jamk.fi

Optroniikan testausjärjestelmä

Niko Hakala

Opinnäytetyö
Marraskuu 2017
Tekniikan ja liikenteen ala
Insinööri (AMK), automaatiotekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Hakala, Niko	Opinnäytetyö, AMK	16.11.2017
	Sivumäärä 29	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Optroniikan testausjärjestelmä

Tutkinto-ohjelma
Sähkö- ja automaatiotekniikka, Insinööri (AMK)

Työn ohjaaja(t)
Markku Ström

Toimeksiantaja(t)
Millog Oy, Optroniikka, Lievestuore

Tiivistelmä

Opinnäytetyön tarkoituksena oli kehittää olemassa olevasta automaattisesta typetyslaitteesta optroniikan testausjärjestelmä lisäämällä uusia ominaisuuksia ja parantamalla käyttöliittymää. Uusilla ominaisuuksilla pyrittiin automatisoimaan vielä manuaalisesti tehtäviä testauksia optroniikkalaitteille. Lisäksi kehittämistehtävän puitteissa selvitettiin, miten kyseinen optroniikan testausjärjestelmä voidaan liittää SQL-tietokantaan.

Optroniikan testausjärjestelmän ohjelmointi suoritettiin etätyönä ja testaus sekä käyttöönotto tehtiin Millog Oy:n Lievestuoreen optroniikkayksikössä. Optroniikan testausjärjestelmän tulee olla helposti laajennettavissa, joten kaikki ohjelmat tehtiin uusiksi parametroituja ohjelmia käyttäen. Sovellusten ohjelmointi ja käyttöliittymä ohjelmoitiin TIA Portal V13 ympäristössä.

Lopputulokseksi saatiin toimiva optroniikan testausjärjestelmä, joka helpottaa optroniikkalaitteiden testausta ja poistaa manuaalisesti suoritettavia työvaiheita. Järjestelmä on myös helposti laajennettavissa, mikäli useampia laitteita halutaan testata tulevaisuudessa ohjelman yhdellä suorituskerralla.

Avainsanat (asiasanat)
TIA Portal, HMI, Optroniikka

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Hakala, Niko	Bachelor's thesis	16.11.2017
		Language of publication:
	Number of pages 29	Permission for web publication: x

Title of publication

Optronics testing system

Degree Programme

Electrical and Automation Engineering

Supervisor(s)

Ström Markku

Assigned by

Millog Oy, Optronics, Lievestuore

Abstract

The aim of this thesis was to develop an optronics test system from an existing nitrogen purging device by adding new features and improving the user interface. The new features were aimed to automate the manual testing of optronics devices. In addition, it was useful to find out how the optronics test system can be linked to the SQL database.

The programming of the optronics test system was carried out as remote work. The testing and the introduction were carried out at the optronics department in Millog Oy. The optronics test system should be easily expanded in the future, therefore, all the programs were made by using parameterized programs. The applications and the user interface were programmed in the TIA Portal V13 environment.

In conclusion, this development work produced a practical optronics test system whereby the testing of optronics devices is easier and some of the manual work steps are no longer needed. The system can also be easily expanded if more than one device needs to be tested at a single execution of the program.

Keywords/tags (subjects)

TIA Portal, HMI, Optronics

Miscellaneous

Sisältö

Kuviot

Johdanto

Tässä opinnäytetyössä tavoitteena on kehittää olemassa olevasta typetyslaitteesta optroniikan testausjärjestelmä yhteistyössä tehtävän toimeksiantajan, Millog Oy:n, kanssa. Patria konserniin kuuluva Millog Oy on erikoistunut Puolustusvoimien asejärjestelmien, elektroniikan, optiikan, ajoneuvojen sekä suojeluvälineiden huoltoon ja kunnossapitoon. Millog Oy on myös Pohjoismaiden suurin optroniikka-laitteiden valmistaja. (Millog Oy 2017.) Yrityksessä on jo aiemmin kehitelty optroniikan testausjärjestelmää, ja aiheesta on tehty yksi opinnäytetyö. Valkonen (2010) on kirjoittanut opinnäytetyönsä typetyslaitteen automatiosointiin liittyen. Tässä opinnäytetyössä tarkoituksena on päivittää typetyslaitteen ohjelma ja käyttöliittymä sekä lisätä tarpeelliset sovellukset. Järjestelmän jatkokehittämisen kannalta työn teoriaosuudessa selvitetään, mitä vaaditaan optroniikan testausjärjes-telmän liittämiseen SQL-tietokantaan.

Työssä käytetään Siemensin logiikkaa. Toimeksiantajalla on tarvikkeet kahteen testausjärjestelmään, joten sain toisen järjestelmän logiikan sekä moduulit mukaan, mikä mahdollisti ohjelman tekemisen osittain etätyönä. Optroniikan testausjärjestelmän käyttöönotto tapahtui Millog Oy:n optroniikka-yksikössä Lievestuoreella. Optroniikan testausjärjestelmän rakentaminen ja kytkeminen kuuluvat toimeksiantajalle.

1 Optroniikan testausjärjestelmä

1.1 Optroniikka

Optroniikka kuvaa laitteiden ja järjestelmien ominaisuuksia. Käsitteessä yhdistyy optiikka, optomekaniikka ja optoelektroniikka. Optiikka tarkoittaa linssejä. Kun optiikka on yhdistetty mekaanisiin rakenteisiin, esimerkiksi tietokoneen osoitinlaite eli hiiri, puhutaan optomekaniikasta. Jos laite sisältää myös elektronisia komponentteja, esimerkiksi lasereita, kyseessä on optoelektroniikka-laite. Optroniikka-järjestelmissä hyödynnetään prosessoriteknologiaa ja niitä ohjataan tietokoneohjelmistoilla. Optroniikka-laitteita ovat esimerkiksi valonvahvistimet, lämpökamerat ja erilaiset tähtäinjärjestelmät. (Patria 2014.)

1.2 Optroniikan testausjärjestelmän toiminta

Optroniikan testausjärjestelmällä tarkoitetaan järjestelmää, jolla voidaan testata ja huoltaa optroniikka-laitteita. Tämän opinnäytetyön järjestelmällä optroniikka-laitteille voidaan suorittaa tiiveystesti, typetys, virranmittaus ja tarkistaa IR-valon toiminta. Tiiveystestissä laitteet paineistetaan typellä laitekohtaiseen raja-arvoon, jonka jälkeen odotellaan laitteelle määrätty pitoaika. Pitoajan jälkeen tarkastellaan laitekohtaisia tuloksia. Mikäli laitteen sisäinen paine on laskenut liikaan pitoajan kuluessa, tulee laitteen vuotokohta korjata, ja sen jälkeen tiiveystesti suoritetaan laitteelle uudestaan. (Koskinen 2017.)

Typetyksessä laitteelle suoritetaan ylipaine-alipaine huuhtelu, jonka tarkoituksena on poistaa laitteen sisältä normaalia ilmaa ja täyttää laite typellä. Laitteet täytetään typellä, jottei optroniikka-laitteen optiset osat huurru. Tämä perustuu siihen, että typen kastepiste on niin alhainen, ettei sitä saavuteta laitteen normaalissa käytössä. Typetys aloitetaan paineistamalla laite laitteen ylipaineen raja-arvoon. Paineistuksen jälkeen laitteen paine tyhjennetään avaamalla laitteen venttiili ja tyhjennysventtiili,

jonka kautta paine pääsee purkautumaan pois laitteesta. Paineen poistamisen jälkeen käynnistetään alipainepumppu ja alipaineistetaan laite laitekohtaiseen raja-arvoon. Kun alipaine on saavutettu, aloitetaan laitteen täyttäminen taas typellä ja paineistaan laite ylipaineen raja-arvoon. Jokaisessa vaiheessa on tärkeää, ettei laitteen sisäinen virtausnopeus kasva liian nopeaksi, jolloin laitteen sisäiset roskat voisivat lähteä liikkeelle ja kulkeutua optisille pinnoille, mikä haittaisi laitteen toimivuutta. Virtausnopeutta rajoitetaan paineistuksessa ja alipaineistuksessa säätöventtiilillä ja paineen tyhjennyksessä tyhjennysventtiilin linjaa kuristamalla. (Koskinen 2017.)

Virranmittauksella selvitetään laitteen virrankulutusta kahdessa eri vaiheessa. Ensimmäisessä vaiheessa IR-valo ei ole päällä, ja toisessa vaiheessa IR-valo on päällä. Molemmissa vaiheissa laitteella on eri raja-arvot, joiden sisällä laitteen virrankulutuksen täytyy olla, että laite toimii oikein. Virranmittauksella voidaan tarkastella myös IR-valon toimintaa. Kun IR laitetaan päälle, tietyn ajan kuluttua valonvahvistinputken pitäisi välähtää merkiksi siitä, että laite toimii oikein. Välähdyksen aikana laitteen virrankulutus laskee huomattavasti, jolloin virrankulutusta tarkastelemalla voidaan havaita välähdys katsomatta laitetta. Aikaisemmin välähdyksen tarkastelu on suoritettu katsomalla laitteen kuvaa ja odottamalla, tapahtuuko välähdystä vai ei. On mahdollista, että uudessa järjestelmässä tämä voidaan hoitaa automaattisesti, eikä työntekijän tarvitse enää odotella ja tarkkailla, tapahtuuko välähdystä laitteen kuvassa. (Koskinen 2017.)

2 Ohjelmointi

2.1 Human Machine Interface

Human Machine Interface (HMI) tarkoittaa käyttöliittymää, joka toimii ihmisen ja ohjelmoitavan logiikan rajapintana. Käyttöliittymän kautta voidaan ohjata ja käyttää laitetta, esimerkiksi syöttää ja valvoa lämpötilaa tai painetta, jota logiikan tulee ylläpitää prosessissa. Käyttöliittymä voi kertoa tekstin tai grafiikan muodossa laitteen tai järjestelmän tilasta ja toiminnoista. (Tzafestas & Tzafestas 2001.)

Käyttöliittymä voi olla yksinkertaisimmillaan painonappi, josta logiikan ohjaama toiminta käynnistetään ja sammutetaan, tai laajempi sovellus, kuten kosketusnäyttö, paneelikäyttöliittymä tai tietokoneella oleva valvomo-ohjelmisto. Kosketusnäyttöön voidaan ohjelmoida sovelluksen kannalta tarvittavat painikkeet ja toiminnot. Paneelikäyttöliittymä voi sisältää sekä kosketusnäytön että painonappeja, joita voidaan ohjelmoida sovelluksen käyttötarkoituksen mukaisesti. Tietokoneella oleva valvomo-ohjelmisto soveltuu hyvin jatkuvan prosessin ohjaukseen ja valvontaan. Silloin käyttöliittymän kautta saadaan tietoa prosessista, ja tiedon pohjalta tarvittaessa suoritetaan toimenpiteitä. Paneelikäyttöliittymien ja tietokone-valvomoiden etuna on monitoroinnin selkeys ja helppokäyttöisyys sekä muunneltavuus. (Kippo & Tikka 2008, 46.)

2.2 TIA Portal

TIA Portal on Siemensin ohjelmointityökalu, joka tulee sanoista Totally Integrated Automation. TIA Portal:ssa on yhdistetty kolme eri ominaisuutta, jotka ovat Simatic Step 7, Simatic WinCC ja Simatic StartDrives. TIA Portal:lla voidaan suorittaa logiikkaohjelmointi, käyttöliittymän suunnittelu ja taajuusmuuttajien ohjelmointi. TIA Portal -ohjelmointityökalun etuna on, että kaikki tarvittavat automaation suunnittelutoiminnot sijaitsevat samassa käyttöliittymässä. Vastaavasti yhtenäinen tietorakenne

nopeuttaa suunnittelutyötä ja vähentää virheiden mahdollisuutta. TIA Portal -ohjelmistoa käytetään Siemensin uusissa logiikoissa, kuten Simatic ET200SP -logiikassa. (Siemens 2017.)

2.3 Parametroitu ohjelmointi

Parametroidussa ohjelmoinnissa käytetään muodollista parametria absoluuttisen osoitteen sijaan. Absoluuttista osoitetta tarvitaan ainoastaan pääohjelman kutsussa, jossa se määritetään todelliseen parametriin. (Ström 2017.) Muodollista parametria käytettäessä ohjelmakoodia on helpompi lukea ja ymmärtää. Muita hyötyjä ovat, ettei kommentteja tarvitse kirjoitella jokaiseen muuttujaan, vaan muuttujaa kuvaava nimi kertoo yleensä riittävästi. Lisäksi tiedon käyttö on nopeampaa. Muutettaessa muuttujan nimeä tai osoitetta päivittyy se automaattisesti kaikkialle, missä sitä on käytetty. (Siemens 2014.)

Absoluuttisessa osoitteessa osoite kirjoitetaan esimerkiksi muotoon %I16.4, joka tarkoittaa Input 16.4. Vastaavasti %IW4 on Input Word 4. Absoluuttisen osoitteen tunnistaa osoitteen prosenttimerkistä. Muodollinen parametri voisi olla esimerkiksi ”Motor” tai ”Value”, jolloin muuttuja on helposti luettavissa ja ymmärrettävissä, eikä lisäkommentteja tarvita. Myös data block:n osoittaminen on helpompaa muodollista parametria käytettäessä, esimerkiksi ”Motor”. Value tarkoittaa Motor data block:a, ja Value on Motor data block:n muuttuja. Absoluuttista osoitetta käytettäessä osoite voisi olla esimerkiksi %DB1.DBX1.0. Absoluuttinen osoite ei kerro käyttäjälle mitään kyseisestä osoitteesta, ja voi siksi aiheuttaa virheitä helpommin kuin muodollinen parametri. (Siemens 2014.)

Muodollista parametria käytettäessä ohjelman voi monistaa helpommin, kun tarkkoja absoluutti osoitteita ei ole käytetty. Esimerkiksi funcktion block, joka on tehty muodollisia parametreja käyttäen, voidaan monistaa pääohjelmaan useita kertoja. Pääohjelmassa voidaan määrittää jokaiselle monistetulle ohjelmalle omat absoluutti osoitteet. (Siemens 2014.) Ohjelman ylläpito on myös helpompaa, koska yksi ja sama ohjelma voi monistettuna ohjata useita samanlaisia testauspisteitä. Jos halutaan

muuttaa testauspisteiden toimintaa, muutos tarvitsee tehdä vain yhteen ohjelmaan, joka vaikuttaa jokaiseen testauspisteeseen. (Ström 2017.)

2.4 Sekvenssi ohjaus

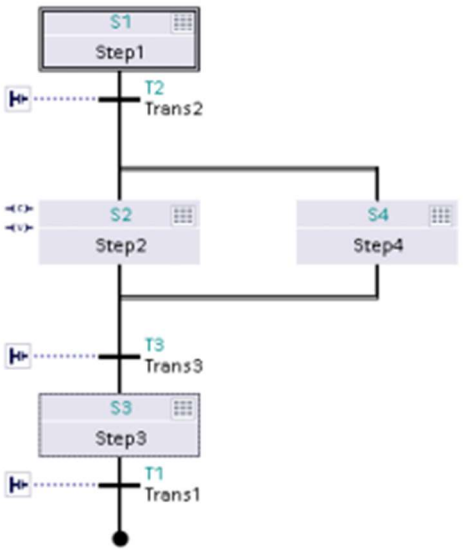
TIA Portal:ssa ohjelman voi kirjoittaa sekvenssi muotoon käyttämällä GRAPH ohjelmointi kieltä. Sekvenssi voi suorittaa itsenäisiä tehtäviä tai jakaa monimutkaisen ohjelman useisiin sekvensseihin. Mikäli sekvenssi käsittelee itsenäistä tehtävää, se suoritetaan rinnakkain ohjelmavirrassa. (Siemens 2014.)

Sekvenssi ohjelmoinnissa ohjelma jaetaan yksittäisiin askeleisiin. Yksinkertaisin tapaus on, että askeleet suoritetaan lineaarisesti toinen toisensa jälkeen. Käyttämällä vaihtoehtoisia tai samanaikaisesti suoritettavia askeleita voidaan tehdä monimutkaisempia rakenteita. Ohjelman suoritus alkaa askeleesta, joka on määritetty ensimmäiseksi askeleeksi. Ensimmäisenä suoritettava askel voi sijaita muuallakin kuin ohjelman järjestyksen mukaisessa ensimmäisessä askeleessa. (Siemens 2014.)

Sekvenssissä askeleen ollessa aktiivinen suoritetaan askeleeseen määritetyt toiminnot. Haaroittamalla pääsekvenssi voidaan suorittaa useita askeleita samanaikaisesti. Askeleen ollessa aktiivinen tarkastellaan siirtoehtoja seuraavaan askeleeseen. Mikäli kaikki siirtoehdot täyttyvät eikä virheitä ole, siirrytään seuraavaan askeleeseen, joka muuttuu aktiiviseksi. Uuden askeleen muuttuessa aktiiviseksi, vanha askel ei ole enää aktiivinen eikä vaikuta ohjelman kulkuun. Sekvenssin voi päättää hyppyyn tai lopettaa viimeiseen askeleeseen. Hypyllä voidaan siirtyä samassa sekvenssissä olevaan askeleeseen tai toiseen sekvenssiin. Tämä mahdollistaa sekvenssin syklisen suorittamisen. (Siemens 2014.)

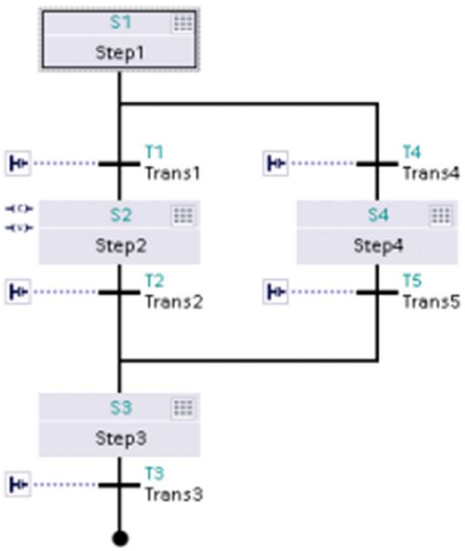
Samanaikaisesti suoritettava haara toimii JA-funktiona. Tämä tarkoittaa, että yhdellä siirtoehdolla siirrytään useisiin askeleisiin, joiden toiminnot suoritetaan. Samanaikaisesti suoritettavat haarat alkavat aina askeleella. Seuraavaan päähaaran askeleeseen siirrytään vasta, kun samanaikaiset haarat ovat suoritettu loppuun. Kuviossa 1 esitetään samanaikaisesti suoritettava haara. Siirtoehdon T2 jälkeen siirrytään askeleisiin

S2 ja S4. Kun molemmat askeleen on suoritettu, siirrytään tarkastelemaan siirtoehtoa T3. (Siemens 2014.)



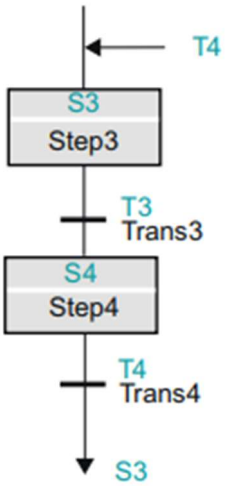
Kuvio 1. Samanaikaisesti suoritettava haara.

Vaihtoehtoisilla haaroilla voidaan luoda ohjelmaan TAI-funktio. Vaihtoehtoisessa haarassa haara alkaa siirtoehdolla. Sen vaihtoehtoisen haaran, jonka siirtoehdot toteutuvat ensimmäisenä, askeleet suoritetaan. Kuviossa 2 ensimmäisen askeleen jälkeen tulee kaksi siirtoehtoa, T1 ja T4. Se haara suoritetaan, kumman siirtoehdot täyttyvät ensimmäisenä. (Siemens 2014.)



Kuvio 2. Vaihtoehtoinen haara.

Sekvenssi ohjelmoinnissa hyppyjä käyttämällä voidaan jatkaa mistä tahansa askeleesta. Hypyn voi sijoittaa päähaaran loppuun, samanaikaisesti suoritettavaan haaraan ja vaihtoehtoisiin haaroihin. Kuviossa 3 on käytetty hyppy, joka hyppää ohjelmassa siirtoehdon T4 jälkeen takaisin askeleeseen S3. (Siemens 2014.)



Kuvio 3. Sekvenssiohjauksen hyppy.

3 Optroniikan testausjärjestelmän liittäminen SQL-tietokantaan

Opinnäytetyöhön kuului teorian osalta selvittää, mitä vaaditaan kyseisen optroniikan testausjärjestelmän liittämiseen SQL-tietokantaan. SQL on lyhenne sanoista Structured Query Language, joka tarkoittaa strukturoitua kyselykieltä. SQL on tietokantakieli, joka on tarkoitettu tietokantojen käsittelyyn. (Hovi 2004.)

Tämän hetkisillä komponenteilla SQL-tietokantayhteyttä ei saa luotua. Helpoin tapa luoda SQL-serverin tiedonkeruu on päivittää nykyinen WinCC Basic WinCC RT Professionaliksi, jolloin ohjelmointi tapahtuu niin ikään TIA Portal ohjelmistolla. WinCC RT Professionalin mukana tulee myös SQL-Server Standard lisenssi, jolloin työskentely SQL-serverille on helppoa, osoita ja klikkaa tapaista. (Kivekäs 2017.)

Toinen vaihtoehto olisi päivittää WinCC Basic WinCC RT Advanced versioon. WinCC RT Advanced tukee sekä SQL-tiedonkeruuta ODBC-linkin kautta että Visual Basic script:ä. Kun ODBC-linkki SQL-serveriin on luotu, voidaan Visual Basic script:llä siirtää tietoa SQL-tietokantaan. Visual Basic script:t ohjelmoidaan TIA Portal ympäristössä. (Kivekäs 2017.)

4 Menetelmä ja työn tavoitteet

4.1 Kehittämistyö

Tämä opinnäytetyö on kehittämistyö, eli toiminnallinen työ. Opinnäytetyössä suunnitellaan ja rakennetaan olemassa olevasta typetyslaitteesta optroniikan testausjärjestelmä. Työn taustaksi olen perehtynyt kirjallisuuteen, jonka pohjalta olen kirjoittanut työn teoriaosuuden. Olen perehtynyt myös aiheesta aiemmin tehtyyn opinnäytetyöhön ja olen saanut jonkin verran suullista tietoa työn toimeksiantajalta.

Kehittämistyöni muodostuu kahdesta osasta, jotka ovat toiminnallinen työ ja kirjallinen raportti. Toiminnalliseen vaiheeseen kuuluvat ohjelmointityö sekä laitteen testaus ja käyttöönotto Millog Oy:llä. Kirjallisessa raportissa avataan työn taustaa käsittelevää teoriaa, kuvataan kehittämistyön vaiheet sekä lopuksi pohditaan ja arvioidaan prosessin kulkua ja kehittämistyön tulosta. (Liukko 2017.)

4.2 Kehittämistyön tavoitteet

Kehittämistyölle asetettuja tavoitteita ovat testausjärjestelmän ohjelmiston rakentaminen ja ohjelman sovitus uuteen logiikkaan aiemman typetyslaitteen pohjalta sekä käyttöliittymän kehittäminen. Ohjelmassa on tärkeää, että se jatkaa kulkua vain testin läpäisseiden laitteiden osalta. Uutena sovelluksena tulee lisätä virranmittaus, jolla saadaan testausjärjestelmä laajennettua kokonaisvaltaisemmaksi testiasemaksi. Toisena uutena ominaisuutena on tarkoitus lisätä selftest, jolla tarkistetaan tietyn väliajoin laitteiston kompponettien toiminta. Ohjelman tulee olla myös helposti laajennettavissa. Käytännön kannalta kehittämistyö tähtää siihen, ettei työntekijöiden tarvitsisi valvoa koko ajan testausprosessia, vaan työntekijät vapautuisivat muihin tehtäviin optroniikka-laitteiden testauksen ajaksi. Järjestelmän jatkokehittämisen kannalta opinnäytetyön toimeksiantaja pyysi selvittämään työn teoriaosuudessa, mitä vaaditaan optroniikan testausjärjestelmän liittämiseen SQL-tietokantaan.

4.3 Tiedonhankinta

Opinnäytetyöni on tehty yhteistyössä työn toimeksiantajan kanssa, ja työn kannalta keskeinen tiedon lähde oli suullinen tiedonanto ja keskustelut Millog Oy:n Tuomas Koskisen kanssa. Lisäksi etsin lähteitä työn teoria-osuutta varten Google Scholar -tietokannasta ja Jyväskylän ammattikorkeakoulun kirjastosta. Tiedonhankinnassa hakusanoina käytin seuraavia asiasanoja: ”optronics”, ”optroniikka”, ”Human Machine Interface”, ”käyttöliittymä”, ”TIA Portal” ja ”SQL”.

Opinnäytetyöni aihe on harvinainen, eikä tiedonhaku tuottanut kovin montaa työhöni sopivaa ja luotettavaa lähdettä. Optroniikan testausjärjestelmän kehittämiseen liittyen löysin vain yhden lähteen, joka oli samasta aiheesta aiemmin tehty opinnäytetyö. Ohjelmointiin liittyvään teoriaan tärkeimmäksi lähteeksi osoittautui Siemensin Step 7 Professional V13 SP1 System -manuaali. Lisäksi olin yhteydessä Siemens Oy:n tekniseen tukeen.

5 Työn toteutus

Opinnäytetyö aloitettiin pitämällä aloituspalaveri Millog Oy:llä, jossa käytiin läpi opinnäytetyön tavoite ja tarkennettiin työn vaatimuksia. Aloituspalaverin jälkeen tutustuimme vielä vanhan typetyslaitteen toimintaan ja ohjelmakoodiin. Millog Oy oli hankkinut kaksi Siemensin logiikkaa ja kaksi kosketusnäyttöä, joten pystyin tekemään opinnäytetyötä etänä. Laitteen toimintaa kävin testaamassa työn eri vaiheissa paikan päällä Millog Oy:llä.

5.1 Ohjelmointi

Opinnäytetyön ohjelmointi suoritettiin Siemensin Tia Portal V13 -ohjelmointityökalua hyödyntäen, ja logiikkana oli Siemensin Simatic ET200SP. Logiikkaan lisättiin Profibus DP moduuli, johon yhdistettiin työssä käytettävä kosketusnäyttöpaneeli ja digitaalisia sekä analogisia tuloja ja lähtöjä.

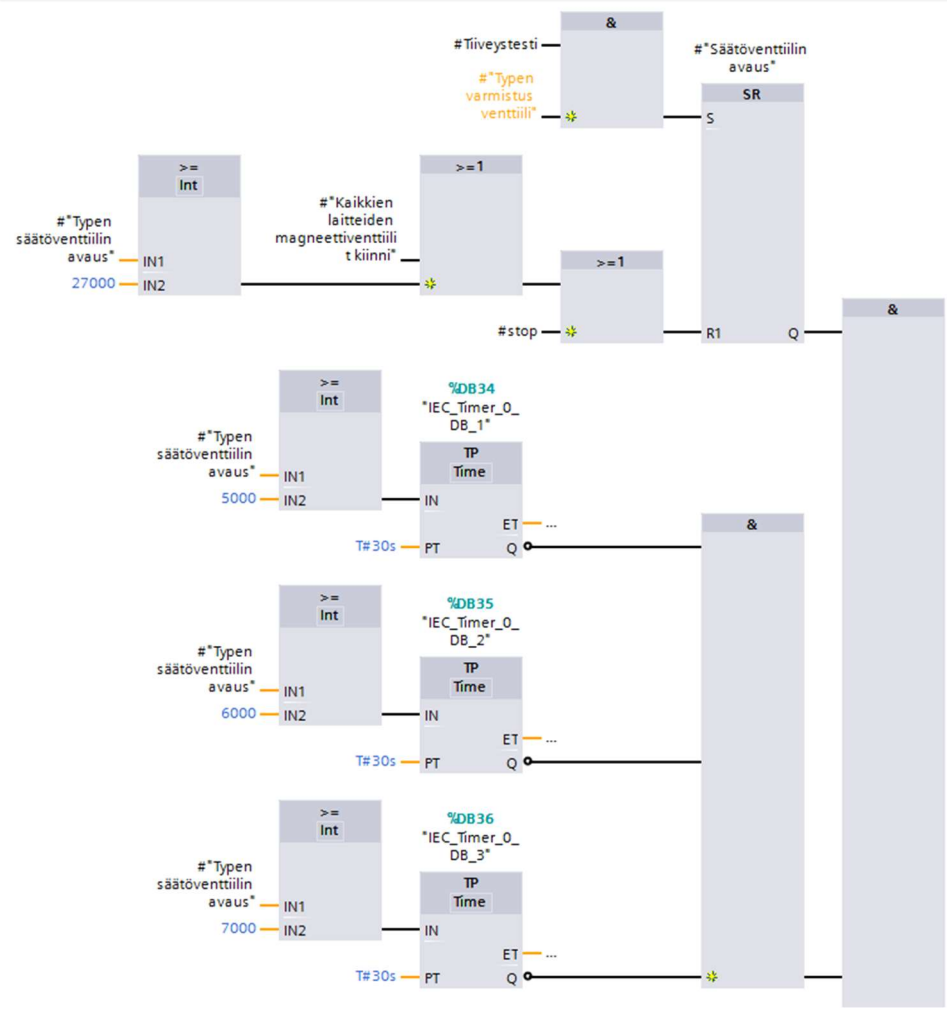
Ohjelmointi työ aloitettiin Hardware konfiguroinnilla, jossa logiikkaan määritettiin siihen lisätyt moduulit. Moduulit DQ 16x24 VDC 0.5A ja AI 8xI 2-/4-wire eivät olleet ohjelmiston valmiissa laitelistassa. Nämä moduulit täytyi lisätä lataamalla Siemensin internet-sivustolta hardware support packages tiedostot HSP0126 ET200SP Modules BA V1.0 ja HSP0162 ET200SP Modules DI DQ STV1.1. Nämä tiedostot sai asennettua TIA Portal ohjelmistoon menemällä options -> support packages -> installation of support packages -> Install. Kun tiedostot oli asennettu, sai myös viimeiset moduulit lisättyä hardware konfigurointiin. Hardware konfiguroinnin jälkeen pystytiin aloittamaan optroniikan testausjärjestelmän ohjelmointi.

Optroniikan testausjärjestelmän vaatimuksena oli, että se testaa 1-10 laitetta yhtä aikaa täysin itsenäisesti, jolloin se vapauttaa työntekijän muihin tehtäviin ja ilmoittaa lopuksi luotettavasti, mitkä laitteet suorittivat testauksen hyväksytysti ja mitkä vaativat huoltotoimenpiteitä. Tässä testauksessa suoritetaan laitteen tiiveystesti ja typetys.

Ohjelman tuli olla myös helposti laajennettavissa, joten päädyin käyttämään parametrisoitua ohjelmointia. Parametrisoidulla ohjelmoinnilla saavutettiin ohjelman helppo laajentaminen, sillä kanavia ohjelmaan saa lisättyä kopioimalla tiiveystestin ja typetyksen funcktion blockeja pääohjelmaan, jossa sitten määritetään kanavan muuttujat. Kopioimalla funcktion blockeja pääohjelmaan luodaan samalla jokaiselle ohjelmalle data block, jonne ohjelma tallentaa muuttujien tiedot.

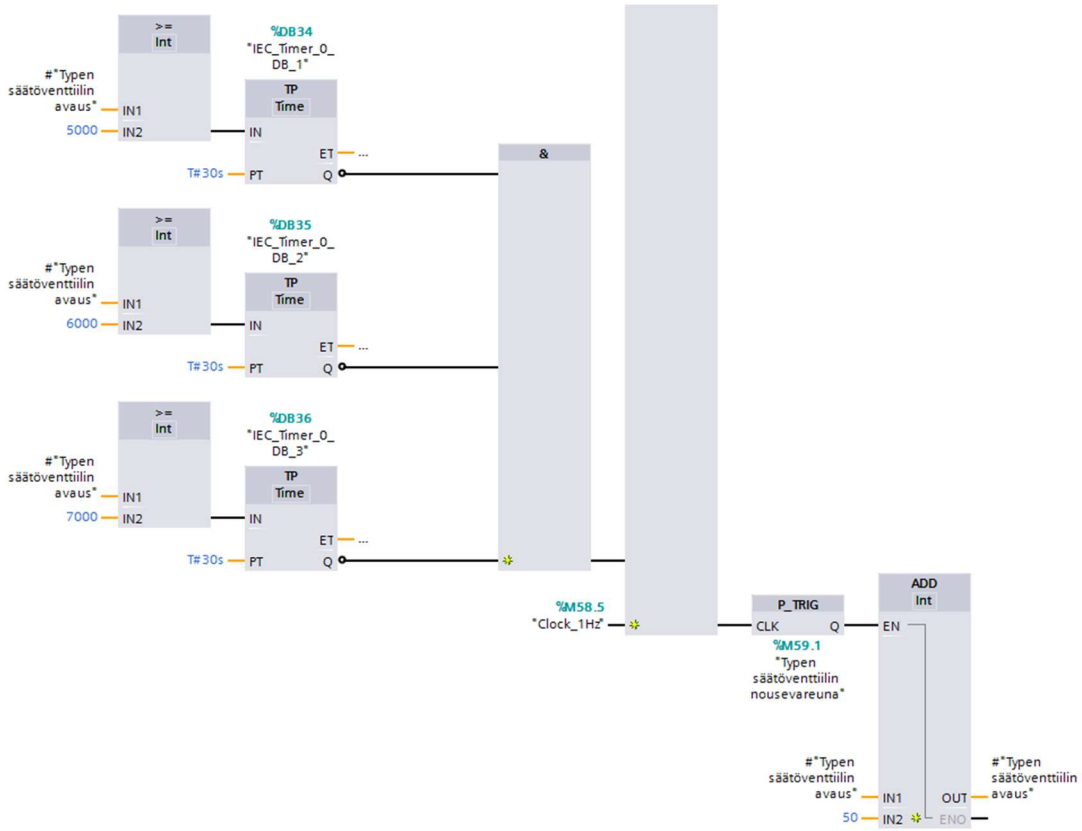
Tiiveystestin ohjelman päätin toteuttaa funcktion blockeja käyttämällä, ja kieleksi valitsin FBD:n (Fucnktion Block Diagram). Valitsin nämä, koska FBD koodi on helppoa ymmärtää ja ne olivat ennestään tuttuja minulle. Tiiveystestissä jokaisella kanavalla on sama ohjelma, mutta niillä on oma data block, johon tallennetaan muuttujien tiedot. Tein oman ohjelman säätöventtiilille, joka säätää typen virtausta laitteeseen. Tämä siksi, ettei jokainen kanava yhtä aikaa ohjaa säätöventtiiliä, jolloin säätöventtiili saattaisi saada kaksi eri ohjauskomentoa, ja siitä seuraisi ohjelman jumittuminen.

Tiiveystestin säätöventtiilin ohjauksessa päädyin käyttämään clock memory byte:ä eli tavua, jonka eri bitit muuttavat tilaansa tietyllä taajuudella. Taajuuksien vaihtoehdot olivat 10Hz, 5Hz, 2.5Hz, 2Hz, 1.25Hz, 1Hz, 0.65Hz, ja 0.5Hz. Säätöventtiilin ohjaukseen valitsin 1Hz taajuuden, jolloin säätöventtiilin ohjaus kasvaa aina sekunnin välein ja säätöventtiilin avausnopeus on helpompi hahmottaa ja hallita. Tämän tavun sai otettua käyttöön hardware konfiguroinnista, eli Project tree -> PLC -> device configuration -> system and clock memory -> enable the use of clock memory byte. Säätöventtiilin ohjauksessa oli tärkeää, että typpi ei pääse virtaamaan liian voimakkaasti, joka voisi aiheuttaa laitteen sisäisten roskien liikkeelle lähtemisen. Lisäksi laitetta oli tärkeää täyttää hitaasti, jolloin paine ehtii tasaantua paremmin laitteen sisällä. Kuvioissa 1 ja 2 on ohjelma säätöventtiilin avauksesta. Kuviossa 4 olevassa säätöventtiilin avauksessa annetaan lupa säätöventtiilin avaamiselle, mikäli tiiveystesti on valittu, typen varmistusventtiili on auki, jonkin laitteen magneettiventtiili on auki ja säätöventtiili on auki vähemmän kuin 27000 eikä stoppia ole painettu. Mikäli nämä kaikki ehdot toteutuvat, voidaan säätöventtiiliä avata.



Kuvio 4. Säätöventtiilin avaus osa 1.

Kuviossa 5 olevilla ajastimilla ”IEC_timer_0_DB_1”, ”IEC_timer_0_DB_2” ja ”IEC_timer_0_DB_3” pyritään hidastamaan typen liian nopeaa virtausta laitteeseen, jolloin laitteen sisäinen paine tasaantuisi pyydettyyn painearvoon nopeammin. M58.5 oleva kellobitti on aktiivinen sekunnin välein, jolloin säätöventtiilin ohjausta nostetaan sekunnin välein. Positiivisen reunan tunnistusta P_TRIG tarvitaan, jottei ohjelma kasvata venttiilinohjausta kuin kerran sekunnissa. Ilman positiivisen reunan tunnistusta ohjelma lisäisi ohjausta useita kertoja kellobitin aktiivisen sekunnin aikana.

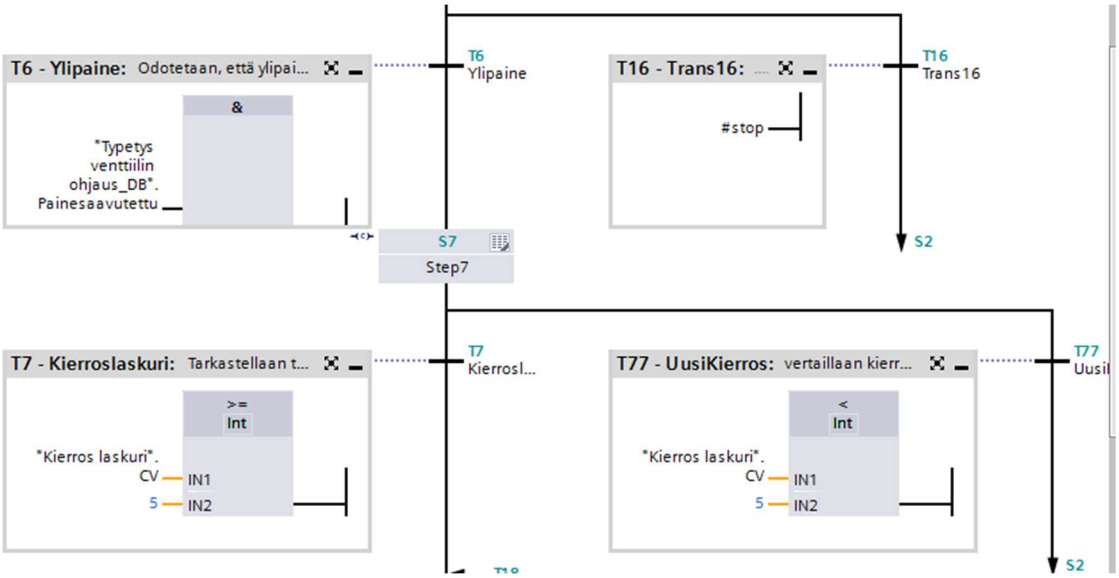


Kuvio 5. Säätöventtiilin avaus osa 2.

Tiiveystestissä laitteen paineen täytyi ehtiä tasaantua ennen kuin tiiveystestin pitoaika voi alkaa. Paineen tasaantuminen toteutettiin käyttämällä paineen tasausaikaa. Tämä tarkoittaa, että odotetaan tietty aika, jonka jälkeen tarkistetaan, onko paineet laskeneet. Mikäli paineet ovat laskeneet, ne täytetään uudelleen. Laitteet täytetään maksimissaan kolme kertaa, jonka jälkeen ohjelma siirtyy eteenpäin.

Paineen tasauksen jälkeen kirjoitetaan jokaiselta kanavalta lähtöarvo kanavan omaan data blockiin ja aloitetaan tiiveystestin pitoaika. Kun pitoaika on ohi, kirjoitetaan tulos muistiin kanavan omaan data blockiin, jonka jälkeen nähdään, onko laite suorittanut tiiveystestin hyväksytysti. Kun tulos on kirjoitettu muistiin, aloitetaan laitteiden paineen tyhjennys. Mikäli typetystä ei ole valittu, ohjelma on valmis. Jos typetys on valittu, jatkaa ohjelma automaattisesti typetykseen tiiveystestin läpäisseiden laitteiden kanssa. Tiiveystestistä typetykseen siirryttäessä oli tärkeää, että vain tiiveystestin läpäisseet laitteet suorittavat typetyksen, jolloin typpeä ei kulu turhaan vuotaviin laitteisiin.

Typetyksessä kanava kohtainen ohjelmointi toteutettiin sekvenssi ohjauksella. Päädyin sekvenssi ohjaukseen, koska typetyksessä suoritettavat toimenpiteet etenevät aina samassa järjestyksessä, jolloin sekvenssi ohjaus on helppo toteuttaa. Kuviossa 6 näkyy typetyksen sekvenssiohjausta. T6 on siirtymäehto askeleeseen 7, ja T16 on siirtymäehto askeleeseen 2. Mikäli ylipaine on saavutettu, siirrytään askeleeseen 7. Jos stop muuttuja on aktiivinen, siirrytään askeleeseen 2. Siirtoehdossa T7 tutkitaan, onko kierroslaskuri saavuttanut arvon 5 tai enemmän. Jos näin on, siirrytään seuraavaan askeleeseen, ja mikäli kierroslaskurin arvo pienempi kuin 5, palataan siirtoehdon 77 mukaan askeleeseen 2.

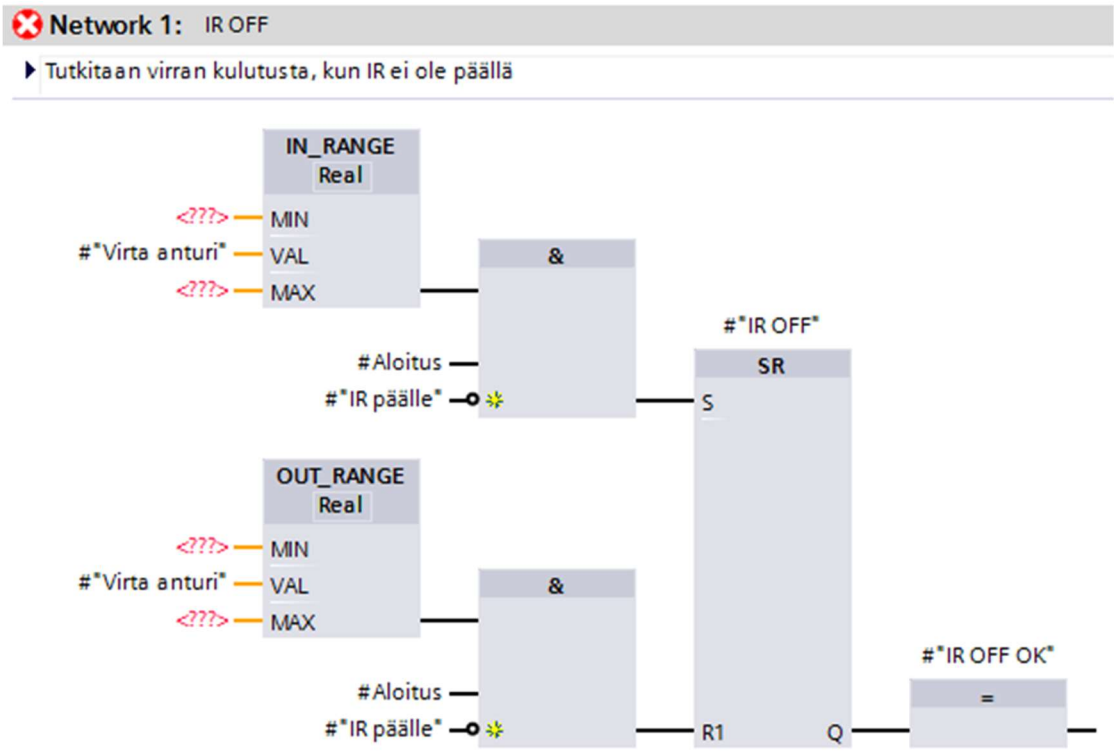


Kuvio 6. Typetyksen kierrosmäärän tarkkailu.

Typetys ohjelmalla on oma venttiilinohjaus ohjelma, jolla ohjataan venttiilit oikeaan asentoon oikeassa järjestyksessä ja vältetään ristikkäisohjauksia. Venttiilinohjaus on toteutettu funcktion block:lla, ja ohjelmointikielenä on käytetty FBD:tä.

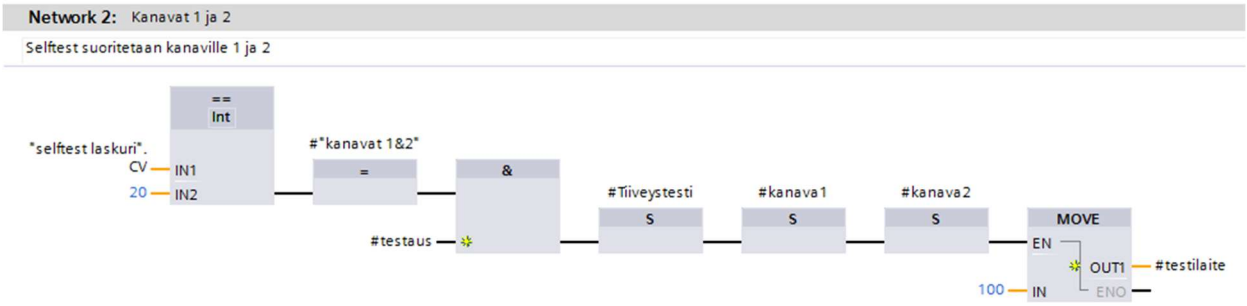
Järjestelmään luotiin myös virranmittaus ohjelma, jolla mitataan laitteiden virtoja. Laitteiden virtoja mittaamalla selvitetään virrankulutusta ja IR-valon toimintaa. Ohjelmassa laitteiden virtaa tarkastellaan kahdessa eri vaiheessa. Ensimmäisessä vaiheessa laitteiden IR-valo ei ole päällä, ja toisessa vaiheessa IR-valo kytketään päälle. Virrankulutuksen täytyy olla laitteeseen määritetyissä rajoissa molemmissa vaiheissa, jolloin laite on hyväksytty. IR-valon toimintaa tutkitaan mittaamalla virtaa IR-valo

päällä. Mikäli äkillinen virran alentuma havaitaan, tapahtuu putkessa välähdys, jolloin laite toimii oikein. Kuviossa 7 esitetään virranmittauksen ohjelma, kun IR-valo ei ole päällä. IN_RANGE lohkossa tutkitaan, onko virrankulutus määritettyjen arvojen välissä, ja OUT_RANGE lohkossa tutkitaan, onko virta-arvo rajojen ulkopuolella.



Kuvio 7. Virranmittaus, kun IR-valo ei ole päällä.

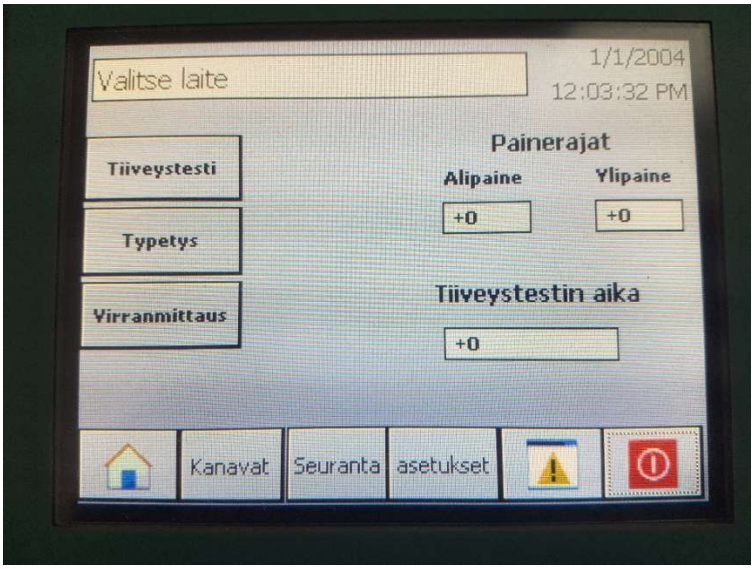
Optroniikan testausjärjestelmälle täytyi luoda ohjelma myös laitteiston testaamiselle, jotta varmistetaan, että järjestelmän komponentit toimivat oikein. Ohjelma ilmoittaa tietyin ohjelmasuoritus kertojen jälkeen, että järjestelmä on testattava. Tällöin valitaan järjestelmän testaus, jolloin ohjelma määrittää kaksi testattavaa kanavaa. Liitetään testilaite kahteen ohjelman määräämään kanavaan, jonka jälkeen painetaan start-painiketta, jolloin testaus käynnistyy. Testauksen jälkeen tarkistetaan, onko järjestelmätestauksen tulos hyväksytty. Tällä pystytään toteamaan, toimivatko järjestelmän venttiilit ja paineanturit. Kuviossa 8 esitetään kanavan 1 ja 2 järjestelmän testaus. Kun selftest-laskuri on tosi, näytöltä voidaan aktivoida testaus-muuttuja, joka asettaa tiiveystestin ja kanavat 1 ja 2 aktiivisiksi sekä valitsee testilaitteen. Tällöin käyttäjän tarvitsee vain kiinnittää testilaitteet ja käynnistää testaus.



Kuvio 8. Selftest

5.2 Käyttöliittymä

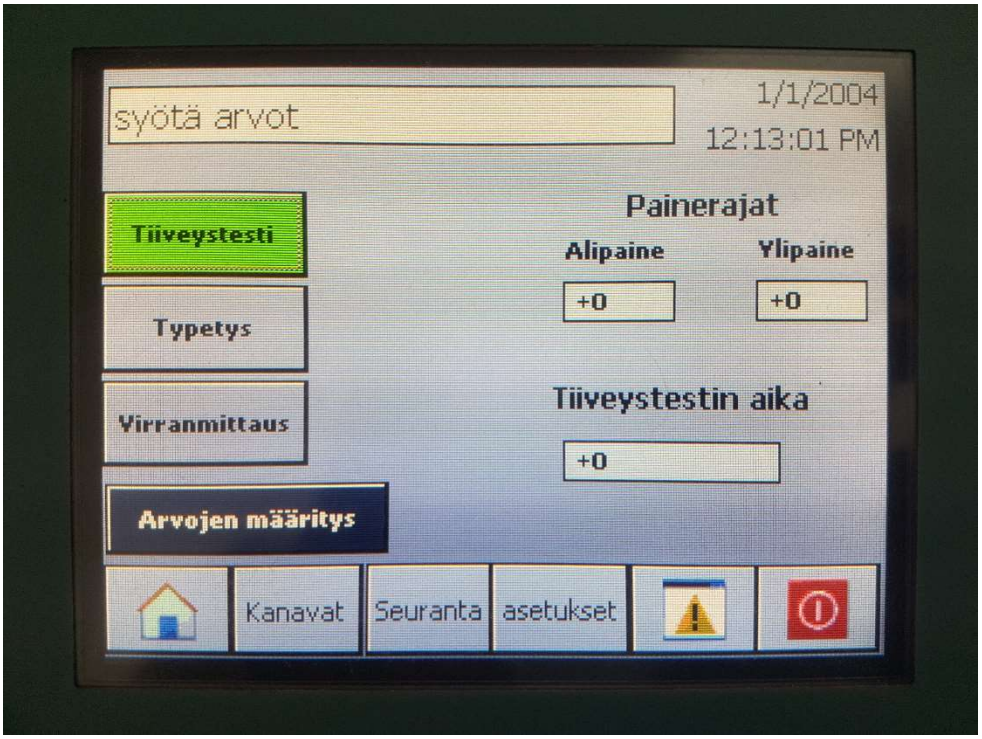
Tässä työssä käyttöliittymä tehtiin Siemensin KPT600 Basic Color DP kosketusnäytölle. Käyttöliittymän ohjelmointi suoritettiin Tia Portal V13 -ohjelmistolla. Kosketusnäytön ja logiikan välisenä yhteytenä toimi Profibus DP. Käyttöliittymän päivitykset ladattiin tietokoneelta, jolloin yhteydeksi tuli muuttaa MPI (Multi-Point Interface). Kuviossa 9 on näkyvillä käyttöliittymän aloitus sivu, josta pystytään määrittämään suoritettavat ohjelmat ja valitsemaan laite. Valitun laitteen raja-arvot näkyvät näytön oikeassa reunassa. ”Valitse laite” -valikosta (kuvio 10) voi valita myös ”syötä arvot” kohdan, jolloin käyttäjä saa itse määrittää laitteelle raja-arvot.



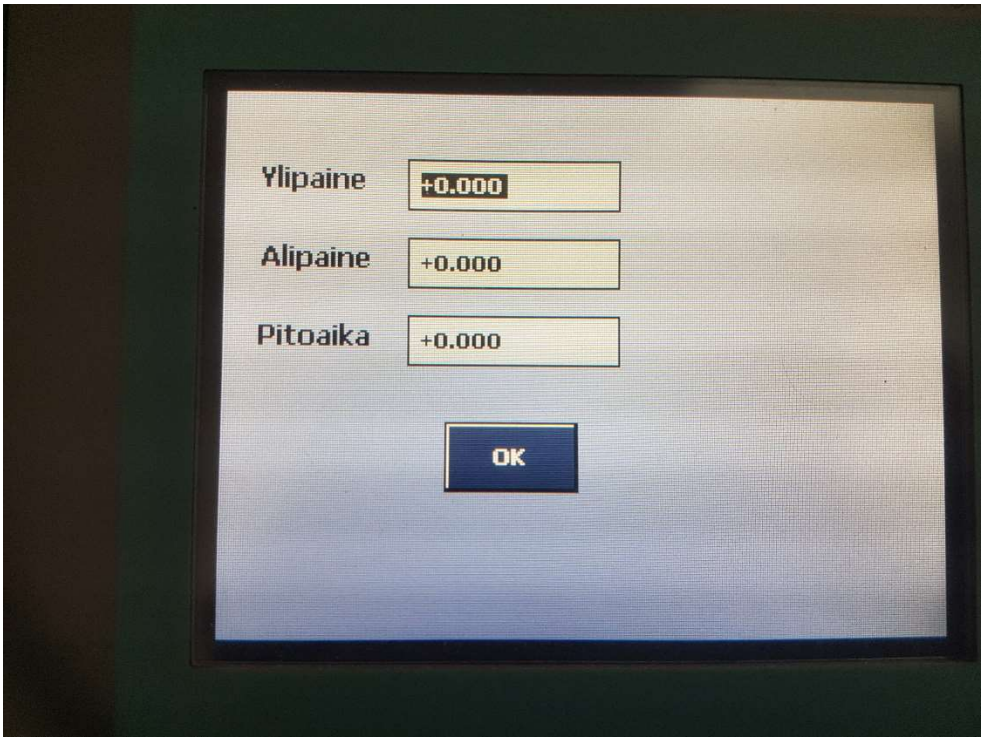
Kuvio 9. Aloitus sivu

Kuvio 10. Laitevalikko

Kuviossa 11 on valittu laitteeksi ”syötä arvot”, jolloin virranmittauksen alapuolelle ilmestyy arvojen määritys nappi, josta pääsee määrittämään halutut raja-arvot. Kuviossa 12 määritetään halutut raja-arvot.



Kuvio 11. Syötä arvot.

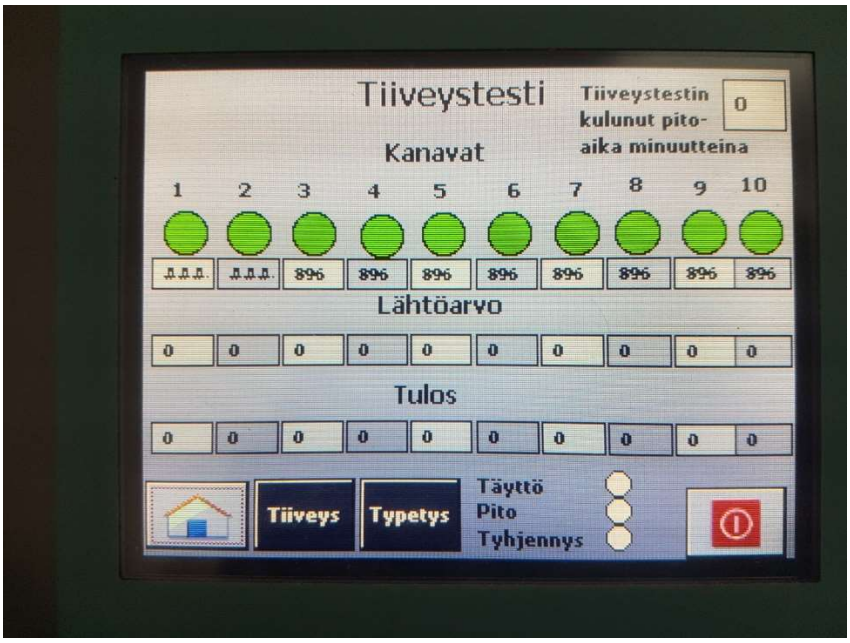


Kuvio 12. Arvojen määritys.

Kanavat-sivulta määritetään kanavat, joille ohjelma suoritetaan. Kanavia voidaan valita 1-10. Mikäli yhtään kanavaa ei ole valittu, start-painike ei ole näkyvissä eikä painettavissa (Kuvio 13).

Kuvio 13. Kanavat sivu.

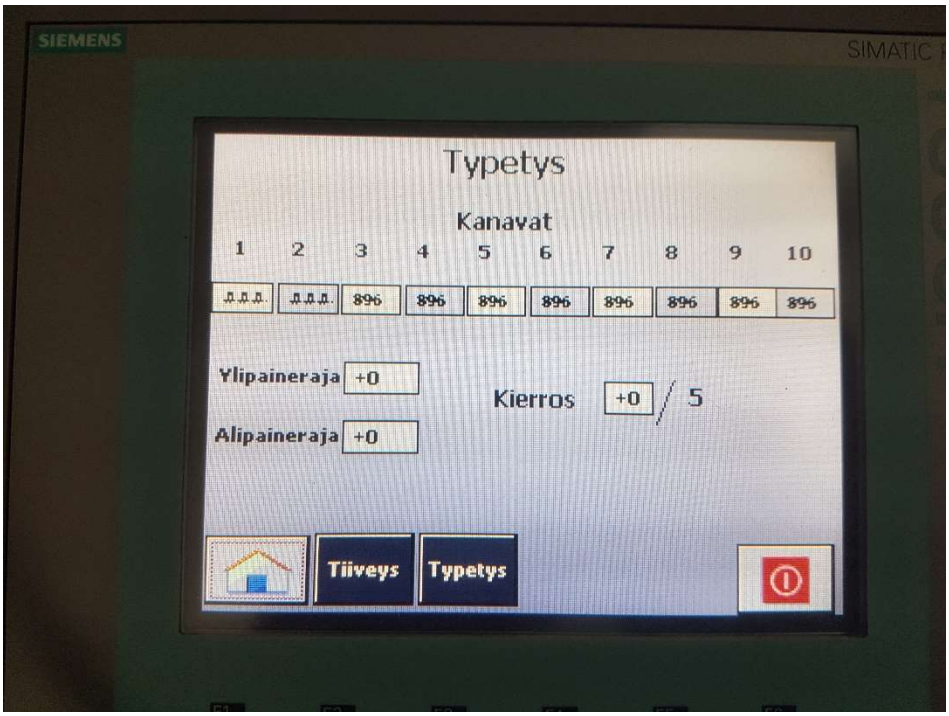
Kun kanavat on valittu ja painetaan starttia, aukeaa tiiveystestin seuranta -sivu automaattisesti (kuvio 14). Tiiveystestin seuranta -sivulta näkee laitteessa olevan paineen, tiiveystestin lähtöarvon ja tuloksen sekä paljonko tiiveystestin pitoajasta on kulunut. Sivun alareunassa on näkyvillä myös vaihe, jota ohjelma suorittaa. Tämä helpottaa ohjelman seuraamista. Tiiveystestin loputtua kanava kohtainen led-valo palaa joko vihreänä tai punaisena riippuen siitä, onko laite suorittanut testin hyväksytysti. Ohjelman päätyttyä vasempaan yläkulmaan ilmestyy ”ohjelma valmis” -painike, josta kuitataan ohjelma päättyneeksi ja samalla resetoidaan muuttujat, jolloin järjestelmä on valmiina seuraavalle testaukselle (kuvio 15). Edellisen testauksen tulokset eivät kuitenkaan resetoidu, vaan ne nollaantuvat vasta seuraavan testin alussa. Näin tulokset eivät katoa ja ovat nähtävillä, kunnes aloitetaan seuraava testaus.



Kuvio 14. Tiiveystestin seuranta.

Kuvio 15. Ohjelma valmis.

Typetyksen seuranta -sivulta (Kuvio 16) näkee laitteissa olevan paineen, alipaineen ja ylipaineen raja-arvot sekä montako kierrosta on suoritettu. Typetyksen suorittamisen jälkeen ilmestyy vasempaan yläkulmaan sama ”ohjelma valmis” -nappi, josta kuitataan ohjelma valmiiksi ja resetoidaan muuttujat.

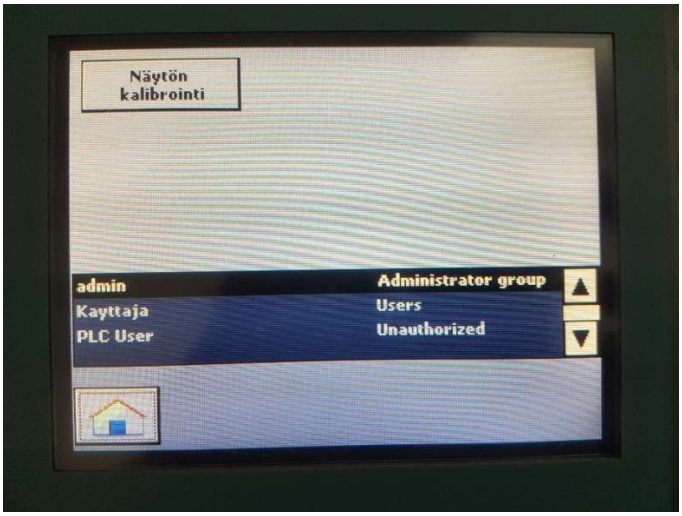


Kuvio 16. Typetyksen seuranta.

Virranmittauksen seurannan (Kuvio 17) ensimmäisessä vaiheessa seurataan virta-arvoja, kun IR-valo ei ole päällä. Mikäli virta-arvot eivät ole raja-arvojen sisällä, palaa kanavalla punainen led, ja jos arvot ovat raja-arvojen sisällä, palaa vihreä led. IR-painiketta painamalla raja-arvot muuttuvat, jolloin seurataan virta-arvoja IR-valon ollessa päällä. Flash:ssä seurataan, tapahtuuko laitteessa huomattavaa virran alentumaa tietyn ajan kuluessa, ja mikäli näin tapahtuu, laite toimii oikein ja vihreä led syttyy. Näytön alareunassa näkyy laitteiden virta-arvot.

Kuvio 17. Virranmittauksen seuranta.

Asetukset-sivulta käyttäjä pääsee kalibroimaan kosketus näytön ja lisäämään käyttäjiä (kuvio 18). Asetukset-sivulle pääseminen vaatii admin-tason käyttöoikeudet.



Kuvio 18. Asetukset sivu.

5.3 Käyttöönotto

Käyttöönottovaiheessa ongelmia ilmeni laitteiden täytössä, sillä laitteiden sisäinen paine ei ehtinyt tasaantua. Tämä aiheutti ongelmia tiiveystestissä, kun laitteet saavuttivat paineen raja-arvon ja venttiilit sulkeutuivat. Tämän jälkeen paine laitteen sisällä alkoi laskemaan nopeasti, koska paine ei ollut ehtinyt tasaantua laitteen sisällä täytön aikana. Paineen tasaus -ongelma ratkaistiin hidastamalla typen virtausta laitteeseen täytön aikana, jolloin paine ehtii tasaantua paremmin, sekä lisäämällä ohjelmaan paineen tasauskerrat. Paineen tasauskerroilla järjestelmä täyttää laitteet ja odottaa tietyn ajan, jonka jälkeen ohjelma tarkistaa, ovatko paineet laskeneet. Mikäli paineet ovat laitteissa laskeneet, ne täytetään uudelleen. Uudelleen täyttö laitteille tehdään maksimissaan kolme kertaa, jonka jälkeen ohjelma siirtyy eteenpäin.

Käyttöön otossa havaittiin myös, ettei ohjelma aina kirjoita tiiveystestissä lähtöarvoa eikä tulosta niiden muuttujiin. Lisäksi osa venttiileistä jäi auki, vaikka niiden pitäisi olla kiinni. Tämä ongelma ratkaistiin lisäämällä ohjelmaan viiveitä, sillä logiikka ei aiemmin ehtinyt kirjoittaa muuttujiin arvoja ohjelman mennessä jo eteenpäin, mikäli kaikki kymmenen kanavaa oli käytössä. Viiveitä lisäämällä nämä ongelmat ratkesivat.

Ohjelman keskeyttäminen aiheutti myös ongelmia. Järjestelmä jäi siihen tilaan, jossa ohjelma oli keskeytetty, ja vaati logiikan resetointia, mikäli halusi aloittaa uuden testauksen. Ohjelmaa muutettiin niin, että ohjelman keskeyttäminen palauttaa järjestelmän lähtöpisteeseen, jolloin ohjelma on valmis seuraavalle testaukselle.

Lisäksi käyttöliittymään tehtiin pieniä parannuksia, jotka selkeyttivät järjestelmän käyttöä. Tiiveystestin seurantaan lisättiin vaiheet, joita ohjelma suorittaa. Tämä helpotti tiiveystestin seuraamista.

6 Tulokset

Optroniikan testausjärjestelmästä tuli kehittämistyön tuloksena toimiva kokonaisuus. Päivitetyllä järjestelmällä voidaan suorittaa testattaville optroniikka-laitteille seuraavat ohjelmat: tiiveystesti, typetys, virran kulutuksen mittaus ja IR-valon toiminta. Järjestelmään lisättiin uusina ominaisuuksina virran kulutuksen mittaus ja IR-valon toiminta. Nyt järjestelmä on ohjelmoitu suorittamaan tiiveystesti ja typetys automaattisesti, ja ohjelma jatkaa typetykseen vain tiiveystestin läpäisseiden laitteiden kanssa. Työntekijän ei tarvitse ohjata ja valvoa toimenpidettä. Selftest-ominaisuudella voidaan nyt tarkistaa järjestelmän komponenttien toiminta. Testausjärjestelmän ohjelma on nyt helposti laajennettavissa, mikäli järjestelmään halutaan lisätä kanavien määrää. Käyttöliittymä on selkeä ja helppokäyttöinen.

7 Pohdinta

Tämän opinnäytetyön tavoitteena oli luoda toimiva, helppokäyttöinen ja helposti laajennettavissa oleva optroniikan testausjärjestelmän ohjelmointi ja käyttöliittymä, joka helpottaisi laitteiden testausta ja vapauttaa työn tekijän muihin tehtäviin testauksen ajaksi. Mielestäni tavoitteisiin on päästy ja kaikki työn vaatimukset täyttyvät. Optroniikan testausjärjestelmän toimintaa pystyisi vielä parantelemaan optimoimalla ohjelmassa olevia viiveitä ja säätämällä typen virtausnopeutta laitteisiin, millä pystyttäisiin vähentämään laitteiden uudelleen täyttö kertoja, mikä taas lyhentäisi ohjelman suoritusaikaa.

Käyttöliittymästä tuli mielestäni helppokäyttöisempi ja selkeämpi kuin vanhan typetyslaitteen käyttöliittymä. Laitteet ja ohjelmat on helppo valita, ja ohjelmaa on helppo seurata ohjelman tilatietojen ja tiiveystestin kuluneen pitoajan kanssa. Käyttöliittymän kosketusnäytön liitäntänä olisi voinut olla ethernet-liitäntä, jolloin ei olisi tarvinnut ostaa logiikkaan erillistä moduulia Profibus DP:lle, vaan logiikan ja kosketusnäytön välinen yhteys olisi hoidettu myös ethernetillä. Tämä olisi helpottanut myös ohjelman lataamista tietokoneelta, jos olisi voinut käyttää ethernet-liitäntää. Nykyisessä järjestelmässä joudutaan vaihtamaan joka kerta Profibus DP yhteys MPI yhteyteen ja määrittämään sen asetukset käyttöliittymää päivitettäessä.

Oman ammatillisen osaamisen kannalta tämä opinnäytetyö oli erittäin opettavainen ja vahvisti ammatillista osaamistani. Etenkin TIA Portal -ympäristö tuli erittäin tutuksi, kun aiempaa kokemusta TIA Portal:sta ei juurikaan ollut. Koulussa on käytetty pääasiassa Siemensin vanhempaa ohjelmisto versiota. Uskon, että TIA Portal:n käytöstä on hyötyä työelämässä. Työskentely oli mielenkiintoista, kun sai olla mukana kehittämässä todellista työelämän projektia.

Lähteet

Hovi, A. 2004. SQL-opas. E-kirja. Jyväskylä: Docendo.

Kippo, A. K. & Tikka, A. 2008. Automaatiotekniikan perusteet. Helsinki: Edita.

Kivekäs, P. 2017. Tekninen tuki, Siemens Oy. Sähköpostiviesti. 11.10.2017.

Koskinen, T. 2017. Millog Oy, Lievestuore. Suullinen tiedonanto 1.4.–10.10.2017.

Liukko, S. 2017. Opinnäytetyön raportointi. Jyväskylän Ammattikorkeakoulu. Viitattu 26.10.2017. https://oppimateriaalit.jamk.fi/raportointiohje/tag/kehittamistyo/.

Millog Oy 2017. Tietoa meistä. Viitattu 29.8.2017. www.millog.fi/portal/fi/tietoa_meista/.

Patria 2014. Tekniikka: Pimeässä näkee loistavasti. Viitattu 28.8.2017. www.patriamagazine.fi/pimeassa-nakee-loistavasti/.

Siemens 2017. TIA Portal - teollisuusautomaation ohjelmistoalusta. Viitattu 22.8.2017. www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/tia_portal.php.

Siemens 2014. Step 7 Professional V13 SP1 System Manual.

Ström, M. 2017. Jyväskylän ammattikorkeakoulun lehtori. Sähköpostiviesti 26.10.2017.

Tzafestas, S. G. & Tzafestas, E. S. 2001. Human-Machine Interaction in Intelligent Robotic Systems: A Unifying Consideration with Implementation Examples. Journal of Intelligent and Robotic Systems 32, 119–141.

Valkonen, O. 2010. Typetyslaitteen automatisointi. Opinnäytetyö, AMK. Jyväskylän ammattikorkeakoulu, Kone- ja tuotantotekniikan koulutusohjelma. Viitattu 6.5.2017. http://urn.fi/URN:NBN:fi:amk-2010101413707.