ERÄÄN TOIMILAITTEEN TOIMINNALLINEN TESTAUS JA LAADUNVARMISTUS

Mikko Santala

Opinnäytetyö
Syyskuu 2015
Auto- ja kuljetustekniikka
Auto- ja työkonetekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Auto- ja kuljetustekniikka
Auto- ja työkonetekniikka

MIKKO SANTALA:
Erään toimilaitteen toiminnallinen testaus ja laadunvarmistus

Opinnäytetyö 61 sivua, joista liitteitä 18 sivua
Syyskuu 2015

Opinnäytetyön tilaajana oli tamperelainen konepajayhtiö Tasowheel Gears Oy. Tehtävänä oli suunnitella ja toteuttaa yritykselle uuden, vasta tuotantoon tulossa olevan, tuotteen toiminnalliseen testaukseen ja laadunvarmistukseen soveltuva testilaitteisto ja ohjelmisto. Järjestelmän toteuttamiseen käytettiin National Instruments:in laitteita ja niiden kanssa yhteensopivaa Labview – ohjelmaa. Testausjärjestelmän tuli olla mahdollisimman automaattinen, jotta testaaminen sitoisi itseensä mahdollisimman vähän työvoimaa ja niin helppokäyttöinen, että lähes kuka tahansa osaisi testata tuotteen sillä. Sen tuli siis pystyä mittaamisen lisäksi ohjaamaan toimilaitetta itsenäisesti. Järjestelmän tuli myös pystyä päättelemään mittaustulosten perusteella, onko tuote läpäissyt testin ja muodostaa tuloksista testiraportti. Testiraportti lähtisi sitten tuotteen mukana loppuasiakkaalle. Laitteistosta ja ohjelmasta laadittiin myös asianmukaiset käyttöohjeet ja muut dokumentit.

Tuotteen testaaminen ja laadunvarmistus piti sisällään mm. toimilaitteen liikematkan, liikeajan, vuotoarvojen ja liikkeellelähtöpaineen mittaamisen ja ulkoisten vuotojen tarkastuksen. Ohjelmasta tehtiin parametrisonnilla mahdollisimman hyvin toiminta-arvoiltaan muokattava, sillä monista toiminta- ja raja-arvoista ei ollut vielä tietoa, koska tuotteita ei ollut testattu käytännössä lainkaan ennen tämän testausjärjestelmän tekoa. Mittausdatan ja järjestelmän toiminta-arvojen hallinta toteutettiin SQL – tietokannalla.

Yritykselle täysin uudenlaisen tuotteen testaamisen aloittaminen oli haastava projekti. Koko testausprosessia kehitettiin opinnäytetyön tekemisen aikana, joten muutoksia ja lisäyksiä sekä laitteistoon, että ohjelmaan tuli jatkuvasti. Tämä aiheutti runsaasti haasteita. Lisäksi projektin jo ollessa pitkällä, ilmeni, että tuotteen valmistuksen tulisi olla toimilaitetyyppiä koskevan standardin mukaista. Standardissa määriteltiin vaatimuksia myös testaamiselle, joten muutoksia tuli sieltäkin. Testausjärjestelmästä tuli kuitenkin lopulta käyttötarkoitukseensa soveltuva ja yrityksen toivomusten mukainen, ja se meni oikeaan käyttöön tuotantoon.

Opinnäytetyön julkisen version sisällöstä on poistettu kaikki toimilaitteen tyyppiin tarkemmin viittaavat tiedot. Nämä tiedot on asetettu salatuiksi liitteiksi.

Asiasanat: testaus, automaattinen, toimilaite, Labview, ohjelmointi, SQL, tietokanta.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Automobile and Transport Engineering
Automobile and Industrial Vehicle Engineering

MIKKO SANTALA:
Functional testing and quality assurance of an actuator

Bachelor's thesis 61 pages, appendices 18 pages
September 2015

The assignment for this thesis came from Tasowheel Gears Oy. The task was to design and implement a test system for functional testing and quality assurance of an actuator. Test system would include physical test equipment and software for PC. The system was implemented with National Instruments hardware and their Labview software. Test system was intended to be as automatic as possible so there would be minimum amount of manpower and time consumed in the testing process. This meant that the test system should also be capable of controlling the actuator in addition to measuring. It should also be able to determine autonomously whether the actuator under test passed the test or not based on the measured data. A test report would be printed at the end of test and delivered to customer as validation of passed test. Appropriate documents about the software and hardware were made for staff training and system maintenance.

The actuator under test was a new product that was just coming to production. The company had no experience in testing this product so there were quite a lot of adjusting and experimenting different things when developing the test process. The functional testing and quality assurance included measuring the actuator's pressure leakage, range of movement, stroke time, moving pressure and external leakage. The system was made to be decently modified by parametrization. That was done by adding SQL database to the system to contain all the actuator dependent limits and system's general functioning values such as data acquiring rates, test pressures and directional valve coil voltages.

The project was quite challenging because the product was new and it had not been tested before. The test principle and setup were changed a number of times and a lot was added to the system during the process which made it difficult to maintain a well-organized structure in the software. After all, the implemented test system came out really functional and filled its purpose. It meets the company's and standard's requirements and exceeds them in some areas. It gives added value to the company and is expected to stay in active use in the Tasowheel Gears Oy production.

All information about the actuator's specific nature has been deleted from the public version of this thesis and set as hidden attachments.

Key words: test, automatic, actuator, Labview, programming, SQL, database

SISÄLLYS

LYHENTEET JA TERMIT

Labview	graafisen ohjelmointikielen ohjelmisto
VI	virtuaalinen instrumentti, Labview – kielellä ohjelmoitu ohjelma
SQL	structured query language, relaatiotietokantakieli
loop	Labview – koodin rakenne, joka toistaa koodiaan kunnes se suljetaan

1 JOHDANTO

1.1 Tehtävänanto

Opinnäytetyön aiheena on tuotteen testaamiseen ja laadunvarmistukseen vaadittavan mittausprosessin luominen. Työnimenä oli erään toimilaitteen toiminnallinen testaus ja laadunvarmistus. Työn tilaaja on tamperelainen konepajayritys Tasowheel Gears Oy, joka on osa Tasowheel Group – konsernia. Tasowheel Gears on erikoistunut voimansiirron komponenttien valmistukseen ja suunnitteluun.

Tehtävänä oli suunnitella ja toteuttaa toimilaitteen testaamiseen ja laadunvarmistukseen soveltuva testausjärjestelmä, joka koostuu tietokoneohjelmasta ja fyysisestä mittalaitteistosta. Testaamisen kohteena oli kuitenkin yritykselle osittain täysin uudenlainen tuote. Tuote on toimilaite, jota on kuusi eri kokoa, ja joiden ääripäillä on hyvin erisuuruiset liikenopeudet suuren kokoeronsa vuoksi. Järjestelmän toiminta-arvoja tuli siten pystyä muuttamaan testattavan toimilaitekoon mukaan. Mittaukset sisälsivät paineen ja aseman mittaamista. Toimilaitteesta testattaisiin järjestelmällä paineenpitokykyä, ulkoisia vuotoja ja liikematkaa. Tarkoituksena oli, että testausjärjestelmä tuottaisi mittauspöytäkirjan tai testiraportin, jolla voitaisiin asiakkaalle todentaa tuotteen olevan vaatimusten mukainen sekä suorituskyvyltään, että laadultaan. Testauslaitteiston ja – ohjelman lisäksi yritykselle laadittaisiin myös asianmukaiset käyttöohjeet ja dokumentit laitteistosta ja ohjelmasta.

Mittausjärjestelmän tuli olla teollisuuskäyttöön soveltuva, kestävä ja kohtuullisen helppokäyttöinen. Tehtävänannon mukana saatiin alustava hahmotelma testausprosessin kulusta, jonka pohjalta järjestelmää lähdettiin suunnittelemaan. Yrityksen toivomuksena oli, että järjestelmästä tehtäisiin mahdollisimman automaattinen, jolloin testaukseen sitoutuisi mahdollisimman vähän työvoimaa. Tämä tarkoitti siis sitä, että mittaamisen lisäksi laitteiston tuli pystyä myös ohjaamaan muita laitteita.

1.2 Opinnäytetyön tavoitteet

Opinnäytetyön tavoitteena oli tuottaa asiakasyritykselle sen toivomuksia mahdollisimman hyvin vastaava lopputuote, joka olisi niin toimiva ja käyttötarkoitukseensa soveltuva, että se myös jäisi käyttöön ajan kuluessa. Opiskelijan näkökulmasta tavoitteena oli tämän lisäksi oppia millaista tuotteen testaaminen ja laadunvarmistaminen tuotannossa on, ja mitä kaikkea sen toteuttaminen vaatii. Kyseessä oli yritykselle melko uuden tyyppinen tuote, joten oli myös mahdollisuus päästä näkemään minkälaisia haasteita uuden tuotteen kehittäminen ja tuotantokuntoon saattaminen yritykselle asettaa. Tämän aiheen parissa oli tilaisuus päästä opiskelemaan testaus – ja ohjausjärjestelmien toteuttamisessa monikäyttöisen Labview – ohjelmiston käyttöä. Relaatiotietokantakin tuli täysin uutena asiana mukaan projektiin.

2 TEORIA

2.1 Mittausjärjestelmän suunnittelu

Testausjärjestelmän suunnittelussa ja kehittämisessä sovellettiin Olli Aumalan esittämiä periaatteita hyvän lopputuloksen saavuttamiseksi. Hänen mukaansa mittausten onnistumisen kannalta järjestelmä tulee olla suunniteltu niin, että käytettävä mittalaite soveltuu mittaussuureen mittaamiseen, käyttäjä osaa käyttää mittausjärjestelmää. Mittalaitteen tulee olla riittävän tarkka ja mitattavalle suureelle sopiva, mutta tunteeton käyttöolosuhteille ja muille kuin mitattavalle suureelle (Aumala, Mittaustekniikan perusteet, 1989).

Mittaustehtävän työvaiheet:

1. tehtävän määrittely
2. suoritusvaihtoehtojen etsiminen ja vertailu
3. tarkoituksenmukaisuuden tarkistus
4. epävarmuuden ennakkoarviointi
5. mittalaitteiden tarkistukset ja kalibroinnit
6. mittausten suoritus
7. tulosten edustavuuden arviointi
8. tulosten käsittely
9. tulosten kelvollisuuden arvostelu
10. dokumentointi ja tulosten informointi asianomaisille

(Aumala, Mittaustekniikan perusteet, 1989)

2.2 Labview – ohjelmointi

National Instruments:n Labview – ohjelma käyttää sen omaa avointa graafista ohjelmointikieltä. Se koostuu ikoneista ja niitä yhdistävien johtimien kokonaisuuksista ja niitä ympäröivistä koteloista ja rakenteista. National Instruments valmistaa myös modulaarisia mittalaitteita, joita yhdessä käyttämällä voidaan luoda yksilöllisiä testaus- ja ohjausjärjestelmiä sekä sulautettuja järjestelmiä. Labview:n käyttömahdollisuudet

ovat sen laajuuden vuoksi lähes rajattomat, mutta ohessa on lyhyesti selvitetty tämän opinnäytetyön aiheena olleen sovelluksen kannalta tärkeimpien elementtien ja rakenteiden perusteita.

2.2.1 Ohjelmointiympäristö

Labview:n ohjelmointiympäristö koostuu etupaneelista ja lohkokaaviosta. Niiden lisäksi ohjelmassa on vielä projektinäkymä, jossa voidaan hallita suuremman ohjelmakokonaisuuden hierarkiaa ja kansiorakenteita. Kun ohjelmointi aloitetaan, sekä etupaneeli ja lohkokaavio ovat tyhjiä tauluja, joissa ei ole mitään. Jos ohjelma sisältää graafisen käyttöliittymän, ohjelman tekeminen aloitetaan usein käyttöliittymän suunnittelusta ja tekemisestä etupaneeliin. Labview:ssä on valmiita näyttöjä, säätimiä ja nappuloita, joita yhdistelemällä voidaan helposti tehdä näyttäviäkin käyttöliittymiä.

Lohkokaavion puolella tehdään varsinainen ohjelmointi. Siellä lohkokaavioon raahataan ikoneja ja kuvakkeita, ja yhdistellään niitä erivärisin johtimin. Sinne ilmestyy myös etupaneeliin sijoitettujen asioiden kuvakkeet. Johtimen väri ja paksuus riippuu käytetystä datatyypistä. Johtimet siirtävät tietoa ikonien ja kuvakkeiden, eli solmujen, välillä. Solmu sisältää aina jonkin koodin, joka suorittaa jonkin toiminnon.

Tieto kulkee Labview:ssä vuokaavion lailla pääsääntöisesti vasemmalta oikealle, joten kooditkin tehdään siinä suunnassa luettaviksi. Koodien suoritusjärjestyksiä voidaan kuitenkin pakottaa ja varmistaa mm. virhejohtimella tai tietyillä jaksotusrakenteilla. Tämä on hyödyllistä esimerkiksi, jos halutaan tehdä koodista helppolukuinen jäsentelemällä eri toimintoihin liittyviä koodeja vaikka omille riveilleen.

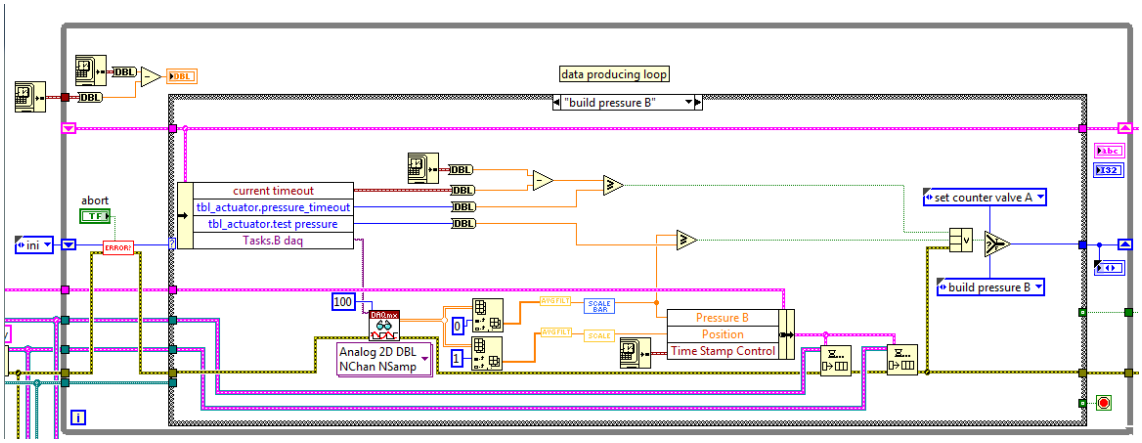
2.2.2 Virtuaalinen instrumentti

Labview:ssä yksi ohjelma on nimeltään virtuaalinen instrumentti, eli VI. Yksinkertainen sovellus voi koostua jopa vain yhdestä VI:stä, mutta laajemmissa sovelluksissa niitä tarvitaan useita, sillä kaikki koodi ei millään mahdu yhteen näytölliseen. Koodia voidaan siistiä kääntämällä koodin osia alemman tason VI:ksi. Laajoissa sovelluksissa

on usein yksi päätason VI, jonka alla on sitten useita perustason VI:tä. VI:tä voidaan yhdistellä toisiinsa johtimilla, jolloin tieto saadaan liikkumaan niiden välillä. VI:n etupaneelissa on liitintaulu, johon voidaan määrittää VI:n sisään – ja ulostulot.

2.2.3 Tilakone

Tilakone koostuu nimensä mukaisesti tiloista. Tilat sisältävät Labview:n tapauksessa palan koodia, joka suorittaa jonkin toiminnon. Yleensä koodi on luonteeltaan sellainen, että sitä voidaan toistaa, eli pysyä samassa tilassa, kunnes jokin asetettu ehto täyttyy. Alla olevassa kuvassa (KUVA 1) on testausohjelman koodi, jossa mitataan kahta signaalia niin kauan, kunnes paine B on suurempi tai yhtä suuri kuin asetettu tavoitepaine. Tilaan on myös lisätty laskuri, joka varmistaa, että tilasta siirrytään seuraavaan, jos asetettua painetta ei ikinä saavutetakaan. Seuraavan tilan valintaa voidaan monipuolistaa lisäämällä siihen esimerkiksi Boolen logiikkaa, kuten alla on tehty. Jos ehto ei täyty, pysytään samassa tilassa, ja sen täyttyä siirrytään seuraavaan. Monesti tilakoneessa on myös siirtorekisteri, jossa voidaan kuljettaa tietoa tilasta toiseen.

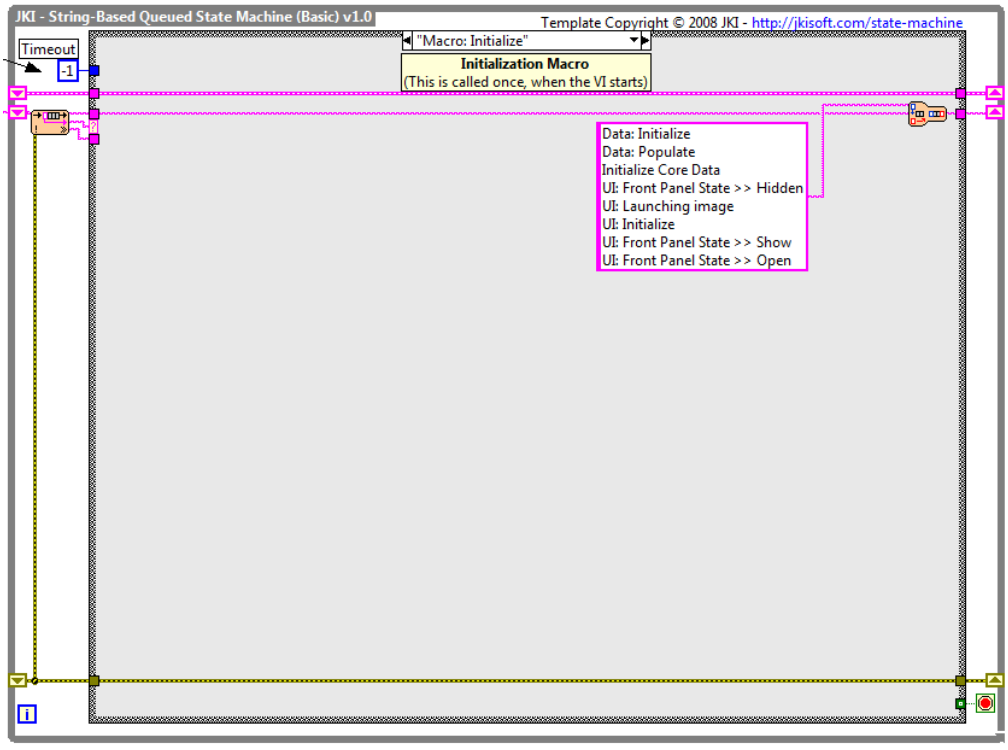


KUVA 1. Tilakone Labview – koodina (Mikko Santala 21.8.2015)

Siirtorekisterissä voidaan siirtää tilasta toiseen mitä vain tietoa kunhan se on määritelty ohjelman alustamisessa, joka tapahtuu yleensä ohjelman ensimmäisessä tilassa. Kun tieto laitetaan siirtorekisteriin, se on saatavilla sieltä milloin tahansa ja säilyy siellä,

kunnes sitä muutetaan tai poistetaan. Usein tietoa kuljetetaan klustereissa, jotta tiedot ovat siellä selkeämmin ja jäsennellysti.

JKI – tilakone on JKI - firman tekemä, hieman viritetty tilakone. Siinä tilojen järjestys ja ohjelman toteutusjärjestys määritellään listamuotoon kirjoitetuilla tilojen nimillä ja käskyillä (KUVA 2). Tämä mahdollistaa modulaarisen koodaamisen ja tekee ohjelman muuttamisen helpoksi, koska tilojen läpikäyntijärjestyksen muuttamiseksi koodia ei tarvitse muuttaa, vain kirjoitettua listaa. JKI on hyvä arkkitehtuuri etenkin käyttöliittymien ja päätason VI:den toteuttamiseen. JKI – tilakoneen saa valmiina koodina Labview:iin lataamalla laajennusosan.



KUVA 2. JKI – tilakone (Mikko Santala 21.8.2015)

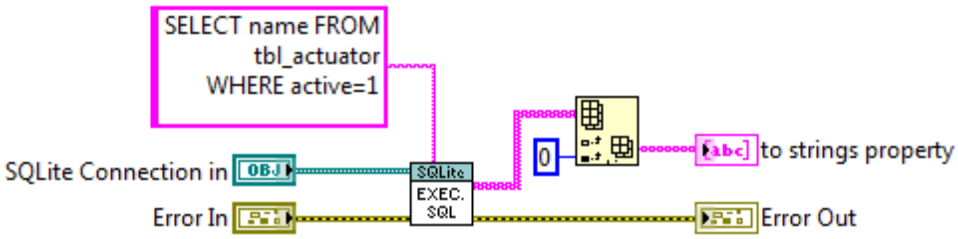
Tilakoneista voidaan tehdä tapahtumapohjaisia lisäämällä siihen tila, jossa on tapahtumakotelo. Sitä käytetään VI:n etupaneelin tapahtumien havaitsemiseen. Tapahtumalle lisätään oma kotelo, jonka sisältämä koodi tai toiminto suoritetaan, kun määritelty tapahtuma esiintyy. Sellainen tapahtuma voi olla yksinkertaisimmillaan painonapin painaminen tai ikkunan sulkeminen sen yläkulmassa olevasta ruksista.

2.2.4 Jono

Jono – rakennetta käytetään tiedon siirtämisessä loopista toiseen. Siihen liittyy olennaisesti tuottaja – käyttäjä – arkkitehtuuri. Siinä yksi loop tuottaa ja lisää jonoon dataa ja yksi tai useampi ottaa dataa pois jonosta ja käyttää sitä. Tehdyssä mittaussovelluksessa tuottajassa kerätään mittausdata ja ensimmäisessä käyttäjässä tallennetaan se ja toisessa näytetään se näytöllä kuvaajassa. Jonolle täytyy määrittää ohjelman alussa siinä siirrettävä datatyyppi, minkä jälkeen se on valmis käytettäväksi. Tehdyssä ohjelmassa jonon datatyypiksi määritettiin neljän elementin klusteri, joka sisältää yhden näytteen paineesta A, paineesta B, asemasta ja aikaleiman. Jono toimii First in first out – periaatteella eli ensimmäisenä sinne laitettu tieto tulee myös ensimmäisenä sieltä pois. Ohjelmaa suljettaessa jono tuhotaan joko pakotetusti, eli heti, tai kunnes se on tyhjä. Mittaussovelluksen ollessa kyseessä, odotetaan kunnes jonosta on saatu kaikki data tallennettua ja tuhotaan se vasta sitten, jotta kaikki mitattu data saadaan talteen.

2.2.5 Tietokantatoiminnot

Labview sisältää valmiita koodeja toimintojen tekemiseen tietokannan kanssa. Niillä voidaan tavallisella SQL – kielellä tehdä kyselyjä ja muita toimintoja tietokantaan. Alla (KUVA 3) pyydetään SELECT – lauseella tietokannasta kaikkien aktiivisten tuotteiden nimet, joita syötetään testausohjelmassa mm. tiputusvalikoihin, joissa valitaan toimilaitekokoa. Ohjelman alussa täytyy ensin avata tietokantayhteys, jonka jälkeen tietokantaan voidaan tehdä hakuja, tai sinne voidaan lisätä tietoa. Lopuksi yhteys täytyy myös sulkea. Periaatteena on, että jokaista yhteyden avaamista kohden tulee olla yhteyden sulkeminenkin.



KUVA 3. Esimerkki SQL – toiminnosta (Mikko Santala 21.8.2015)

2.3 SQL – kieli ja relaatiotietokanta

Relaatiotietokannat perustuvat tutkija E.F. Codd:in vuonna 1970 julkaisemaan relaatiomalliin. Siinä on määritelty relaatiotietokantojen teoreettinen pohja, joka perustuu joukko-oppiin, matematiikkaan ja predikaattilogiikkaan. Relaatiomalli koostuu kolmesta osasta, jotka ovat rakenne, käsittely ja eheyssäännöt. (Hovi, 2004, 5.)

Ennen relaatiomallia tietokannoissa käytettiin hierarkkista mallia tai verkkomallia. Nykyään relaatiomalliin pohjautuvat tietokannat ovat yleisimpiä. Relaatiotietokantojen kielenä käytetään SQL – kieltä, jolla tietokantaan voidaan tehdä kyselyjä. (Hovi, 2004, 5.) Ohessa on käsitelty tärkeimmät testausohjelmassa käytetyt tietokantatoiminnot.

2.3.1 Tietokannan rakenne

Testausohjelma oli sovelluksena sen tyyppinen, että tietokantaa käytti vain yksi käyttäjä tai ohjelma kerrallaan, ja aina samalta tietokoneelta. Siten tietojenhallintaan ei tarvittu mitään serveripohjaista tietokantaa, vaan tähän sovellukseen riitti SQLite – tietokanta, joka koostuu yhdestä tiedostosta.

Relaatiotietokanta koostuu tauluista, joihin kaikki kannassa oleva tieto on talletettu. Taulu koostuu riveistä ja sarakkeista. Jokaiselle sarakkeelle määritetään tietotyyppi ja pituus. Tietotyyppi voi olla esimerkiksi teksti, numeerinen arvo tai aikaleima. Testausjärjestelmän tietokantaan tehtiin omat taulut yleisille asetuksille, toimilaitekohtaisille asetuksille ja raja-arvoille, testin tuloksille, mittausdatalle ja tuotteiden varianteille. Taulujen tarkempi rakenne selviää liitteestä yksi.

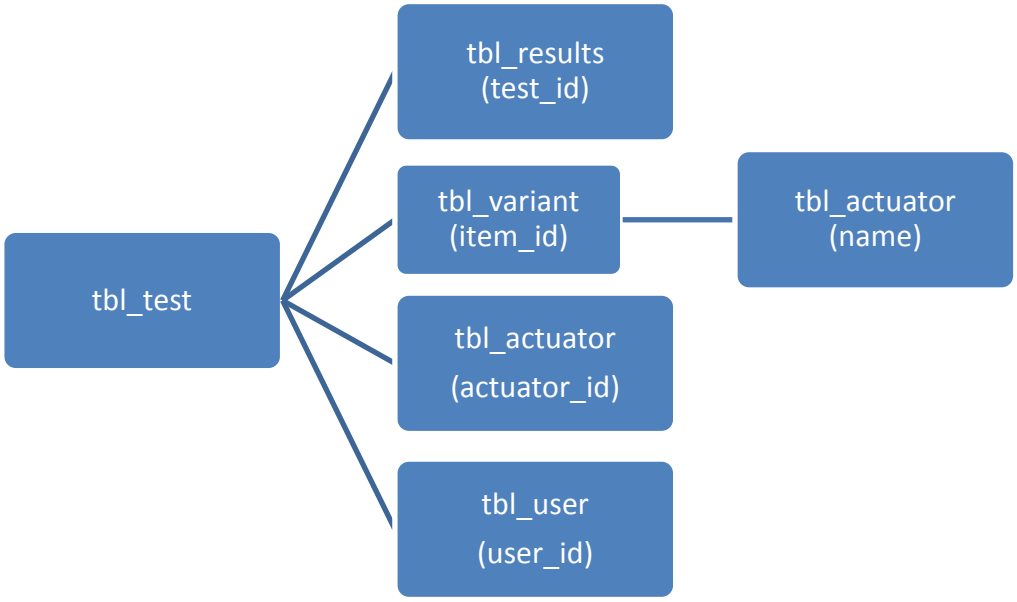
Lisäksi kantaan tehtiin optioksi käyttäjätieto – taulu, jos yritys kokee jossain vaiheessa aiheelliseksi tehdä testaamisesta sallittua vain niille, joilla on tunnukset ohjelmaan. Asetus – tauluun tehtiin sarake, jonka arvoa vaihtamalla käyttäjänhallinta saadaan käyttöön tai pois käytöstä. Salasanan salaus toteutettiin Labview:ssä MD5 Message Digest – koodilla, joka muuntaa syötetyn salasanan täysin tunnistamattomaksi ja epämääräiseksi merkkijonoksi. Näin salasanoja ei saada selville edes muilla

tietokantaohjelmilla käyttäjä – taulua katseltaessa. Käyttäjänhallintaan ei kuitenkaan suuremmin panostettu, sillä testaustoiminta oli vielä niin alkuvaiheessa, että sitä ei toistaiseksi nähty ehdottomana toimintona.

Tietokantaan tuli siis tauluja yhteensä kuusi ja niiden välille luotiin yhteyksiä eli relaatioita. Jokaisella taululla on tunnisteena perusavain, joka on kohdistettu johonkin taulun sarakkeista. Perusavain on yksilöivä eli uniikki, sarakkeessa ei saa siis olla kahta tai useampaa samaa arvoa eri riveillä. Perusavain yksilöi taulun rivit jonkin määrätyn sarakkeen perusteella. Testausohjelman tietokannassa taulujen perusavaimiksi asetettiin usein id – numero, joka tehtiin vielä automaattisesti kasvavaksi sellaisissa tauluissa, joihin ohjelma lisäsi rivejä.

Kun taulujen välille luodaan relaatioita, niihin määritetään viiteavain. Viiteavainta voidaan kutsua myös vierasavaimeksi. Viiteavaimella luodaan yhteys eri tauluissa olevien sarakkeiden välille. Viiteavaimella luotua yhteyttä kutsutaan yksi – moneen – yhteydeksi tai isä – lapsi – yhteydeksi, jossa isällä voi siis olla monta lasta, mutta lapsella ei voi olla kuin yksi isä. Jotta viittaus toimii, on viittaavan ja viittauksen kohteena olevan kentän sisällettävä sama tieto. (Hovi, 2004, 6 – 7.)

Sovellukseen tehdyssä tietokannassa toteutuu yksi – moneen – yhteys esimerkiksi toimilaite – taulun ja testi – taulun välillä. Siinä toimilaitekokoja on vain yksi, mutta yksi toimilaite voi liittyä useampaan testiin. Samalla periaatteella on yhdistetty nimen kautta variantti – ja toimilaite – taulut. Niissä yhteen toimilaitekokoon liittyy monta eri varianttia. Samoin myös mittausdata – taulun ja testi – taulun välille on luotu yhteys testin id – numeron avulla. Myös käyttäjätieto on yhdistetty testiin, jotta testiin voidaan liittää käyttäjä, jos käyttäjänhallinta otetaan käyttöön. Kun se ei ole käytössä, kenttään syötetään oletusarvona tuotannon yleistunnus. Taulujen väliset relaatiot on kuvattu kuviossa (KUVIO 1).



KUVIO 1. Taulujen relaatiot

Relaatiotietokantoihin oleellisesti liittyvien avaimien täytyy täyttää kaksi sääntöä. Eheyssäännön toteutumiseksi perusavaimen arvo ei saa olla tyhjä NULL – arvo. Eli jos testi – tauluun lisätään uusi testi, sille on lisättävä myös id – numero, muuten siihen testiin ei voi osoittaa kuuluvaksi dataa mittausdata – taulussa. Toinen sääntö on viite-eheys, joka tarkoittaa, että testi – taulusta ei voi poistaa riviä, jos siihen on osoitettu kuuluvaksi dataa mittausdata - taulussa. Jos rivi poistettaisiin, syntyisi orporivejä, mikä tarkoittaisi viite-eheyden särkymistä.

2.3.2 Taulujen perustaminen

Pääpiirteissään taulujen perustamisessa määritellään taulun nimi ja sen sisältämät sarakkeet. Sarakkeille määritetään nimi, tietotyyppi, tiedon maksimipituus ja desimaalien määrä. Tarpeen mukaan sarakkeille määritetään rajoitteita. NOT NULL – ehdolla NULL – arvo eli tyhjä arvo ei ole sallittu. Sarakkeelle voidaan asettaa oletusarvo, joka siihen lisätään, jos lisäyksessä siihen ei erikseen tuoda arvoa. Oletusarvoja ovat monesti käyttäjätieto tai aikaleima. UNIQUE – määre tarkoittaa, että sarakkeen arvon täytyy olla yksikäsitteinen eli toista samanlaista ei saa löytyä muilta sarakkeen riveiltä. Lisäksi määritetään halutut perus- ja viiteavaimet. (Hovi, 2004, 102 – 103.)

Viite-eheyden säilyttämiseksi ja takaamiseksi on myös asianmukaista määrittää mitä tapahtuu kun avaimia sisältäviä tauluja päivitetään tai poistetaan. Päivityksessä voidaan isätaulun perusavaimen päivitys asettaa vyörymään lapsitauluun. Vyöryminen määrätään SQL - kielessä ON UPDATE CASCADE – määreellä. Rivien poiston varalta viite-eheyden säilyminen voidaan taata määreellä ON DELETE CASCADE. Tällöin kun isätaulusta poistetaan rivi, poisto vyöryy siihen liitettyyn lapsiriviinkin. (Hovi, 2004, 113 – 114.) Testausohjelman tietokantaa tehtäessä, päivitys – ja poistosääntöihin ei kiinnitetty juurikaan huomiota, sillä ohjelma vain lisää rivejä tauluihin, eikä muokkaa itse kantaa millään tavalla.

Tauluja voidaan perustaa ns. manuaalisesti SQL – kielellä, mutta sitä varten on saatavilla myös ohjelmia, joiden graafiseen käyttöliittymään määritellään taulun asetukset ja ohjelma muodostaa DDL – lauseen sen pohjalta. Sellainen on mm. testausohjelman tietokannan perustamisessa apuna käytetty ilmainen SQLite Studio – ohjelma.

2.3.3 Tietojen käsittely

Relaatiotietokannan tietoja käsitellään joukko-opillisesti. Tietokannassa oleva taulu koostuu joukosta rivejä, jolloin siihen voi kohdistaa joukko-operaatioita. Niiden käyttö tekee tietokannan tietojen käsittelystä tehokasta, sillä operaatiot voidaan kerralla kohdistaa yhteen tai useampaan tauluun ja suureen määrään rivejä. (Hovi, 2004, 8.)

Joukko – operaatiot toteutetaan SQL – kielellä, joka koostuu englanninkielisistä sanoista ja merkeistä. Operaatioita ovat mm. valinta, projektio, yhdiste, leikkaus, erotus ja liitos. Kielen avulla toteutetaan operaatioista koostuvia kyselyitä kantaan ja sillä määritellään tietokannan rakenne ja tarvittaessa muutetaan tai päivitetään sitä. SQL – kielen osaa, jolla määritellään ja muutetaan kannan rakennetta, kutsutaan DDL:ksi. Se tulee sanoista Data Definition Language. DML:ksi kutsutaan puolestaan osaa, joka sisältää tietojen kyselyt, lisäykset, päivitykset ja poistamiset. Se tulee sanoista Data Manipulation Language. (Hovi, 2004, 6;8;14.)

Kun tauluun syötetään uusi tietorivi, voi olla, että johonkin sarakkeista ei ole vielä saatavilla arvoa. Silloin sarakkeeseen tallentuu NULL – arvo, eli tyhjä arvo. Tätä hyödynnetään etenkin silloin, kun sarakkeen tietotyyppi on numero ja tietojen pohjalta tehdään matemaattisia operaatioita. Tällöin tyhjä arvo ei vääristä tulosta samalla tavalla kuin nollan tallettaminen sarakkeeseen jos tietoa ei ole saatavilla. Esimerkiksi voidaan ottaa binäärilogiikan bitti, jossa bitillä on kaksi mahdollista arvoa, 1 ja 0. Tyhjän arvon ansiosta bitillä voisi olla tietokannassa myös tyhjä arvo. Tätä kutsutaan kolmiarvoiseksi predikaattilogiikaksi. (Hovi, 2004, 7.)

Testausohjelmassa NULL – arvoa hyödynnettiin sallimalla ne testi – taululle, jolloin testin alussa pystyttiin synnyttämään uusi rivi tauluun lisäämällä jo saatavilla olevat tuote ja päivämäärätiedot omiin sarakkeisiinsa. Tällöin tuloksia sisältävät sarakkeet jäivät vielä ilman arvoa. Jos NULL – arvo olisi ollut kielletty, olisi puuttuvien arvojen paikalle jouduttu syöttämään nolla – arvot. Testausohjelmaan tehtiin osa, jossa voidaan tarkastella kaikkien tehtyjen testien statistiikkaa, kuten maksimiliikematkojen ja paineen alenemien vaihteluväliä ja keskiarvoa, testien läpäisyprosentteja ja useammin kuin kerran testattujen tuotteiden määrää. Näiden matemaattisten operaatioiden luotettavien tulosten saavuttamiseksi oli hyvä, että ainakin testi – taulussa oli sallittu NULL – arvo.

2.3.4 Kyselyt

Tietokantaan voidaan tehdä SQL – kielellä kyselyjä, jotka tuottavat riveistä koostuvan tulosjoukon. Ohessa on esitelty testausohjelmassa eniten käytetyt kyselyt muutamin esimerkein. SELECT – käsky on yksi keskeisimmistä ohjelmassa käytetyistä SQL – käskyistä. Kyselylauseessa määritellään mitä sarakkeita haetaan, monestako taulusta, mitkä rivit, miten tulokset ryhmitellään tai järjestetään. Testausohjelmassa käytettiin SELECT – käskyä esimerkiksi asetusten ja raja-arvojen hakemisessa.

Toimilaitekohtaisia raja-arvoja haettiin lauseilla, kuten

```
SELECT *
FROM tbl_actuator
WHERE id=1
ORDER BY id, id DESC
```

Esitetty kysely palautti kaikki sarakkeet, jotka liittyivät id – numerolla yksi olevaan toimilaitteeseen. Kuvaajien muodostamiseksi mittausdata – taulusta haettiin tiettyyn testiin liittyvä data seuraavasti

```
SELECT pressure_A ,pressure_B, travel, time
FROM tbl_results
WHERE test_id=1
ORDER BY time, time DESC
```

Jos olisi käytetty sarakkeiden nimen sijasta tähti – merkkiä, olisi tuloksiin tullut mukaan testin id – numero, jota ei kuvaajien muodostamiseen tarvittu.

Kyselyn tuloksia voidaan pyytää tietyssä järjestyksessä tai lajitella. Tämä onnistuu ORDER BY – lauseella, johon liitetään järjestyksen osoittava ASC tai DESC – määrite. ASC lajittelee tulokset nousevaan ja DESC laskevaan järjestykseen. Yllä esitetyssä kyselyssä mittaustulokset pyydettiin ajan mukaan laskevassa järjestyksessä, jolloin data on oikeassa järjestyksessä kuvaajan muodostamista varten. Jos järjestystä ei määriteltäisi, rivit saattaisivat palautua mielivaltaisessa järjestyksessä. Lajittelua voidaan tehdä myös useamman sarakkeen mukaan kohdistamalla ORDER BY- ja DESC- tai ASC – lauseet eri sarakkeisiin.

Muita käytettyjä kyselylauseita ja operaattoreita ovat mm. BETWEEN, IN ja matemaattiset funktiot AVG, SUM, MIN, MAX ja COUNT. Näitä hyödynnettiin etenkin vanhojen testien hakemisessa ja testitulosten muodostamisessa.

2.4 Käytetyt mittaustekniikat

Testausjärjestelmässä käytettiin prosessisuureiden mittaamiseen antureita. Anturi muuttaa mitattavan prosessisuureen siihen verrannolliseksi sähköiseksi viestiksi. Anturi koostuu tuntoelimestä ja anturiosasta. Tuntoelin eli ottomuuttaja havaitsee mitattavan suureen ja anturiosa muuttaa mittaustuloksen sähkösuureeksi. Näiden lisäksi anturi voi sisältää vielä lähetinosan, joka muuntaa anturin antaman viestin standardiviestiksi, joka on 4 – 20 mA tasavirtasignaali. (Fonselius, 1988, 6.)

2.4.1 Paineen mittaaminen

Paine on suure, joka kuvaa nesteen tai kaasun aiheuttamaa voimavaikutusta sitä rajoittavaa seinämää kohtaan. Sitä käsitellään siis voimana pinta-alayksikköä kohti. Paineen tuntoelimet perustuvat kimmoiseen muodonmuutokseen. Sähköiset paineanturit sisältävät yleensä venymäliuskan, joka mittaa tuntoelimen jännitystä. Venymäliuskan vastus muuttuu suhteessa muodonmuutokseen. Sähköiset paineanturit sisältävät integroidun vahvistimen, joka sillan syöttövirran ja mitatun lähtösignaalin. (Aumala, Mittaustekniikan perusteet, 1989, 92 – 93.)

2.4.2 Aseman mittaaminen

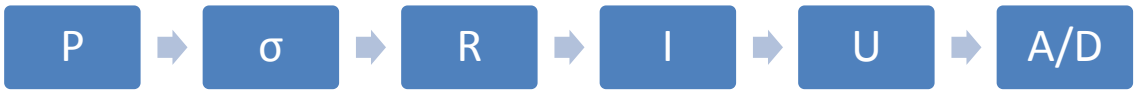
Potentiometriä voidaan käyttää analogisena siirtymäanturina. Se koostuu vastuselementistä ja mittakärkeen kiinnitetystä liukukoskettimesta. Vastuselementtejä valmistetaan useista materiaaleista kuten kuparilangasta käämimällä, hybriditekniikalla tai johtavasta muovista. Potentiometrin ulostulona on jännite, joka vaihtelee lineaarisesti nollan ja siihen tuodun jännitteen välillä riippuen liukukoskettimen sijainnista. Potentiometri sopii hyvin hitaisiin liikenopeuksiin ja on luotettava. Sen heikkous on kuluttava, mekaaninen kosketus vastuksen ja liukukoskettimen välillä, sillä se lyhentää anturin käyttöikää. Potentiometrit ovat kuitenkin verrattain halpoja. (Fonselius, Anturit, 1988, 92.)

2.5 Signaalinkäsittely

2.5.1 Mittausketju

Mittausketjulla havainnollistetaan mitattavan suureen muunnokset anturin tuntoelimestä mittalaitteelle saakka. Se esitetään yleensä lohkokaavion avulla. Mittausketjuun kuuluvat kaikki mittaustoiminnassa mukana olevat muuntavat elimet. (Aumala, Mittaussignaalien käsittely, 1998, 243.)

Paineen mittaamisen mittausketju on esitetty alla kuviossa (KUVIO 2). Siinä anturin tuntoelimeen vaikuttava paine P aiheuttaa tuntoelimeen jännityksen σ, joka puolestaan muuttaa virtapiirin vastusarvoa R. Anturin lähetinosa muuttaa signaalin vastusarvoa vastaavaksi virraksi I. Mittalaitteella mitataan jännitettä piiriin asetetun vastuksen yli, jolloin virtaviesti muuttuu jännitteeksi U.



KUVIO 2. Paineenmittausketju

Vastaavasti aseman mittaamisen mittausketju on esitetty alla (KUVIO 3). Siinä liikematka s vaikuttaa potentiometrin vastukseen R, ja edelleen anturilta mittalaitteelle kulkevaan sähkövirtaan I. Ennen mittalaitetta virtaviesti muutetaan jälleen jännitteeksi U vastuksen yli mittaamalla. Mittalaitteessa tapahtuu vielä muunnos analogisesta muodosta digitaaliseen muotoon eli A/D – muunnos.

KUVIO 3. Asemanmittausketju

2.5.2 Signaalinmuunnokset

Mitattu sähköinen suure saadaan muutettua tietokoneella halutuksi SI – järjestelmän lisäyksiköksi, kuten baareiksi tai asteiksi, kun sille määritetään kerroin. Paine- ja asema-anturit ovat luonteeltaan lineaarisia, joten kertoimen määrittäminen on melko yksinkertaista. Se saadaan anturin mittausalueesta ja sen sähköisen mittaussuureen ääriarvoista yhtälön 1 mukaisesti.

$$k = \frac{\Delta y}{\Delta x} \tag{1}$$

jossa Δy = anturin mittausalue ja Δx = viestin ääriarvojen erotus

Esimerkiksi jos paineanturin mittausalue on 100 bar ja se käyttää standardia 4 – 20 mA virtaviestiä, ja se mitataan 250Ω vastuksen yli jännitteenä mittalaitteelle, signaalinmuunnos menisi kaavan 2 mukaisesti.

$$\mathrm{k} = \frac{\Delta y}{R * \Delta I} = \frac{100\ bar}{250\Omega * (20-4) * 10^{-3} A} = \frac{100\ bar}{(5-1) * V} = \frac{100\ bar}{4V} = 25 * \frac{bar}{V} \tag{2}$$

jossa R = vastus ja I = virta

3 TESTAUSJÄRJESTELMÄ

3.1 Suunnitteluperusteet

Järjestelmän suunnittelussa tärkeä lähtökohta oli saatavilla olevat laitteistot ja ohjelmistot. Liikkeelle haluttiin päästä nopeasti ja mahdollisimman vähin hankinnoin, joten käyttöön otettiin National Instrumentsin laitteita ja niiden kanssa yhteensopiva Labview – ohjelmisto, koska ne olivat heti saatavilla koululta. Lopuksi yritykselle ostettiin vastaavat omat laitteet.

Yritykseltä saadun mittausprosessin hahmotelman mukaan testaus alkaisi sillä, että käyttäjä valitsee toimilaitetyypin, jonka perusteella ohjelma hakee testille toimilaitekohtaiset toiminta- ja raja-arvot tekstitiedostosta. Testattavaa tuotetta oli kuusi eri kokoa, joille tuli toimilaitekohtaiset raja-arvot. Lisäksi käyttäjän tulisi voida syöttää tarvittavat perustiedot testistä tekstikenttiin. Varsinainen testisykli alkaisi tämän jälkeen.

Testisyklissä toimilaitetta ajettaisiin ensin toisesta portista tiettyyn asemaan ja paineeseen, ja kun ne on saavutettu, pidetään siinä asemassa tietty ajanjakso ja tarkkaillaan paineen alenemaa. Ajanjakson kuluttua ajetaan toimilaitetta toisesta portista vastakkaiseen suuntaan, jälleen tiettyyn asemaan ja paineeseen, ja pidetään samanpituinen ajanjakso. Testisykli päättyisi viimeisimpään asemaan.

Testin päätyttyä mittaustulokset luettaisiin tekstitiedostosta mittauspöytäkirjapohjaan. Mittauspöytäkirja sisältäisi numeeriset tulokset maksimipaineista molempiin suuntiin ajettaessa, paineen alenemat pidon aikana, maksimi liikematkan ja kuvaajan, jossa esitetään paineet ja liikematka ajan funktiona.

Tämän kuvauksen pohjalta määritettiin asiat, jotka ohjelman ja laitteiston tuli vähintään pystyä suorittamaan:

- paineen mittaaminen kahdesta eri kohteesta tietyllä painealueella
- aseman mittaaminen yhdestä kohteesta tietyllä liikematkalla
- mittausdatan tallentaminen jatkokäyttöä varten
- testiraportin muodostaminen mittausdatan pohjalta
- toimilaitteen ajaminen, jotta järjestelmästä saadaan automaattinen
- erikokoisten tuotteiden testaukseen mukautuminen

Näiden vaatimusten pohjalta alettiin valita komponentteja, jotka toimivat yhteen National Instrumentsin laitteiden kanssa ja hahmottelemaan ohjelman rakennetta.

3.2 Fyysinen laitteisto

3.2.1 Alusta

Laitteiston tyypiksi valikoitui koululta löytyvien laitteiden perusteella National Instrumentsin C – sarjan laitteet. Ne ovat verrattain yksinkertaisia ja edullisia, USB – yhteyden kautta tietokoneeseen liitettäviä mittalaitteita. Laitteita on erikokoisia riippuen tarvittavasta sisään – ja ulostulojen määrästä. Tyypillisesti laite koostuu alustasta, siihen liitettävästä mittakortista ja mittakorttiin liitettävästä anturista tai toimilaitteesta (KUVA 4).

KUVA 4. Mittalaitteiston alusta kortteineen (www.ni.com 21.8.2015)

Mittakortti valitaan halutun sisään - tai ulostulon tyypin mukaisesti. Mittakortteja on mm. analogisia ja digitaalisia sisään – ja ulostuloja, CAN – ja LIN – yhteyskortteja, ajastukseen ja GPS – paikannukseen sovellettavia kortteja. National Instruments ilmoittaa verkkosivuillansa, että C – sarjan laitteisiin saa jopa 50 erilaista korttia (www.ni.com 8.8.2015). Mittakorteissa on sen tyypistä riippuen vaihteleva määrä kanavia, joihin laitteita voidaan liittää.

Tässä sovelluksessa tarvittiin kahta erilaista ja yhteensä kolme mittakorttia, joten alustaksi valittiin National Instruments cDAQ-9174, johon saa liitettyä neljä mittakorttia. Alustaan jäi siis vielä yksi korttipaikka varalle, jos järjestelmää halutaan tulevaisuudessa laajentaa. Se tultaisiin todennäköisesti täyttämään venymäliuskaa mittaavalla NI9219 – kortilla toimilaitteen momentin tuoton testaamiseksi.

Alustan lisäksi laitteistoon tuli liitettäväksi sisään – ja ulostuloja, jotka vaativat toimiakseen 24V jännitteen, joten laitteistoon tarvittiin jännitelähde. Alusta vaatii toimiakseen 5 – 30VDC ja anturit 24VDC herätejännitteen, joten jännitelähteeksi valittiin National Instrumentsin PS-15. Siitä saatiin riittävät 120 wattia tehoa 24 VDC jännitteellä syöttämällä siihen 230VAC verkkovirtaa laitekaapelilla. Jännitelähteelle luvattiin ohjekirjassa 20 % jatkuvaa ekstratehontuottoa, mikä tulikin tarpeeseen, kun laitteisto laajeni muutamalla ohjattavalla venttiilillä testausprosessia sujuvammaksi kehitettäessä.

3.2.2 Sisääntulot

Mitattavia suureita testauksessa olivat paine ja asema. Koska kyseessä oli sähköinen mittausjärjestelmä, mittaaviksi tuntoelimiksi valikoituivat sähköiset anturit. Antureiden viestityypiksi valittiin virtaviesti, jotta muissa kaapeleissa ja laitteissa kulkevat jännitteet eivät häiritsisi viestiä, kuten jänniteviestin kanssa saattaisi tapahtua.

Sisääntulojen mittakortiksi valittiin NI-9205, joka soveltuu analogiseen jännitteen mittaamiseen, joten virtaviesti piti vahvistaa ja muuttaa ennen mittakorttia jännitteeksi. Se tapahtui asentamalla tarkkuusvastus johtimeen, jossa virtaviesti kulkee, ja mittaamalla mittakortilla jännitettä vastuksen yli. Vastuksen ohmimäärä mitoitettiin antureiden virtaviestin suuruuden perusteella kaavan

$$R = \frac{U}{I} \tag{1}$$

jossa R = vastus
U = jännite
I = virta

mukaisesti. Antureiden virtaviesti on määritelty standardiksi ja sen suuruus on 4 – 20 milliampeeria. Mittakortti puolestaan pystyy mittaamaan jännitteitä alueelta 0-10 volttia, joten vastuksen suuruus saatiin laskettua seuraavasti

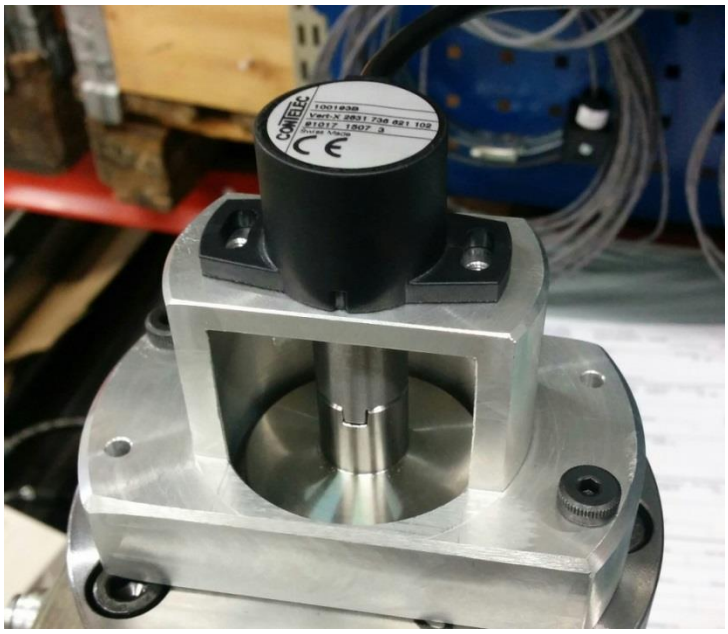
$$R = \frac{Vmax}{Imax} = \frac{5V}{0{,}020A} = 250\Omega \tag{2}$$

jossa *Vmax* = haluttu mitattava maksimijännite
Imax = virtaviestin suurin arvo

Vastuksen ohmimäärän ollessa 250Ω, pienin mitattava jännitearvo on 1 voltti ja suurin 5 volttia, jotka mahtuvat hyvin mittakortin mittausalueelle.

Paineanturit valittiin mittausalueeltaan vain hieman testipainetta suuremmiksi, jotta painetta saadaan mitattua tarkemmin. Mitoituksessa otettiin myös huomioon painelähteen maksimipaine, jotta jos testipaine muuttuu, saadaan kuitenkin mitattua mikä tahansa tuotettavissa oleva paine.

Asema – anturin tuli pystyä mittaamaan pyörimisliikettä suuruudeltaan alle 360 astetta. Siten anturiksi valittiin lineaarinen potentiometri – tyyppinen kulma-asema – anturi, jonka mittausalue on 0 – 360 astetta. Valintaan vaikutti myös liitettävyys testattavaan toimilaitteeseen. Anturille tehtiin siisti, hyvin asemoituva teline, joka sopii kaikkien tuotekokojen standardiin anturikiinnitykseen (KUVA 5).



KUVA 5. Asema-anturi telineessään

3.2.3 Ulostulot

Testausohjelman tuli pystyä ohjaamaan testattavaa toimilaitetta, joten tarvittiin myös analoginen mittakortti, josta saadaan jännitettä ulos. Analoginen ulostulokortti on malliltaan NI-9263. Niitä tarvittiin laitteistoon kaksi kappaletta, sillä ulostulokanavia tarvittiin yhteensä kahdeksan ja yhdessä kortissa on neljä kanavaa.

Toimilaitteen liikettä ohjataan painelähteessä olevalla sähköisellä suuntaventtiilillä. Suuntaventtiili on tyypiltään 4/3 – suuntaventtiili, jossa on molemmin puolin ohjauskela, eli sen ohjaamiseen tarvittiin kaksi ulostulokanavaa. 4/3 tarkoittaa sitä, että venttiilissä on neljä liityntää ja kolme asentoa.

Venttiilin kelojen ohjausjännite on 24VDC ja mittakortin maksimi ulostulojännite on 10VDC, joten ohjaus toteutettiin releen kautta. Releiksi valittiin 5/24VDC puolijohdereleet. Mittakortista syötetään releelle 5V, joka kytkee jännitelähteen 24V ohjattavalle venttiilille.

Samalla tavalla, relekytkennällä, toteutettiin molempiin painelinjoihin, suuntaventtiilin jälkeen sijoitettujen vastaventtiileiden ohjaus. Vastaventtiilit olivat tyypiltään 2/2, jousipalautteisia venttiileitä, joten ne sisälsivät vain yhden ohjattavan ohjauskelan venttiiliä kohden. 2/2 - vastaventtiilissä on kaksi liityntää ja kaksi asentoa. Vastaventtiileitä tarvittiin paineen pitämiseen toimilaitteella pidon aikana, sillä suuntaventtiilissä oli avoin keskiasento. Avoin keskiasento mahdollistaa sen, että kun testaus lopetetaan niin, liitäntäletkut ovat varmasti paineettomat ja toimilaite voidaan turvallisesti irrottaa. Painelinjoissa oli myös paineakut, joilla taataan, että paineistettavassa linjassa pysyy paine. Niiden eteen sijoitettiin toiset vastaventtiilit, jotta testiprosessia voidaan muokata tarvittaessa sellaiseksi, että paineakut saadaan halutessa pois käytöstä.

Yksi ulostulokanava tarvittiin paineenrajoitusventtiilille, jolla säädellään painelähteeltä saatavaa maksimipainetta. Paineenrajoitusventtiilille syötettiin suoraan jännitelähteeltä 24V herätejännite ja ulostulokortilta 0-10V signaali. Venttiilin mukana toimitettiin käyrä, josta voitiin lukea tietyn paineen saamiseksi tarvittava signaalin taso.

Testausprosessia kehitettäessä päätettiin, että toimilaitteet tulevat kokoonpanosta tyhjänä ja ne täytetään väliaineella testilaitteiston painelähteellä. Painelähteen tankkiin lisättiin turvatoimena pintavahti, jolle syötetään yhdestä ulostulokanavasta jännite, jota sitten mitattiin yhdellä sisääntulokanavalla. Pintavahti oli päälle kytkevä malli eli kun nestetaso laskee liian alas, se sulkee virtapiirin ja jännite nousee mittaavassa kanavassa. Ohjelma asetettiin vain ilmoittamaan nestepinnan alenemisesta, ettei painelähde pääse kavitointirajalle asti. Lisäämällä ulostulojen määrää oltaisi voitu toteuttaa painelähteen virran katkaisu ohjelmallisesti.

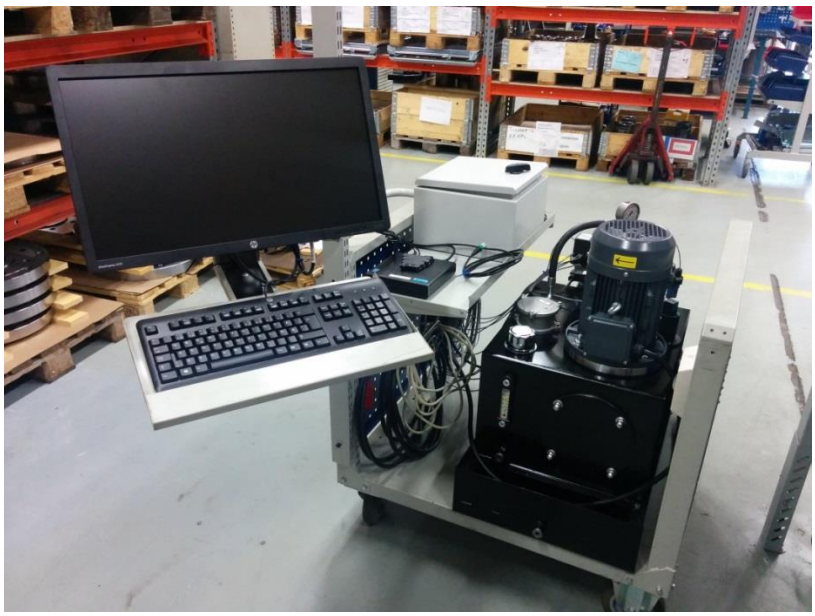
3.2.4 Laitteiston muu rakenne

Mittauslaitteisto rakennettiin metallisen sähkökaapin sisään sen suojaamiseksi lialta ja toimintavarmuuden takaamiseksi konepajaympäristössä, jossa laitteisto voisi muuten joutua kosketuksiin leikkuunesteen, öljyn ja muun lian kanssa. Sähkökaapissa oli irrotettava välipohjalevy sisällä, jonka päälle mittauslaitteen alusta, jännitelähde, releet ja riviliittimet saatiin siististi kiinnitettyä DIN – kiskoon.

Kaapelit laitteistoon tehtiin itse. Niihin valittiin liittimiksi suojatut, kierreholkki - kiinnitteiset M12 – liittimet, jotka ovat IP40 – suojattuja. Sisääntulojen liittimet tehtiin 5 – napaisilla liittimillä ja ulostulot 3 – napaisilla vaarallisten väärinkytkemisten välttämiseksi. Lisäksi laitteen etupaneeli merkittiin tekstitarroin. Alla olevassa kuvassa näkyy etupaneelin merkinnät ja kaapeliliittimien tyyppi (KUVA 6). Kaikkien kaapeleiden johtimet ovat suojattuja ja sisältävät suojajohtimen, jolla saadaan vähennettyä häiriötä anturin signaalissa.

KUVA 6. Mittalaitteiston liitännät (Mikko Santala 20.8.2015)

Toimilaitteelle ei ollut vielä tehty tuotannon kokoonpano-osastolle varsinaista omaa työpistettä, joten testilaitteisto päätettiin kasata liikuteltavaan kärryyn (KUVA 7). Siihen saatiin siistiin pakettiin painelähde, mittalaite ja tietokone, jotta testaus voidaan sijoittaa mihin tahansa, kun kokoonpanon työpiste on tehty.



KUVA 7. Testilaitteisto kärryssä (Mikko Santala 20.8.2015)

3.3 Testausohjelma

3.3.1 Ohjelman suunnittelu

Tärkeimpinä ohjelman suunnitteluperusteina pidettiin muunneltavuutta ja helppokäyttöisyyttä. Ohjelman toiminta-arvoja piti pystyä muuttamaan jälkikäteen, sillä lopulliset arvot muodostuisivat vasta ajan kuluessa, kun nähdään mitkä parhaiten toimivat millekin tuotekoolle. Toiminta-arvoja ovat mm. näytteenottotaajuus, turva-ajastimet, laskurit, aloitusasema, testipaine ja pitoaika.

Ohjelman suunnittelu aloitettiin määrittelemällä ja listaamalla mitä toimintoja ohjelman pitää pystyä suorittamaan. Osa toiminnoista tuli suoraan yrityksen alkuperäisestä testausprosessin hahmotelmasta. Ohjelman pääelementit olivat itse painetesti, vanhojen

testien hakeminen ja tarkastelu, asetusten muokkaaminen ja toimilaitteen manuaalinen ajaminen. Määritellyt toiminnot jaettiin omiin moduuleihinsa, joista tehtiin omat VI:t. VI:n sisällä sille määrätty toiminto pilkottiin edelleen riittävän pieniin osiin, jotta niistä saatiin tilakoneen tiloja. Ohjelman suunnittelussa hyvä työtapa oli tehdä siitä vuokaavio. Kun vuokaavion teki huolella, sen pohjalta oli vaivatonta tehdä itse koodaaminen. Liitteessä 2 on suunnittelun alussa tehdyt vuokaaviot testausprosessista sivulla 1, testistä sivulla 2 ja vanhojen testien hakemisesta sivulla 3.

Ohjelma yritettiin tehdä mahdollisimman helppokäyttöiseksi ja selkeäksi niin, että uudet ohjelmat avautuvat perusnäkymän päälle ja suurin osa käyttäjän toimista olisi vain vastauksia kysymyksiin, joihin voi vastata kyllä tai ei. Ohjelmalle suunniteltiin selkeä perusnäkymä, joka on päätason VI:n käyttöliittymä. Se on esitetty liitteen 3 ensimmäisellä sivulla. Se sisältää painikkeet, joilla päästään siirtymään muihin ohjelman osiin kuten testaamaan, muuttamaan asetuksia ja tarkastelemaan vanhoja testituloksia. Perusnäkymää hallitsevat vanhojen testien hakuehdot ja hakutuloslista. Ohjelman avautuessa tietokannasta haetaan listaan uusimmat testit valmiiksi.

3.3.2 Tietokanta

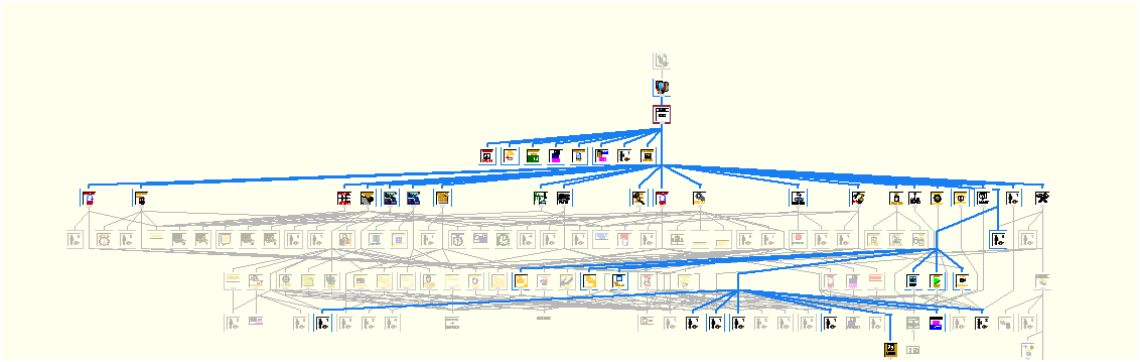
Muutama asia ohjelmassa päätettiin tehdä toisin, kuten mittausdatan ja toiminta-arvojen tallentaminen tekstitiedoston sijasta tietokantaan. Alkuperäisessä suunnitelmassa oli ajateltu, että mittausdata ja toimilaitteiden toiminta- ja raja-arvot tallennetaan tekstitiedostoon ja luetaan sieltä ohjelmaan ja mittauspöytäkirjaan. Siinä perusideana oli varmasti, että arvot olisivat kohtuudella muokattavissa. Sen idean pohjalta näihin toimintoihin paremmaksi ratkaisuksi todettiin SQLite – tietokannan lisääminen ohjelmaan.

SQL – tietokannan lisäämisellä saatiin kaikki ohjelman käyttämä ja tuottama data yhteen tiedostoon talteen omiin tauluihinsa ja niitä voidaan sieltä katsella, muokata tai varmuuskopioida. Lisäksi tietokantaan pääsee käsiksi muilla tietokantaohjelmilla tarpeen vaatiessa. Toiminnan takaamiseksi tehtiin VI, joka tarkistaa onko tietokanta olemassa ja sitä kautta toiminta-arvot saatavilla. Jos haetulla nimellä ei löydy

tietokantaa, VI muodostaa uuden tietokannan, joka sisältää kaikki taulut ja oletusarvot raja- ja toiminta-arvoille. Jos tietokanta löytyy, VI vain palauttaa sen polun ja päätason VI:ssä avataan tietokantayhteys.

3.3.3 Ohjelman hierarkia

Ohjelma koostuu useista, eri tason virtuaalisista instrumenteista. Koko ohjelman rakenteessa tavoiteltiin pyramidin muotoa, jossa päätason VI on ylimpänä, ja sen alla portaittain kasvava määrä ohjelmia ja niiden aliohjelmia. Tällöin VI:t, etenkin päätasolla, pysyvät siisteinä ja yksinkertaisina. Tavoiteltu rakenne saavutettiinkin melko hyvin, kuten alla olevasta kuvasta nähdään (KUVA 8). Pyramidin kapenevasta alaosasta nähdään se, että aliohjelmia käytetään monissa eri ylemmän tason ohjelmassa, eikä jokaiselle ole tehty omaa aliohjelmaa turhaan.



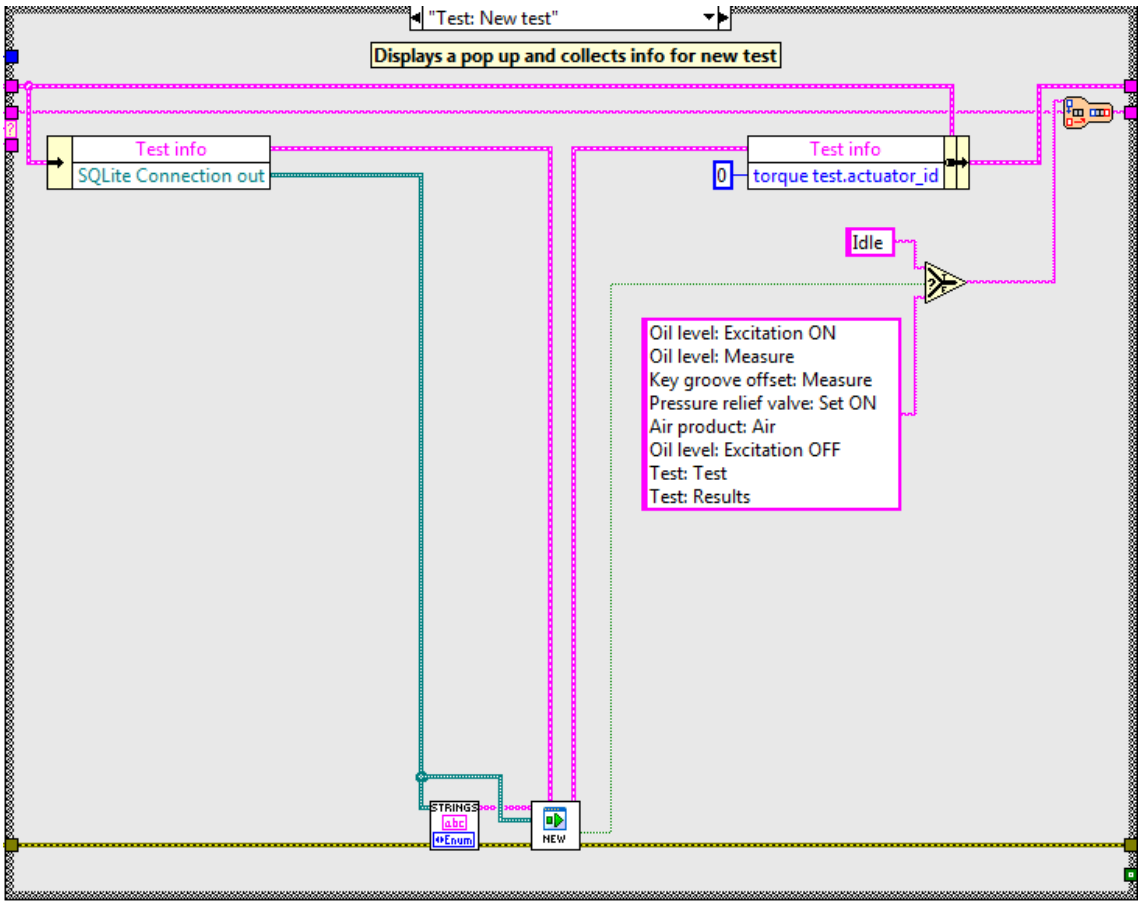
KUVA 8. Ohjelman hierarkia

Perustason ohjelmien alapuolella on vielä joukko aliohjelmia, jotka sisältävät pieniä, yksinkertaisia koodinpätkiä, jotka eivät vaadi käyttäjältä toimia. Tällaisia ovat esimerkiksi datatyyppien muunnokset ja mittaussignaalien muunnokset suureesta toiseen. Niitä tehdään sellaisista toiminnoista, joita käytetään usein eri paikoissa.

3.3.4 Arkkitehtuurit

Suurin osa ohjelman VI:stä käyttää tavallisen tilakoneen arkkitehtuuria ja ne jotka vaativat käyttäjältä toimia, ovat lisäksi tapahtumapohjaisia. Niissä on jonkinlainen käyttöliittymä, joka sisältää tekstikenttiä, säätimiä ja nappeja, joiden arvojen muuttumista tarkkaillaan tapahtuma – rakenteella. Testi – VI on toteutettu tuottaja – käyttäjä – arkkitehtuurilla. Se sisältää tuottajan lisäksi tallennus – loopin, jossa jonosta otetaan mittausdataa ja tallennetaan tietokannan mittausdata – tauluun uudeksi riviksi. Rivi sisältää testin id – numeron, paineen A, paineen B, aseman ja aikaleiman. Toisessa käyttäjä – loopissa mitattua dataa syötetään etupaneelin kuvaajaan ja numeerisiin näyttöihin.

Päätason VI on toteutettu JKI – tilakoneen arkkitehtuurilla. Sen avulla saatiin aikaiseksi sujuva ja toimiva käyttöliittymä ohjelmalle. Lisäksi sillä on helppo määrittää mitkä kaikki toiminnot sisältyvät testiin. Alla olevassa kuvassa (KUVA 9) on komentolistalla määritelty mitä tapahtuu sen jälkeen kun testin aloitustiedot on kerätty käyttäjältä. Ensimmäisenä käydään syöttämässä pintavahdille herätejännite ja mittaamassa pintavahdin signaali. Toisena käydään mittaamassa toimilaitteen asema, mistä määritellään eräiden kohdistusta vaativien osien vaihe-ero. Tämän jälkeen syötetään paineenrajoitusventtiilille signaali, sitten ajetaan toimilaitteen ilmausajo, jonka jälkeen päästään itse testiin. Testin päätyttyä siirrytään testitulosten tarkasteluun. Tulosten tarkastelun sulkeuduttua toimilaite ajetaan vielä toimitusasentoon ja käydään sulkemassa paineenrajoitusventtiili. Ohjelman kaikissa vaiheissa on myös varattu mahdollisuus keskeyttää toiminto, jolloin siirrytään perusnäkymään eli Idle – tilaan. JKI – tilakoneen edut tulivat hyvin esiin, kun testausprosessia kehitettäessä jouduttiin lisäämään erilaisia toimintoja ohjelmaan. Silloin ei tarvinnut kuin tehdä toiminnolle tila ja lisätä se tuollaiseen komentolistaan.



KUVA 9. Tilojen järjestyksen määrittely

3.3.5 Testausprosessi ja ohjelman toimintaperiaate

Toimilaitteen tuotantotestaus oli vielä aivan alkutekijöissään, joten testausprosessia mietittiin ja hiottiin jatkuvasti muodosta toiseen, kun nähtiin mikä toimii ja mikä ei. Testausprosessi käsitti kaiken tuotteen käsittelystä testauspaikalla mittausperiaatteeseen, joten kehityskohteita ja muuttujia riitti runsaasti. Testausprosessin tarkka kuvaus ohjelman osalta on esitetty liitteen 4 sivulla 12, joka on osa testausjärjestelmästä laadittua työohjetta. Ennen testiohjelman käynnistämistä toimilaite on tuotu testauksen työpisteeseen tyhjänä, ja tarvittavat letkut ja asema-anturi on liitetty toimilaitteeseen. Tarkoituksena oli minimoida tuotteen toimituskuntoon saattamiseen vaadittava aika ja työvaiheet. Nyt laite täytetään väliaineella testiohjelman ilmausajossa, jolloin se ei sido työvoimaa, eikä vaadi ylimääräisiä toimilaitteen painelähteeseen liittämisiä, jotka kuitenkin aina vievät jonkin verran aikaa. Kun ohjelma on testin lopuksi ajanut toimilaitteen toimitusasentoon, letkut ja asema – anturi telineineen irrotetaan siitä, ja tuote on toimitusvalmis.

3.3.6 Signaalinkäsittely

Antureiden analogisessa signaalissa huomattiin koekäytöissä jännitepiikkejä, etenkin asema – anturin, joten niiden suodattamiseksi tehtiin oma aliohjelma ja vaihdettiin hieman näytteenoton periaatetta. Aluksi näytteitä otettiin yksi kerrallaan ja syötettiin jonoon, eikä signaalia varsinaisesti suodatettu mitenkään. Jännitepiikkien suodattamiseksi koodattiin keskiarvosuodin, joka ottaa joukon mittausarvoja ja laskee niiden keskiarvon, jolloin lopputuloksena on jännitepiikeistä suodatettu arvo. Joukon suuruus ja näytteenottotaajuus mitoitettiin sellaiseksi, että lopputuloksena oli sama näytteenottotaajuus kuin ennen suodattimen lisäämistä. Tämä paransi toistettavuutta ja tulosten tasaisuutta huomattavasti, sillä kun jännitettä muutetaan useiden kymmenien suuruisella kertoimella toiseksi suureeksi, aiheuttaa pienikin jännitepiikki jo suuria eroja signaalin maksimi- ja minimiarvoissa, joita tietokantakomennoilla haetaan kun määritetään tuloksia.

Suuremuunnoksiin käytetyt kertoimet määritettiin kaavan 2 mukaisesti. Paineantureiden mittausalue oli 250 bar:ia, joten niiden kerroin saatiin seuraavasti

$$\mathrm{k} = \frac{\Delta y}{\Delta U} = \frac{250\ bar}{(5-1)*V} = \frac{250\ bar}{4V} = 62{,}5 \tag{3}$$

Asema-anturin mittausalue oli 360 astetta ja sama standardiviesti, joten sen muunnoskerroin aseteiksi saatiin seuraavasti

$$\mathrm{k} = \frac{\Delta y}{\Delta U} = \frac{360\ deg}{(5-1)*V} = \frac{360\ deg}{4V} = 90 \tag{3}$$

Antureiden näyttämiä tarkistettiin kokeellisesti. Paineantureista varmistettiin, että ne näyttävät nollaa bar:ia, kun ne ovat pöydällä ilmanpaineessa. Lisäksi yrityksellä oli kalibroituja mekaanisia painemittareita, joiden näyttämiin antureiden näyttämiä verrattiin. Asema-anturin näyttämä tarkistettiin kääntämällä sen akselia 180 asteen urassa molemmin päin, ja se näytti 180 astetta niin kuin kuuluukin. Lisäksi tehtiin sarja toistettavuuskokeita, ajamalla samalla toimilaitteella useita testejä peräkkäin ja

tarkistamalla, että etenkin liikematka on joka kerralla sama. Yksi sarja tehtiin niin, että asema-anturi irrotettiin testien välissä toimilaitteesta ja pistettiin takaisin. Keskiarvo – suodattimen lisäämisen jälkeen toistettavuuskokeen tulokset olivat hyviä.

3.3.7 Testin lopputulos

Testauksen kohteena oleva toimilaite on valmistettu sen tyyppisiä toimilaitteita koskevan standardin SFS EN 15714-4 mukaisesti, joten testaamisen, kuten myös tuotannon, tuli olla standardin mukaista. Standardissa ilmoitettiin asiat, jotka valmistajan on pystyttävä ilmoittamaan asiakkaalle. Standardissa oli eritelty tyyppitestauksessa ja tuotantotestauksessa vaadittavat asiat. Tässä tapauksessa kyse oli tuotantotestauksesta.

Standardi vaati tuotantotestaukselta, että valmistajan on pystyttävä ilmoittamaan tuotteesta seuraavia asioita:

- liikeaika molempiin suuntiin ilman kuormaa käyttöpaineella
- minimi liikkeellelähtöpaine
- painetestin pitoaika oltava vähintään 3 minuuttia, painearvot otettava talteen vähintään pidon alussa ja lopussa
- testiraportissa oltava näkyvissä laitteen tyyppi, testilaitteisto ja mittalaitteet
- tuotantotesteissä testattava vähintään vuoto ja liikematka, momentti ja kestoikä testataan vaadittaessa
- vuototestin testijakso ei saa alkaa ennen kuin laite on eristetty painelähteestä ja paine on tasaantunut
- laite ei saa vuotaa ulospäin öljyä

Näiden vaatimusten ja yrityksen hahmotelman pohjalta muodostettiin testiraporttipohja, joka on esitetty liitteessä 3. Se sisältää toimilaitteen tunnistetiedot, standardin vaatimat numeeriset tulokset taulukkona ja kuvaajat mitatusta datasta. Excel – pohjaan tehtiin Boolen logiikkaa hyväksi käyttäen kaavat, joilla ilmoitetaan selkeästi ”PASSED” ja ”FAILED” teksteillä testin läpäistyt ja ei läpäistyt osa – alueet.

3.3.8 Ohjelman muut toiminnot

Yrityksen pyynnöstä ja osaksi omien ideoiden pohjalta ohjelmaan tehtiin muitakin toimintoja, kuin pelkkä testi. Yritys halusi mahdollisuuden toimilaitteen manuaaliajoon PC:ltä, koska se mahdollistaa kaikenlaisten kokeilujen tekemisen, joita toimintaa kehitettäessä ja tuotetta testattaessa tarvitsee tehdä. Pyynnön pohjalta tehtiin ohjelmaan osa, jossa toimilaitetta voi ajaa tiettyyn paineeseen tai tiettyyn asemaan. Siinä on myös manuaalisesti käytettävissä kaikki painelähteen sisältämät, ohjattavat vastaventtiilit. Manuaaliajon käyttöliittymä ja ohjeet on esitetty liitteen 4 sivulla 10.

Testausohjelman perusnäkymää hallitsee lista vanhoista testeistä. Kun ohjelma käynnistyy, listaan haetaan aina tietokannasta uusimmat testit. Toimilaitemallin ja testausajankohdan sisältävillä hakuehdoilla voidaan listaan hakea muita vanhoja testejä. Testituloksia voidaan tarkastella samanlaisessa näkymässä, kuin mikä ilmestyy testin päätyttyä näytölle tai se voidaan suoraan viedä Exceliin tulostettavaksi.

Toimilaitteiden ja testin raja – ja toiminta – arvojen muokkaamiseen tehtiin oma osansa ohjelmaan. Sieltä päästään säätämään kaikkia muokattavissa olevia arvoja. Kun arvoja on muokattu, ne päivitetään tietokantaan, ja ne ovat ajantasaisina käytettävissä, jos muokkaamisen jälkeen aloitetaan esimerkiksi uusi testi. Oman idean pohjalta ohjelmaan tehtiin työkaluvalikkoon statistiikka – ohjelma, jolla voidaan tarkastella tehtyjen testien lukumäärää, läpäisy – ja hylkäysprosentteja, maksimi liikematkan vaihtelua ja muutamia muita mahdollisesti laadunvalvonnan kannalta kiinnostavia lukuja.

Optiona ohjelmaan tehtiin osio, jossa sille voidaan opettaa kohdistettavien osien vaihe – eron vertailuarvo. Siinä periaatteessa vain mitataan asema – anturin signaali halutussa vaihe – eron nollakohdassa, ja mitattu arvo tallennetaan tietokantaan vertailuarvoksi. Vaihe – eron mittaaminen on toistaiseksi vain optio, sillä vaihe – eroa ei toistaiseksi pystytty kokoonpanoteknisistä syistä vielä toteamaan asema – anturin mittauspisteestä. Kaikki oheistoiminnot on esitelty liitteessä 4 sivuilla 5 – 9.

3.4 Parannusehdotukset

Itse koodissa huomaa jatkuvasti siistimisen varaa ja tapoja miten asioita voisi koodata järkevämmin ja yksinkertaisemmin, koska koodatessaan kehittyy jatkuvasti. Kuitenkin opinnäytteen opintopistemäärän, ja sitä myötä siihen laitettavan työmäärän, ollessa melko rajallinen, ei voi koko ajan koodata vanhoja koodeja uudelleen vain siksi, että niistä tulisi siistimpiä tai yksinkertaisempia. Ja kun niitä ei tee uudelleen, on koodeista nähtävissä oma kehityskaari, joka on ollut koko projektin aikana nousujohteinen. Lopulta EXE – muotoon käännetystä ohjelmasta näitä koodejakaan ei päästä edes tarkastelemaan, joten siltäkin kannalta niiden ylenpalttinen siistiminen ei ole järkevää, ellei aio käyttää koodeja myös muihin sovelluksiin jatkossa.

Testi – VI:n voisi tehdä päätason VI:n tapaan JKI – tilakoneen mallilla, jolloin ohjelman laajempi muunneltavuus tilojen järjestyksen osalta olisi mahdollista sen EXE - muotoon kääntämisen jälkeen. Se toimisi niin, että kirjoitettaisiin esimerkiksi tekstitiedostoon tiloja luettelona, ja tekstitiedosto luettaisiin kun testi aloitetaan. Se vaatisi testin muokkaajalta perehtymistä, mutta ei kalliin Labview – lisenssin omistamista ja koko ohjelman uudelleen koodaamista. Testausperiaatetta vaihdeltiin koko opinnäytetyöprosessin aikana useampaan kertaan, joten ohjelman muuttamiselle saattaa vieläkin tulla tarvetta, kun testaaminen saadaan kunnolla käyntiin.

Vanhojen testien arkistoiminen hoidetaan nyt tietokannan uudelleen nimeämisellä, jolloin ohjelma seuraavan kerran käynnistyessään luo uuden. Uudelleen nimeämisen sijasta olisi voinut vielä tehdä koodin, joka ohjelmaa käynnistettäessä vertaa, onko kannassa esimerkiksi tietty määrä testejä tai onko jokin tietty määrä aikaa kulunut. Jos ehto täyttyy, ohjelma siirtäisi vanhat testit niin sanottuun historiatauluun ja poistaisi ne varsinaisesta testitaulusta. Näin testi – taulu ei paisuisi ylettömän suureksi ja kaikki testit säilyisivät silti tallessa samassa kannassa. Toisaalta jos tuotantomäärät, ja sitä kautta testausmäärät pysyvät maltillisina, ei kannan pitäisi kasvaa lyhyessä ajassa kovinkaan suureksi.

POHDINTA

Opinnäytetyö opetti tekijäänsä suunnattoman paljon. Lähtötasoni Labview -ohjelmoinnissa oli käytännössä olematon. Tehdessä kuitenkin oppi jatkuvasti uutta ja kehittyi paremmaksi ohjelmoijaksi. Nyt vastaava ohjelma syntyisi huomattavan paljon nopeammin ja koodikin olisi varmasti siistimpää ja tehokkaampaa. Lisäksi SQL – tietokannan mukaan ottaminen ja sen opettelu oli erinomainen lisä kokonaisuuteen. Molempia taitoja voi varmasti käyttää ja soveltaa muuallakin.

Samoin tiedot laitteistopuolesta olivat melko pintapuolisia. Opin antureista, mittaamisesta, laitteiston rakentelusta ja suunnittelusta paljon. Senkin huomasi, että laitteiston suunnitteluun, komponenttien valintaan ja rakenteluun vaikuttaa yllättävän paljon käytettävissä oleva aika ja budjetti. Lisäksi pitää olla tietämystä tai taitoa etsiä millaisia komponentteja markkinoilla on ylipäätään olemassa käytettäväksi ja muodostaa kustannustehokkaita, mutta käyttötarkoitukseensa sopivia ratkaisuja niiden pohjalta.

Vaikka projekti vaikutti tekijän lähtötasosta johtuen aluksi melko haastavalta, koen, että laadukkaan ohjauksen ja pitkäjänteisen työn ansiosta saavutettiin erinomainen lopputulos. Tästä on kiittäminen projektia ohjannutta Mikko Vaaralaa, jolta sain hyviä vinkkejä ja apua etenkin projektin alussa, kun oma Labview – osaamiseni oli vielä melko rajallista. Ilman hänen ohjaustaan lopputuloksena olisi todennäköisesti ollut huomattavasti vaatimattomampi ja melko staattinen testausjärjestelmä nykyisen muunneltavan sijaan.

Kerätyn palautteen perusteella testausjärjestelmä on yrityksen toiveiden mukainen ja paikoitellen jopa ylittää ne. Esimerkiksi tiedonhallinta tietokannalla on huomattavasti kehittyneempi ja helppokäyttöisempi kuin alkuperäisessä hahmotelmassa ollut tekstitiedostoon tallentaminen ja sieltä lukeminen. Yritys on mahdollisesti soveltamassa tietokantoja muihinkin järjestelmiinsä huomattuaan sen tehokkuuden ja monipuolisuuden. Testausjärjestelmä tuo selkeää lisäarvoa yritykselle jo ihan senkin vuoksi, että ilman asianmukaista testausta tuotteen valmistus ei olisi standardin

mukaista. Lisäksi automatisoitu testaus vapauttaa tuotteita valmistavan ja testaavan asentajan tekemään muita töitä, kuten seuraavaa tuotetta, testaamisen ajaksi.

Testausprosessi saatiin projektin aikana jo melko yksinkertaiseksi, vaikka loppuvaiheessa saatavilla ei ollut juurikaan tuotteita testattavaksi. Kompaktiin muotoon pakattu fyysinen laitteisto ja vähäiset asennukset ennen testausta tekevät testausprosessista vaivattoman. Jos tuotteen tuotanto olisi ollut täysipainoisesti käynnissä, olisi prosessia päästy hiomaan vielä enemmän.

Uuden tuotteen tuotantokuntoon saattamisesta jäi mieleen, että se on yllättävän hidasta, ja se sisältää paljon oheistyötä. Jos tuotteita ei tilata, ei kehitystoimintakaan aina etene, koska on muitakin töitä tehtävänä. Työn edistyminen ei ole siis välttämättä aina vain itsestä kiinni. Tuotteen tuotantokuntoon saattaminen on pitkä prosessi, josta opinnäytetyöni oli vain pieni osa, mutta melko tärkeä, sillä ilman testausjärjestelmää tuotteen valmistaminen ei olisi toimilaitteen valmistusta koskevan standardin vaatimusten mukaista.

Voin suositella vastaavan opinnäytetyön aiheen vastaanottamista kenelle tahansa konetekniikan opiskelijalle. Tekemällä oppii paljon uutta ja pääsee syventämään koulussa käytyjä asioita. Eikä aluksi haastavalta ja laajalta vaikuttavaa tehtävää kannata vieroksua, vaan nähdä se nimenomaan mahdollisuutena oppia ja kehittyä. Labview – ohjelmointitaidot ovat hyödyllinen lisä minkä tahansa tekniikan alan insinöörin taitoihin. Etenkin testauksesta kiinnostuneille Labview – osaaminen voi avata työmahdollisuuksia.

LÄHTEET

Aumala, O. 1989. Mittaustekniikan perusteet. Helsinki: Otatieto Oy.

Aumala, O. 1998. Mittaussignaalien käsittely. Tampere: Pressus Oy.

Fonselius, J. 1988. Anturit. Helsinki: Opetushallitus.

Hovi, A. 2004. SQL – opas. Jyväskylä: Docendo Finland Oy