilanteita kytkemällä antureita irti, ja toimintoja testataan asetusarvoja muuttamalla. Hälytysten viiveitä joutuu yleensä toimintakokeissa muuttamaan, jotta hälytysten aktivoitumista ei tarvitse odotella. Rakennusautomaatiojärjestelmän toimintakokeissa käydään yleensä myös läpi, että suunnitellut toiminnot toimivat halutulla tavalla.

Vasta hyväksytysti suoritettujen toimintakokeiden jälkeen päästään järjestelmät säätämään, ja ilma- ja vesivirtojen säätötoimenpiteiden jälkeen automaatiourakoitsija voi ryhtyä virittämään säätöpiirejä. Viritykset on tehtävä sellaisissa olosuhteissa, että viritetyissä säätöpiireissä on kuormaa, eli karkeasti jäähdytykset viritetään kesällä ja lämmitykset talvella. Virityksiä jää siis usein myös urakan luovuttamisen jälkeen takuuaikana tehtäväksi. Säätöpiirien virittämisestä tehdään virityspöytäkirja, sekä trendiajo. Trendiajolla tarkoitetaan säätöpiirin mittauksesta piirrettyä arvokäyrää, josta nähdään säädön toiminta. Ilmanvaihtokoneissa tren-

dikäyrällä on hyvä näkyä koneen käynnistys- ja nopeudenvaihtotilanteet. Lämmönjakohuoneen vaihdinpaketin virityspöytäkirjaan kirjataan asetusarvojen säätökäyrien ja viritysparametrien lisäksi vallitsevat olosuhteet, eli kaukolämmön lämpötilat ja paineet, sekä ulkolämpötila, jotta nähdään missä olosuhteissa säätö on viritetty. Ilmanvaihtokoneiden virityspöytäkirjaan kirjataan kanavapainesäätöjen ja lämpötilasäätöjen asetusarvot, sekä viritysparametrit. Pöytäkirjoissa on myös hyvä esittää säätölaitteiden, eli venttiilien, toimilaitteiden, taajuusmuuttajien valmistajat ja mallit. (ST 711.04 2020, 10)

2.6 Käytönopastus

Rakennusautomaatiourakkaan kuuluu harvoja poikkeuksia lukuun ottamatta myös järjestelmän käyttökoulutus. Usein urakkapapereissa on urakkaan määritetty esimerkiksi yksi päivä tai kahdeksan tuntia opastusta. Opastus annetaan yleensä huoltohenkilöstölle, mutta joissain tapauksessa myös kiinteistön muille käyttäjille annetaan koulutus, esimerkiksi huonesäätöjen käytöstä. Koulutuksella pyritään siihen, että loppukäyttäjä osaa käyttää järjestelmää oikein.

Koulutus tapahtuu yleensä kohteessa suoraan automaatiokeskuksen edessä, jolloin opastettavan asian voi samalla näyttää käytännössä. Laajemmat koulutukset voidaan toki pitää myös mukavimmissa oloissa siihen tarkoitukseen varatussa neuvotteluhuoneessa (Rakennusautomaatiojärjestelmät 2018, 254). Käytännön urakoissa käyttökoulutus jää yleensä varsin suppeaksi, projektin ohjelmoijan itsensä antamaksi paketiksi, jonka pääpaino on urakan kohteena olevan järjestelmän yksilölliset toiminnot. Tällaisesta koulutuksesta opastettaville ei yleensä jää mitään materiaalia, johon tukeutua myöhemmin. Ohjelmoijalla on kuitenkin paras käsitys itse ohjelmoimastaan kokonaisuudesta, ja sen vuoksi hänen on myös helppo vastata koulutettavien kysymyksiin järjestelmän osista. Kiinteistöhuollon ammattilaisilla alkaa nykyään olla laaja käyttökokemus rakennusautomaatiojärjestelmistä, eikä itse automaatiojärjestelmän koulutus ole välttämättä tarpeen. On toki myös huoltohenkilöitä, joilla ei ole lainkaan, tai vain vähän kokemusta laitteiden käytöstä (Kiinteistöjen Valvomojärjestelmät 2017, 80).

Asianmukainen koulutus voi kestää useammankin päivän, ja koulutuksesta suurin osa tapahtuu kohteessa. Koulutuksen sisältöön pitäisi sisältyä päivittäiseen

käyttöön tarpeelliset toimenpiteet, kuten hälytysten läpikäynti ja kuittaus, sekä hälytysrajojen muutokset ja prosessien muu tarkkailu. Laajimmillaan osana koulutusta opastetaan myös järjestelmän perusohjelmointia, kuten I/O-pisteiden lisäämistä tietokantaan ja energiaraporttien luomista olemassa olevista mittauksista. Järjestelmästä on hyvä tehdä myös urakkakohtainen yksilöity käyttöohje loppukäyttäjälle tueksi. (Kiinteistöjen Valvomojärjestelmät 2017, 87).

3 ITSENSÄ OHJELMOIVAN JÄRJESTELMÄN TOTEUTUS

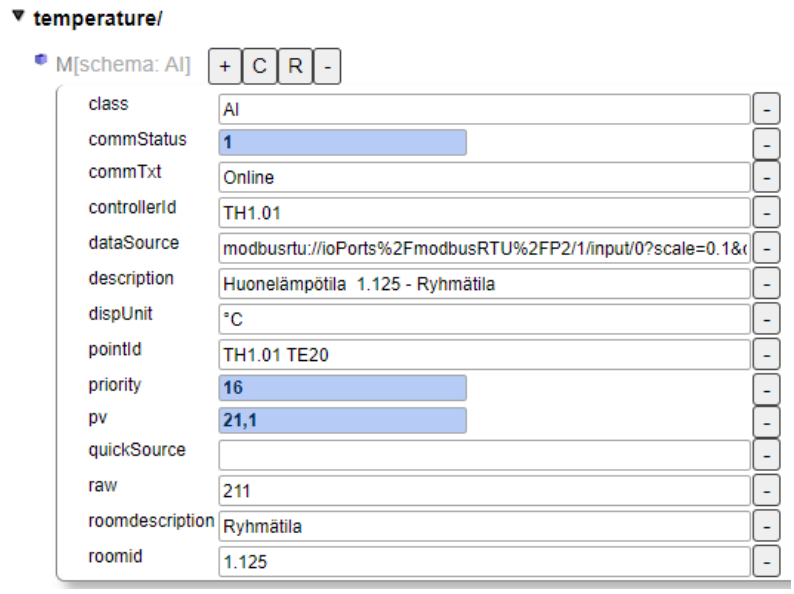
Useimmissa automaatiojärjestelmissä ohjelmointi täytyy ainakin joiltakin osin tehdä käsin. Samoja toimilohkoilla tehtyjä toiminnallisuuksia voidaan tehdä useampaan järjestelmään, mutta esimerkiksi mittaustulojen ja ohjauslähtöjen liittäminen oikeille paikoilleen jää usein ohjelmoijan tehtäväksi. Myös hälytys- ja ohjausrajat jäävät usein käsin aseteltaviksi. Jotta hälytysrajat, asetusarvot ja muut parametrit voidaan asetella ohjelmaa luodessa automaattisesti, täytyy järjestelmän olla ohjelmansa kanssa joustava. Esimerkiksi FBD ohjelmoinnissa järjestelmäkokonaisuuksia voi tallentaa asetusarvoineen ja parametreineen, jolloin samanlainen järjestelmä on suhteellisen helppo luoda uudestaan, mutta muutosten tekeminen on aina käsityötä ja esimerkiksi yhden lämpötilahälytyksen kopioiminen eri hälytysrajoilla vaatii yllättävän paljon työtä. Automaattinen ohjelman monistaminen toimilohkokaaviossa on rajoittunutta, jos edes mahdollista. Tekstipohjaisessa ST-kielessä voidaan pieniä osakokonaisuuksia kopioida helposti, ja toteutustavasta riippuen myös hälytysrajat ja asetusparametrit voidaan kuljettaa mukana. ST-kielessä rajoitteeksi muodostuu se, että toimilohkoja ei voi luoda dynaamisesti. Yhden ilmanvaihtokoneen voi ohjelmoida helposti toimilohkoja hyödyntäen, mutta jokaiselle samanlaisellekin koneelle täytyy luoda oma toimilohkon kutsu.

FBD, Ladder ja ST-kieli ovat kaikki IEC61131-3-standardin mukaisia ja niiden kankeus juontaakin juurensa sidonnaisuuksista standardiin. Standardinmukaisuus mahdollistaa sen, että järjestelmillä ohjelmointi voi ainakin teoriassa olla samanlaista eri järjestelmillä, jos ne käyttävät samaa ohjelmointikieltä. Valmiissa järjestelmässä ei kuitenkaan ulospäin näy minkälainen ohjelma sen toimintoja pyörittää. Tämän vuoksi ohjelmoinnin kankeita solmuja voi lähteä purkamaan toisenlaisella lähestymistavalla. Tamperelainen Bithouse on kehittänyt Actiweb-järjestelmän, jonka ohjelmointi on joustavaa ja monipuolista, mutta sen ohjelmointikieli ei ole IEC61131-3 standardin mukainen tyypitetty ohjelmointikieli. Järjestelmän dynaamisuus kuitenkin mahdollistaa ohjelmointitemppuja, jotka muilla järjestelmillä on joko mahdottomia tai vaikeita (Actiweb ohjelmointiopas, sivu 2), ja siksi tämän toteutuksen automaatioalustaksi valikoitui juuri Actiweb.

3.1 Actiweb-automaatiojärjestelmä

Actiweb ohjelma-alusta on oikeastaan osa Bithousen AWM automaatiojärjestelmää. Järjestelmäkokonaisuuteen kuuluu Actiwebin lisäksi Bithousen kehittämä I/O-moduuli Magio816, mutta järjestelmässä voi käyttää mitä tahansa modbus tai BACnet pohjaista I/O:ta. Lähtökohtana järjestelmässä on se, että asioita ei haluta rajoittaa tiettyihin laitteisiin. Actiwebin voi asentaa mihin tahansa Linux-käyttöjärjestelmää käyttävään tietokoneeseen, eikä sen ohjelmointi vaadi erityisiä lisensoituja työkaluja (AWM Automaatiojärjestelmä).

Actiwebin suurin vahvuus sen avoimuuden jälkeen on sen pistetietokanta. Tietokanta näytetään järjestelmässä json-muodossa, ja sitä on erittäin helppo käsitellä grafiikkasivuissa JavaScriptin avulla ja ohjelmointipuolella Lua-taulukoina. Mikä tahansa tietokantapiste voi sisältää mitä tahansa tietoa, jota voidaan hyödyntää Actiwebin omassa ohjelmassa haluamallaan tavalla, mutta esimerkiksi BACnet-verkkoon tietokanta näyttää vain BACnet-yhteensopivat osat. Kuvassa 10 on BACnet-standardin mukainen AI-tyypin objekti, jossa on lisätietokenttinä *controllerId*, *pointId*, *dispUnit*, *roomdescription* ja roomid, joiden avulla pisteestä voidaan näyttää lisätietoja grafiikkasivuilla tai seurantaraporteissa.



KUVA 4. AI-tyypin objekti

Actiwebissä ohjelmakoodia ei tehdä IEC61131-3-standardin mukaisella ohjelmointikielellä. Kaikki sovellusohjelma tehdään Lua-kielellä, joka on tekstipohjainen, kuten ST-kieli, mutta siinä ei ole standardin mukaisia kiinteitä datatyyppejä. Ohjelmoinnin voi halutessaan tehdä selaimessa kirjoittamalla millä tahansa järjestelmän sivulla osoiteriville ”&code” (Actiweb ohjelmointiopas, sivu 4), mutta ohjelmointikielen tunnistava tekstinkäsittelyohjelma, kuten Notepad++ tai Visual Studio Code, helpottaa ohjelmoijan työtä. Ohjelmatiedostot löytyvät järjestelmän kovalevyltä /opt/slc/prg kansion alta.

Lua-kielellä yhdessä Actiwebin ytimen, slcenginen, joustavan tietokantarakenteesen kanssa saadaan luotua poikkeuksellisen monipuolisia ohjelmia, joilla saadaan vähennettyä toistuvia osia ohjelmoinnista. Actiwebin tietokanta voidaan luoda niin halutessa kokonaan ohjelmallisesti ja prosessiohjelmalle voidaan tietokantapisteillä kertoa riittävästi tietoa, jolla ohjelma valitsee oikeat aliohjelmat ja toimilohkot kyseiseen järjestelmään. Ohjelmaan voidaan luoda omia pistetyyppejä ja ohjelmarutiineja, joilla kyseistä pistetyyppiä käsitellään aina ilman erillistä pistekohtaisen ohjelman kirjoittamista. Ohjelmarutiini taas voi vastaavasti luoda uusia pisteitä ja pistekokonaisuuksia, jolloin järjestelmään voi ikään kuin vauhdissa lisätä esimerkiksi valaistusohjauksen, joka saa automaattisesti aikaohjelman ja alkaa heti toimimaan ilman yhdenkään ohjelmarivin kirjoittamista. Actiwebin ohjelmoinnista vaikeaa tekee kuitenkin se, että ohjelmoijan täytyy itse pitää huoli siitä minkä tyyppisinä pisteet ohjelmaan luetaan. Lua kielessä muuttuja voi olla mitä tyyppiä tahansa ja tietokannasta luettuna esimerkiksi kenttä *roomid* tulkitaan ohjelmassa tekstinä ja *pv* numerona. Tämä ilmaistaan kentän taustavärillä. Usein kenttiä lukiessa täytyykin erikseen ohjelmassa tyyppimuunnoksen avulla pakottaa luettu arvo oikeaan muotoon. (Actiweb ohjelmointiopas, sivu 10).

Käyttöliittymäohjelmointi Actiwebissä tehdään myös selainpohjaisella työkalulla, joka on järjestelmässä sisäänrakennettuna. Grafiikkatyökaluun pääsee Actiwebin selainkäyttöliittymästä kirjoittamalla osoiterivillä olevan osoitteen perään ”&edit”. Grafiikkakuvat ovat luontivaiheessa XML-muotoisia, mutta selaimelle se näytetään HTML5-tekniikan mukaisessa muodossa. (Actiweb Käyttöliittymäoh-

jelmointi). Grafiikkasivujen automaattista luomista varten Actiwebissä on *pageengine*-niminen työkalu, joka luo uuden grafiikkatiedoston aiemmin luodun mallitiedoston pohjalta.

Grafiikkasivujen, ohjelmakoodin ja tietokannan yhteen liittäminen täytyy kuitenkin suunnitella huolellisesti, jotta työn määrä lopulta vähenee.

3.2 Järjestelmän suunnittelu

Automaattisesti itsensä ohjelmoivaa järjestelmää suunniteltaessa täytyy rajata järjestelmäohjelmat siten, että kaikki tuotettava ohjelma on mahdollisimman monikäyttöistä. Suunnitelmista ei yleensä saa liikaa poiketa, joten ohjelman täytyy myös helposti mukautua eri tavoilla suunniteltuihin järjestelmiin. Kaikkea ei voi, eikä kannata ottaa huomioon ennakkoon, mutta uudet suunnitelmissa mahdollisesti yleistyvät ratkaisut pitää pystyä lisäämään järjestelmään siten, että vanha toiminnallisuus säilyy.

Järjestelmän toteutuksen peruskiviksi muodostuivat seuraavat kolme sääntöä:

1. Loppukäyttäjälle näkyvin osa järjestelmästä on sen käyttöliittymägrafiikka. Käyttöliittymägrafiikan tulee olla helppokäyttöinen ja selkeä. Automaattisesti luotavan grafiikan täytyy olla mahdollisimman yleispätevä, jotta sitä voidaan käyttää useammassa kokonaisuudessa.

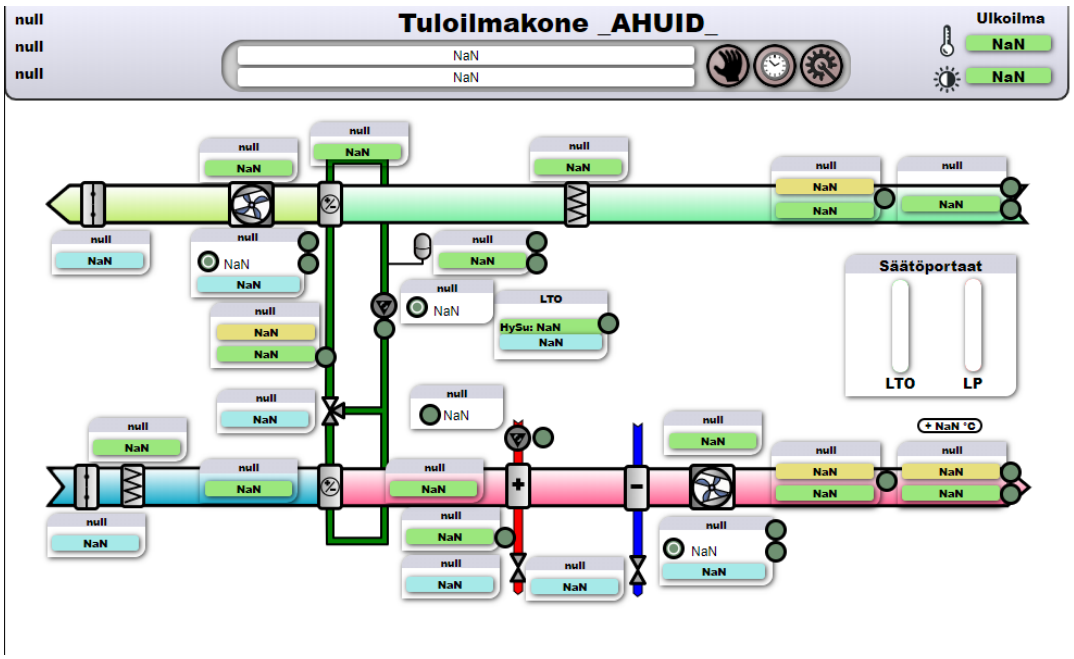
2. Järjestelmän pistetietokanta on rajapinta fyysisen maailman, prosessiohjelman ja lopulta käyttöliittymän välillä. Tietokanta täytyy rakentaa loogisella tavalla, jotta ohjelmaan on helppo poimia tarvittavat tiedot. Tietokannan rakenne ei kuitenkaan saa olla liian jäykkä, jotta ohjelmoinnin vapaus säilyy. Tietokantapisteiden nimeämiskäytännön pitää olla samanaikaisesti kiinteä ja mukautuva.

3. Prosessiohjelma ei näy käyttäjälle, mutta sen täytyy olla selkeä. Perusohjelma täytyy tehdä kerralla hyvin, mutta erikoisohjelmat täytyy voida lisätä liitännäistyyppisesti prosessin mihin vaiheeseen tahansa. Perusohjelmakin tulee pilkkoa pienempiin helposti yhdisteltäviin osiin.

3.3 Monikäyttöinen käyttöliittymägrafiikka

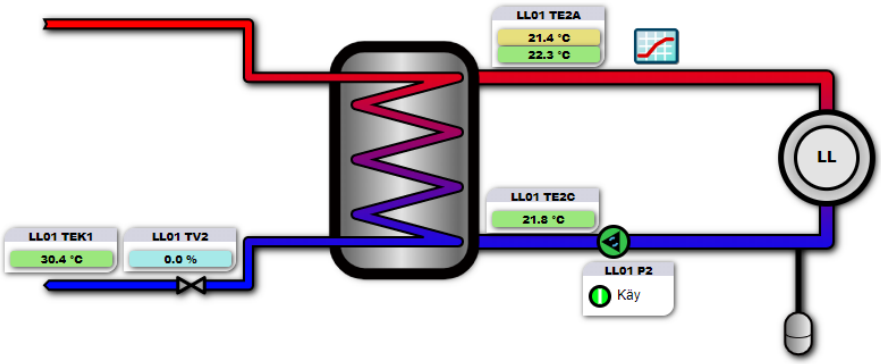
Nykyaikaisissa rakennusautomaatiojärjestelmissä on lähes poikkeuksetta kosketusnäytöllä varustettu käyttöliittymä. Myös käyttöliittymägrafiikka täytyy voida luoda automaattisesti, mutta sitä varten tarvitaan mallitiedostot, jonka mukaan grafiikka luodaan. Käyttöliittymän ulkoasussa huomioidaan suositusten mukaiset värit ja symbolit, mutta sen täytyy kuitenkin näyttää modernilta. Kuvien välillä navigointia varten Actiweb järjestelmä luo valikkorakenteen automaattisesti grafiikkasivujen kansiomallin mukaan, eikä ohjelmoidessa tarvitse tehdä erikseen valikkoa. Valikossa näkyvän tekstin voi vaihtaa lisäämällä grafiikkasivun muokkaustilassa sivulle nimen *Page name* kenttään.

Mallitiedoston tulee sisältää kaikki mahdolliset perustoiminnot, jotka lopullinen järjestelmä voi tarvita. Ylimääräiset poistetaan joko manuaalisesti tai automaattisesti, kun ohjelma on lopullisen sivun luonut. Esimerkiksi ilmanvaihtokoneen grafiikkasivulle piirretään valmiiksi lämmön talteenotto, lämmityspatteri, jäähdytyspatteri ja kiertoilmapelti. Lisäksi kaikki paine- ja lämpötilamittaukset. Erikoisuuksia, kuten kostutuslaitteistoa ei välttämättä tarvitse tehdä mallipohjaan, sillä ne ovat paljon harvinaisempia. Kuvassa 11 on glykolilämmöntalteenotolla varustetun ilmanvaihtokoneen mallitiedosto. Kuvassa on näkyvillä kiinteänä tekstiosuutena avainsana _AHUID_, joka korvataan ilmanvaihtokoneen tunnuksella, kun sivu luodaan ohjelmassa.



KUVA 5. Mallikuva Ilmanvaihtokoneesta, jossa on Glykolilämmöntalteenotto

Lämmönjakohuoneen grafiikkasivumalli tehdään verkostokohtaisesti siten, että verkosto yksinkertaistetaan havainnolliseksi, eikä suunnitelmaa orjallisesti noudattavaksi. Näin samaa kuvaa voidaan käyttää taustana mihin tahansa verkostoon. Joissain tapauksissa voi olla tarpeen tehdä havainnollisempi kuva, mutta sitä ei voi tehdä automaattisesti. Kuvassa 12 on esimerkki lattialämmitysverkostosta, joka on luotu automaattisesti. Verkosto voisi olla mikä tahansa lämmitysverkosto, sillä ainoa sen lattialämmitykseksi määräävä tekijä on ”LL”-tunnus. Taustakuvassa hyödynnetään LVI-piirrosmerkkejä muistuttavia symboleita.



KUVA 6. Valmiiksi generoitu lattialämmitysverkosto

Mallitiedostoissa kaikki tietokantaan viittaukset tehdään korvattavilla avainsanoilla, eli tageillä (Kuva 13). Avainsanat muutetaan Actiwebin sivugeneraattorilla, pageenginellä, vastaamaan oikeita tietokantapisteitä. Avainsanoihin voidaan viitata suoraan myös suoraan ohjelmassa, jolloin voidaan etsiä tietokannasta tietyllä tavalla nimetyt pisteet automaattista grafiikkasivun luontia varten.

Tietokantatagit			
Tähän tiedostoon täydennetään kaikki grafiikkakuvissa ja ohjelmoinnissa tarvittavat tagit.			
Lisäksi listataan nyt käytössä olevat korvaavat tekstit, jos ne ovat kiinteitä			
Näiden tagien mukaan voidaan luoda ID:t myös ohjelmointia varten			
	Tagi	Korvaava teksti	Kuvaus
Ylätason tagit	**tag**	**replacement**	**description**
	ROOTPATH	hvac	Juurihakemisto kaikille pisteille.
	HVAC	hvac	Juurihakemisto kaikille LVI pisteille
	ELECTRIC	electric	Juurihakemisto kaikille sähköpisteille
	VENTILATION	ventilation	Ilmanvaihto/Ilmanvaihtoverkosto
	HEATING	heating	Lämmitysjärjestelmä. Esimerkiksi lämmityspatteri
	COOLING	cooling	Jäähdytys
	SETTINGS	settings	Asetustietokannan polku
	STATUS	status	Tilatekstien polku
	PROJECT	project	Projektin tiedot sisältävä polku
	DISTRICT	district	Kaukolämpö/- kylmä

KUVA 7. Ote tietokannan avainsanoista

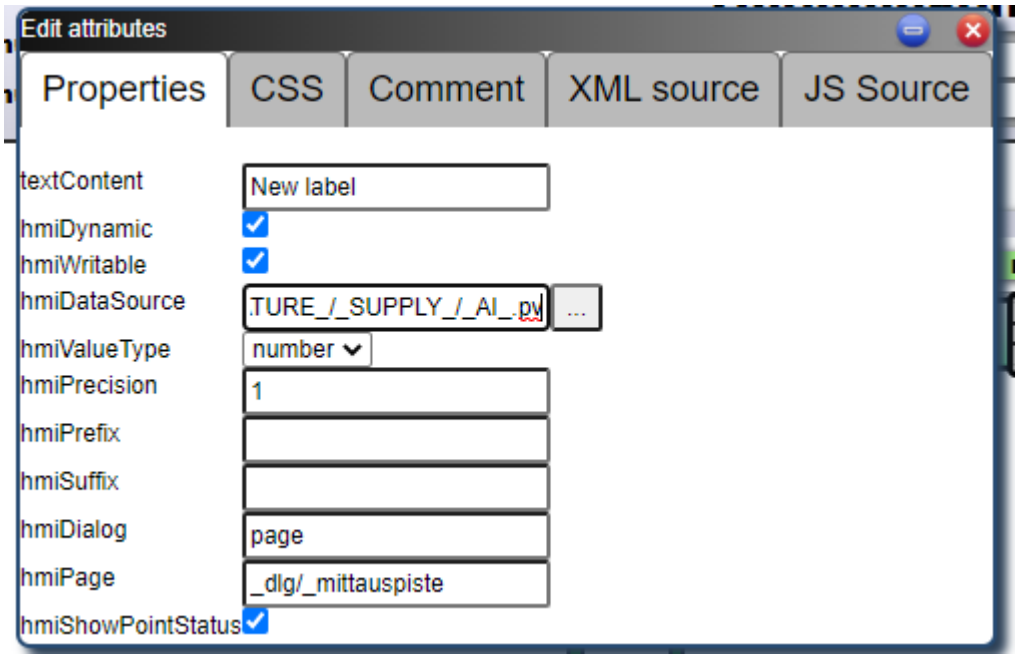
Ilmanvaihtokoneiden asetussivun automaattinen luominen on tällä hetkellä haasteellista, sillä Actiwebissä ei ainakaan vielä pysty kokoamaan sivua pienemmistä paloista yhdistelemällä. Asetussivuja käytännössä pitäisi tehdä aina kaikille variaatioille omansa ja ohjelmallisesti valita oikea mallitiedosto. Tällä hetkellä asetussivu täytyy vielä tehdä käsin.

3.4 Tietokannan rakentaminen

Tietokantapisteiden nimeäminen tietyllä tavalla on tärkeää, jotta Automaattinen luominen voi toimia. Useissa järjestelmissä tietokannassa olevat pisteet ovat suoraan suunnitelmista saatuja pistetunnuksia, jotka voivat olla esimerkiksi muotoa RAKTUNNUS_VAKTUNNUS_JÄRJTUNNUS_LAITETUNNUS_TYYPPITUNNUS, eli rakennuksen tunnus, alakeskuksen tunnus, järjestelmätunnus, laitteen tunnus ja lopuksi I/O-pisteen tyyppitunnus. Tietokannan rakentaminen tällä tavalla kuitenkin hankaloittaa pisteiden automaattista tulkintaa.

Actiwebissä on mahdollista antaa tietokantapisteelle lähes rajaton määrä erinimisiä kenttiä, jolloin kaikki suunnitelmissa haluttu tieto voidaan upottaa pisteeseen itseensä, ja nimetä piste siten, että ohjelman on sitä helppo tulkita. Esimerkiksi ilmanvaihtokoneen tuloilman lämpötilamittauksen tietokantapisteen koko positio tageilla esitettynä on ”_HVAC_/_VENTILATION_/_AHUID_/_TEMPERATURE_/_SUPPLY_/_AI_”. Tästä ohjelman on helppo tulkita, että piste on tuloilman lämpötila, vaikka suunnittelija olisi käyttänyt aivan mielivaltaista laitepositiointia. Tietokantapisteen tunnus muodostuu pääjärjestelmästä _HVAC_, eli LVI-järjestelmät, alijärjestelmästä _VENTILATION_, eli ilmanvaihto ja _AHUID_, eli ilmanvaihtokoneen uniikki tunnus. Tämän perään lisätään laitetyyppi, esimerkissä _TEMPERATURE_, eli lämpötilamittaus, _SUPPLY_ sijoittaa mittauksen tuloilmakanavaan ja lopuksi _AI_ määrittää pisteen olevan analoginen sisääntulo, eli mittaus.

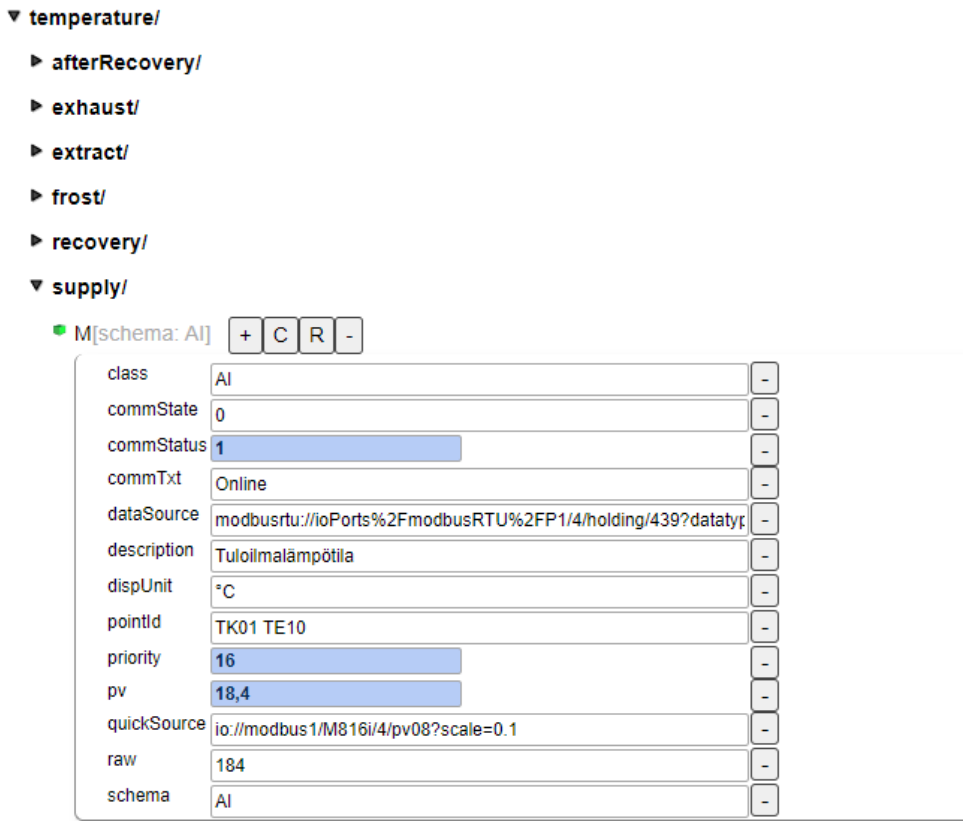
Kuvassa 14 näkyy grafiikkatyökalun *hmiDataSource*-kentässä tageista muodostettu tietokantapisteen tunnus, jota ei vielä ole ohjelmallisesti käännetty uuteen muotoon.



KUVA 8. Avainsanat mallitiedostossa

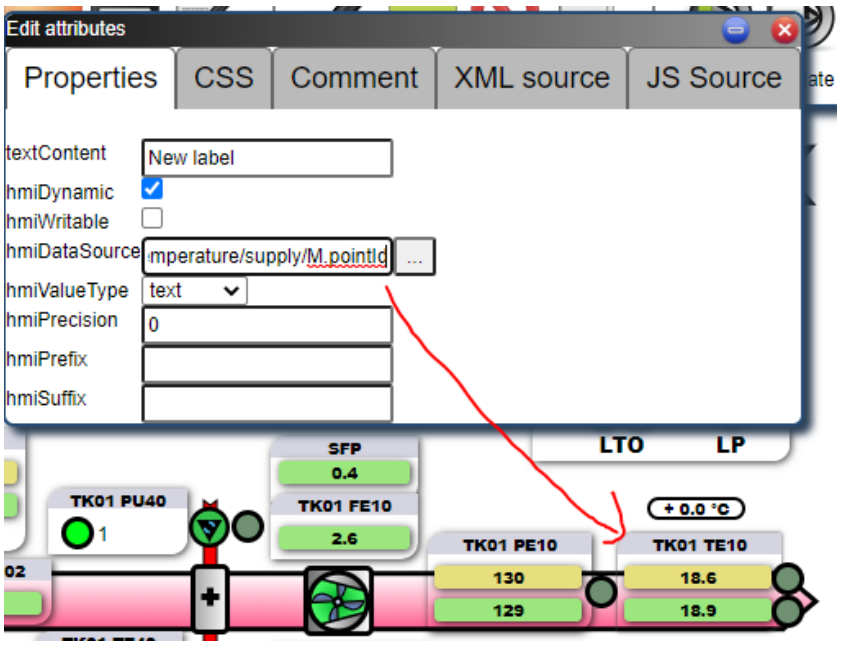
Alaviivamerkeillä esitetyt avainsanat korvataan ohjelmassa kuvassa 5 näkyvällä *replacement* kentän tekstillä. Näin siksi, että pisteiden nimeämiskäytäntöä voi

myöhemmin muuttaa rikkomatta ohjelman toiminnallisuutta. Yllä esitetty tuloilman lämpötila on ohjelman tulkkaamana ” hvac/ventilation/TK01/temperature/supply/M”. Tietokantapisteen sisälle on upotettu sen suunnitelmista saatu laitepositio *pointId*-kenttään (Kuva 15).



KUVA 9. Tuloilman lämpötilamittaus tietokannassa

Kuten ohjelmassa, myös grafiikkasivulla voidaan määrittää mikä kenttä tietokantapisteestä kuvassa näytetään. Yllä oleva tuloilman lämpötilamittaus saadaan pistetunnuksineen käyttöliittymägrafiikalle luomalla sille dynaaminen tekstielementti, joka sisältää *pointId*-kentän. Kuvassa 16 kyseinen tekstielementti on muotoiltu css-tyyleillä laatikoksi ja samaan laatikkoon on lisätty tuloilman lämpötilan mittaus sekä asetusarvo. Grafiikan muokkaustilassa pisteen yksikkö ei näy, mutta käyttäjän näkymässä yksikkö noudetaan pisteen *dispUnit* kentästä.



KUVA 10. Pistetunnuksen esittäminen grafiikalla

3.5 Tietokannan käsittely ohjelmassa

Prosessiohjelmat täytyy kirjoittaa siten, että tietokannassa käytettäviin tunnuksiin viitataan vain yhdessä paikassa, ja silloinkin tunnukset muodostetaan ohjelmallisesti avainsanojen avulla. Excelissä luotu avainsanatiedosto tallennetaan XML muodossa, jolloin se voidaan lukea prosessiohjelmaan järjestelmän XML ohjelmakirjaston avulla. Excelin kehittäjätyökaluista löytyy XML-apuohjelma, jolla taulukko saadaan muutettua itse tehdyn mallin mukaiseksi XML-tiedostoksi. Kun kuvan 13 taulukko muutetaan XML-muotoon, sen sisältö on kuvan 17 mukainen. XML-muotoon muuttamista varten on kuitenkin ensin tehtävä halutun mukainen XML-malli, jonka avulla Excel osaa yhdistää oikeat solut oikeiden XML-avaimien kohdalle. Malliin tehdään samat XML-avaimet, kuin lopulliseen XML-tiedostoon halutaan, tässä tapauksessa *item*, *tag*, *replacement* ja *description*. Excelin kehitystyökalujen XML-määrityksissä valitaan mitkä sarakkeet vastaavat mitäkin avainta, jonka jälkeen Excel osaa tuottaa oikeanlaisen tiedoston.

```
<item>
    <tag>_ROOTPATH_</tag>
    <replacement>hvac</replacement>
    <description>Juurihakemisto kaikille pisteille. </description>
</item>
<item>
    <tag>_HVAC_</tag>
    <replacement>hvac</replacement>
    <description>Juurihakemisto kaikille LVI pisteille</description>
</item>
<item>
    <tag>_ELECTRIC_</tag>
    <replacement>electric</replacement>
    <description>Juurihakemisto kaikille sähköpisteille</description>
</item>
<item>
    <tag>_VENTILATION_</tag>
    <replacement>ventilation</replacement>
    <description>Ilmanvaihto/Ilmanvaihtoverkosto</description>
</item>
```

KUVA 11. Avainsanat XML-muodossa

Actiwebin XML-ohjelmakirjasto mahdollistaa XML-muotoisen datan käsittelyn sellaisenaan, mutta sen käyttö on hieman mukavampaa, kun se muunnetaan ensin taulukkomuuttujaksi. Lua:n taulukoissa voi taulukon indeksin nimenä olla mitä tahansa, joten tässä tapauksessa on loogista nimetä se avainsanan mukaan, näin taulukosta löytää haluamansa tiedot helposti. Aliohjelma *request_taglibrary* kuvassa 18 muodostaa taulukon, jossa on sisällä kaksi taulukkoa *tags* ja *replacements*, sekä avainsanan mukaan indeksoituna korvaavat tekstit. Muuttujasta *tagXMLtable* voidaan lukea haluttu korvaava teksti viittaamalla korvattavaan avainsanaan, esimerkiksi tagXMLtable["_TEMPERATURE_"] Taulukot *tags* ja *replacements* voidaan antaa suoraan syötteenä grafiikkasivujen luontiohjelmaan, jolloin se osaa korvata tekstit grafiikkakuvapohjasta.

```
function request_taglibrary()
    tagXMLtable = {}
    tagXMLtable.tags = {}
    tagXMLtable.replacements = {}
    fname = "/opt/slc/lib/ta_lib/taglibrary.xml"
    taglibrary = Xml.loadAsLuaDOM (fname)
    if taglibrary ~= nil then

        for i, node in ipairs (taglibrary) do
            local tag = (node.tag and node.tag._text or node._name)
            local replacement = (node.replacement and node.replacement._text or nil)
            if replacement ~= nil then
                table.insert(tagXMLtable.tags,tag)
                table.insert(tagXMLtable.replacements,replacement)
                tagXMLtable[tag]=replacement
            end
        end
    else
        print ("no xml")
    end
    return tagXMLtable or {}
end
```

KUVA 12. XML tietojen luenta ohjelmaan

Tietokantaan viittaamista varten tietokantapisteiden tunnukset täytyy vielä ohjelmallisesti rakentaa, jotta niihin on helppo viitata ohjelmassa. Helpoin keino tähän on tehdä aliohjelma, jolla tunnukset tallennetaan taulukon sopivasti nimettyihin soluihin. Kuvassa 19 aliohjelma luo tunnukset lämmönjakoverkostolle. *Prefix* muuttuja on ainoa ei-kiinteä osa ohjelmaa. Se sisältää verkoston nimen uniikin osan, jotta useampi verkosto voi käyttää samaa ohjelmakoodia. Tietokantapisteeseen voi nyt viitata ohjelmassa yksinkertaisesti käyttämällä *idTable* taulukon kenttiä koko tietokantatunnuksen sijaan, kuten kuvan 19 ehtolauseessa, joka tarkistaa pumpun olemassaolon.

```
--Lämmönjaon pisteiden nimeämiskäytäntö
function f_generateHeatingIds(input)
    prefix = input.networkPath
    local idTable =
        {
        outTemp = tagXMLtable["_ROOTPATH_"].."/"..tagXMLtable["_OUT_"].."/"..tagXMLtable["_TEMPERATURE_"],
        supplyTemp = prefix..tagXMLtable["_TEMPERATURE_"].."/"..tagXMLtable["_SUPPLY_"],
        returnTemp = prefix..tagXMLtable["_TEMPERATURE_"].."/"..tagXMLtable["_RETURN_"],
        pump = prefix..tagXMLtable["_PUMP_"],
        pressure = prefix..tagXMLtable["_PRESSURE_"],
        diffPressure = prefix..tagXMLtable["_DIFFPRESSURE_"],
        valve = prefix..tagXMLtable["_VALVE_"],
        }
        -- Varmistetaan, että pumppua ei turhaan lisätä hälytyspisteisiin, jos pumppua ei ole.
        if not(Data.exists(idTable.pump.."/"..tagXMLtable["_DI_"])) then
            idTable.pump = ""
        end
    return idTable

end
```

KUVA 13. Lämmönjaon tietokantanimet

3.6 Automaattinen järjestelmän tunnistaminen

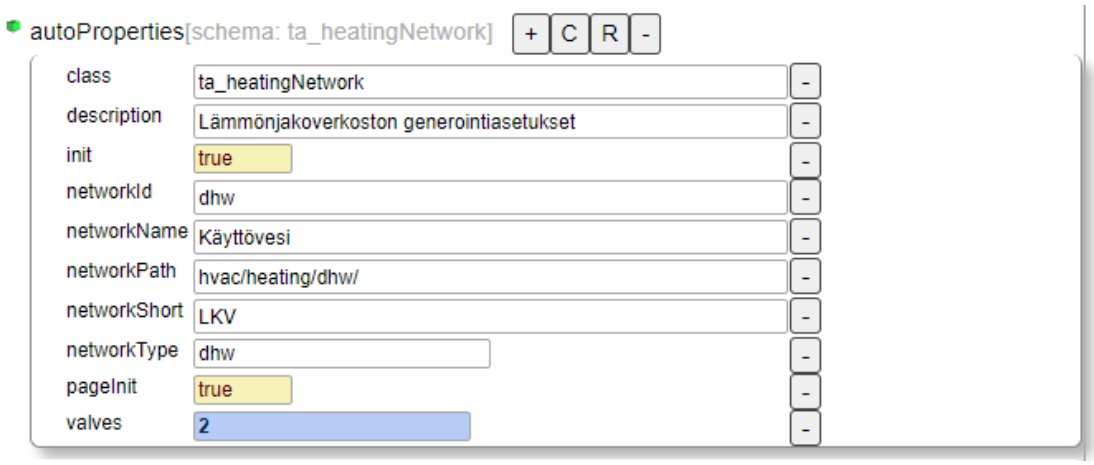
Koska järjestelmän tietokanta on luotu kiinteästi määritellyllä tavalla, voidaan järjestelmä ohjelmallisesti tunnistaa tietokannan rakenteen perusteella. Järjestelmän luonti toteutetaan kaksivaiheisesti, jotta ohjelmoija voi tarkistaa automaattisen tunnistuksen tietokannasta ja tehdä tarvittavat muutokset ennen grafiikkasivujen ja ohjelman luontia. Aliohjelma *initHeatSystem* kuvassa 20 etsii tietokannasta *ta_heatingNetwork* luokan tietokantapisteitä ja tekee niistä taulukon.
Taulukkoon lisätään kaikki uniikit verkostot, joilla ei vielä ole kyseisen tyypin mukaista pistettä olemassa.
Tämän jälkeen kutsutaan aliohjelmaa, joka luo käyttöliittymägrafiikat ja ohjelmat perustuen *ta_heatingNetwork*-pisteen mukaisiin asetuksiin, sekä lisää kyseisen pisteen kaikille niille verkostoille, joilta se aiemmin puuttui.
Ohjelma suoritetaan aina yhden kerran järjestelmän käynnistyessä uudestaan, eli uuden verkoston lisäämisen jälkeen tarvitaan kaksi uudelleenkäynnistystä, jotta se alkaa toimimaan.

```
function initHeatSystem()
    local heatNetworkList = Data.list ("*","class == 'ta_heatingNetwork'")
    -- Käydään läpi olemassa olevat lämmönjaon verkostot.
    -- Haetaan vielä uudet uniikit lämmitysverkostot tietokannasta.
    local idprefix = tagLibrary["_HVAC_"].."/"..tagLibrary["_HEATING_"].."/"
    local datalist = Data.list(idprefix.."*")
    -- Käydään datalistasta läpi uniikit verkostonimet ja luodaan niistä taulukko
    for i, v in ipairs (datalist) do
        v = string.match(v, idprefix..'.-/')
        if not(heatNetworkList[v] == true) then
            heatNetworkList[v] = true
            table.insert (heatNetworkList, v)
        end
    end
    if #heatNetworkList > 0 then
        f_generateHeatingNetwork(heatNetworkList)
    end
    return true
end
```

KUVA 14. Lämmönjakoverkoston tunnistaminen tietokannasta

Tietokantapisteen luokka *ta_heatingNetwork* on järjestelmään itse luotu tietokantapisteen tyyppi, eli *schema*, johon voidaan määrittää tietokantapisteeseen aina lisättävät tietokentät ja niille oletusasetuksia. Kuvassa 21 on käyttövesiverkoston

automaattisesti luotu *ta_heatingNetwork* tyyppinen tietokantapiste. Kentistä ohjelma tietää valita mallitiedostoista käyttövesiverkostoa esittävän taustakuvan kahdella venttiilillä, korvata kuvassa näkyvät tekstit, ja luoda verkostolle asetusarvot ja säätimet, joita se tarvitsee. *Init* ja *pageInit*-kentät kertovat ovatko sivu ja asetukset jo luotu. Asettamalla kentän false-tilaan, saadaan sivu tai oletusasetukset luotua uudestaan. Pisteen kenttä *networkPath* sisältää tiedon, jota kuvan 19 aliohjelma käyttää *prefix* muuttujassa.



KUVA 15. Automaattisesti luodut käyttövesiverkoston parametrit

3.7 Liitännäisillä laajennettavat ohjelmat

Samaa prosessiohjelmaa voi yleensä käyttää kaikissa samanlaisissa järjestelmissä, mutta joskus on tarvetta tehdä poikkeuksia ohjelman toimintaan. Ei ole kannattavaa tehdä omia perusohjelmia jokaista poikkeusta varten, mutta poikkeuksien lisääminen olemassa olevaan ohjelmaan täytyy olla mahdollista.
Esimerkiksi ilmanvaihtokone ei yleensä muutu perustoiminnaltaan mihinkään, mutta kone voidaan haluta vaikkapa pysäyttää erikoistilanteissa.
Kuvan 22 aliohjelma *f_lisaHalytys* lisää minkä tahansa hälytyksen tai tilatiedon ilmanvaihtokoneen hälytystilatekstin tietokantapisteeseen.

```
function f_lisaHalytys(id,text,invert)
    -- Koneen pysäyttäviä hälytyksiä voi lisätä taulukkoon kutsumalla tätä funktiota
    if id ~= nil and Data.exists(id) then
        for _,v in ipairs(haly_tt_taulukko) do
            if v.id == id then
                return true
            end
        end
        local invert = invert or false
        table.insert(haly_tt_taulukko,{id=id,text=text,invert=invert})
        local hairiotaulu = Data.get(IVKone.HalytysTT)
        if hairiotaulu ~= nil then
            for _,v in ipairs (hairiotaulu.stateTexts) do
                if v == text then return true end
            end
            table.insert(hairiotaulu.stateTexts,text)
            hairiotaulu.numStates = hairiotaulu.numStates + 1
            Data.set(IVKone.HalytysTT,{stateTexts=hairiotaulu.stateTexts,numStates=hairiotaulu.numStates})
        end
    end
end
```

KUVA 16. Aliohjelma Ilmanvaihtokoneen pysäyttävien hälytysten lisäämiseen

Tilatekstin tietokantapiste ensin luetaan ”Data.get”-komennolla, jonka jälkeen luettuun tietoon lisätään uusi hälytys aliohjelmakutsun mukaisesti antamalla seurattavan tietokantapisteen tunnus ja grafiikalla näytettävä tilateksti. Invert-muuttujalla kerrotaan aliohjelmalle, että täytyykö tietokantapisteen tilatiedon olla päällä vai pois, jotta kone pysäytetään. Muuttunut tieto kirjoitetaan takaisin tietokantaan ”Data.set”-komennolla. Ehtolauseilla varmistetaan, että hälytys lisätään listaan vain yhden kerran.

Tämän jälkeen kyseinen hälytys voidaan huomioida ilmanvaihtokoneen pysäytystä käsittelevässä aliohjelmassa ja hälytykselle määritetty tilateksti nähdään ilmanvaihtokoneen grafiikkakuvassa. Aliohjelma toimii näin laajennuksena kiinteästi määrätyille koneen pysäyttäville hälytyksille ja mahdollistaa poikkeustapausten lisäämisen muuttamatta pääohjelman toimintaa.

4 VALMIIN JÄRJESTELMÄN TESTAUS

Valmista järjestelmää testattiin melko perinteisessä kerrostaloprojektissa Tampereen Härmälänrannan alueella KOy Konttilukinkatu 11:ssa. Kyseessä oli kahden alakeskuksen projekti, jossa toinen alakeskuksista sijaitsee lämmönjakohuoneessa ja toinen Ilmanvaihtokonehuoneessa. Lämmönjakohuoneen alakeskuksessa on suurin osa projektin sähköpisteohjauksista ja Ilmanvaihtokonehuoneessa on käytännössä vain ilmanvaihtolaitteita. Urakkaan kulunutta aikaa on hankala verrata toisten urakoiden kanssa, sillä kaikki urakat ovat yksilöitä, mutta malliprojektin toteutus ohjelmointityön aloittamisesta testausten aloittamiseen kesti kokonaisuudessaan neljä työpäivää.

4.1 Kytkentäsuunnitelmasta tietokantaan

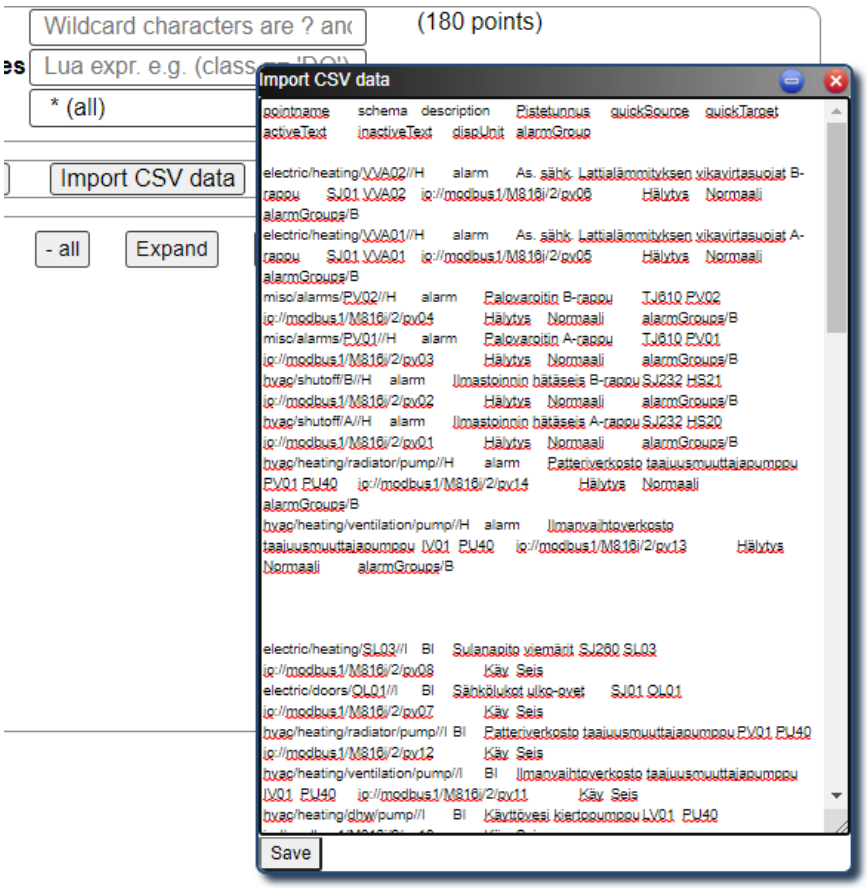
Actiweb on järjestelmänä melko tuore, eikä kaikki siinä käytettävät työkalut ole Bithousen itsensä tekemiä.
Koska tietokannan voi järjestelmään syöttää csv-muotoisena, on mikä tahansa taulukkolaskentaohjelma hyvä siihen tarkoitukseen. Käyttämällä Exceliä saadaan makrojen ja kaavojen avulla rakennettua välttävä työkalu tietokannan luontiin.

Kuvassa 23 on lämmönjakohuoneen kytkentäluettelon Actiwebin käyttöliittymään kopioitava osuus Excelistä.
Sarakkeiden nimet määrittävät mihin tietokantapisteen kenttään kirjoitetaan ja vasemmanpuoleisin sarake on tietokantapisteen nimi, joka on koottu ketjuttamalla erillisistä soluista. Nämä solut täytyy toistaiseksi vielä täyttää käsin, mutta järjestelmän kehittyessä on tarkoitus saada nekin vähintään puoliautomaattisiksi.

pointname	schema	description	pointId	quickSource	quickTarget	activeText	inactiveText	dispUnit
hvac/heating/radiator/pump/H	alarm	taajuusmuuttajapumppu	PV01 PU40	io://modbus1/M816i/2/pv14		Hälytys	Normaali	
hvac/heating/ventilation/pump/H	alarm	taajuusmuuttajapumppu	IV01 PU40	io://modbus1/M816i/2/pv13		Hälytys	Normaali	
hvac/heating/radiator/pump/I	BI	taajuusmuuttajapumppu	PV01 PU40	io://modbus1/M816i/2/pv12		Käy	Seis	
hvac/heating/ventilation/pump/I	BI	taajuusmuuttajapumppu	IV01 PU40	io://modbus1/M816i/2/pv11		Käy	Seis	
hvac/heating/dhw/pump/I	BI	Käyttövesi kiertopumppu	LV01 PU40	io://modbus1/M816i/2/pv10		Käy	Seis	
hvac/heating/radiator/pump/O	BO	taajuusmuuttajapumppu	PV01 PU40		io://modbus1/M816i/1/pv15	Käy	Seis	
hvac/heating/ventilation/pump/O	BO	taajuusmuuttajapumppu	IV01 PU40		io://modbus1/M816i/1/pv14	Käy	Seis	
hvac/heating/district/supply/M	AI	Kaukolämpö menolämpötila	KL01 TE40	io://modbus1/M816i/2/pv16?scale=				°C
hvac/heating/radiator/return/M	AI	Patteriverkosto paluulämpötila	PV01 TE41	io://modbus1/M816i/3/pv16?scale=				°C
hvac/heating/radiator/supply/M	AI	Patteriverkosto menolämpötila	PV01 TE40	io://modbus1/M816i/3/pv14?scale=				°C
hvac/heating/ventilation/pressure/M	AI	Ilmanvaihtoverkosto paine	IV01 PE40	io://modbus1/M816i/3/pv12?scale=0.0006				bar
hvac/heating/ventilation/diffPressure/M	AI	Ilmanvaihtoverkosto paine-ero	IV01 PDIT40	io://modbus1/M816i/3/pv10?scale=0.025				kPa
hvac/heating/ventilation/return/M	AI	paluulämpötila	IV01 TE41	io://modbus1/M816i/3/pv08?scale=				°C
hvac/heating/ventilation/supply/M	AI	menolämpötila	IV01 TE40	io://modbus1/M816i/3/pv06?scale=				°C
hvac/heating/dhw/supply/M	AI	Käyttövesi menolämpötila	LV01 TE40	io://modbus1/M816i/3/pv04?scale=				°C
hvac/heating/district/return/M	AI	Kaukolämpö paluulämpötila	KL01 TE41	io://modbus1/M816i/3/pv02?scale=				°C
hvac/heating/radiator/diffPressure/M	AI	Patteriverkosto paine-ero	PV01 PDIT40	io://modbus1/M816i/4/pv04?scale=0.025				kPa
hvac/heating/radiator/pressure/M	AI	Patteriverkosto paine	PV01 PE40	io://modbus1/M816i/4/pv02?scale=0.0006				bar
hvac/heating/radiator/pump/A	AO	taajuusmuuttajapumppu	PV01 PU40		io://modbus1/M816i/4/pv16?scale=0.1			%
hvac/heating/ventilation/pump/A	AO	taajuusmuuttajapumppu	IV01 PU40		io://modbus1/M816i/4/pv14?scale=0.1			%
hvac/heating/radiator/valve/stage1/A	AO	Patteriverkosto venttiili	PV01 TV40		io://modbus1/M816i/4/pv12?scale=0.1			%
hvac/heating/ventilation/valve/stage1/A	AO	Ilmanvaihtoverkosto venttiili	IV01 TV40		io://modbus1/M816i/4/pv10?scale=0.1			%
hvac/heating/dhw/valve/stage2/A	AO	Käyttövesi venttiili 2	LV01 TV41		io://modbus1/M816i/4/pv08?scale=0.1			%
hvac/heating/dhw/valve/stage1/A	AO	Käyttövesi venttiili 1	LV01 TV40		io://modbus1/M816i/4/pv06?scale=0.1			%

KUVA 17. Import csv työkaluun liitettävä osuus

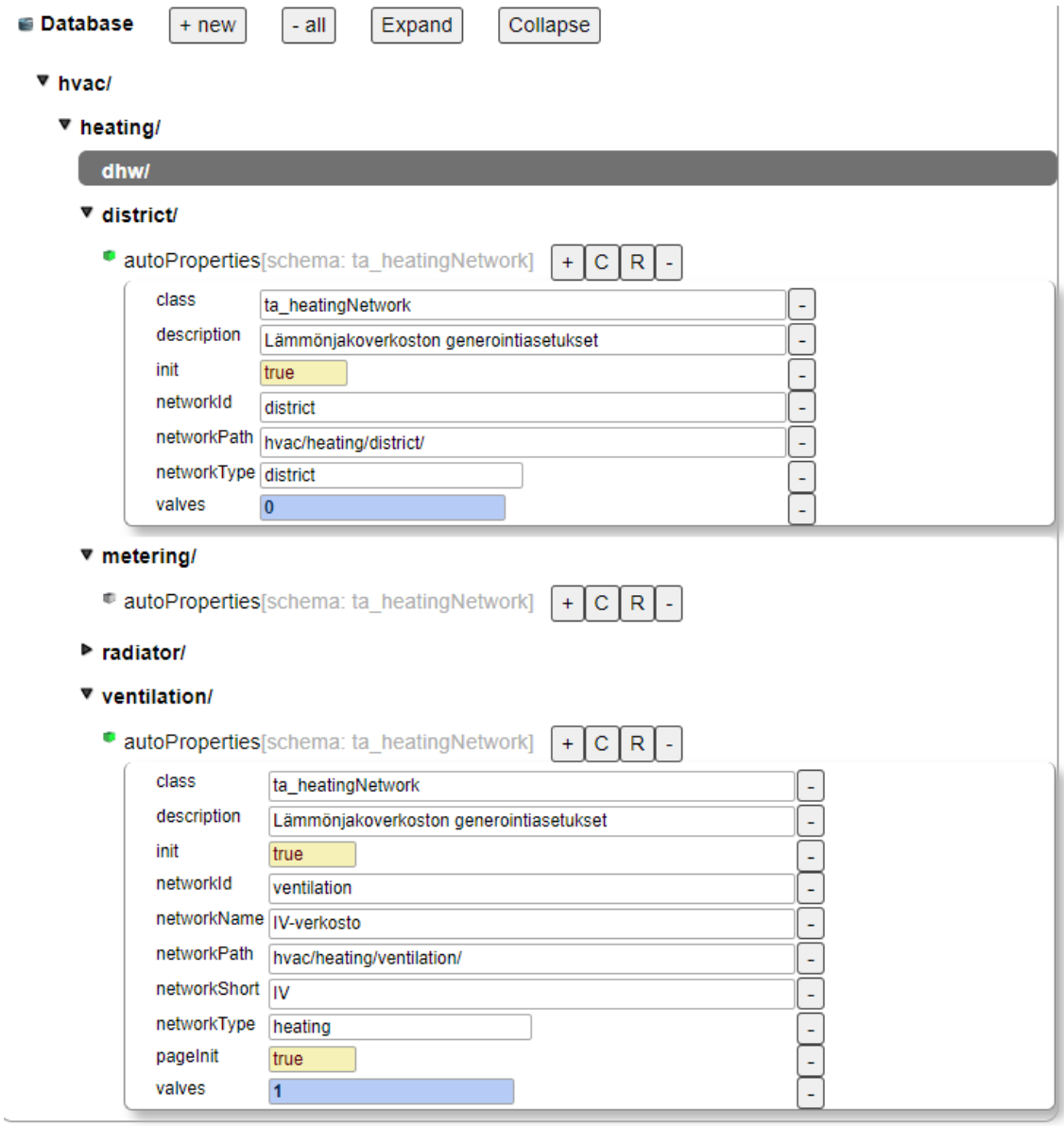
Keltaisella merkitty alue valitaan ja kopioidaan leikepöydälle. Leikepöydältä se liitetään Actiwebin tietokantanäkymästä löytyvään *Import CSV Data* ikkunaan (Kuva 24) ja painetaan tallennuspainiketta. Tietokanta on heti käytettävissä, mutta modbus-kommunikointi alkaa vasta ohjelman uudelleenkäynnistämisen jälkeen. Tämä johtuu siitä, että Actiweb järjestelmässä modbus-rekisterien luentaa optimoidaan siten, että peräkkäisiä rekisterejä pyritään lukemaan yhdellä kutsulla. Rekisterikutsut niputetaan sovelluksen uudelleenkäynnistyksessä. Tällä tavalla voidaan vähentää liikennettä väylässä.



KUVA 18. Tietokannan luonti csv-tiedoista

4.2 Tietokannan automaattinen tulkinta

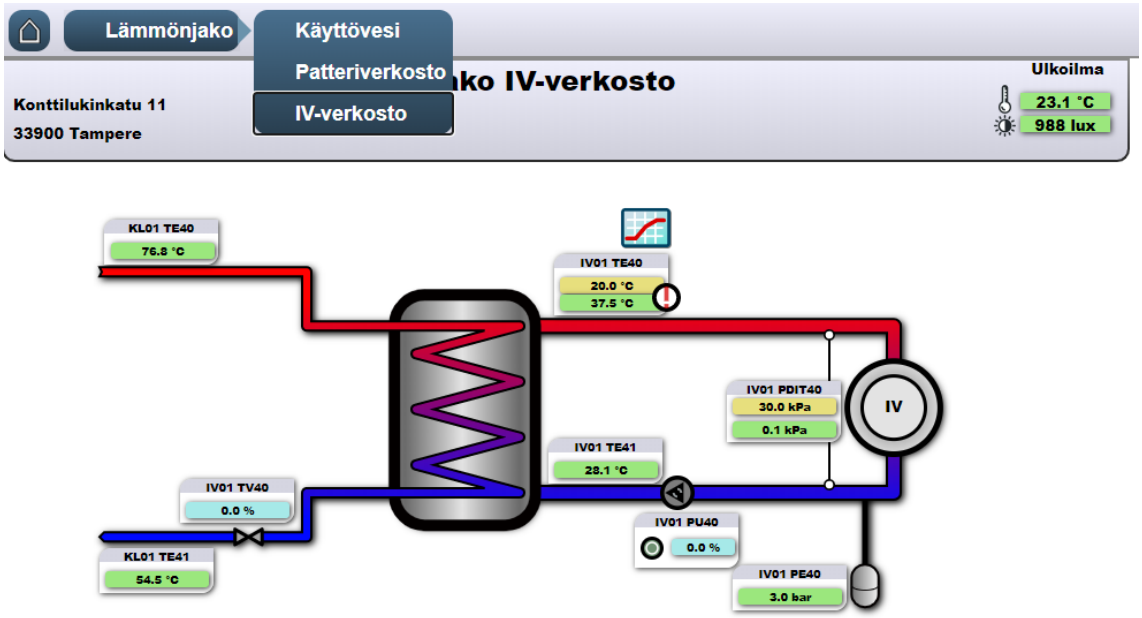
Ohjelman uudelleenkäynnistyksen yhteydessä verkostojen automaattisen luonnin ohjelma luo verkostoille *autoProperties* nimisen kentän. josta voidaan tarkistaa asetusten oikeellisuus ennen grafiikkasivujen luontia (Kuva 25). Järjestelmä on löytänyt polkurakenteen alta kansiot *district*, eli kaukolämpö ja *metering,* mittarointi. Näille ei ole ohjelmassa olemassa kuvamallia, joten ohjelma ei edes yritä luoda niille grafiikkasivua. Verkostot *dhw, ventilation* ja *radiator*, eli käyttövesi, ilmanvaihtoverkosto ja patteriverkosto luodaan niille ohjelmassa määritettyjen oletusarvojen mukaisiksi ja niille tehdään automaattisesti grafiikkasivu.



KUVA 19. Tietokannasta löydetyt verkostot

4.3 Grafiikkasivun viimeistely

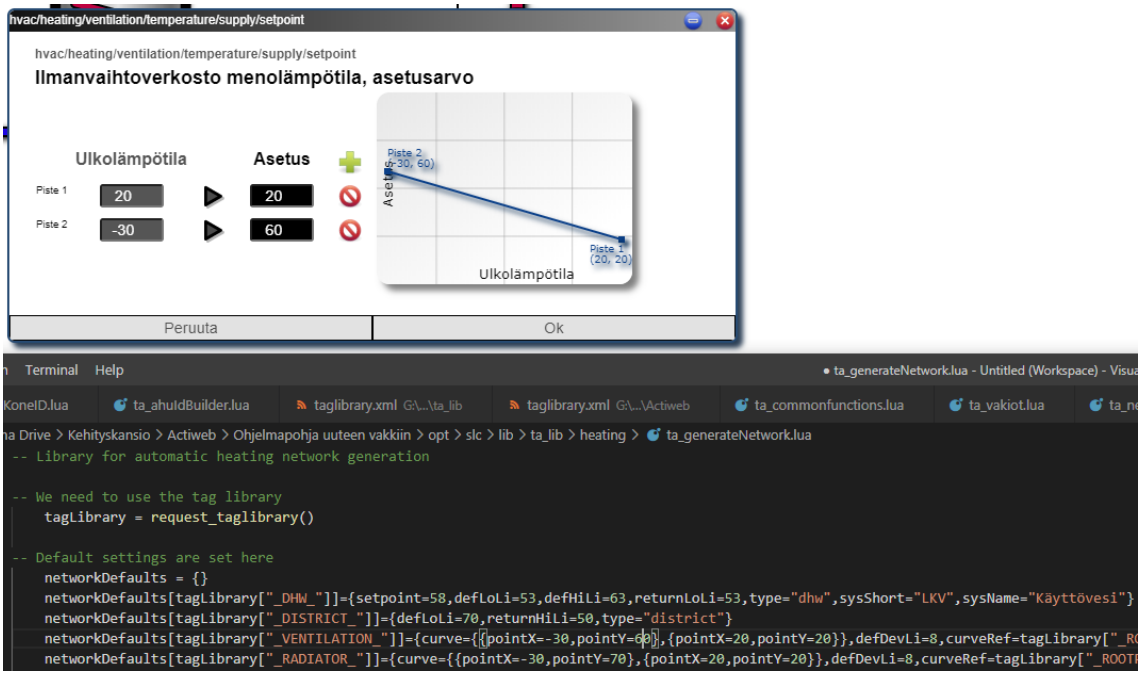
Toisen uudelleenkäynnistyksen jälkeen linkkilistalle ilmestyy linkit lämmönjako-huoneeseen ja sen alasivuihin (Kuva 26). Mallikohteessa sivulle ei jäänyt mitään ylimääräistä poistettavaa, mutta esimerkiksi verkostosta palaavan veden lämpö-tilamittausta ei joka kerta ole. Toistaiseksi ylimääräiset asiat täytyy vielä poistaa käsin sivun muokkaustilassa, mutta järjestelmän kehittyessä siitäkin pitäisi päästä eroon.



KUVA 20. Luotu grafiikkasivu ja linkit

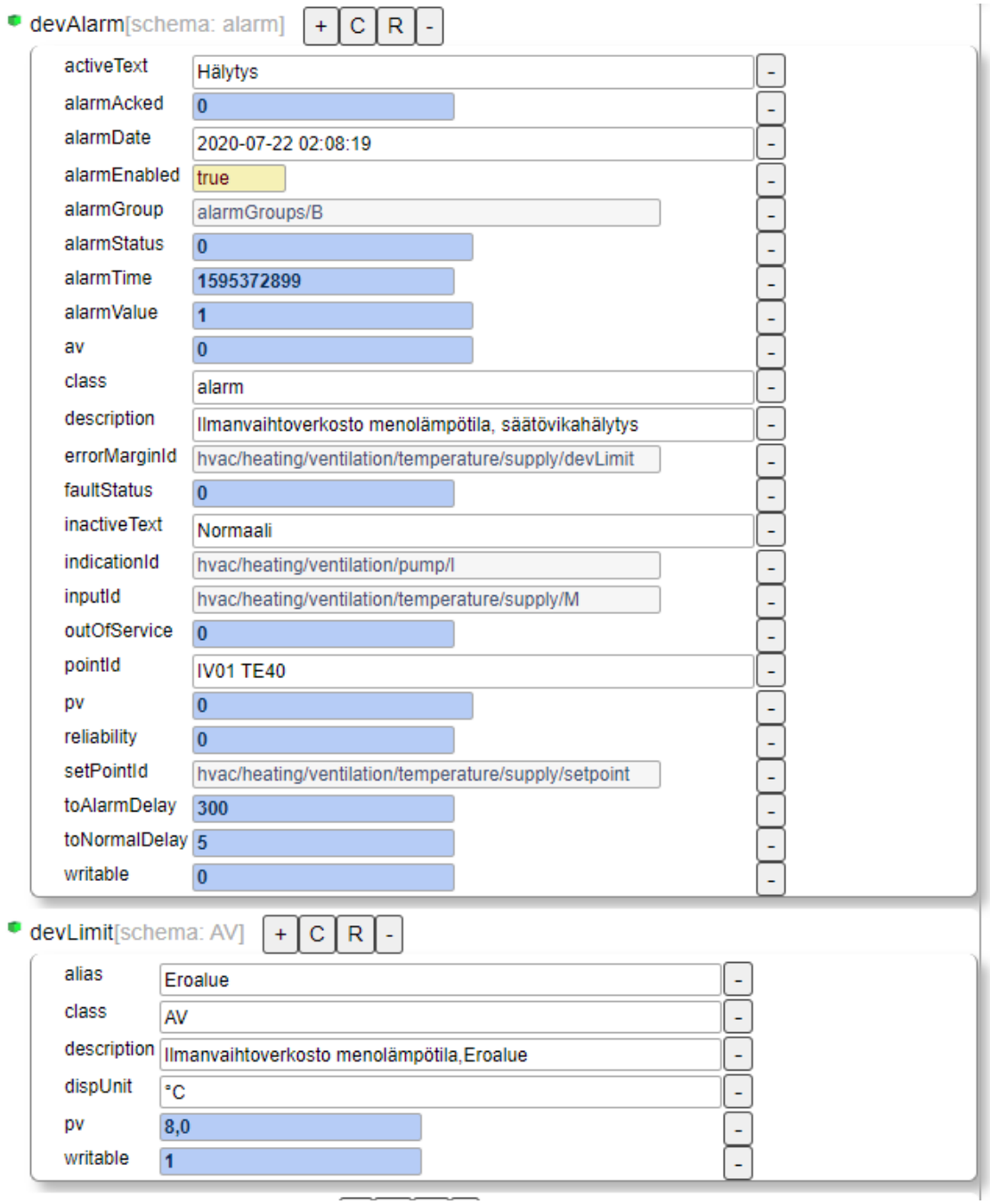
4.4 Viimeistely

Lopuksi on vielä syytä tarkistaa, että muut oletusasetukset, kuten verkoston säätökäyrä on valikoitunut oikein esiasetetuista oletuksista (Kuva 27). Asetusarvot täytyy kuitenkin vielä vaihtaa vastaamaan suunnitelmia. Verkoston säätö alkaa toimimaan heti uudelleenkäynnistyksen jälkeen, mutta viritysparametrit eivät välttämättä heti ole hyvät. Säädön virityksen automatisointia järjestelmässä ei toistaiseksi ole.



KUVA 21. Säätökäyrän oletusarvot

Kaikki ohjelmalliset hälytykset käyttävät Actiwebin omia kirjastoja, ja niistä täytyy tarkistaa, että kaikki parametrit ovat tulleet oikein (Kuva 28). Kentästä *errorMarginId* järjestelmä tunnistaa, että kyseessä on säätövikahälytys. Kenttä *indicationId* kertoo minkä tietokantapisteen täytyy olla aktiivisessa tilassa, jotta hälytys voi aktivoitua. Tässä tapauksessa kyseessä on ilmanvaihtoverkoston pumppu. Asetusarvo, johon mittausta verrataan, luetaan *setPointId* kentässä annetusta pisteestä. Itse mittaus on *inputId* kentässä



KUVA 22. Ilmanvaihtoverkoston säätövikahälytys

Lopuksi kaikki prosessiohjelmat täytyy testata. Vaikka ohjelma tulee automaattisesti generoituna samanlaisena joka kerralla, on testaaminen ainoa keino varmistua järjestelmän toiminnasta. Lämmönjakohuoneen ohjelmassa on säätöjen ja hälytysten lisäksi vain pumppujen kesäpysäytysohjelma ja ulkolämpötilan keskiarvon laskenta. Ilmanvaihtokoneiden ohjelmassa on useita turvatoimintoja testattavaksi, esimerkiksi jäätymissuojan toiminta, Ilmanvaihdon hätäpysäytys ja tulopuhaltimen pumppulukitus. Laajemmista järjestelmistä on syytä tehdä pöytäkirja, eikä testaamista voi laiminlyödä, vaikka omaan ohjelmaan luottaisi täysin.

5 POHDINTA

Tämän työn tavoitteena oli saada luotua ohjelmakirjasto ja ohjelmointimenetelmät, joilla tulevien projektien ohjelmointityöhön käytettävää aikaa saadaan vähennettyä. Suurin osa työhön käytetystä ajasta kului ohjelmointitapojen suunnitteluun, mikä onkin toimivan toteutuksen kannalta tärkeää. Hyvin suunnitellun tietokantamallin, ja siihen sopivan käyttöliittymägrafiikan yhteen liittäminen ohjelmallisesti ei lopulta ollut työlästä. Prosessiohjelmissa ainoa käytännön ero perinteisempään rakennusautomaation ohjelmointiin oli se, että mihinkään I/O-pisteeseen ei viitata suoraan, vaan kaikki kyselyt ohjataan tietokantaan avainsanalistan avulla. Avainsanojen avulla ohjelmointi oli hankalin vaihe omaksua, mutta siitä saavutettu tietokannan muokattavuus on jatkokehityksen kannalta tärkeää.

Valmiin ohjelman testausvaiheessa ensimmäinen oleellinen havainto ohjelmakirjaston toimivuudesta oli se, kuinka vähän ohjelmointityötä projektiin jäi kytkentäsuunnitelmien tekemisen jälkeen. Projektissa säästetty aika osoittautui yllättävän suureksi tyypillisessä kerrostalourakassa. Mallikohteena toimineen KOy Konttilukinkatu 11:n kahden alakeskuksen järjestelmän yhdellä ilmanvaihtokoneella ja lämmönjakohuoneella erillispisteineen pystyi ottamaan työmaalla käyttöön kahden päivän ohjelmointityön jälkeen, kun aliurakoitsija oli tehnyt asennukset työmaalla valmiiksi. Suurin osa työmaalla tapahtuneesta ohjelmoinnista oli joko lisätöitä tai varaukseksi muuttuneita pisteitä. Mallikohteen valmistumisen jälkeen ohjelmaa on käytetty muihinkin projekteihin ja useimmissa tapauksissa Excelissä syötettävät tietokantapisteiden nimet ovat suurin työvaihe.

Vaikka järjestelmässä on vielä paljon kehitettävää, on se silti hyvin lähellä sitä, mikä työn tarkoitus oli. Tyypillisen kerrostalon rakennusautomaatio-ohjelmoinnin pystyy järjestelmällä tekemään suunnilleen samalla kerralla kytkentäsuunnitelmien kanssa. Excel-tiedosto, jolla kytkentäluettelot tehdään, vaatii vielä kehitystä, että päästään eroon mahdollisimman suuresta määrästä käsin tehtävää työtä. Järjestelmällä on erittäin paljon potentiaalia vähentää ohjelmoinnin työn ja ohjelmointivirheiden määrää, kun sitä vielä jatkojalostetaan nykyisestä. Ainakin tällä hetkellä vaikuttaa siltä, että järjestelmällä saadaan toivottua kilpailuetua merkittävästi. Järjestelmä on kuitenkin luonteeltaan sellainen, että se oikeas-

taan koskaan valmistu, sillä tekniikka kehittyy jatkuvasti ja modernin järjestelmän täytyy pysyä kehityksen perässä. Seuraava kehitysaskel voisi liittyä loppudokumentoinnin automatisointiin, sillä luovutusvaihetta järjestelmä ei vielä helpota ollenkaan.

LÄHTEET

AWM-Automaatiojärjestelmä. Verkkoaineisto. https://bithouse.fi/automaatio/awm-automaatiojarjestelma/. Luettu 17.6.2020

Actiweb käyttöliittymäohjelmointi. Verkkoaineisto. Bithouse Oy https://bithouse.fi/wp-content/uploads/2019/11/ActiWeb_grafiikka_V1.pdf Luettu 29.6.2020

Actiweb ohjelmointiopas. Verkkoaineisto. Bithouse Oy https://bithouse.fi/wp-content/uploads/2019/11/ActiWeb_ohjelmointi_V1.pdf Luettu 29.6.2020

Liedes R., 2017, ST-Käsikirja 22, Kiinteistöjen valvontajärjestelmät. Sähkötieto ry

Liedes R., 2018, ST-Käsikirja 17, Rakennusautomaatiojärjestelmät. Sähkötieto ry

Ouflex Tool. Verkkoaineisto. Ouman Oy http://ouman.fi

Piikkilä V., 2017, ST-Käsikirja 21, Kiinteistöjen tiedonsiirtoväylät. Sähkötieto ry

Sahala A., Alilehto J., 2018, ST 711.15, Ohjelmistojen dokumentointi. Sähkötieto ry

Sahala A., 2016, ST-kortti 721.01, Taloteknisten järjestelmien käyttöliittymät. Sähkötieto ry

Sahlstén T., Kähkönen T., 2013, ST-kortti 736.00, Rakennusautomaatioprojektin hallinta. Sähkötieto ry

Sahlstén T., 2020, ST-kortti 711.04, Rakennusautomaatiourakan laadunvarmistus-, valvonta- ja vastaanottomenettelyohjeita. Sähkötieto ry

The RS-485 Desing Guide, Verkkoaineisto. Texas Instruments. https://www.ti.com/lit/an/slla272c/slla272c.pdf?ts=1603520173634. Luettu 24.10.2020