TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

OTTO HIETAMIES

AUTOMATISOITU TESTAUSJÄRJESTELMÄ SULAUTETTUJEN OHJELMISTOJEN REGRESSIOTESTAUKSEEN

Diplomityö

Tarkastaja: professori Karri Palovuori
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan
tiedekuntaneuvoston kokouksessa
4. syyskuuta 2013

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
HIETAMIES, OTTO: Automatisoitu testausjärjestelmä sulautettujen ohjelmistojen regressiotestaukseen
Diplomityö, 48 sivua, 2 liitesivua
Joulukuu 2013
Pääaine: Elektroniikan laitesuunnittelu
Tarkastaja: professori Karri Palovuori
Avainsanat: Automatisoitu testausjärjestelmä, sulautettu järjestelmä, ohjelmistotestaus, regressiotestaus , jatkuva integrointi

Sulautettujen järjestelmien testaus on aikaa vievä prosessi, mutta välttämätöntä laitteiden toiminnallisuuden varmistamiseksi. Perinteinen manuaalisesti suoritettu testaus voi olla hidasta ja vaatii yleensä useita työtunteja testien suorittamiseen. Tämän diplomityön aihe muodostui Espotel Oy:n tarpeesta tehostaa ohjelmistotestausprosessia. Yrityksessä havaittiin, että suuren luokan tuotekehitysprojekteissa ohjelmistotestaus oli merkittävä hidastava tekijä.

Työssä selvitetään, miten toteutetaan kustannustehokas automatisoitu testausjärjestelmä sulautettujen ohjelmistojen testaukseen. Tavoitteena on lyhentää ohjelmiston testaukseen kuluvaa aikaa ja automatisoida osa testauksen raportoinnista.

Työssä käsitellään ohjelmiston- ja automatisoidun testauksen taustoilla olevia teorioita ja esitellään niihin liittyviä käsitteitä. Teoriaosuudessa käydään läpi muun muassa ohjelmistotestauksen tasoja, regressiotestausta ja jatkuvaa integrointia

Toteutettu testausjärjestelmä koostuu kahdesta eri osasta, testauslaitteistosta ja sitä ohjaavasta ohjelmistosta. Aluksi selvitettiin testattavien moduulien testijärjestelmälle asettamat laitteistovaatimukset, joiden pohjalta laitteistoa lähdettiin kehittämään. Järjestelmän ohjelmistoa kehitettiin laitteiston kanssa samanaikaisesti.

Työn tuloksena saavutettiin tavoitteiden mukaisesti ohjelmistoprosessin tehokkuutta parantava automatisoitu järjestelmä. Ohjelmiston testausta pystyttiin nopeuttamaan ja toistettavuutta lisäämään. Ohjelmistovirheiden havainnointi helpottuu ja näin virheet eivät pääse leviämään ohjelmistoprosessin ylempiin tasoihin. Järjestelmän kustannustehokkuus kasvaa automatisoinnin myötä, mutta todelliset kustannukset näkyvät vasta projektin myöhemmässä vaiheessa.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications engineering
HIETAMIES, OTTO: Automated Test System for Regression Testing of Embedded Software
Master of Science Thesis, 48 pages, 2 Appendix pages
December 2013
Major: Design of Electronic Circuits and Systems
Examiner: Professor Karri Palovuori
Keywords: Automated test system, embedded system, software testing, regression testing, continuous integration

Testing embedded systems is a time consuming process, but necessary for verifying the functionality of devices. Conventional manual testing can be slow and require multiple working hours to execute the tests. The Topic of this thesis is based on the needs of Espotel Oy to enhance their software testing process. The company noticed that software testing was a significant factor in slowing down large scale research and development projects.

This thesis resolves how to implement a cost-effective automated test system for embedded software testing. The goal is to shorten the time used for software testing and to automate a part of test reporting.

This thesis examines the theories and concepts behind software and automated testing. The theory part of this thesis covers for instance the levels of software testing, regression testing and continuous integration. The implemented test system consists of two parts, the testing system and the control software that controls it. At first the test requirements of the tested modules were resolved. Based on this the development of the hardware began. The development of the software and the hardware were made simultaneously.

As a result of this thesis an automated test system was created to enhance the software process, which is consistent with the objectives. It was now possible to increase the speed and test repeatability of software testing. Detection of software errors was made easier and therefore these errors can't spread to higher levels of the software process. The cost-efficiency of the system grows due to the automation, but the actual costs can not be seen until the later stage of the project.

ALKUSANAT

Tämä diplomityö tehtiin Espotel Oy:ssä osana tuotekehitysprojektia. Työn tekeminen aloitettiin kesällä 2013. Haluan kiittää työn tarkastajaa professori Karri Palovuorta ohjauksesta ja tuesta työn aikana.

Kiitokset myös Espotel Oy:n Tampereen toimiston väelle kannustuksesta. Erityisesti haluan kiittää Chief Engineer Jarno Mannista työn ohjauksesta ja Account Manager Kari Viitalaa, joka mahdollisti työn tekemisen. Kiitos myös kaikille työn kielelliseen tarkastukseen osallistuneille.

Suuret kiitokset kuuluvat avopuolisolleni Riikka-Mari Tuomelalle ja työn vauhdittajana toimineelle lokakuussa syntyneelle esikoispojallemme.

Tampereella 24. lokakuuta 2013

Otto Hietamies

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

AD	Analog/Digital.
ADC	Analog to Digital Converter, analogia-digitaalimuunnin
ADI	Analog/Digital Input, analoginen/digitaalinen sisäänmeno
ADO	Analog/Digital Output, analoginen/digitaalinen ulostulo
AI	Analog Input, analoginen sisäänmeno
AO	Analog Output, analoginen ulostulo
API	Application Programming Interface, ohjelmointirajapinta
CAN bus	Control Area Network, automaatioväylä, jota käytetään ajoneuvoissa ja teollisuuslaitteissa
CI	Continuous integration, jatkuva integrointi
Comedi	Control and measurement device interface, ajuri- ja kirjastokokoelma tiedonkeruu korteille
CppCheck	C/C++ -koodin analysointityökalu
CppUTest	C/C++ -pohjainen XUnit –testikehys
DAC	Digital to Analog Converter, digitaali-analogiamuunnin
DC	Direct Current, tasavirta
DI	Digital Input, digitaalinen sisäämeno
DO	Digital Output, digitaalinen ulostulo
DPDT	Double Pole Douple Throw, kaksoisvaihtokoskettimellinen releтyyppi
DRV	Driver
DUT	Device under test
Ext OC	Excitation Overurrent, herätejännitteen ylivirta
Ext	Excitation, herätejännite
FAI	Frequency/AnalogInput, taajuus/analoginen sisäämeno
FDI	Frequency/Digital Input, taajuus/digitaalinen sisäämeno
FDO	Frequency/Digital Output, taajuus/digitaalinen ulostulo
FFT	Fast Fourier Transform
FI	Frequency Input, taajuus sisäämeno
GND	Ground, maapotentiaali
HIL	Hardware-in-the-Loop
I/O	Input/Output, sisäänmeno/ulostulo
ID	Identification
Jenkins	Java-pohjainen CI-palvelinohjelma
JUnit	Ohjelmistokehys toistettavien yksikkötestien toteutukseen
K-tyyppi	Termoparianturin tyyppi
Linux	Avoimen lähdekoodin käyttöjärjestelmä
NI LabView	National Instruments LabView, järjestelmäsuunnitteluun tarkoitettu graafinen ohjelmistoympäristö

NI TestStand	National Instruments TestStand, PXI –järjestelmien testien hallintaan suunniteltu ohjelmisto
NULL-osoitin	Komento, joka osoittaa ohjelman tyhjään osoitteeseen
OC	Overcurrent, ylivirta
OV	Overvoltage, ylijännite
PCI	Peripheal Component Interconnect
PSD	Power Supply Driver
PSS	Power Supply System
PT1000	Lämpötila-anturi
PTCL	Procket Test Command Language
PXI	PCI eXtensions for Instrumentation, PCI laajennus instrumentointiin
R&D test platform	Research and Development test platform, tuotekehitystestausalusta
TC	Thermocouple, termopari
TCL Expect	Ohjelmien skriptaamien luomiseen tarkoittettu TCL –liitännäinen
TCL	Tool Command Language
Tcltest	Virallinen TCL yksikkötestauskehys
TCP/IP	Tietoliikenneprotokollaperhe, jossa TCP on kuljetuskerros ja IP on verkkokerros
TEMP	Temperature
TTL	Transistor-Transistor-Logic, transistori-transistori-logiikka
USB	Universal Serial Bus
VBUS	VoltageBUS, USB:n jänniteväylä
Vout	Voltage Output, jänniteulostulo
WB	Wirebreak, johdinkatko
Xenomai	Reaaliaikainen kehityskehys, joka toimii yhteistyössä Linuxin kernelin kanssa
XUnit	Yksikkötestauksen ohjelmistokehys

1 JOHDANTO

Sulautettujen järjestelmien tuotekehityksen aikainen testaus on usein aikaa vievää ja monimutkaista. Testauksen automatisoinnille on tästä syystä suuri tarve. Testauksen merkitys korostuu etenkin sulautetuissa järjestelmissä, joissa elektroniikka tuo lisähaasteita testauksen kattavuudelle ja toistettavuudelle. Valmiita automatisoituja ratkaisuja on tarjolla, mutta ne ovat usein vielä kalliita ja järjestelmän pystyttäminen hidasta.

Perinteinen manuaalisesti suoritettu testaus on hidasta ja vaatii yleensä useita työtunteja testien suorittamiseen. Automatisoinnilla pyritään lyhentämään tuotekehityksen aikana testaukseen kuluvaa aikaa, unohtamatta asiakkaiden tuotteilleen asettamia laatuvaatimuksia. Samalla automatisointi parantaa testauksen toistettavuutta, kun inhimillisten virheiden mahdollisuus pienenee. Automatisointi vaatii kuitenkin lisätyötä, joka on kohdennettava testausohjelmiston kehitykseen ja testausjärjestelmän kokoamiseen.

Tämän diplomityön tarkoituksena oli toteuttaa edullinen ja ohjelmistokehittäjien tarpeisiin soveltuva automatisoitu testausjärjestelmä sulautetun ohjelmiston ja elektroniikan testaukseen. Tarkoitus oli suunnitella järjestelmästä helppo sekä käytettävyydeltään että järjestelmän kokoamiselta. Tällä tavalla järjestelmä olisi helpompi ottaa käyttöön tulevissa projekteissa.

Testausjärjestelmä luotiin apuvälineeksi tuotekehitysprojektiin. Projektin laitteet ovat keskeisessä roolissa testausjärjestelmässä. Testattavista laitteista esitellään tässä työssä kuitenkin vain testauksen kannalta välttämättömät laitteistomääritykset, sen sijaan että kuvattaisiin laitteiden yksityiskohtaista toimintaa.

Luku 2 esittelee tämän työn pohjana olevia ongelmia, joita lähdettiin ratkomaan. Luvussa esitellään toteutettavan järjestelmän valintaperusteita verrattuna valmiisiin ratkaisuihin. Lopuksi kerrotaan sulautettujen järjestelmien perusteita.

Luvussa kolme käydään läpi työn teoriataustaa, esitellään ohjelmiston suunnitteluprosessia ja sulautettuja järjestelmiä. Samalla käsitellään ohjelmistotestauksessa käytettäviä prosessimalleja ja erityispiirteitä sekä teorioita ohjelmistosuunnittelun taustalla.

Tämän jälkeen luvussa neljä käsitellään testauksen automatisointia yleisellä tasolla ja ohjelmistotestauksessa käytettävien automatisointien hyötyjä ja haittoja. Näiden lisäksi esitellään automatisoinnin kannalta järkeviä testitapauksia ja jatkuvaa integrointia.

Luvussa viisi esitellään testijärjestelmän suunnittelua ja komponenttivalintoja. Samalla käydään läpi testilaitteet, testattavat laitteet ja testikytkennät, joilla halutut testaukset saadaan tehtyä. Testikytkennöistä esitellään eri liityntätyyppien ominaisuudet ja kuvina varsinaiset kytkennät.

Luvussa kuusi käsitellään ohjelmistotestauksen tavoitteita, testausstrategiaa ja sen eri vaiheita. Lisäksi esitellään ohjelmistotestauksessa käytettyjä testaustyökaluja sekä tapauskohtaisesti testisekvenssejä ja niiden tuloksia.

Viimeisessä luvussa esitellään yhteenveto tuloksista, jotka tämän työn aikana saavutettiin. Samalla kootaan yhteen toteutetun järjestelmän tuloksia ja verrataan niitä aikaisemmin esitettyihin vaatimuksiin. Lopuksi pohditaan mahdollisia jatkokehitysideoita ja parannusehdotuksia.

2 TAUSTA

2.1 Lähtökohta

Työn suunnittelu aloitettiin, koska Espotel Oy:ssä havaittiin, että suuren luokan tuotekehitysprojekteissa ohjelmistotestaus oli yksi merkittävimmistä hidastavista tekijöistä. Ohjelmistotestausta haluttiin automatisoida asiakkaan tilaamassa projektissa, jossa suunnitellaan sulautetun järjestelmän laitteisto. Ohjelmiston lisäksi projektissa testataan suunniteltua elektroniikkaa ja näiden välistä toimintaa. Projektin laitteiden mittaustarkkuuksien ja niihin liittyvien toteutusosien testauksessa käytetään tuotekehitystestausalustaa (R&D test platform), josta vastaa Espotel Oy:n oma testausosasto.

2.1.1 Tuotekehitystestausalusta

Projektissa käytössä oleva tuotekehitystestausalusta on luotu National Instrumentin PXI-järjestelmän pohjalta. Tämä mahdollistaa valmiiden PXI-testausmoduulien käytön, joiden avulla testausjärjestelmä on muunneltavissa eri tarpeiden mukaan. Testausalustassa käytetään National Instrumentin PXI-järjestelmille suunniteltuja NI Labview- ja NI TestStand -ohjelmistoja. NI LabView on järjestelmäsuunnitteluun tarkoitettu ohjelmisto, joka mahdollistaa graafisen ohjelmointiympäristön. NI TestStand on testien hallintaan suunniteltu ohjelmisto, jonka avulla luodaan, suoritetaan ja otetaan käyttöön testiohjelmia. Sen avulla voidaan luoda myös testisekvenssejä. (Procket; NI-PXI.)

Projektissa tuotekehitystestausalustaa käytetään testattavien moduulien toiminnallisuuden ja ennen kaikkea tarkkuuksien testaukseen. Tärkeimpänä ominaisuutena voidaan pitää testien toistettavuutta, jolloin voidaan verifioida laitteille tehdyt muutokset ja niiden vaikutukset eri ympäristöissä, kuten eri lämpötiloissa. Saatuja tuloksia analysoidaan ja verrataan asiakkaan vaatimusmäärittelyihin.

2.1.2 Järjestelmävalinnat

Testausjärjestelmän suunnittelun tärkeimpänä vaatimuksena pidettiin kustannustehokkuutta. Tällä rajattiin pois valmiita testausjärjestelmiä, joiden hankintakustannukset olisivat olleet jo liian suuret. Toinen kriitiinen tekijä järjestelmävalinnoissa oli soveltuvuus ohjelmiston testaukseen ohjelmistokehittäjille tutuilla ohjelmointikielillä. Tällöin kehitystyö ei vaatinut uusien ohjelmointikielien opiskelua tai asiantuntijoiden palkkaamista, vaan kehitystyö voitiin aloittaa jo heti suunnittelutyön alettua.

Aikaisempia kokemuksia vastaavanlaisista järjestelmistä ja toiminnallisuuksista kartoitettiin yrityksen sisällä ja ilmeni, että Espotelin omalla tuotekehitystestausalustalla ei olisi järkevää toteuttaa järjestelmää. Sen kustannukset olivat budjettiin nähden liian suuret, toteutus veisi paljon aikaa ja NI LabView ja NI TestStand ohjelmointialustana olisivat haasteellisia ohjelmoida ja integroida projektin kehitysympäristöön.

Edellä mainituista syistä päädyttiin toteuttamaan järjestelmä itse. Samalla mahdollistettiin testausjärjestelmän ja varsinaisen ohjelmiston rinnakkainen kehittäminen. Tällöin samat kehittäjät seurasivat molempien toteutusta ja näin kyettiin reagoimaan virheisiin nopeammin. Kappaleissa 5 ja 6 käsitellään tarkemmin järjestelmän suunnittelua ja valintaperusteita.

2.2 Sulautettu järjestelmä

Sulautettu järjestelmä (Embedded system) on laitteiston ja ohjelmiston yhdistelmä. Lisäksi se voi sisältää esimerkiksi mekaanisia tai elektronisia lisälaitteita. Yksittäistä tarkkaa määritelmää sulautetulle järjestelmälle ei kuitenkaan ole ole mahdollista tehdä.

Sulautetun järjestelmän tarkoitus on yleensä suorittaa jotain määrättyä tehtävää, toisin kuin perinteiset tietokoneet, joiden käyttökohteita ei ole ennalta määritelty. Yleensä sulautetut järjestelmät ovat osa jotain suurempaa järjestelmää, jossa niitä voi olla useita. Esimerkkinä nykyaikaiset autot, joissa valtaosa toiminnoista hoidetaan ohjelmallisesti. Tällaisissa isoissa järjestelmissä laitteiden välille luodaan kommunikaatioverkko, esim. CAN-väylän avulla. (Barr & Massa 2006.)

Prosessori ja ohjelmisto ovat tunnusomaisia sulautetuissa järjestelmissä, mutta käyttäjä ei välttämättä huomaa niiden olemassaoloa. Monessa tapauksessa ne tekevätkin paljon taustatyötä, mikä kuitenkin näkyy käyttäjälle vain pienenä muutoksena. Prosessori ja ohjelmisto on mahdollista korvata erillisellä integroiduilla piirillä ja näin toteuttaa halutut funktiot järjestelmässä. Prosessorien ja ohjelmiston käyttöä tukee kuitenkin niiden joustavammat toteutukset, helppokäyttöisyys, hinta ja vähäinen virrankulutus. (Barr & Massa 2006.)

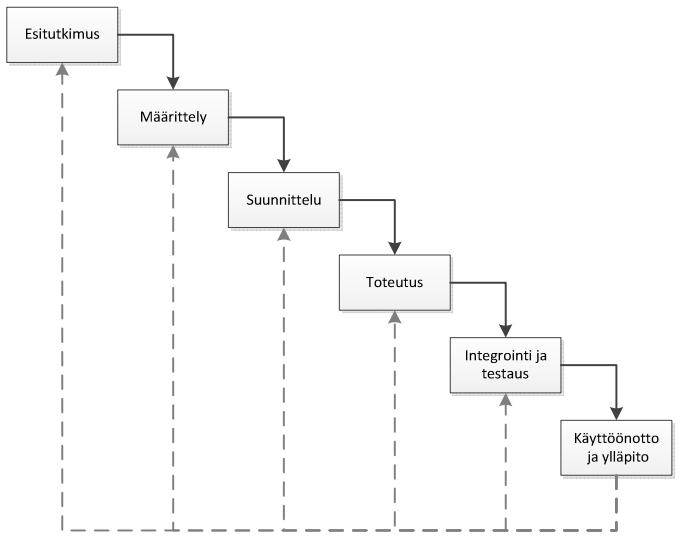
3 OHJELMISTOSUUNNITTELU

3.1 Ohjelmistotestaus

Ohjelmiston testaus on yksi tärkeimmistä osista ohjelmistokehityksessä. Pelkkä koodin kirjoittaminen ja kääntäminen ei takaa luotettavaa tulosta. Testauksen tarkoituksena on todistaa, että ohjelma tekee niin kuin on määritelty, toiminnot toimivat oikein eikä ohjelmassa ole virheitä. Asian voi kiteyttää kuten Glenford J. Myers *The Art of Software Testing* –kirjassaan: “Software testing is the process of executing a program or system with the intent of finding errors” (Myers 2004, s. 11). Toisin sanoen voidaan olettaa, että ohjelmistossa on virheitä ja tarkoituksena on löytää ne testaamalla.

3.1.1 Vesiputousmalli

Ohjelmistotuotanto voidaan kuvata peräkkäisinä vaiheina, josta ilmenee ohjelmiston elinkaari. Prosessi voidaan jakaa eri vaiheisiin kuten esitutkimus, määrittely, suunnittelu, toteutus ja testaus. Lopuksi jäljelle jää valmiin tuotteen ylläpito. Tätä suoraviivaista prosessia kutsutaan vesiputousmalliksi, jonka yksi esimerkki on esitetty kuvassa 3.1. Mallista on tehty useita eri muunnelmia, mutta kaikista niistä on eroteltavissa määrittely-, suunnittelu- ja toteutusvaiheet. (Pressman 2010; Haikala & Märijärvi 2000.)

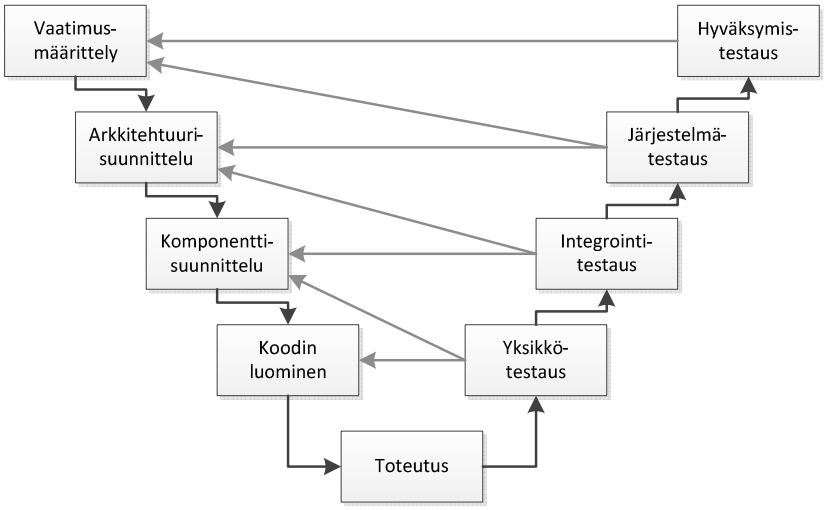


Kuva 3.1 *Esimerkki ohjelmistokehityksen vesiputousmallista (Haikala & Märijärvi 2000)*

Vesiputousmalli on esimerkki suunnitteluohjautuvasta prosessista, jossa prosessin vaiheiden suunnittelu ja aikataulutus on tehtävä ennen kuin niiden toteuttaminen aloitetaan. Seuraavaan vaiheeseen tulisi siirtyä vasta, kun edellinen on saatu valmiiksi. Käytännössä vaiheiden välillä on päällekäisyyttä. Esimerkiksi suunnitteluvaiheessa havaitaan vaatimusten virheet tai toteutusvaiheessa havaitaan suunnittelun virheet. Usein kuitenkin aikataulutus vaatii nopeaa testausta, joten suunnittelua joudutaan karsimaan. (Sommerville 2011.)

3.1.2 V-Malli

Vesiputousmallista on esitetty erilaisia variaatioita, joista yksi on V-mallin prosessimalli (Kuva 3.2). V-mallin mukaisessa prosessissa havainnollistetaan suunnittelun ja testausprosessin välisten vaiheiden yhteys. Testauksen tasoilta todetut ohjelmiston toimivuuden verifiointi ja validointi yhdistyvät aikaisemmin esitettyihin suunnitelmiin ja toteutuksiin. (Pressman 2010.)



Kuva 3.2 *Ohjelmistokehityksen V-malli (Pressman 2010)*

3.2 Ohjelmistotestauksen tasot

Ohjelmistotestauksen toteutukseen on olemassa monia vaihtoehtoisia toteutustapoja. Yksinkertaisin, mutta samalla myös radikaalein, on toteuttaa testaus, kun järjestelmä on saatu muuten valmiiksi. Tällöin testaus suoritetaan koko järjestelmälle ja toivotaan, että ohjelmiston virheet löytyvät. Tällainen toteutustapa on kuitenkin epävakaa ja koodiin jää helposti virheitä. Ratkaisuna voidaan käyttää testausstrategiaa, jossa testaus aloitetaan V-mallin mukaisesti alhaalta yksikkötestauksesta, edeten kohti huippua ja hyväksymistestausta. (Pressman 2010.) Jokainen testauksen taso esitellään erikseen seuraavaksi.

3.2.1 Yksikkötestaus

Yksikkötestauksessa (Unit testing, Module testing) keskitytään ohjelman pienimpien yksikköjen eli komponenttien tai moduulien testaukseen. Ennen kuin ohjelman kokonaistestausta on järkevää testata, jaetaan se pienempiin osiin, joihin yksikkötestauksessa keskitytään. Yksittäistä moduulia testaamalla ohjelmasta on helpompi löytää ja osoittaa virheellinen kohta juuri kyseiseen moduuliin. Testauksen aikaisessa vaiheessa löydetty virhe estää sen siirtymisen ylempiin testauksen vaiheisiin ja näin vähentää siitä johtuvia kustannuksia. Yksikkötestauksen toinen kustannustehokas ominaisuus on moduulien rinnakkaistestaus, jolloin useita moduuleita on mahdollista testata rinnakkain. (Pressman 2010; Fewster & Graham 2003; Myers 2004.)

3.2.2 Integrointitestaus

Integrointitestauksella (Integration testing) tarkoitetaan ohjelmistoarkkitehtuurin rakentamista ja yksikkötestattujen moduulien liittämistä toisiinsa. Tällöin testattavat moduulit liitetään toisiinsa ja testataan liittämisestä aiheutuneiden virheiden löytämiseksi. Tavoitteena on luoda ohjelmistorakenne, joka vastaa ohjelmistosuunnitelussa tehtyjä suunnitelmia. (Pressman 2010; Myers 2004.)

Integrointitestauksessa käytetään tavallisesti kahta erilaista testaustapaa. Ei-inkremantaalinen eli ”big-bang” ja inkrementaalinen tapa. Ei-inkremantaalisessa tavassa jokainen moduuli testataan yksin, jonka jälkeen ne kaikki yhdistetään kokonaisuudeksi. Testaus vaatii jokaiselle moduulille omat ajuri- ja tynkämoduulit, jotka lisäävät testaustavan työmäärää.

Inkrementaalinen testaustapa on ei-inkrementaaliselle tavalle vastakohta. Testaus ja ohjelman kokoaminen tehdään pienissä osissa kasvattaen askelin ohjelman kokoa. Tällöin virheiden havainnointi ja paikannus on helpompaa, jolloin kaikki moduulien väliset liitynnät tulevat testattua tarkemmin. (Myers 2004; Pressman 2010.)

3.2.3 Järjestelmätestaus

Järjestelmätestaus (System testing) jatkaa integrointitestauksessa saatujen komponenttien yhteen liittämistä ja järjestelmän luomista. Testauksessa varmistetaan, että luodut komponentit ovat yhteensopivia, toimivat keskenään oikein, sekä siirtävät oikean datan oikeaan aikaan niiden välillä. Tässä vaiheessa ohjelmistoprosessia voidaan ensimmäistä kertaa liittää eri osapuolten tekemät komponentit yhteen ja testata ohjelmistojärjestelmä kokonaisena. Järjestelmätestauksella on myös tarkoitus todentaa, että ohjelma vastaa sille asetettuihin järjestelmävaatimuksiin. (Sommerville 2011.)

3.2.4 Hyväksymistestaus

Hyväksymistestaus (Acceptance testing) on testauksen taso, jossa verrataan toteutettua ohjelmaa vaatimuksiin ja testataan, vastaako se loppukäyttäjän tarpeita.

Hyväksymistestaus jaetaan kahteen osaan, alfa- ja beta-testaukseen. Alfa-testauksen suorittaa yleensä ohjelmiston kehittänyt osapuoli. Kehittäjien on kuitenkin lähes mahdonta ennustaa, kuinka ohjelmistoa tullaan lopulta käyttämään. Tästä syystä hyväksymistestauksen suorittaa usein asiakas tai loppukäyttäjät. Tätä kutsutaan beta-testaukseksi ja toisin kuin alfa-testauksessa, ohjelmiston kehittäjät eivät osallistu testaukseen. Testaajat kirjaavat testitulokset ja virheet ylös ja raportoivat kehittäjäosapuolta näiden korjaamiseksi. (Pressman 2010; Myers 2004.)

3.2.5 Regressiotestaus

Regressiotestaus (Regression testing) tarkoittaa testausta, jolla varmistutaan, että ohjelmiston aikaisemmin testatut toiminnallisuudet toimivat edelleen. Regressiotestaus ei ole testauksen taso kuten yksikkötestaus tai integrointitestaus, vaan se on ohjelman uudelleen testausta, jota voidaan suorittaa eri testauksen tasoilla.

Ohjelmiston testauksessa pyritään löytämään koodin virheellisiä toimintoja ja sen jälkeen korjaaman ongelmat. Korjaukset ja uusien ominaisuuksien lisääminen voivat kuitenkin aiheuttaa aikaisemmin toimineisiin ominaisuuksiin epähaluttuja toimintoja tai virheitä. Regressiotestauksella pyritään todistamaan vanhojen toiminnallisuuksien toimivuus uudemmissakin ohjelmistoversioissa. Testauksessa ajetaan läpi aikaisemmin suoritetut testitapaukset, mikä voi olla erittäin aikaa vievää, varsinkin kun useita tai suuria ohjelmistojulkaisuja tehdään. Tällöin erityisen tärkeäksi nousee sopivien testitapausten valinta sekä testien automatisointi. (Pressman 2010; Burnstein 2003.)

4 TESTAUKSEN AUTOMATISOINTI

Automatisoinnilla tarkoitetaan jonkin tehtävän suorittamista ilman, että käyttäjän tarvitsee itse suorittaa toimenpiteitä. Testauksen automatisoinnissa testit, jotka käyttäjä muuten tekisi, tehdään koneellisesti. Automatisointi ei kuitenkaan kokonaan poista käyttäjän tarvetta, sillä käyttäjää tarvitaan luomaan testitapaukset ja testiautomatiikka. Pitkälle kehitetty testausautomatiikka vaatii käyttäjää vain testauksen käynnistämiseen. (Fewster & Graham 1999.)

4.1 Testitapaukset

Sekä automatisoidut että manuaalisesti tehtävät testaukset vaativat toimiakseen erilaisia testitapauksia (Test cases). Testitapauksella tarkoitetaan kuvausta siitä, miten jotain testataan. Testitapauksia voi olla ääretön määrä ja niiden kaikkien suorittaminen käytännössä mahdotonta. Testitapauksista on osattava valita tarvittavat ja tärkeät testit, joilla saavutetaan mahdollisimman kattavasti virheiden paikantaminen. Testaajalta, joka toimii usein testitapausten suunnittelijana, vaaditaan taitoa rajata olennaisimmat testitapaukset turhien joukosta.

Testitapausten tärkein ominaisuus on virheiden löytämisen tehokkuus. Toinen tärkeä ominaisuus on yksittäisen testitapauksen kattavuus. Mitä useamman asian testi kattaa, sitä vähemmän tarvitaan erillisiä testitapauksia ja näin saadaan testeistä kustannustehokkaampia. Toisaalta virheiden ilmetessä niiden paikallistamisesta ja syyn selvittämisestä tulee hankalampaa. (Fewster & Graham 1999.)

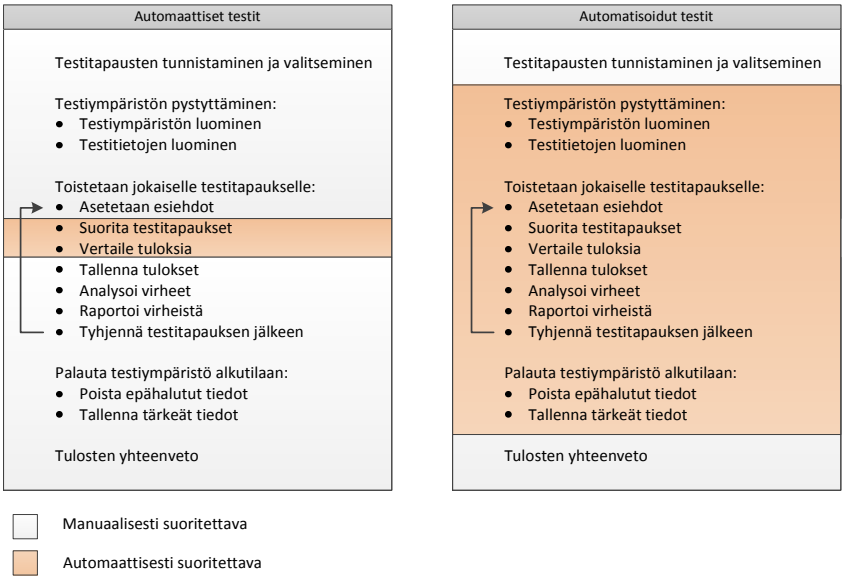
4.2 Testiautomaatio

Testien automatisointi voi yllättää yritykset kustannuksillaan, jotka voivat olla huomattavasti suuremmat kuin manuaalisesti toteutetussa testauksessa. Jotta automatisoinnista on hyötyä, tulee testit suunnitella ja toteuttaa huolella. Automatisoinnin laatu riippuu testien laadusta.

Epäolennaisten asioiden testaus tuottaa epätoivottuja tuloksia, eikä väärien asioiden testauksen automatisointi auta asiaa, vaan tuottaa epäitoivoittuja tuloksia nopeammin. Automatisoinnin toteutuksen jälkeen testaus on taloudellisesti kannattavampaa kuin manuaalisesti suoritettu, ja automatisoinnin myötä uusien testitapausten luomisen pitäisi olla helpompaa ja nopeampaa. Kun huomioidaan automaation ylläpito ja päivityksestä aiheutuvat kulut, kustannukset nousevat suuremmiksi kuin manuaalisesti kaikkien testien suorittaminen. Automatisoinnin suunnittelussa onkin erityisesti huomioitava testien toistettavuus, jolloin testauksesta tulee kustannustehokasta. (Fewster & Graham 1999.)

Testitapaukset vaativat usein esiprosessointia (pre-processing) ennen kuin varsinainen testaus voidaan aloittaa. Tämä pätee sekä manuaalisissa että automaattisissa testauksissa. Samoin testien jälkeen tarvitaan tulosten tallentamista, virheanalyysiä ja niistä raportointia eli jälkiprosessointia (post-processing). Esi- ja jälkiprosessointi ovat hyviä ja tehokkaita automatisoitavia tehtäviä. Testauksen automatisoinnin määrällä on suuri vaikutus testauksen tehokkuuteen.

Kuvassa 4.1 on esitetty kaksi erilaista testauskattavuuden esimerkkiä, automaattiset testit ja automatisoidut testit sekä havainnollistettu niiden välisiä eroja. Ensimäisessä vaihtoehdossa esi- ja jälkiprosessointi tehdään manuaalisesti, mutta itse testiprosessi tehdään automatisoidusti. Testaajalta kuluu aikaa testiympäristön luomiseen ja testien käynnistämiseen sekä testin jälkeiseen prosessiin. Järkevämpi vaihtoehto on automatisoida koko testaus, jolloin testaaja vapautuu muihin tehtäviin. Testaajalle testausprosessista jää vain olennaisten testitapausten tunnistaminen ja valitseminen sekä lopullisten tulosten yhteenveto.



Kuva 4.1 *Automaattisten testien ja automatisoidun testauksen erot. (Fewster & Graham 1999)*

4.3 Automatisoinnin hyödyt

Testien automatisoinnilla on mahdollista toteuttaa testejä, joiden tekeminen manuaalisesti veisi äärettömästi aikaa tai joita olisi jopa mahdotonta tehdä. Regressiotestauksen automatisointi on yksi selkeimmistä hyödyistä automatisoinnille. Testauksessa suoritetaan testitapauksia, jotka ovat jo aikaisemmin ajettu, mutta automatisoinnilla voidaan suorittaa ne halutussa järjestyksessä uudelleen. Tällöin voidaan varmistua etteivät ohjelmaan tehdyt muutokset ole aiheuttaneet virheellisiä toimintoja muualla. Manuaaliseen testaukseen verrattuna automatisointi mahdollistaa useampien testien ajamisen useammin ja lyhyemmässä ajassa. Näin saadaan

luotettavampi kokonaiskuva järjestelmästä. Testijärjestelmällä voidaan suorittaa testejä, joiden tekeminen ilman automatisointia olisi hidasta tai hankalaa. Tämä korostuu varsinkin, kun testataan sulautettuja järjestelmiä, joissa ulkoisten syötteiden määrä voi olla suuri. Sisäänmenevät syötteet pysyvät testikertojen välillä muuttumattomina, jolloin vain ulostulon muutoksia tarvitsee seurata. (Fewster & Graham 1999.)

Ohjelmistoprosessia automatisoimalla saadaan henkilöstöresursseja tehokkaammin hyödynnettyä. Testaus voi olla yksitoikkoista työtä, joka on helposti korvattavissa automatisoinnilla. Tällöin testaaja voidaan vapauttaa suunnittelemaan parempia testejä. Tehokkaampi ja nopeampi testaus lyhentävät myös ohjelmistokehitykseen kuluvaa aikaa ja ohjelmiston julkaisut nopeutuvat. Automatisointia suunnitellessa on huomioitava, että kaikkia testejä ei voida automatisoida. Kuitenkin automatisoinnilla saavutetaan perusteellisempi testaustulos vähemmällä vaivalla sekä parannetaan laatua ja tuottavuutta. (Fewster & Graham 1999.)

4.4 Automatisoinnin haitat

Automatisoinnin hyötyjen lisäksi siinä on myös haittoja. Osa ongelmista ilmenee vasta automatisointia tehtäessä ja ne tulevat yllätyksenä käyttäjälle. Automatisointia suunnitellessa tehdään helposti oletus, että uudet työkalut korjaavat kaikki olemassa olevat ongelmat. Samalla unohdetaan kuinka työkalujen käyttönotto voi olla hidasta ja automatisoidun järjestelmän luominen vie henkilöstöresursseja. Huonosti suunnitellut ja organisoidut testit ja testitapaukset tuottavat huonoja tuloksia. On siis järkevämpää keskittyä löytämään oikeat testit ja siten parantaa testien laatua, kuin parantaa huonojen testien tehokkuutta. (Fewster & Graham 1999.)

Liiallinen luotto testitapauksiin voi antaa virheellisiä tuloksia. Vaikka testauksessa ei löytyisikään yhtään virhettä, se ei tarkoita, että niitä ei voisi olla itse testiohjelmassa. Testiohjelmat on saatava toimimaan oikein ennen kuin luotetaan testituloksiin. Monimutkaisissa laitteissa testauksella ei kuitenkaan päästä kovinkaan hyvään testauskattavuuteen. Automatisoitua testijärjestelmää tulee ylläpitää ja päivittää. Nämä voivat pysäyttää testauksen pitkiksikin ajoiksi ja jopa keskeyttää sen kokonaan. Tällöin huomataan, että manuaalisesti testit oltaisiin jo suoritettu. (Fewster & Graham 1999.)

4.5 Jatkuva integrointi

Ohjelmiston pienet osat tulee jossain vaiheessa koota ensin isommiksi kokonaisuuksiksi ja sen jälkeen näistä koostetaan uusi ohjelmistoversio. Yksittäin testatut ohjelmiston osat tulisi saada toimimaan saumattomasti keskenään, mutta osia integroitaessa voidaan havaita ohjelmiston toimimattomuutta. Testauksen jakaminen eri testauksen tasoille auttaa vikojen paikallistamisessa, mutta usein integrointi tehdään vasta prosessin myöhäisemmässä vaiheessa ja suurissa osissa, jolloin ei ole helppoa todentaa, mikä osa aiheutti vikaantumisen. Jos projektissa integrointivaihetta ei osata aikatauluttaa,

integrointiin kuluva aika voi tulla yllätyksenä ja aiheuttaa viivästyksiä. (Kawalerowicz & Berntson 2011.)

Ongelmiin on ratkaisuna jatkuva integrointi (CI, engl. Continuous integration). Sen perusideana on integroida suunnittelijoidan työtä mahdollisimman aikaisessa vaiheessa ja jatkuvasti. Tämä antaa suunnittelijoille mahdollisuuden muuttaa ohjelmakoodia tietäen, että jos jokin osa hajoaa, siitä saadaan nopeasti palautetta. (Humble & Farley 2010.)

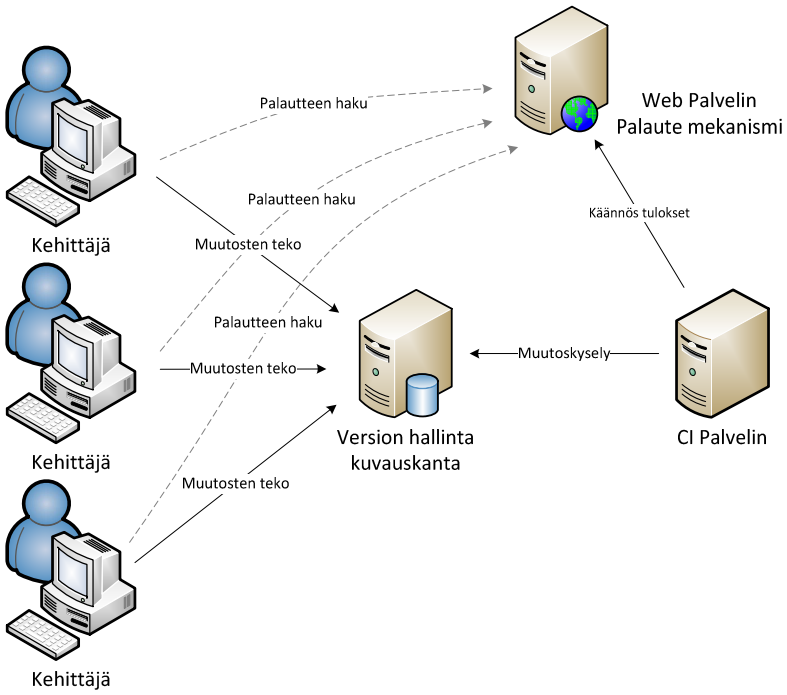
Testauksen tapaan jatkuva integrointi on järkevää automatisoida, jolloin integrointi tehdään automaattisesti jokaisen tehdyn muutoksen jälkeen tai määrättyinä aikoina, esimerkiksi joka yö. Automatisoidun integroinnin osana ajetaan tarvittavat automatisoidut testit, jotta varmistutaan ohjelman toimivuudesta. Ohjelmiston tuotantoprosessi vaatii jatkuvaa testausta, tulosten seurantaa ja tulosten perusteella tapahtuvaa virheiden korjaamista. Kun havaitaan virhe joko ohjelman koostamisessa tai testausprosessissa, ohjelmiston kehitys voidaan keskeyttää ja korjata virheet välittömästi. Tällä tavalla tiedetään, että uusin versio ohjelmasta on toimiva, joten ohjelma pysyy koko ajan toimivassa tilassa. Jatkuva integrointi on ennen kaikkea tapa kommunikoida osapuolten välillä niin, että jokainen voi helposti nähdä järjestelmän tilan ja muutokset, jotka siihen on tehty. (Kawalerowicz & Berntson 2011; Humble & Farley 2010; Fowler 2006.)

4.5.1 Jatkuvan integroinin prosessi

Jatkuvaa integrointia varten markkinoilla on useita tuotteita, jotka tarjoavat pohjan ohjelmiston automaattiselle koostamiselle ja testausprosessille. Yksinkertaisimmillaan jatkuvan integroinnin ohjelma tarkistaa versiohallinnasta, onko muutoksia tehty. Jos muutoksia on tehty, ohjelma hakee viimeisimmän version ja ajaa käännöksen ja testit. Viimeisenä ohjelma ilmoittaa käyttäjälle tuloksista. (Humble & Farley 2010.)

Käytännössä jatkuvan integrointin runkona on CI-palvelin (Continuous Integration Server), jolla on kaksi tehtävää. Ensiksi se suorittaa yksinkertaisia toimenpiteitä säännöllisin väliajoin. Toiseksi CI-palvelin tarjoaa suoritetuista prosesseista tulosten seurannan ja ilmoittaa onnistumisista tai epäonnistumisista. Kuvassa 4.2 on esitetty tyypillinen jatkuvan integroinnin prosessikaavio, jossa CI-palvelimelta lähetetään säännöllisin väliajoin muutoskyselyjä version hallinnan kuvauskantaan (Version control repository). Jos havaitaan muutoksia, kopioidaan tiedostot palvelimelle ja suoritetaan halutut komennot. Tyypillisesti tehdään ohjelmasta käännös ja ajetaan tarpeelliset automaattiset testit. (Humble & Farley 2010.)

Useimmat CI-palvelimet sisältävät web-palvelimen, joka pitää yllä listaa käännösversioista ja mahdollistaa raporttien tutkimisen. Kokonaisuudessaan palvelimet toimivat ohjelman binäärien ja asennustiedostojen varastona, josta testaajien ja asiakkaiden on helppoa hakea uusimmat versiot ohjelmasta. (Humble & Farley 2010; Kawalerowicz 2011.)



Kuva 4.2 *Jatkuvan integroinnin prosessikaavio. (Kawalerowicz 2011)*

4.5.2 Jatkuvan integroinnin työkalut

Jatkuvan integroinnin hallintaan on useita työkaluja, joista tunnetuimpia CruiseControl, DamageControl ja Jenkins. Testausjärjestelmässä päädyttiin Jenkinsiin, koska siitä oli yrityksessämme aikaisempaa kokemusta. Jenkins on Java-pohjainen CI-palvelinohjelma, joka ohjaa toistettavien tehtävien suoritusta. Jenkins keskittyy ohjelmiston kääntämiseen ja testaukseen jatkuva-aikaisesti, mikä automatisoituna parantavaa helppokäyttöisyyttä ja kasvattaa tuottavuutta. (Fowler 2006; Berg 2012; Jenkins.)

5 AUTOMATISOITU TESTAUSJÄRJESTELMÄ

5.1 Testausjärjestelmän laitteistovaatimukset

Sulautettujen ohjelmistojen testausta varten suunniteltiin testausjärjestelmä, jolla pystyttiin automatisoimaan testausta. Suunnittelulähtökohtana oli saada aikaan kustannustehokas, mutta samalla mahdollisimman kattavan testauksen mahdollistava järjestelmä. Valmiit ratkaisut, kuten National Instrumentin PXI-alustan järjestelmät hylättiin, liian suurien kustannusten takia.

Testausjärjestelmän vaatimuksiin vaikuttivat ohjelmiston testauskattavuuden rajaus ja testattavien laitteiden tekniset vaatimukset. Testitapauksien määrää rajoitettiin myös valitsemalla testattaviksi kanaviksi yksi kutakin kanavatyyppiä. Jos samaa kanavatyyppiä oli laitteessa useita, kuten esimerkiksi Analog Input -kanavissa 1-5 (AI1-5, testattiin kanavatyypin toiminta vain yhdellä kanavalla. Tarkemmat laitteistovaatimukset on esitelty taulukossa 1. Laitteistovaatimukset pohjautuvat osittain testattavien moduuleiden vaatimuksiin.

Taulukko 1 *Testausjärjestelmän laitteistovaatimukset.*

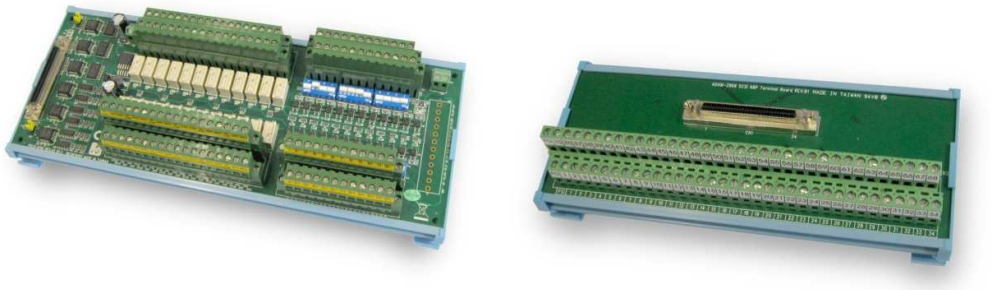
Vaatimus	Tarkennus
Teholähde 24V @ 2A testilaitteiden tehonsyöttöön	Releohjaus jännitteiden päälle/pois kytkemiseen Releohjaus tehonsyötön valintaan (1 tai 2 PSS)
Teholähde 24V @ 2A DRV kanavien tehonsyöttöön	Releohjaus jännitteiden päälle/pois kytkemiseen Releohjaus tehonsyötön valintaan (1 tai 2 PSS)
Mahdollisuus signaalien reititykseen	Releohjaus signaalien reititykseen
Mahdollisuus lukea digitaalisia sisäänmenoarvoja	Jänniteulostulo laitteesta Kytkintyyppinen ulostulo laitteelle
Mahdollisuus luoda analogisia ulostulosignaaleja	4 - 20mA virtasilmukka -10mV - 30mV termopari simulaatio 0-20V jänniteulostulo -500mV - 500mV audiosignaali ulostulo
Mahdollisuus mitata analogisia sisäänmenosignaaleja	Jännitemittaus 0 - 5V Virtamittaus 0 - 200mA

5.2 Testausjärjestelmän laitteisto

Automatisoidun testausjärjestelmän rungoksi valikoitui Linux-työasema-PC, jonka lisäksi laitteistovaatimusten perusteella päädyttiin Advantechin teollisuusautomaation tiedonkeruu- ja ohjausratkaisuihin. Työasema-PC:n rajallinen PCI-paikkojen määrä rajasi automaatiokorttien määrän kolmeen. PCI-kortit liitettiin I/O-terminaaleihin, joiden avulla testilaitteet oli mahdollista kytkeä järjestelmään.

5.2.1 Mittauslaitteisto

Laitteistovaatimusten (Taulukko 1) mukainen releohjaus toteutettiin Advantechin PCI-1751-AE Digital I/O -kortilla, joka on kytketty SCSI-68 -kaapelilla PCLD-8761 Digital input / Relay -korttiin. Yhdistelmä nimettiin Relemoduuliksi. Analoginen ulostulo-moduuli (AO-moduuli) toteutettiin PCI-1723 Analog output -kortilla, joka yhdistettiin ADAM-3968 I/O-terminaali -korttiin. Analoginen sisäänmeno-moduuli (AI-moduuli) toteutettiin kytkemällä PCI-1711L Multifunction kortti ADAM-3968 terminaaliin. Kuvassa 5.1 on esitetty laitteistossa käytetyt Advantechin terminaalikortit.



Kuva 5.1 *Advantechin PCLD-8761 ja ADAM-3968 terminaalikortit.*

Mittausmoduulien määrittelyt on esitetty taulukossa 2. Relemoduulin ohjauskortissa PCI-1751 on 48 digitaalista I/O-kanavaa, jotka jakautuvat PCLD-8761 moduulissa 24:ään isoloituun digitaaliseen sisäänmenoon ja 24 releeseen. Digitaalisen sisäänmenon jännitealue on 0-30V ja releiden jännitteenkesto 30Vdc @ 1A.

AO-moduulin ohjauskortissa PCI-1723 on kahdeksan analogista ulostuloa, jotka toimivat sekä jännite- että virtalähtöinä. Jännitealue on säädettävissä -10V - +10V alueelta ja virta-alue 0 – 20mA alueelta. Digitaalisia I/O-kanavia moduulissa on 16 kappaletta ja logiikkatasoina käytetään 5V:n TTL-tasoja.

AI-moduulissa on 16 analogista sisäänmenokanavaa, joiden mittausalue on -10V - +10V. Digitaalisia kanavia on yhteensä 32, joista 16 on sisäänmenoja ja toiset 16 ulostuloja. Kanavien logiikkatasot ovat 5V/TTL yhteensopivia.

Taulukko 2 *Advantechin korttien määrittelyt*

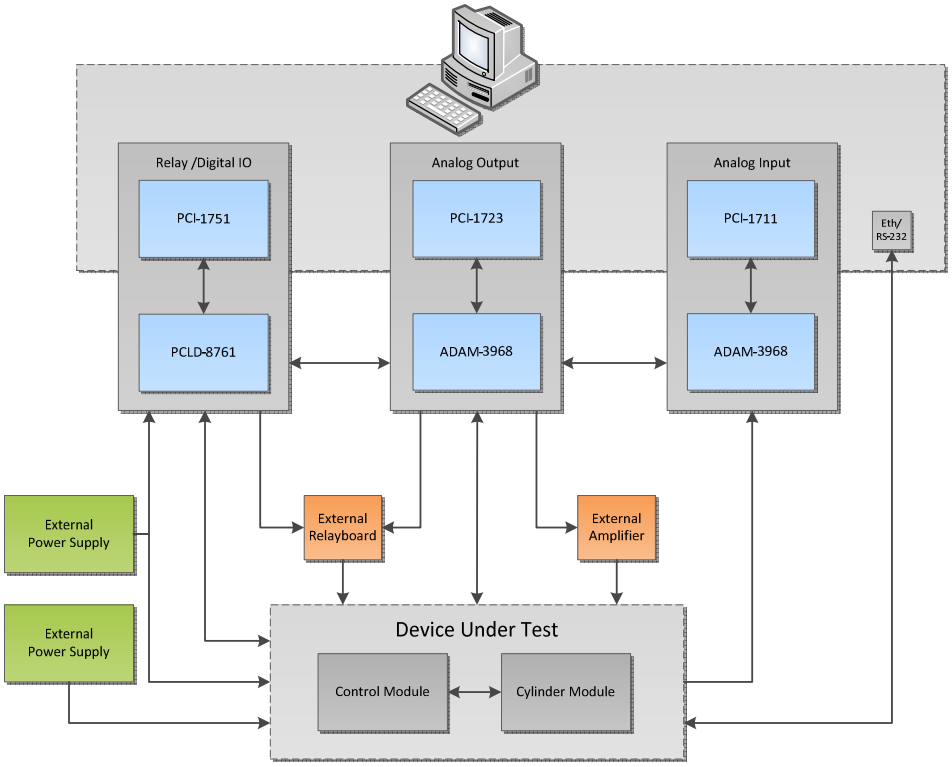
Module	PCI Card/ I/O Terminal	Channels	Input/Output Range
Relay Module	PCI-1751	48 digital I/O	Digital:5V/TTL
	PCLD-8761	24 Digital Inputs 24 Relays (SPDT)	Digital Input: 0-30V Relay: 30Vdc @ 1A
Analog Output Module	PCI-1723	8 Analog Outputs 16 Digital I/O	Analog Output: -10V - +10V 0 - 20mA Digital: 5V/TTL
	ADAM-3968	68-Pin	-
Analog Input Module	PCI-1711	16 Analog Inputs 16 Digital Inputs 16 Digital Outputs	Analog Input: -10V - +10V Digital: 5V/TTL
	ADAM-3968	68-Pin	-

Valmiiden teollisuusautomaatiokorttien lisäksi testijärjestelmään tehtiin itse ulkoinen vahvistinkortti (External Amplifier) ja kolme kappaletta ulkoisia relekortteja (External relayboard). Vahvistinkortissa käytetään OPA2171-operaatiovahvistimia, joille on mahdollista määrittää vahvistus kahdella vastuksella. Testijärjestelmässä käytetään vain yhtä vahvistinkytkentää muiden ollessa varalla. Relekorteissa käytettään DPDT-tyyppisiä (engl. Double Pole Douple Throw) kaksoisvaihtokoskettimellisia releitä. Näillä mahdollistetaan kahden eri signaalin reititys yhdellä ohjaussignaalilla.

5.2.2 Järjestelmän tehonsyöttö ja kommunikointi

Koko testausjärjestelmän tehonsyöttö tapahtuu kahden eri teholähteen avulla. Toinen teholähde syöttää 24V -jännitteen testattaville moduuleille ja antaa lisävirtaa relemoduulille. Toisella teholähteellä syötetään 24V -jännite toisen testattavan moduulin lisäjännitteeksi. Testausjärjestelmän yksinkertaistettu lohkokaavio on esitetty kuvassa 5.2.

Testien suoritusta ohjataan testi-PC:ltä käsin, joka kytkeytyy järjestelmän mittausmoduuleihin PCI-väylän ja SCSI-68 -kaapelien kautta. Kommunikointi testattavien laitteiden ja testi-PC:n välillä tapahtuu ethernetin ja RS-232-sarjaportin kautta kuten kuvassa 5.2 on esitetty.



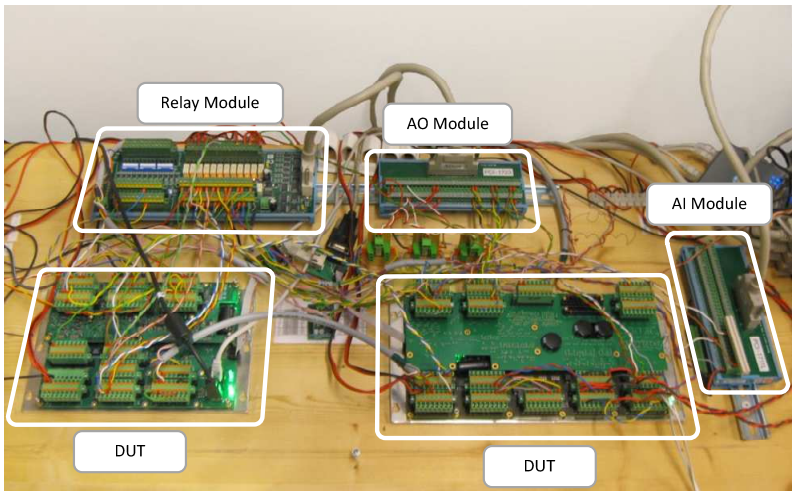
Kuva 5.2 *Testausjärjestelmän lohkokaavio*

5.2.3 Testattavat laitteet

Tässä työssä käsitellään testattavia laitteita (DUT, engl. Device Under Test) osana testausjärjestelmää. Testattavien laitteiden tarkempaa toimintaa ei esitetä, mutta tärkeimmät laitteiden laitteistomäärittelyt esitellään testikytkentöjen yhteydessä.
Testattavana on kaksi erillistä laitetta, jotka on kytketty testausjärjestelmän lisäksi myös toisiinsa. Testattavina laitteina käytetään yhtä ohjausmoduulia (Control module) ja yhtä sylinterimoduulia (Cylinder module).

5.2.4 Toteutettu järjestelmä

Testijärjestelmän laitteisto koottiin puulevyn päälle, jotta sen siirtäminen tulevaisuudessa olisi helpompaa. Advantechin moduulit kiinnitettiin metallisiin U-kiskoihin. Sekä kiskot että testattavat moduulit kiinnitettiin ruuveilla puulevyyn. Ulkoisista teholähteistä kaapeloinnit kiinnitettiin erillisiin kytkentärimoihin, joista tehonsyötöt jaettiin haluttuihin paikkoihin. Kuvassa 5.3 on esitetty toteutettu testijärjestelmä, johon on merkitty tärkeimmät komponentit.



Kuva 5.3 *Toteutettu testausjärjestelmä*

5.3 Testijärjestelyt

Toteutus tehtiin Hardware-in-the-Loop (HIL) -testauksena, jossa ohjausjärjestelmä on toteutettu fyysisesti, mutta anturit korvattiin simuloiduilla signaaleilla. Testeissä käytetään kahta erilaista testityyppiä, signaali- ja diagnostiikkatestejä. Signaalitesteissä testataan kanavan toiminta-alue mahdollisimman kattavasti. Diagnostiikkatesteissä testataan kanavatyypissä olevat mahdolliset diagnostiikkatoiminnallisuudet. Virheellistä käyttäytymistä, kuten kanavien välisiä ristiinkuulumisia ei ole tarkoitus testata. Testit tehdään kanavatyyppi kerrallaan ja vain yksi testitapaus kerrallaan. Esimerkiksi kanavaan tehdään ensin signaalialueen testaus, jonka jälkeen tehdään ylivirtatilanteiden testaus.

Testausympäristön kehityksessä panostettiin kytkentöjen yksinkertaisuuteen ja luotettavuuteen. Testikytkennät tehtiin käyttäen monisäikeisiä kaapeleita, joiden päät holkitettiin kytkentöjen helpottamiseksi. Mittauskorttien päissä johtimet kiinnitettiin ruuviliittimiin ja testattavien moduulien päässä käytettiin Phoenix TFKC 2,5/ 8-STF-5,08 jousiliittimiä (Kuva 5.4).

Kuva 5.4 *Testikytkennöissä käytetty Phoenix TFKC 2,5/ 8-STF-5,08-liitin (Phoenix)*

5.3.1 Signaalitestit

Signaalitesteillä simuloidaan kanaviin kytkettäviä oikeita antureita. Simulaatioiden etuna verrattuna oikeisiin antureihin on niiden mahdollisuus testata laajemmin koko mittausalue. Anturit tarvitsevat ulkoisia muutoksia, kuten lämpötilan tai paineen muutoksia, simulaatio puolestaan voidaan toteuttaa signaalia muuttamalla. Signaalitestit tehdään käyttäen tyypillisiä kuormia, eikä testeissä ole tarkoitus testata kanavien diagnostiikkaa.

Analogisten sisäänmenokanavien signaalitestit tehdään pääsääntöisesti tuomalla kanavaan haluttu jännitesignaali, jolla voidaan tehdä koko jännitealueen pyyhkäisyjä. Testauksen aikana havaittiin, että virtasisäänmenoja syöttävän AO-moduulin virranajokyky rajoittui 15 mA:iin, eikä 20 mA:iin kuten moduulin määrittelyt ilmoittivat. Testitapauksissa virta-alue rajattiin 4-15 mA:iin. Analogiset virtaulostulot testataan mittaamalla tunnetun vastuksen yli jännitettä ja laskemalla saaduista arvoista virta-arvo.

Kanavissa testataan myös kalibrointien toiminta. Kalibrointireferenssi saadaan testi-PC:ltä, joten kyseessä ei ole varsinainen tarkkuuskalibrointi vaan kalibroinnin toiminnallinen testaus. Digitaalisissa I/O-kanavissa käytetään joko releitä kytkiminä tai mittausmoduulien digitaalisia I/O-lähtöjä. Myös kanavien sisäisiä takaisinkytkentöjä hyödynnetään mittausdatan talteen ottamisessa ja analysoinnissa.

5.3.2 Diagnostiikkatestit

Diagnostiikkatesteissä testataan mitattavista kanavista suojauskytkentojä ja niiden rajoja. Testattavia tilanteita ovat ylivirta (OC, engl. overcurrent), ylijännite (OV, engl. overvoltage), johdinkatko (WB, engl. wirebreak) ja herätejännitteen ylivirta (Ext OC, engl. excitationvoltage overcurrent). Virhetilat ja –rajat vaihtelevat kanavakohtaisesti, mutta samantyyppisissä kanavista testataan vain yksi kanava, poikkeuksena kanavat, joiden sisäisissä toteutuksissa on eroja.

Ylivirtatilanteet luodaan pääsääntöisesti nostamalla syötettävän signaalin tasoa niin, että se ylittää liipaisutason (Trigger level). Herätejännitteiden ylivirtatilanteet luodaan kytkemällä sopiva vastus herätejännitelähdön ja maapotentiaalin (GND) väliin.

5.4 Ohjausmoduulin testikytkennät

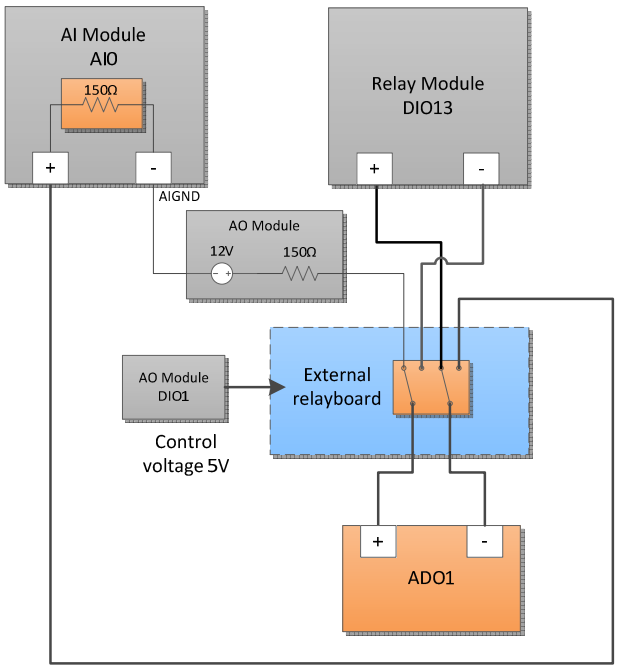
5.4.1 ADO – Analog/Digital Output

Analog/Digital Output (ADO) on ulostulo, jossa on kaksi eri moodia, analoginen- ja digitaalinen. Analogisessa virtamoodissa (mA-moodi) ulostuloalue on 0-24mA ja digitaalisen moodin (DO-moodi) ulostulo on ON/OFF-tyyppinen ja jännitasot ovat 0/24V.

Kanavan molemmat moodit on mahdollista testata samalla kanavalla ADO1 kuvan 5.5 mukaisella kytkennällä. Kanavakohtaisen moodin valinnan lisäksi testikytkennässä on signaalien valinnalle ulkoinen relekortti, jota ohjataan AO-moduulin digitaalisesta lähdöstä DIO1. Kumpikin moodi vaatii toimiakseen ulkoisen jännitelähteen, joka digitaalisessa moodissa on relemoduulissa ja analogisessa virtamoodissa AO-moduulissa.

Analogisessa moodissa AO-moduulista syötetään 12V -lisäjännite 100Ω:n sarjavastuksen kautta. Ulostulon virta-arvo säädetään ohjelmallisesti 4-20mA -alueelta ja luetaan AI-moduulin AI0 -sisäänmenosta 150Ω:n tehovastuksen yli. Digitaalisessa

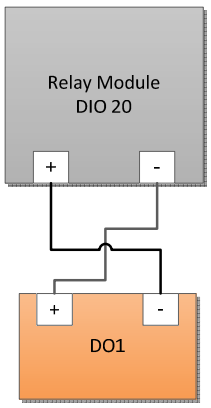
moodissa kanavan ulostuloa kytketään päälle ja pois ohjelmallisesti. Ulostulon arvo luetaan relemoduulin digitaalisesta sisäänmenosta DIO13.



Kuva 5.5 *ADO kanavan testikytkentä*

5.4.2 DO – Digital Output

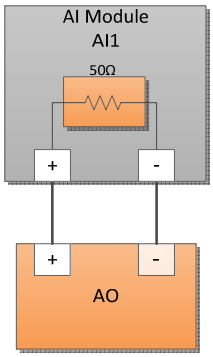
Digital Output -kanavassa on vain yksi moodi, joka on ON/OFF-tyyppinen digitaalinen 0/24V:n ulostulo. Kanavatyypin testaus tehdään DO1-kanavalla kuvan 5.6 mukaisella kytkennällä. Kanavan ulostulo kytketään päälle ja pois. Kanavan tila tulkitaan relemoduulin digitaalisesta sisäänmenosta DIO20.



Kuva 5.6 *DO kanavan testikytkentä*

5.4.3 AO – Analog Output

Analog Output on virtalähtöinen analoginen ulostulo, jossa ulostuloalue on 4-20mA tai 0-200mA riippuen valitusta moodista. Kummatkin moodit voidaan testata samalla kytkennällä. Kanavan testikytkentä (Kuva 5.7) koostuu AI-moduulin sisäänmenosta AI1, josta mitataan kanavan ulostulon virta-arvo 50 Ω tehovastuksen yli.



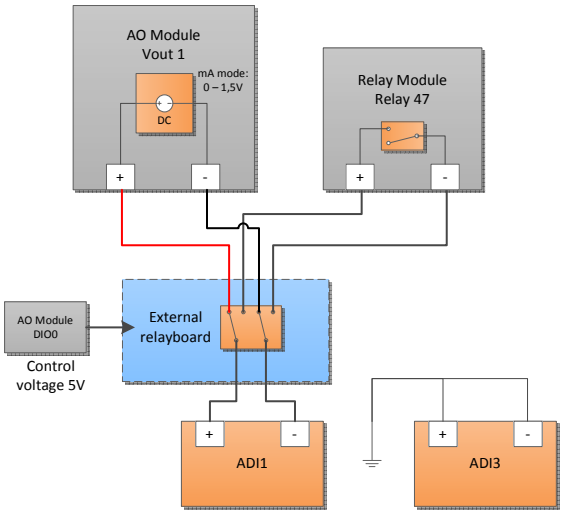
Kuva 5.7 *AO kanavan testikytkentä*

5.4.4 ADI – Analog/Digital Input

Analog/Digital Input on sisäänmenokanava, jossa on analoginen virtamoodi (mA-moodi) ja digitaalinen moodi (DI-moodi). Kanava kytkenee mittaamaan virtaa 0-24 mA alueelta ja digitaalisen moodin sisäänmenoalue 0-32 V. ADI-kanavassa on ylivirtasuoja, joka suojaa kanavan mittauskomponentteja.

Kummankin moodin testaamista varten testikytkentä on tehty ADI1-kanavaan, jossa on ohjelmallisen moodin valinnan lisäksi ulkoinen relekortti signaalien valintaa varten. Relettä ohjataan AO-moduulin DIO0 -lähdöllä. Testikytkentä on esitetty kuvassa 5.8. Analogisessa virtamoodissa virta-arvo syötetään AO-moduulin jännitelähdöstä Vout1. Jännitetaso valitaan 0-1,5 V-alueelta. Ylivirtasuojan liipaisutaso on 3,6 V, joka voidaan syöttää samasta Vout1 jännitelähdöstä. Digitaalinen moodi testataan kytkemällä relemoduulin relettä 47 päälle ja pois. Kytkimen tila tulkitaan ohjelmallisesti.

ADI3-kanavaan on tehty kytkentä sisäänmenon kohinatason mittausta varten. Kanavan kummatkin sisäänmenopinnit on kytketty maapotentiaaliin, jolloin +pinnistä mitataan kohinatasoa.

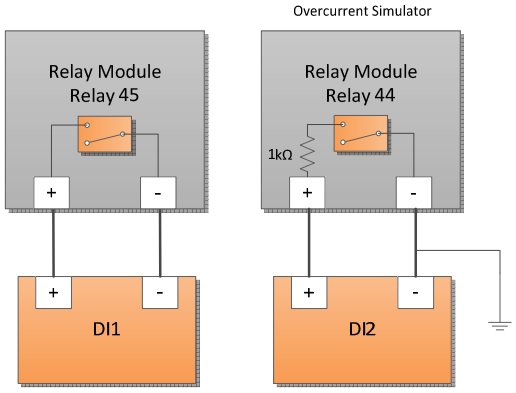


Kuva 5.8 *ADI kanavan testikytkentä*

5.4.5 DI – Digital Input

Digital input on yhden moodin digitaalinen sisäänmenokanava. Sisäänmenoalue on 0-32 V ja kanava on suojattu ylivirtasuojalla. Testikytkentä on esitetty kuvassa 5.9 ja koostuu kahdesta kanavasta, joista DI1 -kanavalla testataan sisäänmenon toiminta-aluetta ja DI2 ylivirtadiagnostiikka.

Signaalitestaus tehdään kytkemällä relemoduulin relettä 45 päälle ja pois. Sisäänmenon tila tulkitaan ohjelmallisesti. Ylivirtasuojan diagnostiikka reagoi sisäänmenon jännitteen laskiessa alle 20 V:n. Tämä saadaan aikaan lisäämällä 1kΩ alasvetovastus sisäänmenon +pinnin ja maapotentiaalin GND väliin. Vastus voidaan kytkeä päälle ja pois releellä 44.



Kuva 5.9 *DI-kanavan testikytkentä*

5.4.6 AI – Analog Input

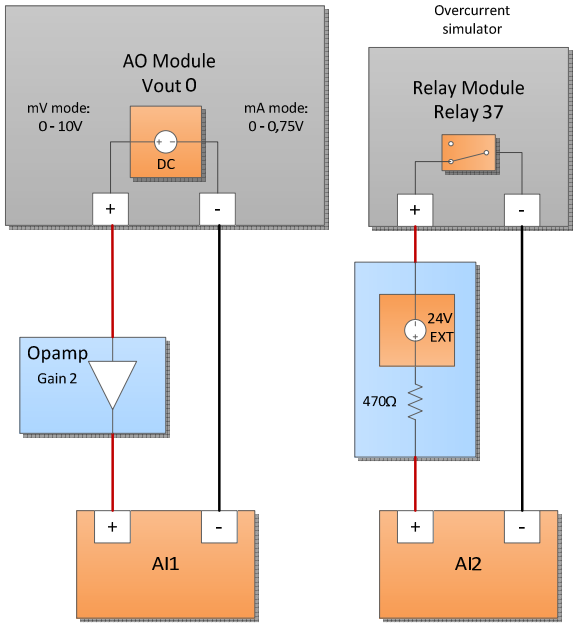
Analog input -kanava on analoginen sisäänmeno -kanavatyyppi, jossa on kaksi eri moodia, jännitemoodi (mV-moodi) ja virtamoodi (mA-moodi). Jännitemoodissa sisäänmenoalue on 0-20 V ja virtamoodissa 4-20 mA. Kanavassa on aktiivinen ylivirtasuoja, joka toimii virtamoodissa suojaamassa mittauskomponentteja.

Testauksessa käytetään kahta eri kanavaa AI1 ja AI2 kuten kuva 5.10 osoittaa. Ensimmäisellä kanavalla testataan molempien moodien signaalialueet ja AI2 -kanavalla testataan diagnostiikka.

Signaalialueen testauksessa käytetään AO-moduulin jännitelähtöä Vout0, jonka ulostulon rajoittuu 10 volttiin. Tämä signaali on vahvistettu ulkoisella operaatiovahvistinkytkennällä kaksinkertaiseksi, jotta saavutetaan jännitemoodin koko jännitealue 0-20 V.

Virtamoodissa Vout0 jännitelähtöä säädetään 0-0,75 V:n alueella, joka vahvistetaan operaatiovahvistimella kaksinkertaiseksi. Näin voidaan kattaa koko haluttu jännitealue 0-1,5 V, joka vastaa virta-aluetta 0-15 mA .

Ylivirtadiagnostiikan testausta varten AI2 -kanavaan on kytketty ulkoinen 24 V:n jännite ja virran rajoitusta varten 470 Ω:n vastus. Relemoduulin releellä 37 kytketään ulkoinen jännitelähde kanavaan ja aiheutetaan ylivirtatilanne.



Kuva 5.10 *AI-kanavan testikytkentä*

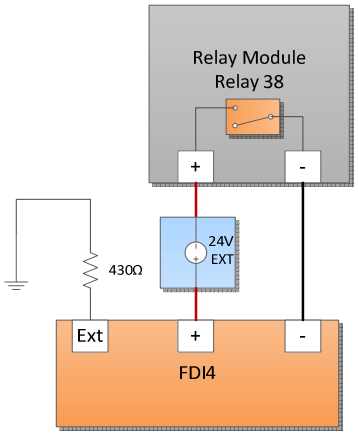
5.4.7 FDI – Frequency/Digital Input

Frequency/Digital Input (FDI) on digitaalinen kanava, jossa on kaksi eri moodia taajuusmoodi (FI-moodi) ja digitaalinen moodi (DI-moodi). Tässä tapauksessa testataan vain digitaalinen moodi ja kanavan diagnostiikka, jolloin molempien testaus suoritetaan FDI4 -kanavan testikytkennällä, joka on esitetty kuvassa 5.11.

Digitaalisen moodin testausta varten sisäänmenoon on kytketty ulkoinen 24 V:n jännitelähde, jota kytketään päälle ja pois relemoduulin releellä 38. Kytkimen tila tulkitaan ohjelmallisesti.

FDI -kanavissa on erillinen ulostulo 24 V:n herätejännitteelle (Excitation voltage), jossa on 47 mA:n virtaraja. Diagnostiikkatestissä voidaan todeta sekä herätejännitteen

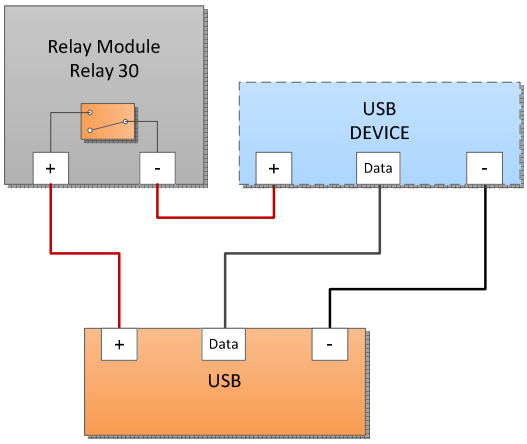
ohjauksen toimivuus että ylivirtasuojan toimivuus. Testikytkennässä ylivirtatila luodaan kytkemällä 430 Ω:n vastus herätejännitteen (Ext) ja maapotentiaalin (GND) väliin ja kytkemällä herätejännite päälle.



Kuva 5.11 *FDI-kanavan testikytkentä*

5.4.8 USB

USB-portin testausta varten on tehty kuvan 5.12 mukainen testikytkentä. Testissä käytetään relettä 30, jonka asentoa muuttamalla voidaan kytkeä USB-porttiin VBUS -jännite päälle ja pois. USB:n tila voidaan tulkita ohjelmallisesti. Testi vastaa tilannetta, jossa USB-laite fyysisesti irroitetaan tai kytketään USB-porttiin.



Kuva 5.12 *USB testikytkentä*

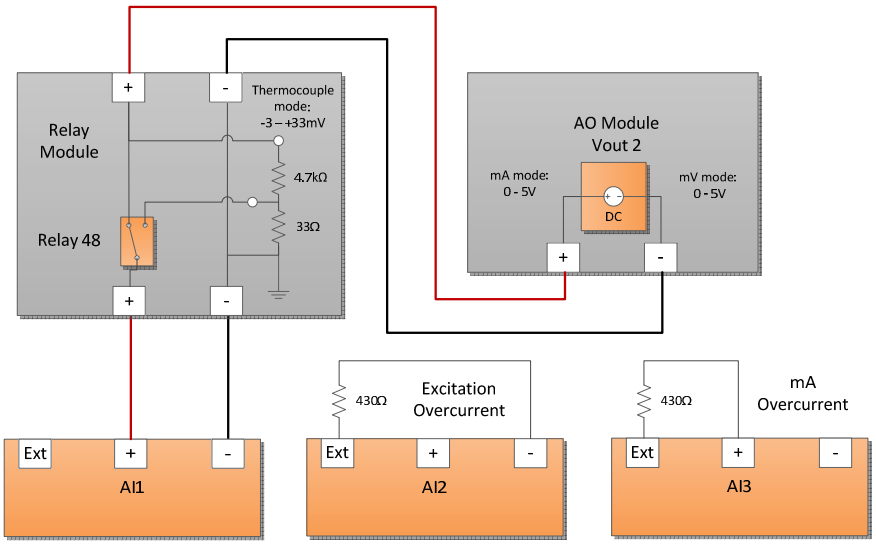
5.5 Sylinterimoduulin testikytkennät

5.5.1 AI – Analog Input

Analog input -kanava on analoginen sisäänmeno -kanavatyyppi, jossa on kolme eri moodia, jännitemoodi (mV-moodi), virtamoodi (mA-moodi) ja lämpötilaa mittaava termoparimoodi (TC- moodi). Jännitemoodissa testattava sisäänmenoalue on 0-5 V, virtamoodissa 4-15 mA ja termoparimoodissa -3 - +33 mV (-50 - +800 °C) . Kanavassa on erillinen 24 V:n herätejännitelähtö, jolle on oma ylivirtasuoja. Virtamoodissa kanavaa suojaa aktiivinen ylivirtasuoja.

Testikytkentä koostuu kolmesta AI-kanavasta, joista AI1-kanavalla tehdään signaalialueen testaus ja kanavilla AI2 ja AI3 testataan diagnostiikka. Ohjelmallisen moodin valinnan lisäksi kytkentään on lisätty relemoduulin rele 48, jolla voidaan valita termoparimoodia varten tarkempi jännitealue.

Kanavissa AI2 ja AI3 testataan ylivirtasuojaukset. AI2-kanavan herätejännitteen ylivirtatila luodaan kytkemällä herätejännitteen sisäänmenon −pinniin 430 Ω:n vastus, jonka jälkeen herätejännite kytketään päälle. Virtamoodin ylivirtatilannetta varten on kytketty herätejännitteen ja kanavan +sisäänmenon väliin 430 Ω:n vastus. Molemmissa tapauksissa suojauksen toiminta varmistetaan ohjelmallisesti. Kanavan testikytkentä on esitelty kuvassa 5.13.



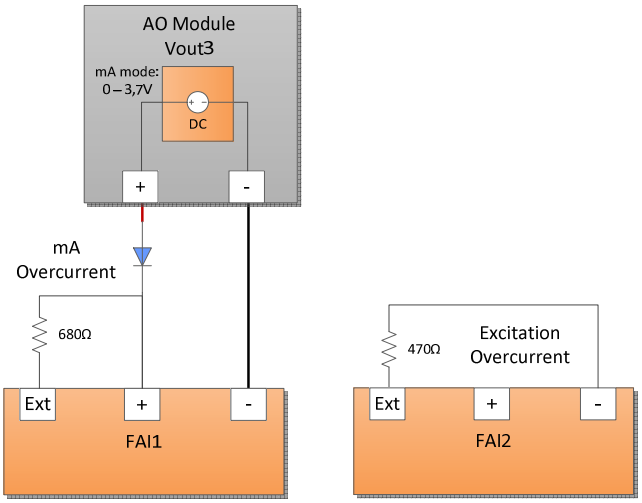
Kuva 5.13 *AI-kanavan testikytkentä*

5.5.2 FAI – Fast Analog Input

Fast Analog Input -kanava on analoginen virtasisäänmeno kuten AI-kanavatkin, mutta näytteistystaajuus on suurempi. Testauksessa nopeutta ei kuitenkaan huomioida erikseen, vaan testaus suoritetaan AI-kanavien tapaan virtamittauksena. Diagnostiikasta testataan signaalilinjan ja herätejännitteen ylivirtasuojaukset.

Testaus suoritetaan kahdella FAI-kanavalla, joista FAI1:llä tehdään signaalialueen testaus ja ylivirtasuojan testaus. FAI2-kanavalla testataan herätejännitteen ylivirtatila. Testauskytkentä on esitelty kuvassa 5.14. Sisäänmenoalue testataan säätämällä AO-moduulin Vout3 jännitelähtöä haluttuun arvoon. Kanavan virta-arvo tulkitaan ohjelmallisesti. Signaalin ylivirtatila luodaan FAI1 +pinnin ja herätejännitteen (EXT) väliin kytketyllä 680 Ω:n vastuksella. Kytkemällä herätejännite päälle jännite nousee kanavan sisäänmenossa liipaisutason 6,8 V yli. Suojauksen tila tulkitaan ohjelmallisesti.

Herätejännitteen ylivirtadiagnostiikkaa varten on kytketty 470 Ω:n vastus herätejännitteen ja kanavan FAI2 –pinnin väliin. Ylivirtatila luodaan kytkemällä herätejännite päälle ja suojauksen tila tulkitaan ohjelmallisesti.



Kuva 5.14 *FAI-kanavan testikytkentä*

5.5.3 FDI – Frequency/Digital Input

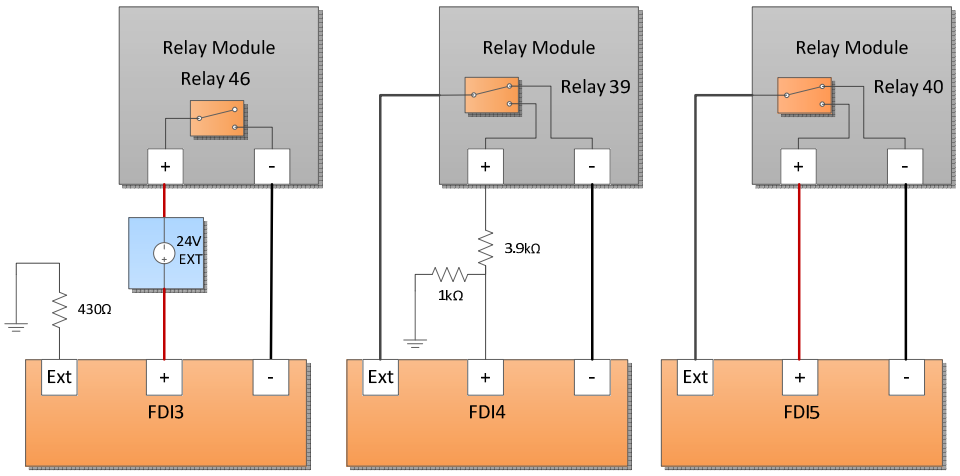
Frequency/Digital Input on sisäänmenotyyppinen kanava, jossa on kaksi eri moodia, taajuusmoodi (FI-moodi) ja digitaalinen moodi (DI-moodi). Näistä testataan vain digitaalinen moodi ja sisäänmenon liipaisutasot. Diagnostiikasta testataan herätejännitteen ylivirtasuoja, kanavan kellutus (Floating) ja sisäänmenon suojauskytkennän toiminta. Testauskytkentä on esitetty kuvassa 5.15.

Digitaalinen moodi testataan FDI3-kanavalla, johon on kytketty ulkoinen 24 V:n jännitelähde ja relemoduulin rele 46. Rele toimii ON/OFF-kytkimenä, joka kytkee jännitteen päälle ja pois. Kanavan tila luetaan ohjelmallisesti. Samaan kanavaan on

kytketty 430 Ω:n vastus herätejännitteen (EXT) ja maan (GND) väliin. Kytkennällä testataan herätejännitteen ylivirtasuojan toiminta. FDI-kanavien herätejännitteellä virtaraja on 47 mA. Kanavalla FDI5 testataan ulostulon liipaisutasojen synkronointi, jota varten herätejännite on kytketty releen 40 kautta. Releellä valitaan haluttu testitapaus liipaisutasojen testin ja diagnostiikkatestin väliltä.

Sisäänmenon FDI -signaali voidaan joko sitoa maapotentiaaliin tai kelluttaa. Kelluttamisen testausta varten herätejännitteestä on tehty kahdella vastuksella (3.9 kΩ ja 1 kΩ) jännitejako, josta signaali syötetään FDI4 +pinniin. Ohjelmallisesti kytketään herätejännite ja vaihdetaan kanava kelluvaan tilaan. Tilan muutos varmistetaan kanavan ulostulon muutoksesta ohjelmallisesti.

FDI-kanavissa on suojauskytkentä suojaamassa FDI –sisäänmenon sisäisiä kytkentöjä kanavan ollessa kelluttamattomassa tilassa (kytkettynä maapotentiaaliin). Ylijännitesuoja reagoi jännitteen noustessa 5,7 V rajan yli ja kytkee automaattisesti kanavan kelluvaan tilaan. Testausta varten herätejännite on kytketty releen kautta testattavan kanavan −pinniin. Suojaustesti tehdään FDI4- ja FDI5-kanavalla, koska niiden sisäisessä toiminnassa on eroavaisuuksia.



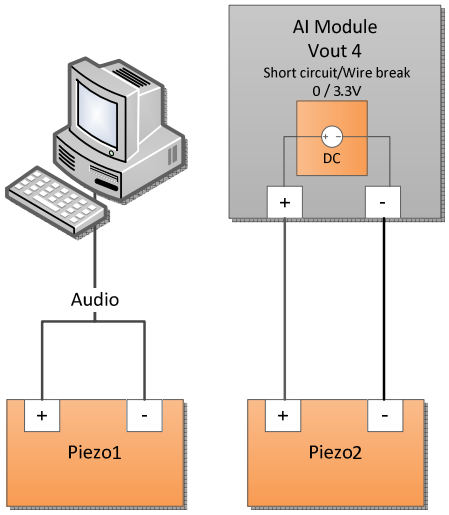
Kuva 5.15 *FDI-kanavan testikytkentä*

5.5.4 Pietso

Pietso (engl. Piezo) on analoginen sisäänmenokanavatyyppi, jolla mitataan korkeaimpedanssista (>1 MΩ) pietsosähköistä anturia. Sisäänmenon jännitealue on 0-1000 mVp-p AC taajuusalueella 500 Hz-2.5 kHz. Kanavan testauksessa käytetään kahta eri kanavaa, joiden testikytkentä on esitetty kuvassa 5.16.

Pietso1-kanava on kytketty tietokoneen äänikortin ulostuloon. Tietokoneella generoidaan sinisignaalia pistetaajuuksina 500 Hz-8 kHz:n alueella. Signaalille tehdään FFT-muunnos (engl. Fast fourier transform), josta tulkitaan kanavan tulkitseman signaalin taajuus. Pietso2-kanavalla testataan diagnostiikan reagointi oikosulku- ja johdinkatkotiloihin.

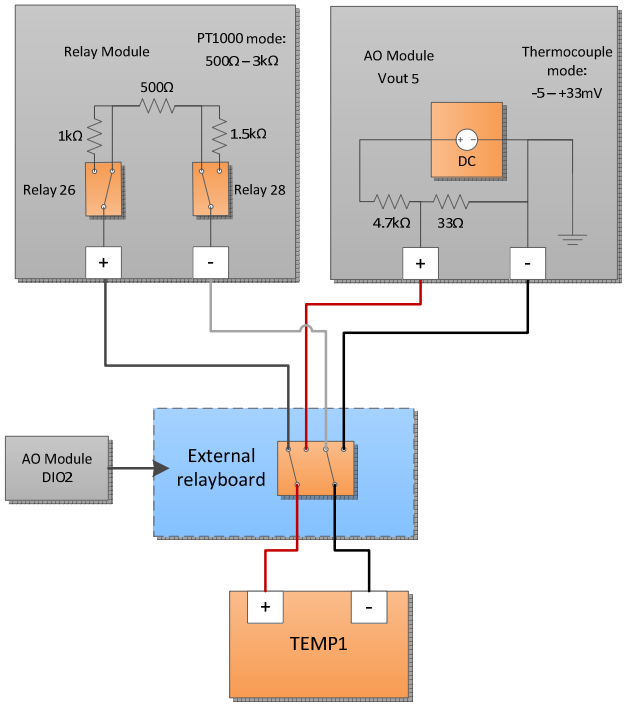
Oikosulkutila luodaan asettamalla AO-moduulin Vout4 -jännitelähtö 0 V:iin ja johdinkatko luodaan säätämällä jännite lähelle 3.3 V:a. Molemmat vikatilat tulkitaan AD-muuntimen ulostuloarvosta ohjelmallisesti.



Kuva 5.16 *Pietso kanavien testikytkentä*

5.5.5 Temp – Temperature

Temp-kanava on lämpötila-antureita hyödyntävä kanavatyyppi. Lämpötila-antureina kanavassa käytetään PT1000-antureita ja K-tyypin termopariantureita. Molempia varten on luotu testikytkentä (Kuva 5.17) jolla simuloidaan anturien mittausaluetta. PT1000-moodin sisäänmenoalue on 500 Ω - 3 kΩ, joka vastaa lämpötila-aluetta -50°C - +850°C. Termoparin sisäänmenoalue on -3 mV – 33 mV , joka vastaa lämpötila-aluetta -50°C - +800°C.



Kuva 5.17 *Temp testikytkentä*

Testattavan moodin valinta tapahtuu ohjelmallisesti ja ulkoisen relekortin tilaa vaihtamalla. Relekorttia ohjataan AO -moduulin digitaalisesta lähdöstä DIO2.

Termoparimoodissa AO-moduulin Vout5 -jännitelähdöllä asetetaan haluttu jännitetaso, joka ajetaan jännitejaon läpi. Jännitejaolla lasketaan syötetty jännite termoparin sisäänmenon mukaiselle (-3 mV – 33 mV) testausalueelle. Jännitearvo muunnetaan ohjelmallisesti celsiusasteiksi, mikä tulkitaan ohjelmallisesti.

PT1000-testaus tehdään käyttäen kolmea kiinteäarvoista vastusta. Haluttu vastusarvo valitaan käyttäen releitä 26 ja 28. Releiden asentoa muuttamalla saadaan haluttu vastuskombinaatio kattaen resistanssit 500Ω, 1,5kΩ, 2kΩ, 3kΩ taulukon 3 mukaisesti.

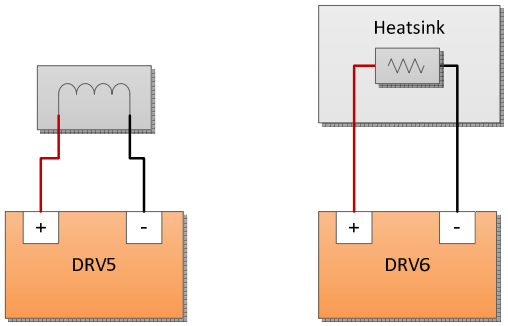
Taulukko 3 *PT1000 vastuskombinaatiot*

		Rele 26	
	Tila	NO	NC
Rele 28	NO	500Ω	1,5kΩ
	NC	2kΩ	3kΩ

NO	Normaalisti auki (Normal open)
NC	Normaalisti kiinni (Normal closed)

5.5.6 DRV – Driver

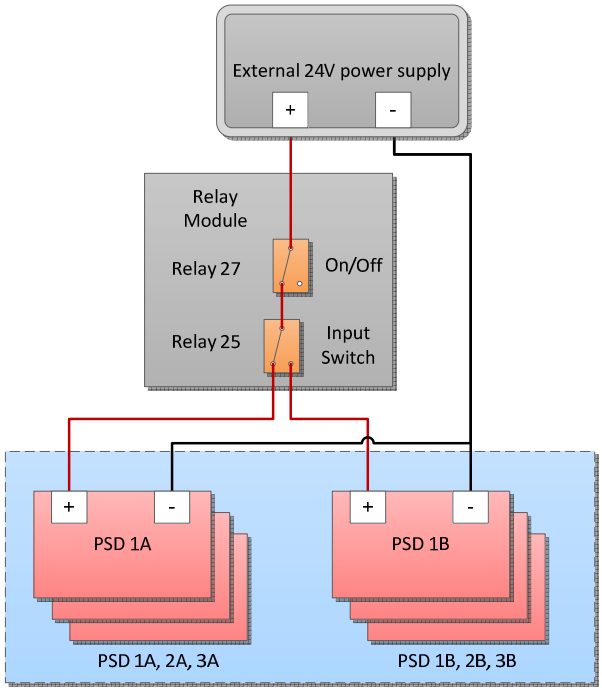
Driver-kanava (DRV) on ulostulokanava, jolla voidaan ajaa suuria virtoja induktiiviseen kuormaan. Resistiivisen kuorman kanssa kanava toimii digitaalisena ulostulona. Kanavatyypin testikytkennässä (Kuva 5.18) eri kuormatyypit testataan omissa kanavissaan. Induktiivinen kuorma on kytketty kanavaan DRV5 ja resistiivinen kuormavastus on kytketty kanavaan DRV6. Kuormavastuksen lämpenemisen rajoittamiseksi siihen on lisätty ulkoinen jäähdytyselementti.



Kuva 5.18 *DRV-kanavien testikytkennät*

5.5.7 PSD – Tehonsyöttö

Sylinterimoduulissa on erillinen tehonsyöttö (PSD engl. Power Supply Driver) isovirtaisille DRV-kanaville, jotka on jaettu kolmeen eri DRV-pankkiin (engl. DRV-bank). Jokaiselle kolmelle DRV-pankille on omat tehonsyötöt, jotka on kahdennettu pankkikohtaisesti. Tehonsyötön testikytkennässä (Kuva 5.19) ohjataan kaikkien pankkien tehonsyötöt yhtä aikaa joko A- tai B-sisäänmenoon. Valinta tehtään relemoduulin releellä 25. Pankkijännitteiden tehonsyöttö voidaan katkaista kokonaan muuttamalla releen 27 asentoa.

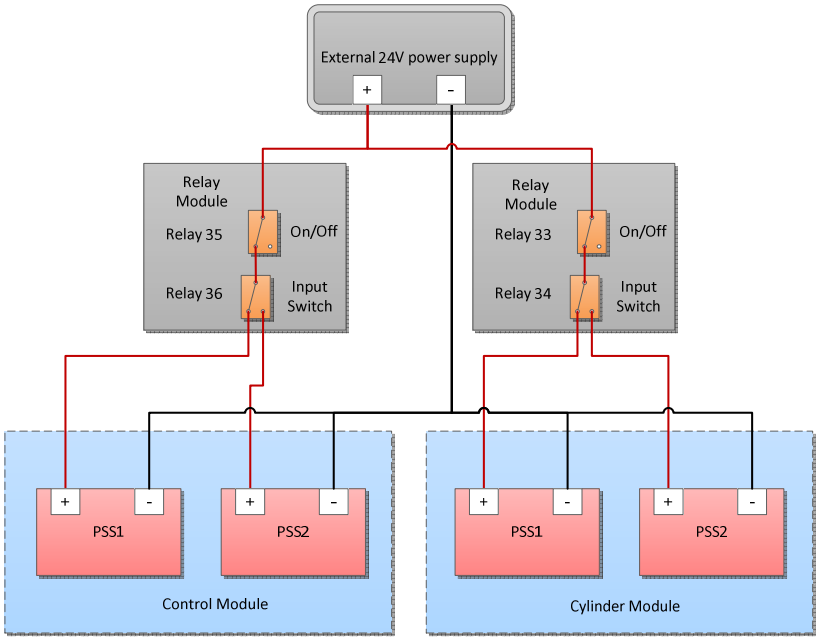


Kuva 5.19 *PSD-tehonsyötön testikytkentä*

5.6 Moduulien yhteiset testikytkennät

5.6.1 PSS – Tehonsyöttö

Ohjaus- ja sylinterimoduulin PSS-tehonsyöttö (Power Supply System) on kytketty samasta ulkoisesta 24 V:n teholähteestä 2 A:n virtarajalla. Relemoduulin releitä 35 ja 33 käytetään kytkemään virta testattaville moduuleille. Kummassakin moduulissa on kaksi erillistä tehonsyöttöliitintä, jotka voidaan valita releiden 36 ja 34 asentoa vaihtamalla. Testikytkentä on esitetty kuvassa 5.20.



Kuva 5.20 *PSS-Tehonsyötön testikytkentä*

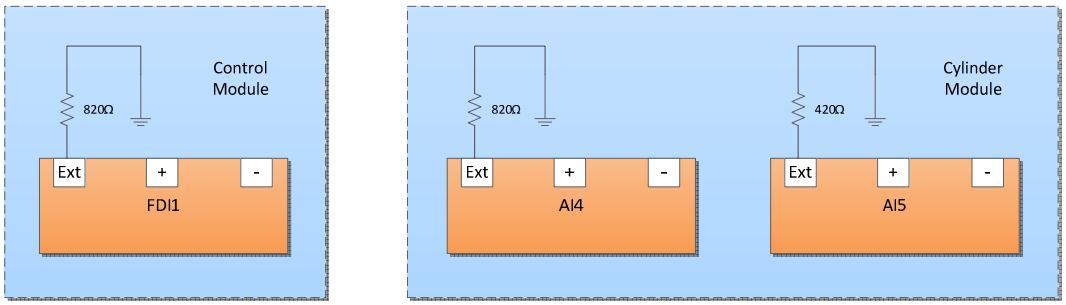
5.6.2 PSS – Tehonsyöttöjen kuormituskytkentä

PSS-tehonsyötön kuormitustesteissä lisätään ylimääräistä virrankulutusta moduuleita syöttävälle teholähteelle. Kaikissa tapauksissa kanavan herätejännitteen (Ext) ja maapotentiaalin (GND) väliin on kytketty kuormavastus, jolla aiheutetaan virrankulutukseen nousu. Kuormitus saadaan aikaiseksi kytkemällä herätejännite päälle. Testikytkentä on esitetty kuvassa 5.21.

Ohjausmoduulin PSS-tehonsyötön kuormitustestissä FDI1-kanavaan on kytketty 820 Ω:n vastus. Vastuksella aiheutetaan noin 30 mA:n nousu virrankulutukseen, joka tulkitaan ohjelmallisesti.

Sylinterimoduulin kuormitustesti tehdään kahdella AI-kanavalla, jolloin virrankulutuksen nosto voidaan tehdä kahdessa vaiheessa. AI4-kanavaan on kytketty 820 Ω:n vastus, joka aiheuttaa noin 30 mA:n nousun virrankulutukseen. AI5-kanavaan

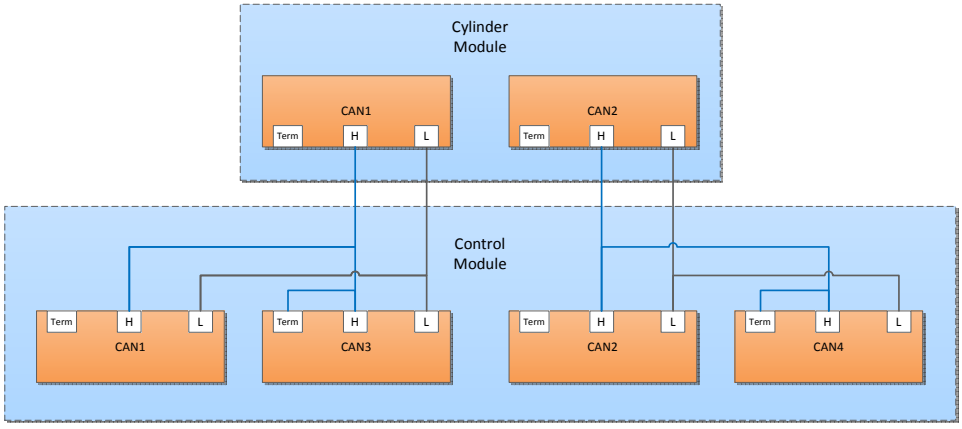
on kytketty 420 Ω:n vastus, jonka kuormitus vastaa noin 60 mA:n nousua virrankulutukseen.



Kuva 5.21 *Tehonsyöttöjen kuormitusten testikytkennät*

5.6.3 CAN – väylä

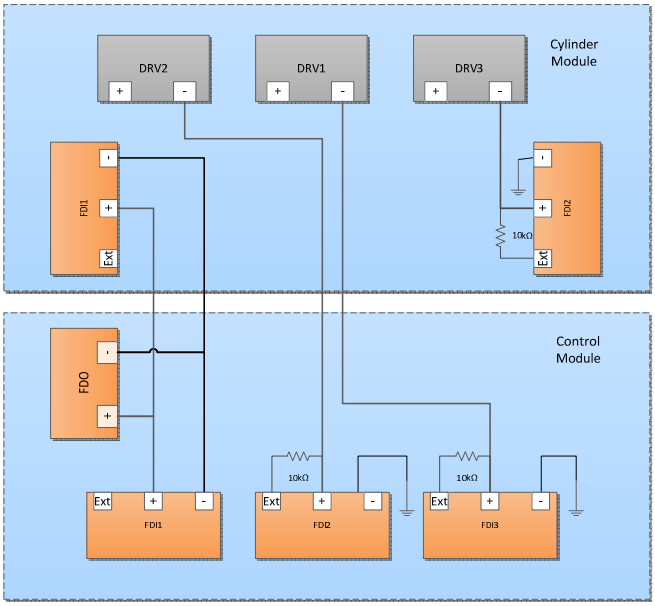
CAN-väylää testataan molempien moduulien välillä kuvan 5.22 mukaisella testikytkennällä. Moduulien CAN-portit on jaettu niin, että ohjausmoduulin CAN-kanavat 1 ja 3 sekä sylinterimoduulin kanava 1 on kytketty keskenään suljettuun silmukkaan. Vastaavasti ohjausmoduulin kanavat 2 ja 4 ja sylinterimoduulista on kytketty yhteen. Kanavat on sisäisesti terminoitu.



Kuva 5.22 *CAN-väylän testikytkentä moduulien välillä*

5.6.4 Pulssijonon testaus

Pulssijonon testaus on testitapaus, joka yhdistää molemmat testattavat moduulit yhdeksi suuremmaksi kokonaisuudeksi. Testikytkentä (5.23) on tehty vain testattavien moduulien välille, joita ohjataan testi-PC:llä.

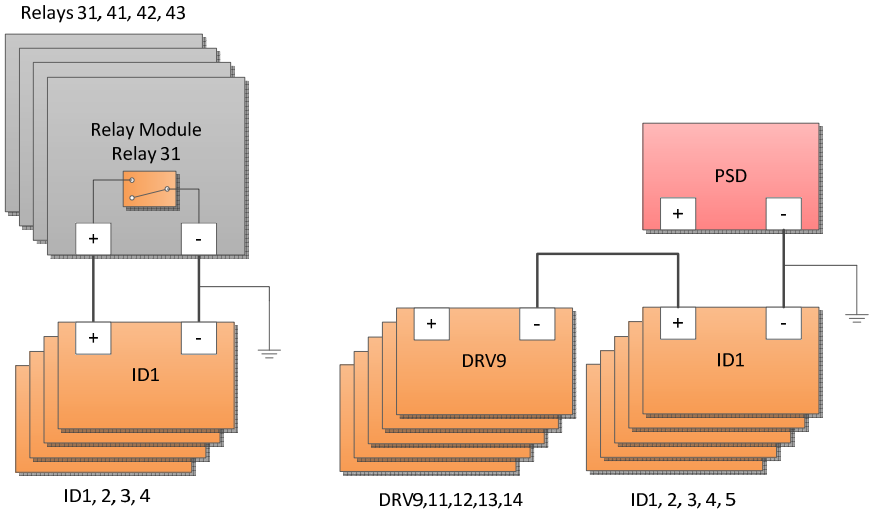


Kuva 5.23 *Pulssijonon testikytkentä*

Testin tarkoituksena on luoda tasainen kanttipulssijono ja siihen puuttuva pulssi. Kanttipulssijono tehdään ohjausmoduulin FDO-kanavalla (Frequency Digital Output), jonka ulostulon jännitealue on 0/24 V ja taajuusalue 0-25 kHz. FDO-kanava on kytketty molemmissa moduuleissa FDI1-kanavaan. Ohjausmoduulin FDI1-kanavalla monitoroidaan pulssijonon tarkkuutta ja sylinterimoduulin FDI1-kanavasta signaali ohjataan sisäisesti DRV-kanaviin 1, 2 ja 3. Signaaliin luodaan puuttuva pulssi kytkemällä DRV-kanavan ohjaus päälle yhden pulssin ajaksi. DRV-kanavat 1 ja 2 on kytketty −pinneistä ohjausmoduulin FDI-kanaviin 2 ja 3, joilla tulkitaan muutettua pulssijonosignaalia. Vastaavasti DRV3-kanavan -sisäänmeno on kytketty sylinterimoduulin FDI2-kanavaan. DRV-kanavien signaaleihin on lisätty 10 kΩ:n ylösvetovastukset, jotka on kytketty FDI-kanavien 24 V:n herätejännitteisiin.

5.6.5 ID – Identification pins

Identification -pinneillä (ID-pinni) muodostetaan testattavasta moduulista tunnistuskoodi. Ohjausmoduulissa on neljä kappaletta ID-pinnejä ja sylinterimoduulissa viisi. Ohjausmoduulissa kaikilla ID-pinneillä on yhteinen maapiste, joka testikytkennässä (5.24) on esitetty –sisäänmenona. Testauksessa relemoduulin releillä 31, 41, 42 ja 43 kytketään ohjausmoduulin ID-pinnejä 1, 2, 3, 4 järjestelmän maapontetiaaliin (GND).



Kuva 5.24 *ID-pinnien testikytkennät. Vasemmalla ohjausmoduulin ja oikealla sylinterimoduulin testikytkentä.*

Sylinterimoduulin ID-pinnien testaukseen käytetään moduulin omia DRV-kanavia, jotka toimivat releiden tapaan kytkiminä. ID-pinnien +sisäänmenot on kytketty DRV-kanavien –sisäänmenoihin. Kanavien maapisteet on kytketty samaan potentiaaliin yhdistämällä PSD-tehonsyötön maapiste ja ID-pinnien yhteinen maapiste järjestelmän maapotentiaalin (GND).

6 TESTAUSJÄRJESTELMÄN OHJELMISTO

Tässä kappaleessa esitellään automatisoituun testausjärjestelmään luotua ohjelmistoympäristöä ja testauksen tavoitteita. Samalla määritellään ja esitellään ohjelman testauksessa käytetyt menetelmät, joilla varmistutaan eri ohjelmistokerrosten toimivuudesta.

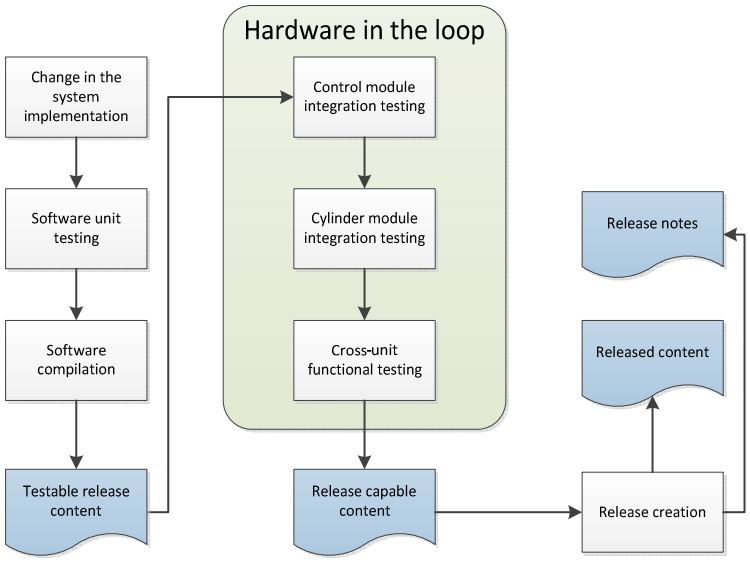
6.1 Testauksen tavoitteet

Ohjelmiston testauksen automatisointia lähdettiin suunnittelemaan tavoitteena koko ohjelmistoprosessin nopeuttaminen ja laadun parantaminen. Projekti, jonka sivutuotteena testausjärjestelmä suunniteltiin, on suuri, joten testauksen tiedettiin olevan monimutkaista. Ohjelmiston testauksen avuksi luotiin automatisoitu testausjärjestelmä, jolla luotiin rajapinta testattavien laitteiden elektroniikan ja ohjelmiston välille. Vaikka pääasiallinen tarkoitus oli luoda testausautomaatio ohjelmiston testaukseen, saatiin myös palautetta testattavien moduuleiden toiminnasta.

Testauksen tavoitteina oli saavuttaa kohtuullisen korkea testauskattavuus ja todentaa ohjelmiston oikeellinen toiminta. Virheiden päätyminen julkaistaviin ohjelmistoversioihin pyrittiin estämään huolellisella suunnittelulla ja regressiotestauksella. Testauksen automatisoinnilla pyrittiin minimoimaan käyttäjän tarve puuttua testaukseen. Samalla tavoitteena oli minimoida testi-PC-ympäristön toteutuksen monimutkaisuutta. Testitapaukset suunniteltiin ja optimoitiin yksinkertaisiksi, nopeasti ja useasti toistettaviksi, jotta testaukseen ja palautteen saamiseen olisi mahdollisimman nopeaa.

6.2 Testausstrategia

Testausstrategian pohjana käytetään jatkuvaa integrointia, joka mahdollistaa pienien muutosten nopean testauksen ja tarkastelun. Testaus automatisoitiin niin, että muutos ohjelmistotason toteutuksessa laukaisee automaattisesti testien suorittamisen. Testiajo koostuu usean tason testeistä alkaen ohjelman yksikkötestauksesta päättyen moduulien väliseen toiminnallisuustestiin. Ohjelmistotestauksen työnkulkukaavio on esitetty kuvassa 6.1.



Kuva 6.1 *Ohjelmistotestauksen työnkulkukaavio*

Testausstrategian ensimmäisenä askeleena on ohjelmiston staattinen analyysi ja yksikkötestaus, joka suoritetaan paikallisesti käyttäjän tietokoneella. Staattisella analyysillä tarkoitetaan analysointia, jolla pyritään löytämään virheitä ohjelman rakenteesta ilman sen suorittamista. Testitapaukset on luotu yksikkötestaukseen tarkoitetulla sovelluskehyksellä. Tämä helpottaa testaustilanteita, jotka normaalisti olisi hankala toteuttaa.

Siinä vaiheessa kun yksikkötestauksesta saadaan hyväksyttyä testauskelpoinen julkaisu, siitä kootaan ja käännetään automaattisesti versio palvelimelle. Tuloksena saadaan ohjelmisto, joka voidaan ladata palvelimelta testattavaan laitteistoon. Koko prosessi on automatisoitu ja tarvittavat ohjelmistotyökalut on integroitu prosessiin.

Seuraava testauksen vaihe on hardware-in-the-loop -testaus, jossa ohjelmistoa testataan testilaitteistolla. Testilaitteiston integraatiotestauksessa tehdään moduuleille yksitellen toiminnallisuustestaus, jossa varmistetaan moduulien oikeellinen toiminta. Testausjärjestelmällä simuloidaan ulkoisia antureita, joiden signaaleja tulkitaan suurpiirteisesti. Mittaustarkkuuksia ei ole tarkoitus testata, vaan niitä varten projektissa on erikseen tuotekehitystestausalusta.

Kun testattavien moduulien toiminnallisuus on varmistettu, siirrytään testausksessa korkeammalle tasolle toiminnallisuuksissa. Moduulien väliset monimutkaisemmat funktiot testataan käyttäen moduulien omia sisäisiä toimintoja. Testauksessa voidaan vähentaa huomattavasti testausjärjestelmän tarvetta, koska testattavissa moduuleissa itsessään luodaan testauksessa tarvittavat signaalit. Testitapausten suunnitteluun ja toteutukseen on kiinnitettävä huomiota, jotta ne tuottavat luotettavia tuloksia.

Kun kaikki testit on hyväksytysti läpäisty, testatusta sisällöstä voidaan luoda varsinainen julkaisu mikäli kaikki tarvitta materiaali on valmiina. Samalla voidaan luoda julkaisuseloste (engl. Release notes), josta selviää julkaisuversiolle tehdyt testit, niiden tulokset ja ohjelmaan tehdyt muutokset.

6.3 Testaustyökalut

Testausjärjestelmän testi-PC:n käyttöjärjestelmänä käytetään Ubuntu Linuxia, johon on asennettu erilaisia testaustyökaluja ohjaamaan ja hallitsemaan prosessia.Ohjelmiston testauksessa käytettävät ohjelmistotyökalut on on esitelty taulukossa 4.

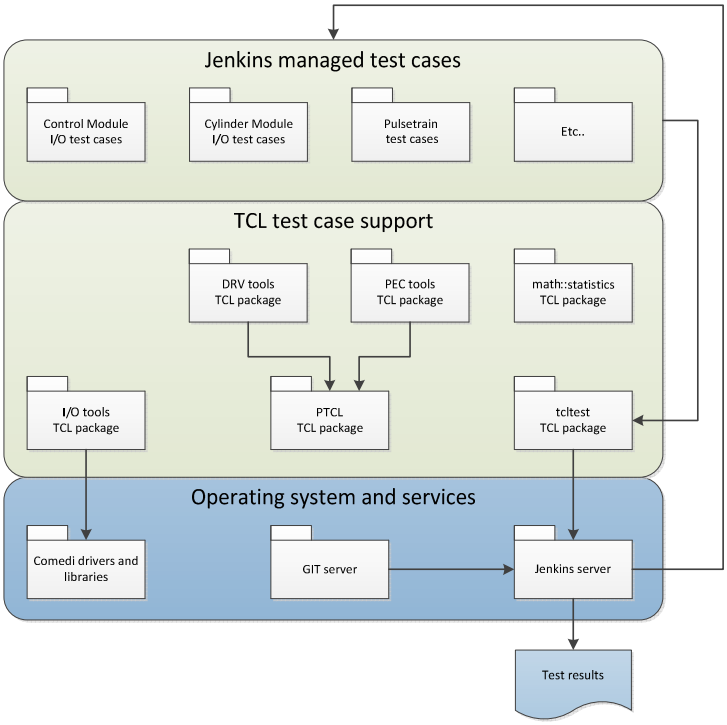
Taulukko 4 *Ohjelmiston testauksessa käytetyt ohjelmistotyökalut*

Task	Used software
Test PC operating system	Ubuntu Linux
Test execution control	Jenkins
C code unit testing	CppUTest
C code static analysis	CppCheck
Test unit control	PTCL wrapper written as TCL package Expect scripts
Test I/O control	Comedi libraries, utility applications, TCL package wrapper
Hardware in loop tests	Tcltest, official TCL unit testing framework JUnit report conversion tclunit

Testausohjelmien runkona käytetään jatkuvan integroinnin työkalua Jenkinsiä, jonka sovelluskehys mahdollistaa testitapausten ajamisen sekä yksikkötestauksessa että hardware-in-the-loop -testauksessakin. Jenkinsillä seurataan tehtyjä muutoksia koodin kuvauskannoissa ja reagoidaan niihin suorittamalla automaattisia testejä.

Ohjelmiston yksikkötestaus on toteutettu CppUTest -ohjelmistokehyksellä ja testi-PC:llä käännetään ja suoritetaan testit. CppUTest on C/C++ -pohjainen XUnit -testikehys, jolla tulosten raportointi on mahdollista JUnit -yhteensopivassa muodossa, joka voidaan helposti sisällyttää Jenkinsiin. Staattisen analyysin työkalua CppCheck:iä käytetään koodin lisätestauksessa ja sitä voidaan käyttää Jenkins liitännäisenä. CppCheck on pääasiassa työkalu, jolla pyritään havaitsemaan virheitä, joita tavallisilla kääntäjillä ei havaita. (CppUTest; CppCheck.)

Testausjärjestelmän mittaus- ja ohjausmoduulit perustuvat Advantechin moduuleihin, joita ohjataan testi-PC:lle asennetuilla Comedi-ajureilla ja -kirjastoilla. Yksinkertaisia komentoriviohjelmia tehtiin käyttäen PTCL (Procket Test Command Language) -paketteja, jotka tarjoavat testiskripteille yksinkertaisen yhteyden laitteistoon. Comedi-ajureiden tarjoama tuki järjestelmässä käytetyille advantechin moduleille on erittäin hyvä, mutta joitain muutoksia ajureihin oli tehtävä, jotta ne tukisivat kaikkia testitapauksia. Kuvassa 6.2 on esitetty kaaviokuva ohjelmiston testaustyökalujen pinosta (engl. Stack). (Comedi 2013)



Kuva 6.2 *Kaaviokuva testaustyökalujen pinosta*

Testattavien laitteiden ohjaus toteutettiin TCL -Expect skripteillä, mikä helpottaa Bootloader:n ja komentorivityökalujen keskinäistä toimintaa (Expect 1994). Tästä syystä Expect -skriptejä käytetään testilaitteiden alustukseen, oikeiden binäärien ja testi-image -tiedostojen lataamiseen.

TCL-paketti, joka käsittelee PTCL:n testikomentoja, on kasattu PTCL:ssä TCP/IP-socket:n päälle. Paketti mahdollistaa yksinkertaisen tuen komentojen ja kyselyjen tekemiseen sekä funktioita toleranssiarvojen tarkistukseen. PTCL-pakettia käytetään testitapauksissa, jotka on toteutettu TCL:n sisäänrakennetulla yksikkötestauskehyksellä. Kehys tarjoaa tavanomaiset testit, asetukset ja mekanismin yksittäisten testitapausten toteutukseen. Kehys mahdollistaa myös tiettyjen testitapausten suorittamisen komentoriviltä, jolloin voidaan määrittää yhden testintapauksen vaikutukset. TCL:n yksikkötestauksen raportointiformaatti ei ole Jenkinsissä tuettu, joten erillistä tclunit -ohjelmaa käytetään muuntamaan raportit JUnit -yhteensopivaan muotoon. Näin Jenkinsillä saadaan käsiteltyä testitapaustenkohtaisesti testien tilastot.

6.4 Yksikkötestaus

Yksikkötestaus on ensimmäinen askel ohjelmistosprosessissa, jossa varsinaisesti testataan toteutettua ohjelmaa. Yksikkötestauksen tarkoituksena on testauttaa eri ohjelmointirajapintoja (API, engl. Application programming interface) ja ohjelmistomoduuleja. Testaus keskittyy vain ohjelmiston testaukseen, eli muu laitteisto on suljettu testeista pois. Näin voidaan toteuttaa sisäänmenoarvoja, joiden toteuttaminen testilaitteistolla olisi hankalaa.

Testien tärkein tavoite on todentaa ohjelmiston oikeanlainen toiminta tilanteissa, joissa virheellisiä parametreja käytetään eri API-funktiokutsuissa. Tilanne tulee esiin esimerkiksi silloin, kun funktiota kutsutaan NULL-osoittimella (Null-pointer). Yksikkötestaus mahdollistaa paluuarvojen analysoinnin, jossa tarkastellaan onko paluuarvot oikeanlaisia nähden syötettyihin arvoihin.

Useimmista ohjelmistorajapinnoista testataan vain API-rajapintakerros (API-frontend) eikä varsinaisia toteutuksia, jotka korvataan testeissä malleilla. Niillä mahdollistetaan palautteen keräys funktiokutsuista, josta voidaan varmistaa etupään oikeellinen toiminta.

Testausta varten on toteutettu malleja, joiden avulla ohjelmistoyksiköitä voidaan testauttaa ilman että niihin tarvitsee tehdä muutoksia. Esimerkiksi Xenomai- funktiot on toteutettu yksinkertaisina tynkäfunktioina, jotka mahdollistavat ohjelman suorittamisen testi-PC:llä, jossa ei ole erillistä Xenomai-laajennusta. Xenomai on reaaliaikainen kehityskehys, joka toimii yhteistyössä Linuxin kernelin kanssa(Xenomai).

Ohjelmiston yksikkötestit jaetaan automaattisesti jokaisen muutoksen jälkeen, jotka tehdään ohjelmiston kuvauskantaan. Yksikkötestien toteutuksessa käytetään CppUTest ohjelmistokehyksestä.

6.5 Hardware-In-The-Loop-testaus

Ohjaus- ja sylinterimoduulille tehdään toisistaan riippumattomat integrointitason testaukset osana hardware-in-the-loop-testausta. HIL-testaus on ensimmäinen testausvaihe, jossa ohjelmistoa testataan testijärjestelmän laitteistolla ja testauksen tavoitteena on todentaa moduulien oikeellinen toiminta. Testauksessa käytetyt kytkennät on esitelty kappaleissa 5.4 Ohjausmoduulin testikytkennät, 5.5 Sylinterimoduulin testikytkennät ja 5.6 Moduulien yhteiset testikytkennät. Seuraavaksi esitellään ohjelmistotestauksessa käytetyt testisekvenssit.

6.5.1 Analogiset sisäänmenot

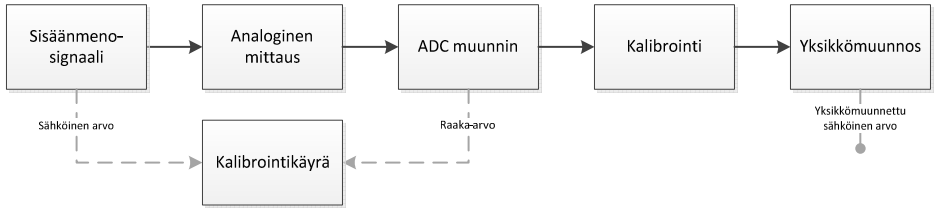
Testattavia analogisia sisäänmenokanavia ovat ohjausmoduulista ADI ja AI, joista jälkimmäisessä kaksi eri moodia, jännite- ja virtamoodi. Sylinterimoduulissa testattavia ovat FAI, Pietso (Piezo Feedback), Temp (termoparimoodi) ja AI (virtamoodi, jännitemoodi, termoparimoodi). Kaikissa tapauksissa sisäänmenosignaali syötetään jännitelähteestä ja kanavakohtaiset testausparametrit on esitetty liitteessä 1.

Jännitealueen ja virta-alueen testeissä generoidaan halutulla sisäänmenoalueelta jännitepyyhkäisy, joka jännitemoodissa vastaa suoraan sisäänmenoarvoa ja virtamoodissa yksikkömuunnettua virta-arvoa.

Aluksi kanavan sisäänmeno asetetaan haluttuun moodiin: jännite- , virta- tai termoparimoodiin riippuen siitä, onko testattavassa kanavassa moodinvalintaa. Sisäänmenoon syötetään 0 V:n jännite ja varmistetaan, että kanava näyttää haluttua arvoa.

Ennen varsinaista signaalitestiä kanavalle ajetaan kalibroinsekvenssi, jossa generoidaan jännitepyyhkäisy signaalialueen läpi. ADC-muuntimen raaka-arvo ja sisäänmenon arvo tallennetaan ja niistä luodaan sovituskäyrä, jota verrataan oletusarvoihin. Sisäänmeno palautetaan 0 V:n tasolle.

Testin varsinainen jännitealueen pyyhkäisy tehdään vastaavasti kuin kalibrointitestin jännitepyyhkäisy. Sisäänmenoon generoidaan haluttu jännitetaso, josta tallennetaan mittausketjun ulostulosta yksikkömuunnettu sähköinen arvo. Sähköistä arvoa verrataan sisäänmenon vastaavaan arvoon. Sisäänmenon jännite lasketaan takaisin 0 V:n jännitteeseen. Testisekvenssin lohkokaavio on esitetty kuvassa 6.3



Kuva 6.3 *Analogisten sisäänmenojen testisekvenssin lohkokaavio*

Testit ovat hyväksytysti suoritettu, kun kaikki vertailut täyttävät niille asetetut kriteerit. Tuloksista luodaan kuvaajat sekä sisäänmenoarvon ja yksikkömuunnetun arvon suhteesta että kalibroinnin sovituskäyrä ADC-muuntimen ja sisäänmenoarvon suhteesta, jotka näytetään Jenkinsissä.

6.5.2 Analogiset ulostulot

Analogisia ulostuloja moduuleissa on vain kahta eri tyyppiä, AO ja ADO. Kummassakin tapauksessa säädetään ulostulon virta-arvoa ja mitataan ulkoisen mittausvastuksen yli jännitettä. AO:ssa on kaksi virtamoodia (4-20 mA ja 0-200 mA) ja ADO:ssa yksi.

Testisekvenssin alussa valitaan kanavasta haluttu moodi ja ajetaan kalibrointisekvenssi. Kalibroinnissa ohjataan DAC-muuntimen raaka-arvoja ja mitataan sähköinen arvo mittausmoduulin AI-sisäänmenosta. Arvot tallennetaan ja sen jälkeen niista generoidaan kalibroinnin sovituskäyrä. Kalibroinnin jälkeen säädetään ulostulo nollatilaan.

Varsinaisessa testauksessa virta-arvoa ohjataan pienin askelin kattaen koko virta-alue. Virta-arvo mitataan tunnetun mittausvastuksen yli jännitearvona, josta tehdään yksikkömuunnos virta-arvoksi. Edellä mainituista luodaan kuvaaja, joka yhdessä kalibrointikuvaajan kanssa tallennetaan Jenkinsiin. Testit ovat hyväksytysti suoritettu, kun vaatimusten mukainen toiminta on virheettömästi saavutettu.

6.5.3 Digitaaliset sisäänmenot

Digitaalisten sisäänmenojen testisekvenssi perustuu ON/OFF-kytkiminä toimiviin releisiin. Ohjausmoduulissa digitaalisia sisäänmenokanavia ovat ADI, DI, USB sekä molemmissa testattavissa moduuleissa FDI ja ID. Sylinterimoduulin IP-pinnien testauksessa releiden tilalla käytetään DRV-kanavia, joiden ulostulot toimivat kytkiminä.

Testauksen alkuun asetetaan kanava haluttuun moodiin. Mikäli kanavassa on erilliset kynnysjännitteet (Threshold), ne säädetään haluttuihin arvoihin. Seuraavaksi ohjataan kytkin päälle ja tarkistetaan, että sisäänmenon tila on muuttunut. Vaihdetaan kytkimen asentoa ja varmistetaan, että sisäänmenon tila on jälleen muuttunut. Testisekvenssi toistetaan useita kertoja.

Testit ovat hyväksytysti suoritettu, kun testisekvenssin jokaisessa vaiheessa kanavan toiminta on vaatimusten mukaista. Jenkinsiin raportoidaan testien läpäisystä (PASS) ja hylkäyksestä (FAIL).

6.5.4 Digitaaliset ulostulot

Digitaalisia ulostulokanavia testattavissa moduuleissa on vain ohjausmoduulin ADO- ja DO-kanavat. Testissä kanavat toimivat kytkiminä, joiden tila tulkitaan relemoduulin digitaalisesta sisäänmenosta.

Kanava asetetaan haluttuun moodiin, jonka jälkeen ulostulo nollataan ja varmistetaan, ettei vikatiloja ilmaannu. Kanavan ulostulo kytketään päälle ja tilanmuutos luetaan testi-PC:ltä. Samalla tarkistetaan, ettei vikatiloja ole ilmaantunut. Seuraavaksi kanavan ulostulo kytketään pois päältä ja varmistetaan, että tilanmuutos näkyy myös testi-PC:llä. Testisekvenssi toistetaan useita kertoja.

Testit ovat hyväksytysti suoritettu, kun testisekvenssin jokaisesta vaiheessa kanavan toiminta on vaatimusten mukaista. Jenkinsiin raportoidaan testien läpäisystä tai hylkäyksestä.

6.5.5 PT1000-lämpötilatestaus

Temp-kanava mittaa lämpötilaa kahdessa eri moodissa, termoparissa ja PT1000:ssa. Termoparin testisekvenssi on esitetty kappaleessa 6.5.1 Analogiset sisäänmenot. PT1000 -testauksessa lämpötila-anturia simuloidaan neljällä vastuksella, joiden kombinaatioita hallitaan kahdella releellä.

Aluksi kanavan sisäänmeno asetetaan haluttuun moodiin ja varmistetaan, että releet ovat halutussa asennossa. Ennen varsinaista signaalitestiä kanavalle tehdään kalibroinsekvenssi, jossa mittausalue käydään läpi eri vastuskombinaatioilla. ADC-muuntimen raaka-arvo mitataan ja tallennetaan, jota verrataan sisäänmenon arvoon ja niistä luodaan kalibroinnin sovituskäyrä.

Kalibroinnin jälkeen tehdään kanavan varsinainen signaalitestaus, joka tehdään samalla periaatteella kuin kalibrointitestauskin. Tällä kertaa verrataan sisäänmenon

tunnettua arvoa ja mittausketjun ulostulon yksikkömuunnettua sähköistä arvoa. Arvoista luodaan kuvaaja, jota tarkastellaan määrättyjen toleranssien rajoissa. Testisekvenssi toistetaan useita kertaa. Testit ovat hyväksytysti suoritettu, kun kaikki vertailut täyttävät niille asetetut kriteerit.

6.5.6 Ylivirtasuojausten testaus

Testattavien kanavien sisäänmenon ylivirta- ja herätejännitteen ylivirtasuojauksen mekanismit testataan aiheuttamalla ulkoisesti suojauksen liipaisutason ylitys. Liipaisutasoissa ja vikojen aiheuttamistavoissa on eroja kanavien välillä ja ne riippuvat kanavatyypistä ja suojausmekanismeista. Sisäänmenon ylivirtasuojatestaus tehdään ohjausmoduulin AI- , ADI- ja DI-kanaville sekä sylinterimoduulin AI- ja FAI-kanaville. Herätejännitteen ylivirtasuojaus testataan ohjausmoduulin FDI-kanavalle ja sylinterimoduulin FDI-, AI- ja FAI-kanaville.

Ylivirtasuojien testauksessa kanavan sisäänmeno asetetaan aluksi haluttuun moodiin. Varmistetaan, että kanava näyttää normaalia arvoa eikä ylivirtasuoja ole kytkeytynyt päälle. Kytketään ylivirtatilanne päälle ja varmistetaan, että ylivirtasuoja on liipaissut. Yritetään nollata ylivirtasuojaus, jonka ei tulisi vaikuttaa ylivirtatilaan. Kytketään ylivirtatilanne pois päältä ja varmistetaan, että ylivirtasuojaus on edelleen päällä. Viimeisenä nollataan ylivirtasuojaus ja varmistetaan, että suojaus on kytkeytynyt pois päältä. Testisekvenssi toistetaan useita kertoja, jotta varmistutaan, että kanavan ohjaukset toimivat joka kerta oikein.

Testit ovat hyväksytysti suoritettu, kun testisekvenssin jokaisessa vaiheessa kanavan toiminta on vaatimusten mukaista. Jenkinsiin raportoidaan vain testien epäonnistumisista.

6.5.7 Pietsokanavien testaus

Pietsokanavalle testaus tehdään pistetaajuuksina sinisignaalilla, joka generoidaan testi-PC:n äänikortilla. Kanavan takaisinkytkennän testisekvenssi tehdään kuten analogisten sisäänmenojen testaus.

Testisekvenssin aluksi asetetaan kanavaan haluttu näytteenottotaajuus ja kytketään mittaus päälle. Seuraavaksi ohjataan testi-PC:llä olevan signaaligeneraattoria ajamaan pistetaajuudet yksi kerrallaan. Jokaisen pistetaajuuden kohdalla suoritetaan FFT-muunnos ja tallennetaan kanavan mittaama sähköinen arvo. Testisekvenssi toistetaan 10 kertaa. Viimeisen mitatun taajuuden jälkeen kanavan mittaus kytketään pois päältä, jotta tuloksiin ei tule turhaa mittausdataa.

FFT-muunnoksen arvoa verrataan sisäänmenevän signaalin taajuusparametreihin ja hyväksytään ennalta määrättyjen toleranssien rajoissa. Testin tulokset raportoidaan Jenkinsiin.

6.5.8 DRV-kanavien testaus

DRV-kanavien testauksessa varmistutaan, että kanavan toiminnallisuus on kunnossa ja varmistutaan virtamonitoroinnin toimivuudesta. Testauksessa käytetään kuormana joko resistiivistä tai induktiivistä kuormaa. Testitapauksissa testisekvenssit ovat lähes samanlaiset. Resistiivisessä tapauksessa kuorma rajoittaa virtaa ja induktiivisen kuorman tapauksessa virtaa reguloidaan.

Aluksi kanava asetetaan haluttuun moodiin. Kanavasta asetetaankulkemaan virtarajoitettu vakiovirta, jonka jälkeen kytketään kanavan ulostulo päälle. Mitataan sisään palaavan virran arvo kanavan virtatakaisinkytkennästä. Asetetaan virtamonitoroinnin maksimivirtaraja suuremmaksi kuin kanavan ulostulovirta. Varmistetaan, ettei monitoroinnin virtarajojen muuttaminen aiheuttanut varoitusta. Kytketään kanavan ulostulo pois päältä ja varmistetaan, ettei se aiheuttanut varoitusta.

Seuraavaksi asetetaan virtamonitoroinnin maksimivirtaraja alemmaksi kuin ulostulon virta-arvo. Kytketään ulostulo päälle ja tarkistetaan, että monitoroinnin varoitus on lauennut. Kytketään ulostulo pois päältä ja tarkistetaan, että virtamonitoroinnin varoitus pysyy edelleen päällä, kunnes kanavan virhenollaus tehdään.

Koko testisekvenssi toistetaan 10 kertaa, jotta varmistutaan virtamonitoroinnin toiminnasta. Testit ovat hyväksytysti läpäisty, kun jokainen testikierros on virheettömästi läpäisty. Jenkinsiin raportoidaan testien läpäisystä tai hylkäyksestä.

6.5.9 Tehonsyöttöjen testaus

Tehonsyöttöjä on kahdenlaisia, laitteiden PSS-tehonsyöttö, joka on kummassakin moduulissa ja sylinterimoduulin DRV-pankkien erillinen PSD-tehonsyöttö. Tehonsyötöille tehdään testauksessa jännitteen ja virran määrättyjen toleranssien rajoissa, kun tehonsyötön sisäänmenoa vaihdetaan laitteen ollessa käynnissä. Toisessa testissä kuormitetaan moduulia ja mitataan virrankulutuksen nousu. PSS-alive -testissä testataan, pysyykö testattava moduuli käynnissä, kun tehonsyöttö katkaistaan määrätyksi ajaksi.

Jännitteen tarkastelutesti aloitetaan varmistamalla, että moduuliin on kytketty jännitteet ja ne on ohjattu releellä haluttuun tehonsyöttöön. Sisäänmenon jännite mitataan ja tallennetaan, jonka jälkeen vaihdetaan tehonsyötön sisäänmenoa ja tarkastellaan kummankin tehonsyötön sisäänmenon jännitteet. Mitattuja arvoja verrataan jännitteen toleranssiarvoihin, joiden rajoissa ne saavat vaihdella. Seuraavaksi vaihdetaan jälleen tehonsyötön sisäänmenoa ja toistetaan mittaukset ja vertailut. Testisekvenssi toistetaan useita kertoja.

Virranmittauksen testisekvenssi on vastaavanlainen kuin jännitteen mittauksessa, mutta jännitteen sijasta tarkastellaan virrankulutuksen muutosta sille asetettujen toleranssien puitteissa. Testisekvenssi toistetaan 10 kertaa. Testit on hyväksytty, kun tehonsyötön mitatut arvot pysyvät toleranssirajojen sisäpuolella.

Virrankulutuksen nousun testisekvenssi aloitetaan mittaamalla ja tallentamalla senhetkinen sisäänmenon virta-arvo. Seuraavaksi kytketään tunnettu kuorma kiinni ja mitataan virta uudelleen. Arvoa verrataan laskennallisesti odotettuun arvoon. Sylinterimoduulin tapauksessa kytketään vielä toinen kuorma kiinni ja toistetaan sekä mittaus että vertailu uudelleen. Testisekvenssi toistetaan useita kertoja. Testit ovat hyväksytysti suoritettu, kun mitatut virrankulutuksen nousut vastaavat määriteltyjen toleranssien sisäpuolella.

PSS-alive -testisekvenssin aluksi varmistetaan, että moduulin tehonsyöttö on kytketty päälle. Sen jälkeen mitataan ja tallennetaan senhetkinen sisäänmenon virta-arvo. Tehonsyöttö kytketään ennalta määrätyksi ajaksi pois päältä ja kytketään jälleen päälle, jonka jälkeen tarkastellaan, pysyikö testattava moduuli käynnissä. Testisekvenssi toistetaan 10 kertaa. Testi on hyväksytysti suoritettu, kun moduuli pysyy käynnissä koko testin ajan.

6.5.10 Pulssijonokytkennän testaus

Työn tässä vaiheessa pulssijonon testauksessa keskitytään vain testikytkennän (ks. Kappale 5.6.4) toiminnan varmistamiseen. Testikytkentä yhdistää ohjausmoduulin ja sylinterimoduulin yhdessä toimivaksi yksiköksi ja testaa FDO-kanavan DO-moodin toiminnallisuutta.

Testisekvenssin aluksi määritetään kaikkien testattavien FDI-kanavien kynnysjännitteet ennalta määriteltyihin arvoihin. Testissä signaalilähteenä käytetään FDO-kanavaa, jonka ulostulo säädetään DO-moodiin ja kytketään päälle. Aluksi varmistetaan, että suoraan FDO-kanavaan kytketyt FDI-kanavat ovat vaihtaneet tilaansa aktiiviseksi. FDO:n ulostulo kytketään pois ja varmistetaan, että FDI-kanavien tila on muuttunut.

Seuraavaksi määritellään DRV-kanavat haluttuun moodiin, jotta ne toimivat kytkiminä. Tällöin varmistetaan, että DRV-kanaviin kytkettyjen FDI-kanavien tila ei ole aktiivisena. Muutetaan DRV-kanavat lepotilaan ja varmistetaan, että FDI-kanavat ovat vaihtaneet tilaansa aktiiviseksi.

Jokaisen FDI-kanavan kohdalla testisekvenssi toistetaan useita kertoja. Jenkinsiin raportoidaan testien läpäisystä tai hylkäyksestä.

7 YHTEENVETO JA JATKOKEHITYSAJATUKSET

Diplomityö tehtiin osana suurempaa tuotekehitysprojektia, jonka ohjelmistotestaukseen tarvittiin automatisointia. Suunnittelu ja kehityskohteena oli edullinen automatisoitu testausjärjestelmä, jolla kyettäisiin luotettavasti ja nopeasti testaamaan ohjelmistoa sen eri kehitysvaiheissa. Kehitystyö tehtiin yhteistyössä ohjelmistosuunnittelijoiden kanssa, joiden avuksi testausjärjestelmää oltiin luomassa. Työn edetessä selvitettiin testauslaitteiston ominaisuuksia ja rajoitteita, joiden perusteella testitapaukset ja testikytkennät suunniteltiin ja toteutettiin. Vaikka testikytkennät ja mittauslaitteisto suunniteltiin tuotekehitysprojektin omien tarpeiden mukaan, saimme työn tuloksena aikaan järjestelmän, joka on mahdollista ottaa käyttöön muissakin projekteissa.

Testausjärjestelmän laitteiston rajoitti joissain tilanteissa sisäänmenojen määrää ja signaalialueita. Laitteiston rajoitukset johtuivat PCI-korttien vanhentuneesta tekniikkasta, joka esimerkiksi mahdollisti vain kolmen PCI-kortin käyttämisen. Tästä syystä PCI-korttipohjaisen laitteiston laajentaminen on haastavaa. Toisaalta laitteiston muuntelu ja ylläpito on yksinkertaista, koska testitapaukset rajattiin koskemaan vain muutamia kanavatyypin kanavia. Järjestelmän laajennettavuuteen ja signaalialueiden rajoituksiin mahdollisena ratkaisuna pidetään PCI-korttien päivittämistä nykyäaikaisempiin ratkaisuihin, kuten USB-liitettäviin moduuleihin. Samalla voidaan saavuttaa laajempi testauskattavuus ja mahdollistaa tarkempien mittausten tekeminen.

Testausjärjestelmän kehitystyötä helpotti, kun ohjelmointikielet ja menetelmät olivat ohjelmistosuunnittelijoille tuttuja. Koska varsinainen ohjelmiston ja testausjärjestelmän kehityksestä vastasi samat henkilöt ja niitä tehtiin samanaikaisesti, testausjärjestelmän kehitykseen ei aina jäänyt tarpeeksi aikaa. Ongelma olisi voitu välttää, jos testausohjelmiston kehitystyötä varten olisi ollut oma henkilö. Kuitenkin laitteiston testauksen ja kehityksen ollessa nyt saman ryhmän vastuulla voitiin testituloksiin reagoida nopeammin kuin, että olisi tarvinnut muodostaa erillisiä ohjelmistojulkaisuja testausryhmälle. Tällä menetelmällä on ohjelmistovirheiden lisäksi löydetty useita laitteistovikoja.

Ohjelmiston regressiotestaus auttoi ohjelmistokehittäjiä huomaamaan koodiin päässeet virheet varhaisessa vaiheessa. Tämä paransi ohjelmiston luotettavuutta ja vähensi myös laitteistotestauksen työmäärää. Tarkkuuksien testaaminen ei toteutetulla järjestelmällä ole mahdollista, joten niiden parantaminen tuottaisi laajemman kuvan laitteiston toiminnasta ja paljastaisi tarkemmin ohjelmistovirheitä. Vaikka järjestelmästä tehtiin pitkälle automatisoitu jäi käyttäjälle paljon tehtävää. Testikattavuuden

saavuttamista ei pysty nykyisessä järjestelmässä järkevällä tavalla todentamaan, vaan työ on tehtävä käsin käyttäjän toimesta.

Testausjärjestelmä on työn tässä vaiheessa vielä osittain kehityksen alla ja ylläpito työn alkuvaiheessa. Tämän takia lopullista työmäärää ei ole mahdollista arvioida. Kehitystyö on vaatinut aikaa, mutta automatisoinnin ansiosta hyödyt näkyvät vasta projektin myöhäisemmässä vaiheessa. Järjestelmästä saatiin toteutettua kustannustehokas verraten ehdolla olleisiin vaihtoehtoihin. Testi-PC:n toiminnan jakautuminen projektissa muihinkin tehtäviin kuin pelkästään testausjärjestelmän runkona toimimiseen, jakaa kustannuksia pienempiin osiin. Testausjärjestelmän käyttö ja ylläpito tulee jatkumaan tuotekehitysprojektin loppuun asti ja auttaa näin ohjelmistokehityksen läpiviemistä eri laitteistoversioiden kanssa.

LÄHTEET

Barr, M., Massa, A. 2006, Programming Embedded Systems: with C and GNU develpment tools 2nd Edition, O'Reilly, 301 p.

Berg, A., 2012, Jenkins Continuous Integration Cookbook, Packt Publishing Ltd., 328 p.

Burnstein, I. 2003, Practical Software Testing: a Process-Oriented Approach, Springer-Verlag, 709 p.

CppCheck. [WWW]. [Viitattu 27/9/2013] http://cppcheck.sourceforge.net/

CppUTest. What is CppUTest [WWW]. [Viitattu 27/9/2013] http://cpputest.github.io/

Comedi. 2013. Comedi linux control and measurement device interface, [WWW]. [Viitattu 28/9/2013] Saatavissa: http://www.comedi.org/index.html

Expect. 1994. Manpage of Expect. [WWW]. [Viitattu 28/9/2013] Saatavissa: http://www.tcl.tk/man/expect5.31/expect.1.html

Fewster, M., Graham, D. 1999, Software Test Automation: Effective use of test execution tools, Addison – Wesley, 572 p.

Fowler, M. , 2006, Continuous Integration, [WWW]. [Viitattu 22/9/2013] Saatavissa: http://www.martinfowler.com/articles/continuousIntegration.html

Haikala, I., Märijärvi, J., 2000, Ohjelmistotuotanto 7. Painos, Satku, 414 s.

Humble, J., Farley, D. 2010, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison – Wesley 463 p.

Jenkins. Meet Jenkins. [WWW]. [Viitattu 24/9/2013] https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins

Kawalerowicz, M., Berntson, C. 2011, Continuous Integration in .NET, Manning, 382 p.

Myers, G. 2004. The Art of Software Testing 2nd Edition. John Wiley & Sons, Inc. 151 p.

NI-PXI. [WWW]. [Viitattu 8/10/2013] Saatavilla: http://sine.ni.com/np/app/main/p/ap/global/lang/fi/pg/1/sn/n24:PXI-FSLASH-CompactPCI/

Phoenix. [WWW]. [Viitattu 30/9/2013] Saatavilla: http://eshop.phoenixcontact.net/phoenix/images/productimages/size1/22741_1000_int_01.jpg

Pressman, R. 2010. Software Engineering: A Practitioner's Approach 7th Edition, R. S. Pressman & Associates, Inc. 895 p.

Procket. Procket Test Platforms [WWW]. [Viitattu 26/9/2013] http://www.espotel.com/procket

Sommerville,I . 2011. Software Engineering 9th Edition, Addison – Wesley 773 p.

Xenomai. Xenomai: Real-Time Framework for Linux. [WWW]. [Viitattu 9/10/2013] Saatavilla: http://www.xenomai.org/

Analogiset ulostulot/sisäänmenot	Ohjausmoduuli						Sylinterimoduuli					
	AI		ADI	AO		ADO	AI			FAI	Temp	Pietso
Parametri	mA	V	mA	mA	mA	mA	mA	V	TC	mA	TC	Feedback
Aloitus jännite/virta	4 mA	-10 V	4 mA	4 mA	4 mA	4 mA	4 mA	0 V	-3mV	4 mA	-3mV	0,1 V
Lopetus jännite/virta	15 mA	10 V	15 mA	200 mA	20 mA	20 mA	15 mA	5 V	33mV	15 mA	33mV	3,3 V
Askeleen koko	0,5 mA	0,5 V	0,5 mA	5 mA	0,5 mA	0,5 mA	0,5 mA	0,1 V	1mV	0,5 mA	1mV	0,1 V
Pyyhkäisyn toleranssi	1 %	1 %	1 %	1 %	1 %	1 %	1 %	1 %	1 %	1 %	1 %	1 %
Oletetut ADC sovitusarvot*	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx	Gain xx Offset xx
Sovitus toleranssi	5 %	5 %	5 %	5 %	5 %	5 %	5 %	5 %	5 %	5 %	5 %	5 %

Ylivirtasuojausten testaus	Ohjausmoduuli				Sylinterimoduuli				
	AI	ADI	DI	FDI-Ext	AI	AI-Ext	FAI	FAI-Ext	FDI
Parametri	mA	mA	V	mA	mA	mA	mA	mA	mA
Normaali arvo	24 mA	24 mA	24V	24 mA	24 mA	24 mA	24 mA	24 mA	24 mA
Suojauksen liipaisutaso	40 mA	41 mA	20V	47mA	36 mA	47mA	30 mA	47mA	47mA
Virran vertailutoleranssi	1 %	1 %	1 %	1 %	1 %	1 %	1 %	1 %	1 %

Digitaaliset sisäänmenot	Ohjausmoduuli				Sylinterimoduuli	
	ADI	DI	FDI	ID	FDI	ID
Parametri	DI	DI	DI	ID	DI	ID
Kytkintyyppi	Rele	Rele	Rele	Rele	Rele	DRV
Nousevan reunan kynnysjännite	-	-	3.0V	-	3.0V	-
Laskevan reunan kynnysjännite	-	-	1.0V	-	1.0V	-

* Sovitusarvot saadaan mittaamalla

PT-1000- Lämpötilatestaus

	Sylinterimoduuli			
	TEMP			
Parametri	PT1000			
Anturisimulaattorin resistanssi	559Ω	1558Ω	2060Ω	3058Ω
Vertailutoleranssi	1 %	1 %	1 %	1 %

Pietsokanavan testaus

	Sylinterimoduuli				
Parametri	Pietso				
Testisignaalin taajuudet	500Hz	1000Hz	2000Hz	4000Hz	8000Hz
Vertailutoleranssi	2 %	2 %	2 %	2 %	2 %

DRV-kanavien testaus

	Sylinterimoduuli	
	DRV	
Parametri	ON/OFF	HSD
Kuormavirta	~0.5A	1.0A
Kuormatyyppi	Resistiivinen	Induktiivinen

Tehonsyöttöjen testaus

	Ohjausmoduuli		Sylinterimoduuli			
Parametri	PSS1	PSS2	PSS1	PSS2	PSDxA	PSDxB
Aktiivisen teholähteen jännitetoleranssi	23.0 - 25.0V	23.0 - 25.0V	23.0 - 25.0V	23.0 - 25.0V	23.0 - 25.0V	23.0 - 25.0V
Passiivisen teholähteen jännitetoleranssi	0.0-0.5V	0.0-0.5V	0.0-0.5V	0.0-0.5V	0.0-0.5V	0.0-0.5V
Virranmittauksen toleranssi	5 %	5 %	5 %	5 %	-	-
Virrankulutuksen kuorma 1	30mA	-	30mA	-	-	-
Virrankulutuksen kuorma 2	-	-	60mA	-	-	-
Virrankulutuksen nousun toleranssi	5 %	5 %	5 %	5 %	-	-