Kalle Kruth

Säteilylähteiden testaussolun master-ohjelmiston toteutus TestStandillä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

25.4.2024

Tiivistelmä

Tekijä:	Kalle Kruth
Otsikko:	Säteilylähteiden testaussolun master-ohjelmiston toteutus TestStandillä
Sivumäärä:	33 sivua + 4 liitettä
Aika:	25.4.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Lehtori Tuomas Leppänen Senior Test Development Engineer Janne Kari

---

Insinöörityössä tutustuttiin Planmecan röntgenlaitteiden säteilylähdeosastolla sijaitsevaan testaussoluun sekä ohjelmoitiin kyseiseen testaussoluun uusi ohjaus- eli master-ohjelma. Työn aikana tutustuttiin NI:n TestStand- ja LabVIEW-ohjelmistoihin, joilla uusi master-ohjelma toteutettiin. Master-ohjelma haluttiin ohjelmoida uudelleen, koska vanha ohjelma ei ollut toimintavarma. Ohjelmaan oli lähes mahdotonta tehdä muutoksia, eikä ohjelma ilmoittanut virhetilanteista.

Uuden masterin ohjelmointiin käytettiin pohjatietona vanhan masterin koodia, sekä olemassa olevaa koodin ulkopuolista dokumentaatiota. Työn kulku eteni ensin tutustumalla testaussoluun perinpohjaisesti. Tutustumisen yhteydessä käytiin läpi säteilylähteiden testausprosessi, testaussolun keskeisimmät komponentit, testaussolun komponenttien väliset kytkennät sekä kommunikaatioon käytetyt protokollat ja käskyt.

Uusi master-ohjelma saatiin toteutettua onnistuneesti ja insinöörityön alussa asetettuihin tavoitteisiin päästiin niiltä osin, miten työn puitteissa ehdittiin testaamaan. Toimintavarmuuden varmistaminen onnistuu vasta, kun ohjelma saadaan täysipäiväiseen käyttöön tuotantoon. Pitkän validointiprosessin takia käyttöönottoa ei ehditty tekemään työn aikana.

Avainsanat: TestStand, LabVIEW, testausautomaatio, graafinen ohjelmointi, hammaslääkärin röntgenlaitteet

---

Abstract

Author:	Kalle Kruth
Title:	Implementing Master Software for a Radiation Source Testing Cell Using TestStand
Number of Pages:	33 pages + 4 appendices
Date:	25 April 2024
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisors:	Tuomas Leppänen, Lecturer Janne Kari, Senior Test Development Engineer

---

This thesis work concerns the testing cell located in the radiation source production department of Planmeca's X-ray machines. The purpose of the work was to program a new control software, also known as master software, for the testing cell. The thesis work acquainted with NI's TestStand and LabVIEW programming environments that were used to program the new master. The master program needed to be reprogrammed because the old program was not reliable, it was nearly impossible to make changes to it and the program did not report error situations.

The old program's code, as well as existing documentation outside the code, were used as reference for the new master. The first step of the thesis work was familiarization with the testing cell. Familiarization of the testing cell included going through the testing process, getting to know the key components of the cell, mapping the connections between the components as well as figuring out the protocols and commands used for communication. After all necessary information was gathered through familiarizing with the testing cell, the master software was programmed with TestStand and LabVIEW using the aforementioned information as guidance.

The new master program was successfully implemented, and the objectives set at the beginning of the thesis work were achieved to the extent that they could be tested within the scope of the work. Ensuring reliability will only be possible once the program is put into full-time production use. Due to the lengthy validation process, the program could not- be put into full-time production use during the project.

Keywords: TestStand, LabVIEW, Test automation, Graphical programming, Dental X-ray machines

Sisällys

Lyhenteet

API: *Application Programming Interface.* Ohjelmointirajapinta.

UDP: *User Datagram Protocol.* Yhteydetön datakommunikaatioprotokolla.

TCP: *Transmission Control Protocol*. Yhteysorientoitunut datakommunikaatioprotokolla.

DAQ: *Data Acquisition.* Datankeruulaite, joka mahdollistaa sähköisten ja fysikaalisten ilmiöiden mittaamisen ja ohjaamisen tietokoneella.

DUT: *Device Under Test.* Testauksessa oleva laite.

VI: *Virtual Instrument.* LabVIEW-ohjelma.

1 Johdanto

Opinnäytetyössä suunnitellaan ja ohjelmoidaan testaussolun master-ohjelmisto, jonka tarkoituksena on ohjata säteilylähteiden testaamisesta vastaavan solun toimintaa. Suunnittelun pohjana käytetään vanhaa master-ohjelmaa ja tarkoituksena on parantaa testaussolun toimintavarmuutta sekä helpottaa tulevien parannuksien lisäämistä. Työssä tutustutaan TestStand- sekä LabVIEW-ohjelmointiympäristöihin, säteilylähteiden testaussoluun sekä master-ohjelman rakenteeseen ja toimintaideaan.

Alkuperäinen master-ohjelma ohjelmoitiin testaussolun suunnittelun ja rakentamisen yhteydessä aikavälillä 2015–2018, mutta ohjelman käytössä ja toimivuudessa ilmenivät ongelmia heti ohjelman elinkaaren alussa. Masteria on päivitetty ajan myötä useita kertoja, mutta vikoja ei ole saatu korjattua kunnolla, joten ohjelma päätettiin ohjelmoida kokonaan uudelleen.

Master ohjelma vastaa testaussolun sisällä olevan robotin käskyttämisestä, säteilylähteiden testien tilan seurannasta sekä testauksesta vastaavien testiohjelmien kanssa kommunikoinnista. Ohjelma lukee säteilylähteiden sarjanumerot sekä näyttää visuaalisesti testisolun statuksen operaattorille.

Opinnäytetyö tehdään muun muassa hammaslääkärin hoitotuoleja sekä röntgenlaitteita valmistavalle Planmeca Oy:lle. Planmeca Oy on vuonna 1971 Heikki Kyöstilän perustama yksityisomisteinen perheyritys. Planmeca Group, johon Planmeca Oy kuuluu, työllistää maailmanlaajuisesti yli 4600 ihmistä, ja yritysryhmän liikevaihto vuonna 2022 oli 1,2 miljardia euroa. [Planmeca Better care through innovation.]

2 Säteilylähteet

Säteilylähde on nimensä mukaisesti röntgenlaitteen osa, joka tuottaa röntgensäteilyä kuvantamista varten. Planmeca myy useita eri röntgenlaitemalleja, jotka tarvitsevat eri tehoisia ja kokoisia säteilylähteitä. Tämä tarkoittaa sitä, että testaussolun on pystyttävä testaamaan kaikki säteilylähdetyypit. Säteilylähdetyyppejä on neljä erilaista.

ProOne

Säteilyteholtaan ja kooltaan pienin säteilylähde on ProOne-panoraamalaitteissa käytetty säteilylähde. Kuvassa yksi vasemmalla on säteilylähde ja oikealla panoraamalaite, jossa säteilylähdettä käytetään.

Kuva 1. Planmecan valmistama panoraamakuvantamiseen tarkoitettu ProOne-röntgenlaite sekä laitteen sisällä oleva säteilylähde [Planmecan 2D-kuvantamisratkaisut].

ProMax

Seuraavaa säteilylähdettä käytetään ProMax-kuvantamislaitteissa. ProMaxia käytetään niin panoraama- kuin 3D-kuvantamisessa. Kuvassa kaksi ProMax säteilylähde vasemmalla ja Planmeca ProMax 2D S2 -röntgenlaite oikealla.

Kuva 2. Planmecan valmistama ProMax 2D S2 -röntgenlaite sekä laitteessa käytetty säteilylähde [Planmecan 2D-kuvantamisratkaisut].

Verity

Verity-säteilylähdettä ei käytetä Planmecan omissa röntgenlaitteissa. Säteilylähdettä käytetään Planmedin, joka on yksi Planmeca Groupiin kuuluvista yrityksistä, kuvantamislaitteessa. Verity-kuvantamislaitetta käytetään muun muassa raajojen kuvantamiseen. Kuvassa kolme on Verity-säteilylähde sekä laite, jossa säteilylähdettä käytetään.

Kuva 3. Muun muassa raajojen kuvantamiseen käytetty Verity-kuvantamislaite, sekä laitteessa käytetty säteilylähde [Planmed Verity Brochure].

120 kV

Neljäs testaussolussa testattava säteilylähde on 120 kV:n säteilylähde, jota käytetään Planmecan 3D-kuvantamiseen tarkoitetuissa Viso-laitteissa. Kuvassa neljä on Viso G7 -röntgenlaite ja 120 kV:n säteilylähde.

Kuva 4. Planmeca Viso G7 -röntgenlaite sekä laitteessa käytetty säteilylähde [Planmecan KKTT-laitteet].

3 Ohjelmistot

Testaussolun master-ohjelmisto päätettiin toteuttaa NI:n tuottamalla TestStand-sekvensserillä, jolla kutsutaan NI:n LabVIEW-ohjelmointikielellä ohjelmoituja aliohjelmia. Pääsyy, miksi TestStand ja LabVIEW valittiin, on se, että yrityksen sisällä käytetään laajasti kyseisiä ohjelmistoja, minkä seurauksena ylläpito tulee helpottumaan jo löytyvän osaamisen ansiosta.

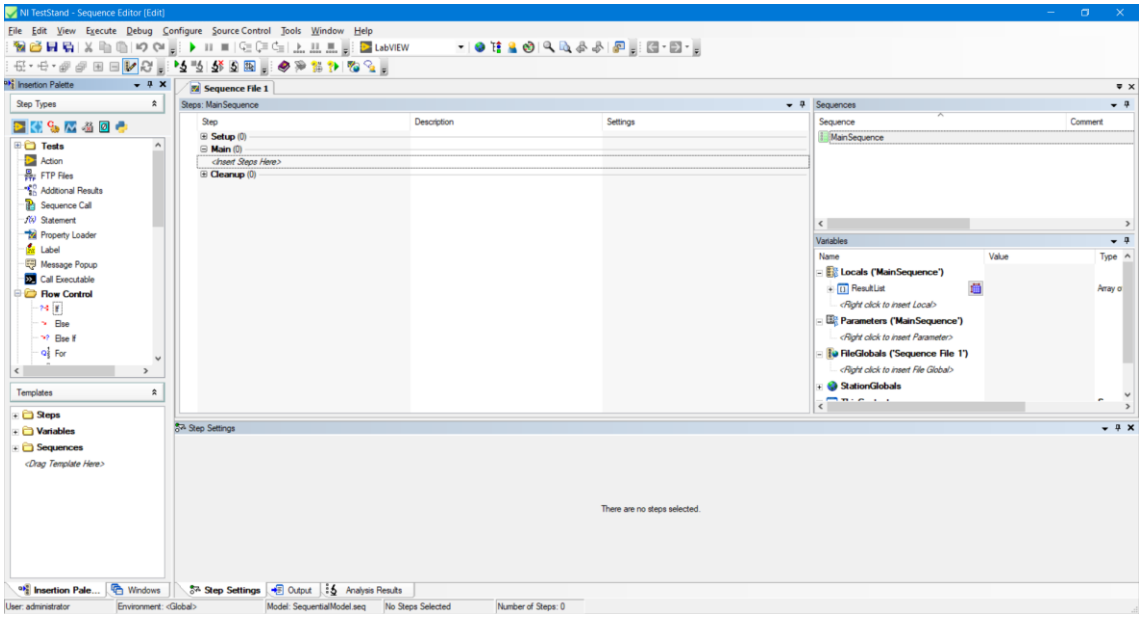
3.1 TestStand

TestStand on NI:n tekemä ATE (automated test executive) -ohjelma, joka organisoi ja suorittaa sekvensseinä uudelleenkäytettäviä koodimoduuleita, jotka voidaan kirjoittaa useilla eri ohjelmointikielillä [TestStand Reference manual 2008: 1–1]. ATE-ohjelma erottaa testauksen suorittavat koodimoduulit (usein laitteistokohtaisia) ja testauksen kokonaisvaltaisen suoritussysteemin (usein kaikille laitteistoille sama) toisistaan. Näin saavutetaan modulaarinen ja helposti skaalautuva arkkitehtuuri, joka on helpommin ylläpidettävä pitkällä aikavälillä. [Test Executive Software – Build or Buy? A Financial Comparison Using NI TestStand 2023.]

TestStand-ohjelma koostuu sekvensseistä, jotka koostuvat stepeistä. Stepit taas voivat suorittaa useita toimia kuten alustaa instrumentin, suorittaa monimutkaisen testin tai kontrolloi sekvenssin suoritusjärjestystä [Getting Started with TestStand 2014]. Sekvenssit koostuvat kolmesta askelryhmästä eli step groupista, jotka ovat setup, main ja cleanup. Setupissa on tarkoitus suorittaa valmistelevat toimet, kuten mittalaitteiston valmistelu. Mainissa suoritetaan sekvenssin päätoiminnallisuus. Cleanup on viimeisteleviä toimintoja kuten laitteiston turvallista alasajoa varten. Cleanup pyritään aina suorittamaan, vaikka setupissa tai mainissa tapahtuisi virhe.

Sekvenssit rakennetaan raahaamalla kuvassa viisi näkyvän käyttöliittymän vasemmasta reunasta haluttu askeltyyppi käyttöliittymän keskellä näkyvään sekvenssi-ikkunaan haluttuun askelryhmään. Askel voidaan konfiguroida käyttöliittymän alareunassa step settings -valikossa. Käyttöliittymän oikeassa

yläreunassa on listattu kaikki sekvenssitiedoston sekvenssit. Sekvenssi-ikkunan alapuolella on sekvenssin muuttujat.



Kuva 5. TestStandin käyttöliittymä.

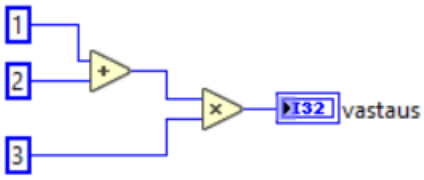
Askeltyyppejä on useita, joista tärkeimmät tämän opinnäytetyön kannalta ovat Action-askel sekä Flow Control- ja Synchronization-askeleet. Action-askelta käytetään kutsumaan LabVIEW-koodimoduleita, Flow Control -askeleet ovat muistakin ohjelmointikielistä tuttuja if-else-, for-, while- ym. -rakenteita, ja Synchronization-askeleet ovat rinnakkaisajon synkronointiin tarkoitettuja askeleita. Synkronointiaskeliin perehdytään tarkemmin luvussa 5.4.

3.2 LabVIEW

LabVIEW on NI:n kehittämä graafinen ohjelmointikieli, joka alun perin kehitettiin helpottamaan datan keräämistä laboratorioinstrumenteilla tiedonkeruujärjestelmiä hyväksi käyttäen. LabVIEW:tä voidaan käyttää datan keräämiseen instrumenteista, datan prosessointiin, datan analysointiin ja instrumenttien sekä laitteiden hallintaan. LabVIEW mahdollistaa insinöörien tuoda dataa ulkomaailmasta tietokoneelle, tehdä päätöksiä saadun datan perusteella ja hallita ulkomaailmassa olevia laitteita datasta saaduilla päätöksillä. [Larsen 2010: 11.]

LabVIEW-ohjelmia kutsutaan VI:ksi, joka tulee sanoista virtual instrument. LabVIEW-ohjelmat koostuvat kahdesta komponentista, front panelista ja block diagramista. Front panel sisältää ohjelman graafisen käyttöliittymän ja näyttää käyttäjälle ohjelman näppäimet ja grafiikan. Block diagram pitää sisällän ohjelman yhteen langoitetut ohjelmointielementit, jotka rakentavat itse ohjelman toiminnallisuuden. [Larsen 2010: 14.]

LabVIEW on niin sanottu dataflow-ohjelmointikieli eli ohjelman suoritusjärjestys määräytyy saatavasta datasta. Esimerkkinä kuvan kuusi numeroiden yksi ja kaksi yhteenlasku suoritetaan ennen kertolaskua, sillä yhteenlaskupalikalla on kaikki suoritukseen tarvittava data. Yhteenlaskun suorituksen jälkeen kertolaskupalikka saa kaiken tarvitsemansa datan, joten palikan koodi voidaan suorittaa ja saadaan vastaus. LabVIEW:n dataflow-periaate tekee rinnakkain pyörivien koodipätkien ohjelmoinnin erittäin helpoksi, mutta kääntöpuolena kyseisten rinnakkaisten koodien suoritusjärjestystä on mahdotonta tietää.



Kuva 6. Yhteenlasku ja kertolasku LabVIEW:lla laskettuna.

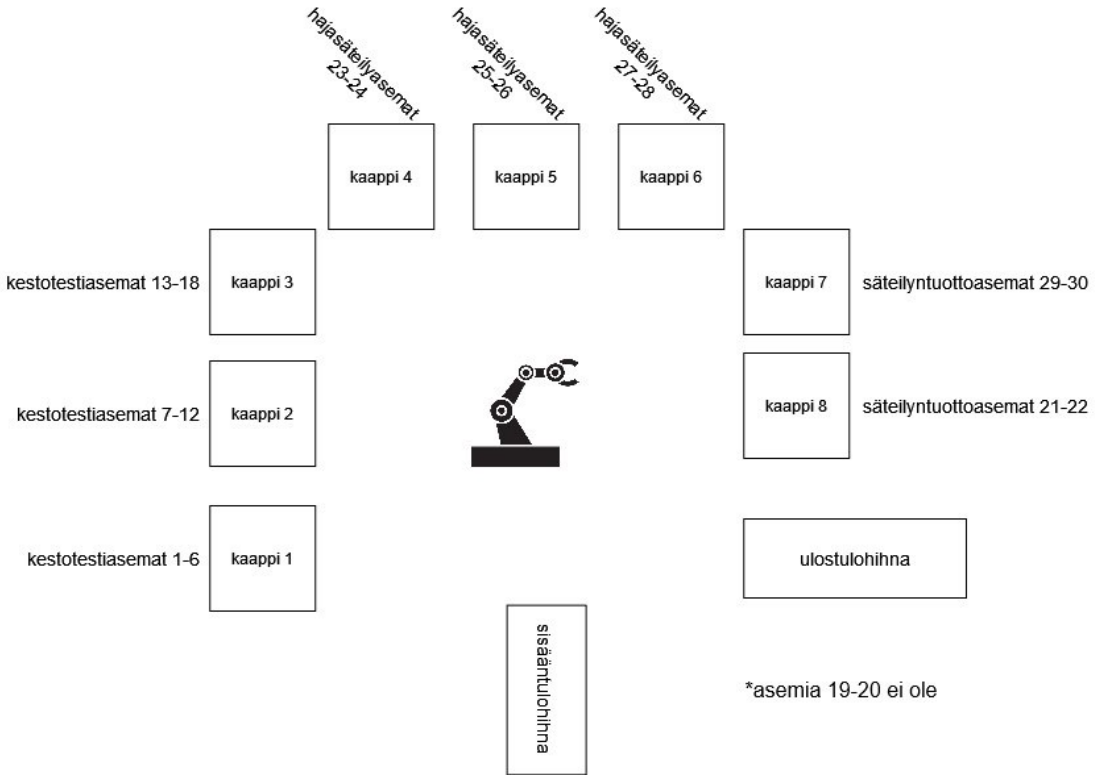
4 Testaussolu

Jokainen röntgenlaitteeseen menevä säteilylähde kulkee testaussolun lävitse, jossa säteilylähde testataan kolmessa eri testissä. Ensimmäinen testausvaihe on kestotesti, jossa säteilylähteellä tehdään ennalta määrätty määrä valotuksia ja katsotaan, että säteilylähde suorittaa valotukset ilman läpilyöntejä tai muita virheitä. Läpilyönneillä tarkoitetaan säteilylähteen sisällä tapahtuvia oikosulkuja, jotka voivat johtua muun muassa kokoonpanovirheistä, epäpuhtauksista öljyn seassa tai öljyn seassa olevasta vedestä.

Testausprosessin toinen vaihe on säteilyntuottotesti. Säteilyntuottotesti on viranomaisten määräämä turvallisuuteen liittyvä testi, jossa mitataan säteilylähteen kykyä tuottaa oikea määrä säteilyä lineaarisesti ja toistettavasti. Testissä testataan myös säteilyn laatua. Testin aikana säteilylähteellä valotetaan eri virta-arvoilla sekä eri pituisilla valotuksilla ja mitataan säteilymittarilla, että säteilylähde tuottaa odotetun määrän röntgensäteilyä.

Testausprosessin kolmas vaihe on toinen viranomaisten määräämä hajasäteilytesti, jossa varmistetaan, että säteilylähteen säteilysuojauksessa ei ole vikoja ja ettei suoja päästä säteilyä liikaa läpi. Käytännössä testi suoritetaan tulppaamalla säteilylähteen säteilyaukko, jolloin säteilylähteen ulkopuolelle, ei pitäisi vuotaa merkittävästi säteilyä. Säteilysuoja ei ole sataprosenttinen, minkä takia on määritelty tarkat säteilymäärät, jotka säteilylähde saa säteillä ympäristöön. Testin aikana säteilylähteellä suoritetaan pitkiä valotuksia ja mitataan säteilyn määrää säteilylähteen eri kulmista.

Testaussolu on täysin automatisoitu solu, joka koostuu Yaskawan robotista, sisään- ja ulostuloliukuhihnoista sekä testikaapeista. Kuvassa seitsemän on kuvitettu testaussolun karkea pohjapiirrustus.



Kuva 7. Testaussolun pohjapiirustus.

Jokaisessa testikaapissa on kaksi lokeroa. Kaappien 1–3 lokeroissa on kolme kestotestiasemaa lokeroa kohti. Kaappien 4–8 lokeroissa on yksi testiasema lokeroa kohti.

Operaattori asettaa säteilylähteen sisääntulohihnalle, jota kautta säteilylähde kulkeutuu testaussolun sisälle. Liukuhihnan päässä olevan valokennon aktivoituessa robotille lähetetään säteilylähteen tunnistuskäsky. Robotti kuvaa tarttujan vieressä olevalla kameralla säteilylähteen ja yrittää tunnistaa säteilylähteen tyypin käyttäen konenäköä. Kun säteilylähde on tunnistettu onnistuneesti, robotti noutaa säteilylähteen liukuhihnalta ja vie sen kestotestiasemaan. Kestotestikaappien lokeroiden täytyy olla täynnä, jotta testit voidaan aloittaa, eli kun robotti on vienyt kolme säteilylähdettä lokeroon, niin lokeron ovi suljetaan ja kestotestit aloitetaan. Kun kaikkien lokerossa olevien säteilylähteiden testit ovat loppuneet, lokeron ovi avautuu ja robotti noutaa säteilylähteet kaapista ja vie ne joko ulostulohihnalle tai säteilyntuottotestiin riippuen testin tuloksesta.

Robotin vietyä säteilylähteen säteilyntuottotestiasemaan, lokeron ovi sulkeutuu ja testi alkaa. Testin päätyttyä lokeron ovi avautuu ja robotti noutaa säteilylähteen ja vie sen joko hajasäteilytestiin tai ulostulohihnalle riippuen testin tuloksesta.

Ennen kuin säteilylähde asetetaan hajasäteilytestiasemaan, täytyy säteilylähteen sädeaukko tulpata. Hajasäteilytestikaappien vieressä on jokaiselle säteilylähdetyypille oma tulppa, jonka robotti asettaa säteilylähteen sädeaukkoon ennen, kun vie sen testiasemaan. Kun säteilylähde on viety asemaan, lokeron ovi sulkeutuu ja testi käynnistyy. Testin päätyttyä lokeron ovi aukeaa, robotti noutaa säteilylähteen, irrottaa tulpan säteilylähteen sädeaukosta ja vie sen ulostulohihnalle. Säteilylähteen orientaatio ulostulohihnalla ilmaisee operaattorille, onko säteilylähde läpäissyt kaikki testit vai onko joku testi hylätty.

4.1 Robotti

Testaussolun robotti on Yaskawan MH50-robotti DX100-ohjaimella. Kommunikaatio robotin ja master-ohjelman välillä tapahtuu digitaalisignaalein. Master-ohjelman tietokoneeseen on kytketty kaksi NI:n valmistamaa DAQ-moduulia, yksi sisääntulo- ja yksi ulostulomoduuli.

Jotkin testaussolun sensoreista kuten sisääntulohihnan valokenno sekä ulostulohihnan valokennot on kytketty robotin ohjauskaappiin. Ohjauskaapin lähdöistä on tuotu johdot master-tietokoneella olevalle DAQ-sisääntulomoduulille. Sisääntulomoduulin kautta liukuhihnojen sensoreiden tilat välittyvät master-ohjelmalle.

Masterin käskyt robotille välitetään ulostulo DAQ-moduulin kautta. Master voi käskeä robottia noutamaan säteilylähteen sisääntulohihnalta ja testausasemista sekä viemään säteilylähteen testausasemaan ja ulostulohihnalle. Master voi käskeä robottia myös aloittamaan viivakoodinlukuliikkeet, jotka ovat hitaita ohivientejä viivakoodinlukijan edestä.

Itse robotin liikesekvenssit ovat valmiiksi ohjelmoituna robottiin eikä master ota niihin kantaa. Master ainoastaan kertoo, milloin pitää hakea ja mistä sekä milloin pitää viedä ja minne.

4.2 Testiasemat ja Robottisolun tietokoneet

Jokaisessa testauskaapissa on kaksi Siemensin LOGO!-ohjelmoitavaa logiikkaa eli yksi logiikka lokeroa kohti. Testiasemien sensorit on kytketty ohjelmoitavien logiikoiden sisääntuloihin, ja ohjelmoitavat logiikat välittävät sensoridatan master-ohjelmalle. Säteilylähteiden virransyöttö tulee releiden kautta, joita ohjelmoitavat logiikat ohjaavat oman ohjelmointinsa mukaan. Logiikat ajavat myös lokeroiden ovia masterin käskyjen mukaan.

Robottisolussa on yhteensä kahdeksan tietokonetta. Seitsemän näistä tietokoneista on varattu säteilylähteiden testaamisesta vastaavien testausohjelmien pyörittämiseen. Kahdeksas tietokone pyörittää master-ohjelmaa. Kommunikaatio tietokoneiden välillä tapahtuu UDP (User Datagran Protocol) -protokollaa käyttäen.

UDP on yhteydetön tiedonsiirtoprotokolla, jota käytetään nopeutta vaativissa käyttötarkoituksissa. Yhteydettömällä protokollalla tarkoitetaan protokollaa, jossa lähettäjän ja vastaanottajan välille ei muodosteta yhteyttä ennen datan lähettämistä. [Reynders & Wright 2003: 131.]

Kolme tietokonetta on varattu kestotestiasemille eli vastaavat kaapeissa 1–3 olevien asemien 1–18 testiohjelmista. Kaksi tietokonetta on varattu kaapeissa 4–6 olevien testiasemien 23–28 testiohjelmille ja kaksi tietokonetta on varattu 7–8 kaapeissa olevien testiasemien 21, 22, 29 ja 30 testiohjelmille.

5 Master-ohjelma

5.1 Masterin rooli ja yleisimmät vikatilanteet

Robottisolun master-ohjelman roolina on ohjata solun toimintaa. Master kommunikoi robotin kanssa ja käskee robottia aloittamaan oikeat toiminnot oikeaan aikaan. Esimerkkinä säteilylähteen tullessa sisääntuloliukuhihnan valokennon kohdalle, master lähettää robotille komennon aloittaa säteilylähteen tunnistus. Master vastaa myös testiohjelmien kanssa kommunikoinnista, eli masterin tehtävä on aloittaa oikeat testit oikeissa asemissa, kun niihin tuodaan säteilylähteitä. Masterin täytyy pitää kirjaa säteilylähteiden sijainnista testaussolun sisällä, jotta oikeiden säteilylähteiden sarjanumerot pystytään yhdistämään oikeisiin testeihin ja näin säilyttämään jäljitettävyys. Master vastaa myös testien turvallisesta suorittamisesta, sillä säteilylähteiden testaamiseen liittyy säteilyä ja säteilylähteiden ei tulisi tuottaa säteilyä ihmisen ollessa testaussolun sisällä. Testaussolun turvapiirin ollessa lauenneena, master ilmoittaa siitä testausohjelmille ja käskee niitä pysäyttämään testit, kunnes turvapiiri on kuitattu.

Nykyinen master-ohjelma on toimintavarmuudeltaan melko heikko ja aiheuttaa säteilylähteiden tuotantoon lähes päivittäisiä katkoksia. Katkokset työllistävät testauslaitteiden ylläpidosta vastaavaa tiimiä turhaan, mikä puolestaan vaikeuttaa tiimin muuta toimintaa.

Yleisimmät vikatilanteet ohjelman ajon aikana ovat jumiutumistilanteet, jolloin ohjelma ei suostu tekemään tarvittavia toimenpiteitä, kuten noutamaan säteilylähdettä, jonka testi on valmistunut testausasemasta. Ajon aikana on esiintynyt myös tilanteita, jolloin master on käskenyt robottia viemään säteilylähteitä vääriin asemiin. Ohjelma ei myöskään ilmoita vikatilanteista operaattoreille millään tavoin, mikä johtaa tilanteisiin, joissa testaussolu on saattanut seistä vikatilassa useamman tunnin ajan ennen kuin vikatilannetta on huomattu.

Nykyinen master on toteutettu LabVIEW-ohjelmoinnilla ja ohjelman arkkitehtuurin takia muutosten teko koodiin, sekä vikojen etsiminen koodista, on erittäin

vaikeaa. Näiden edellä mainittujen syiden takia, master-ohjelma päätettiin ohjelmoida täysin uudelleen.

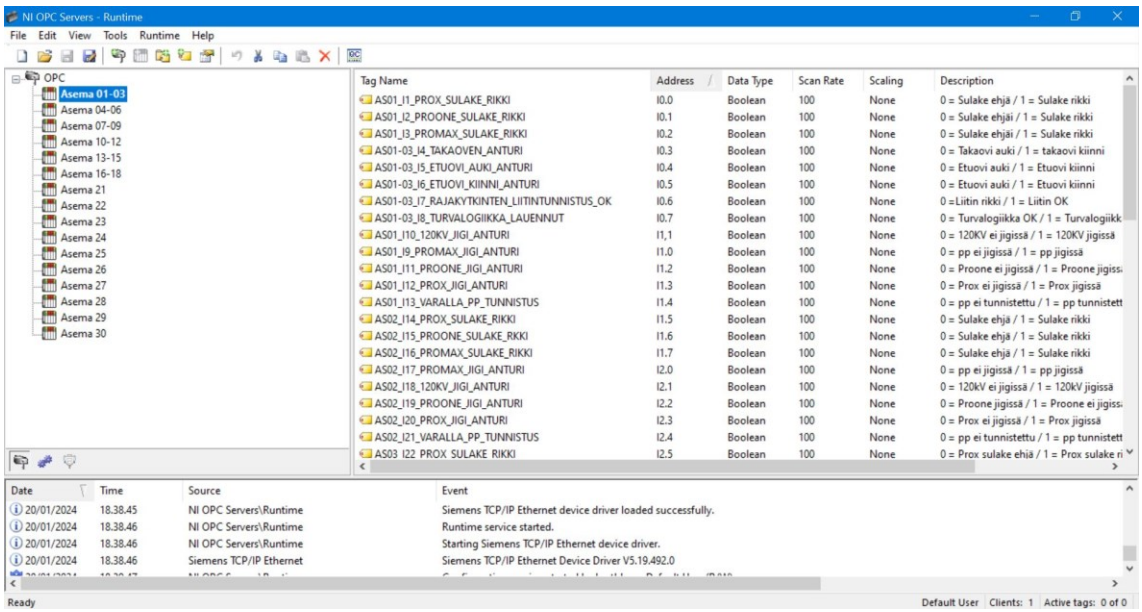
5.2 Lähtötiedot

Uuden master-ohjelman suunnittelun pohjana käytettiin vanhan masterin tietoja. Tärkein tiedon lähde oli ohjelman koodi, jota läpi käymällä saatiin tietoa siitä, miten master kommunikoi testausohjelmien ja robotin kanssa sekä kokonaiskuva ohjelman rakenteesta.

Itse koodin lisäksi ohjelmasta löytyi jonkin verran koodin ulkopuolista dokumentaatiota, joka sisälsi tietokoneiden käyttämät IP-osoitteet sekä testausohjelmissa käytetyt porttinumerot ja digitaaliset kytkennät robotin ja master-tietokoneen välillä. Kommunikaatio masterin ja PLCeiden kanssa oli toteutettu NI:n tekemää OPC Servers -ohjelmaa käyttäen.

Käytännössä OPC Servers toimi välitulkkina PLCeiden ja LabVIEW:n välissä eli OPC Servers lähetti ja vastaanotti datan logiikalta ja LabVIEW taas lähetti ja sai datan OPC Serverssiltä. Nimestään huolimatta OPC Servers ei kommunikoi PLCeiden kanssa OPC:n välityksellä, vaan Siemensin omalla kommunikointiprotokollalla, joka on rakennettu TCP-protokollan päälle.

OPC Servers -ohjelmaan pystyy lisäämään logiikan laitteena ja tälle laitteelle voidaan lisätä tageja. Tageille annetaan Address eli osoite, mikä vastaa logiikalla olevan tagin osoitetta. Tämän jälkeen tietokoneelle saadaan reaaliaikaista dataa logiikoiden tagien tiloista. Kuvassa kahdeksan on esitelty OPC Servers -ohjelma ja osa yhden laitteen tageista.



Kuva 8. OPC Servers -ohjelma

Tärkein apu OPC Serversistä oli sen konfiguraatiotiedostot, jotka sisälsivät listan kaikista logiikoiden kytkennöistä. Tiedostoista näki jokaisen tagin nimen, joka kuvasi melko hyvin, mitä kyseiseen tagiin on liitetty. Nimen lisäksi oli kerrottu osoite eli fyysinen tulo tai lähtö, mihin kyseisen tagin sensori tai toimilaite on kytketty. Nimen ja osoitteen lisäksi tiedostoissa oli jokaiselle tagille kuvaus, jossa avattiin, mitä tagin eri tilat tarkoittavat.

Nämä tiedot osoittautuivat korvaamattomiksi, kun uutta masteria lähdettiin ohjelmoimaan, sillä lopullisessa versiossa päätettiin olla käyttämättä kyseistä ohjelmaa valmistajan tuen loppumisen takia. OPC Serversin sijaan kommunikaatio toteutettiin Hampel Software Engineeringin tekemällä avoimen lähdekoodin S7NetCom LabVIEW -lisäosalla, joka mahdollisti suoran kommunikaation LabVIEW:stä logiikalle.

S7NetCom on kommunikaatioajuri, jonka avulla LabVIEW voi kommunikoida Siemensin S7-logiikoiden kanssa käyttäen Siemensin omaa kommunikaatioprotokollaa. Ajuri on toteutettu LabVIEW:n TCP/IP primitiivejä käyttäen. [Hampel 2022.]

5.3 Uuden masterin toimintaidea

Uuden masterin toimintaidean ytimessä on asynkroninen rinnakkaisajo, mikä tarkoittaa, että samanaikaisesti suoritetaan useita prosesseja omissa säikeissään, jotka toimivat itsenäisesti toisistaan. Rinnakkaisajon avulla ohjelma saadaan toimimaan huomattavasti nopeammin sekventiaaliseen ajoon nähden. Rinnakkaisajo tuo jonkin verran monimutkaisuuttaa koodiin ja prosessien synkronoiminen saattaa olla haasteellista, mutta koin, että rinnakkaisajon edut voittavat mahdolliset haasteet. Esimerkkinä eduista vanhalla masterilla, joka toimii sekventiaalisesti, kestää pahimmassa tapauksessa useita minuutteja reagoida testin valmistumiseen, kun taas rinnakkaisajolla toteutetulla masterilla, reagointi tapahtuu maksimissaan sekunnissa.

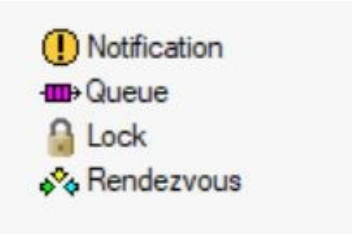
Ohjelma voidaan jakaa kolmeen pääkomponenttiin, jotka ovat pääsekvenssi, testisekvenssi ja käyttöliittymä. Pääsekvenssi vastaa säteilylähteiden siirtelylogiikasta, eli sekvenssi päättää, milloin säteilylähteet noudetaan ja mistä testiasemasta sekä minne säteilylähteet tulee viedä noudon jälkeen. Sekvenssi vastaa säteilylähteen sarjanumeron lukemisesta, sekä sarjanumeron validiteetin tarkastamisesta. Pääsekvenssin vastuulla on kirjanpito säteilylähteiden sijainnista testaussolun sisällä. Sekvenssi vastaa myös itse käskytyskomentojen lähetyksestä robotille, turvapiirin tilan seurannasta ja tilan välittämisestä testausohjelmille sekä lokin kirjoituksesta.

Jokaiselle säteilylähteelle käynnistetään oma testisekvenssi sen jälkeen, kun säteilylähde on viety kestotestiasemaan. Testisekvenssi pyörii pääsekvenssin ja muiden mahdollisten testisekvenssien rinnalla. Testisekvenssi vastaa testiohjelmien kanssa kommunikoinnista, lokeroiden ovien avaus- ja sulkukomentojen lähettämisestä logiikoille sekä säteilylähteiden testien statuksien välittämisestä pääsekvenssille.

Käyttöliittymä on pääosin LabVIEW:llä toteutettu visuaalinen apu operaattoreille kertomaan säteilylähteiden testien tilasta. Käyttöliittymä ilmoittaa myös mahdollisista testaussolun vikatilanteista.

5.4 Synkronointi ja tiedonsiirto säikeiden välillä

Synkronointi ja tiedonsiirto rinnakkain pyörivien sekvenssien välillä toteutetaan käyttäen TestStandin notifikaatioita, jonoja, lukkoja, rendezvousia ja globaaleja muuttujia. Kuvassa yhdeksän on esitelty notifikaatio, jono, lukko ja rendezvous stepit.



Kuva 9. TestStandin synkronointisteppejä.

Notifikaatio

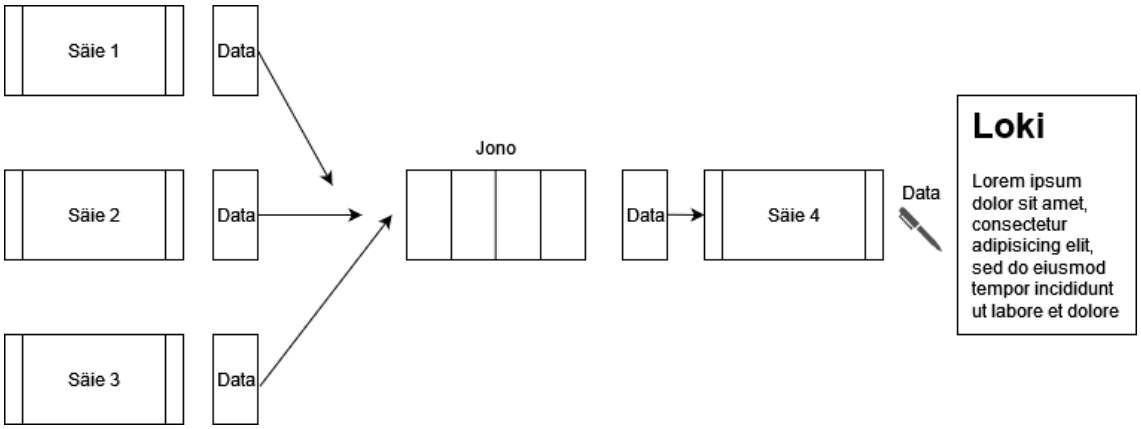
Notifikaatiolla säie pystyy nostamaan ilmoituksen, jonka avulla säie pystyy ilmoittamaan haluamastaan asiasta muille kyseistä notifikaatiota kuunteleville säikeille. Lähettävä säie pystyy pelkän ilmoittamisen lisäksi lähettää dataa kuunteleville säikeille.

Kun halutaan käyttää notifikaatiota, tulee notifikaatio ensin luoda. Luonnin yhteydessä notifikaatiolle annetaan nimi. Notifikaation nimi toimii ilmoituksen tunnisteena ja nimen avulla säikeet voivat päättää, mitä notifikaatiota kuuntelevat tai minkä notifikaation asettavat. Kun notifikaatiota käytetään luonnin jälkeen, ilmoituksen asettava säie laittaa ilmoituksen pystyyn ja jatkaa oman ohjelmansa suoritusta. Ilmoitus pysyy asetettuna, kunnes ilmoituksen elinikä loppuu tai kunnes ilmoituksen lukeva säie laskee ilmoituksen eksplisiittisesti alas. Lukeva säie voi siis päättää luetaanko ilmoitus ainoastaan kerran, vai halutaanko, että useampi säie lukee saman ilmoituksen. Esimerkkinä notifikaatiolla voidaan kertoa, kun operaattori on kuitannut jonkun kuittausta vaativan asian, jolloin ohjelma voi jatkaa suoritustaan.

Jono

Jonolla eli queuella voidaan synkronoida datan luonti ja kulutus useiden säikeiden välillä ilman, että dataa katoaa. Käytännössä dataa luovat säikeet laittavat dataa jonon perälle sitä mukaa, kun ne sitä luovat. Dataa kuluttava tai kuluttavat säikeet lukevat dataa jonon kärjestä. Näin kaikki lähetetty data saadaan käsiteltyä ilman datan häviämistä, vaikka datan luonti tapahtuisi nopeammin kuin datan käsittely.

Jonon käyttöön liittyvät samat periaatteet kuin notifikaatioillakin eli jonoja voi olla useita. Jonot erotetaan toisistaan antamalla jonolle luonnin yhteydessä nimi. Jonoon lisätessä dataa sekä jonosta luettaessa dataa voidaan päättää käsiteltävä jono nimen perusteella. Jonoa voidaan esimerkiksi käyttää kuvassa kymmenen esitetyllä tavalla lokin kirjoitukseen. Kuvassa säikeet 1–3 lähettävät jonoon dataa, jonka säie neljä kirjoittaa lokitiedostoon. Näin säikeiden 1–3 ei tarvitse odottaa lokikirjoituksen valmistumista, vaan voivat jatkaa heti datapaketin lähetyksen jälkeen omaa suoritustaan.



Kuva 10. Jonon käyttäminen lokikirjoituksen apuna.

Lukko

Lukkoja käytetään, kun halutaan rajata koodin osa, jonka vain tietty määrä säikeistä voi suorittaa samanaikaisesti. Ensimmäisen säikeen aloittaessa koodin osan suorituksen säie lukitsee lukon, ja seuraavat säikeet jäävät odottamaan, kunnes ensimmäinen säie on suorittanut koodin ja avannut lukon. Lukon auettua, seuraava odottava säie lukitsee lukon ja aloittaa koodin suorituksen ja niin edelleen.

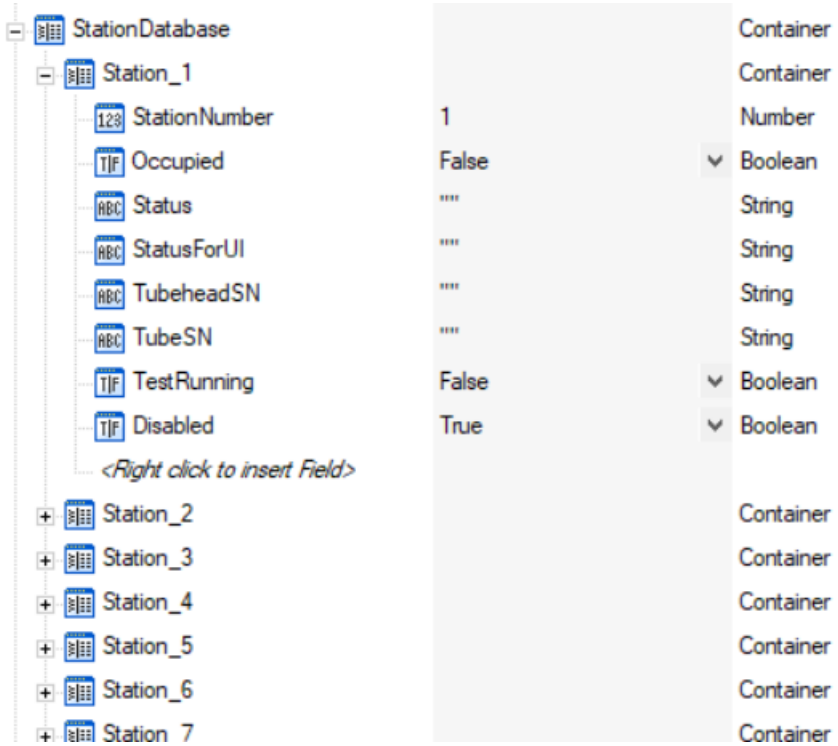
Lukkoa voidaan käyttää myös lukitsemaan koodin osa, minkä jälkeen muut suorittavat säikeet hyppäävät lukitun koodiosan yli. Tämä on hyödyllistä esimerkiksi tilanteessa, jossa vain yhden testiaseman säikeen halutaan käskyttää lokeron oven sulkeminen ja muiden samassa lokerossa olevien testisäikeiden halutaan hypätä käskytyksen yli.

Rendezvous

Rendezvousia voidaan käyttää pakottamaan säikeitä odottamaan toisiaan ja ”tapaamaan” tietyssä kohtaa koodia, ennen kuin ne pääsevät jatkamaan ohjelman suoritusta. Oivallinen rendezvousin käyttökohde on viime luvussa esitelty lukon esimerkki. Ensimmäisen säikeen lukittua koodin osa ja säikeen käskyttäessä lokeron oven sulkua loput, koodinpätkän yli hypänneet, säikeet jäävät odottamaan rendezvous-kohtaan, kunnes lokeron ovi on suljettu. Kun ovi on suljettu ja sulkeva säie on päässyt tapaamiskohtaan, kaikki säikeet jatkavat suoritusta.

Globaalit muuttujat

Globaalit muuttujat ovat muuttujia, jotka on jaettu kaikkien säikeiden kanssa. Kaikki säikeet pystyvät lukemaan ja muuttamaan globaaleja muuttujia. Esimerkkinä käytetyistä globaaleista muuttujista on StationDatabase, joka on konttityyli-nen muuttuja, jonka sisällä jokaiselle testaussolun asemalle on oma muuttuja. Jokainen asemamuuttuja on myös oma konttimuuttuja, joka sisältää kyseisen aseman oleelliset tiedot. Kuvassa 11 on StationDatabase-globaalimuuttuja.



Kuva 11. StationDatabase-globaalimuttuja.

5.5 Pääsekvenssi

Setup

Sekvenssin setupissa eli ohjelman käynnistämisen jälkeen ensimmäisessä suoritettavassa askelryhmässä suoritetaan toimet, jotka tehdään vain kerran masterin käynnistyessä. Setupissa luodaan notifikaatio pääsekvenssin ja käyttöliittymän väliseen kommunikointiin sekä notifikaatio kriittisten virheiden lähettämiseen käyttöliittymään. Kriittisillä virheillä tarkoitetaan virheitä, joista ohjelma ei pysty palautumaan ja ohjelman suoritus täytyy pysäyttää. Notifikaatioiden lisäksi luodaan jono lokiviestien lähettämiseen lokin kirjoittamisesta vastaavalle sekvenssille sekä jono testisekvenssien ja pääsekvenssin väliseen kommunikointiin. Luodaan myös LabVIEW:n sisäinen jono, jota käytetään viestien lähettämiseen TestStandistä LabVIEW-käyttöliittymään. LabVIEW-jonon referenssi tallennetaan globaaliin muuttujaan, johon kaikki ohjelman sekvenssit pääsevät käsiksi mahdollistaen kaikkien sekvenssien lähettää dataa käyttöliittymään.

Setupissa suoritetaan myös TestStandin ulkopuolisten tekstitiedostojen sisällön lukeminen ohjelman muistiin suorituksen ajaksi. Tällaisia tiedostoja ovat errors.csv, config.ini, PLCConfig.ini ja TestPCConfig.ini. Errors-tiedosto pitää sisällään TestStandin ulkopuoliset, ohjelmoijan määrittelemät tapauskohtaisesti mukautetut virhekoodit ja niiden virheviestit. Config.inin kautta suoritetaan ohjelman parametrisointi, eli configin kautta pystyään helposti muuttamaan asetuksia ilman, että itse ohjelman koodia täytyy muuttaa. PLCConfig-tiedostosta luetaan ohjelmoitavien logiikoiden tiedot kuten logiikoiden IP-osoitteet sekä muistipaikkojen osoitteet, mihin sensorit ja toimilaitteet on kytketty. TestPCConfig sisältää testitietokoneiden IP-osoitteet ja testiohjelmien käyttämät UDP-portit.

Setupissa käynnistetään viisi sekvenssiä omiin säikeisiinsä. Sekvenssit pyörivät pääsekvenssin rinnalla koko ohjelman suorituksen ajan. Lokin kirjoitussekvenssi vastaa muiden sekvenssien lähettämien lokiviestien kirjoittamisesta lokitiedostoon. Käyttöliittymä sekvenssi vastaa käyttöliittymän käynnistämisestä ja käynnissäpitämisestä. Turvalogiikan tilan valvoja vastaa turvapiirin tilan seuraamisesta ja ilmoittaa muille sekvensseille, jos turvapiiri laukeaa. Ulostulohihnan

tilan valvoja seuraa ulostulohihnan sensoreita ja ilmoittaa käyttöliittymälle, jos ulostulohihna täyttyy ja vaatii operaattorin huomiota. Viimeinen setupissa omaan säikeeseensä avattava sekvenssi vastaa käyttöliittymästä tulleiden operaattorin kuittausten välittämisestä sekvenssille, joka alun perin lähetti viestin käyttöliittymään.

Main

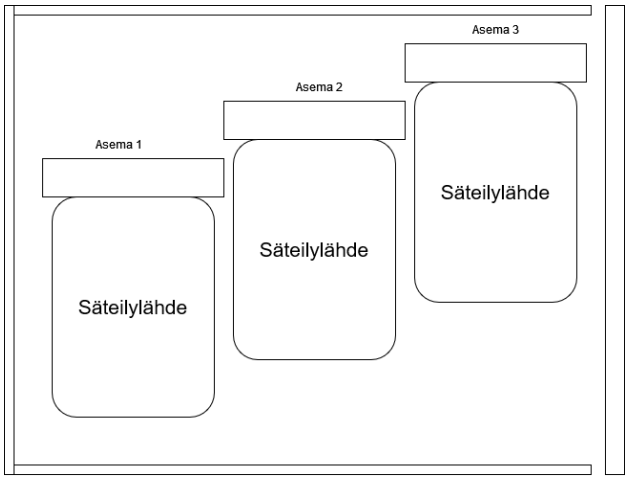
Setupin suorituksen jälkeen pääsekvenssi siirtyy pääaskelryhmään eli mainiin. Main koostuu while-loopista, jossa ohjelma pyörii, kunnes ohjelma terminoituu. Loopin ideana on ensin lukea viestit status jonosta, jotka testisekvenssit ovat lähettäneet pääsekvenssille ja päivittäneet asematietokantaan päättyneiden testien statukset. Tämän jälkeen lähdetään käymään testiasemia läpi viimeisestä testistä eli hajasäteilystä alkaen ja siirretään siirtoa odottavat säteilylähteet. Masterin siirtopäätös perustuu asematietokannassa olevaan testin status-tietoon, sekä tietoon, onko prosessin seuraavassa vaiheessa tilaa. Käytännössä jos testiaseman säteilylähteellä on joko pass- tai fail-status ja seuraavassa vaiheessa on tilaa, master käskee robottia noutamaan säteilylähteen ja kuljettamaan sen seuraavaan vaiheeseen.

Hajasäteilyasemien kohdalla ensin tarkastetaan, onko ulostulohihnalla tilaa lukemalla ulostulohihnan tilatieto robotilta. Jos ulostulohihnalla on tilaa, katsotaan, onko tarkasteltavassa testiasemassa säteilylähdettä. Jos asemassa on säteilylähde, luetaan, onko säteilylähteen testillä statusta. Jos säteilylähteen testillä ei ole statusta, se tarkoittaa, että testi on vielä kesken, joten siirtokomentoa ei lähetetä robotille ja siirrytään tarkastelemaan seuraavaa asemaa. Jos testillä on status, lähetetään robotille käsky noutaa säteilylähde asemasta ja viemään se ulostulohihnalle. Jos taas ulostulohihnalla ei ole tilaa tai tarkasteltava testiasema ei sisällä säteilylähdettä, siirrytään tarkastelemaan seuraavaa hajasäteilyasemaa. Kaikki hajasäteilyasemat käydään läpi edellä mainitulla tavalla, minkä jälkeen siirrytään tarkastelemaan säteilyntuottoasemia.

Säteilyntuottoasemat käydään saman tyylisesti läpi kuin hajasäteilyasemat. Hajasäteilyasemien tarkasteluun nähden säteilyntuottoasemien kanssa täytyy

ottaa huomioon myös testisekvenssin kanssa kommunikointi sekä hajasäteilyasemien tila. Liitteessä yksi on kuvattuna vuokaavion muodossa hakemisen logiikka. Logiikka toimii statuksen tarkastamiseen asti samalla tavalla kuin hajasäteilyasemien kanssa, mutta tarkastuksen jälkeen testin ollessa hyväksytty tarkastetaan, onko hajasäteilytestissä vapaata. Jos hajasäteilyssä on vapaa asema, robotti käsketään noutamaan säteilylähde säteilyntuottoasemasta ja viemään vapaaseen asemaan. Viennin jälkeen testisekvenssille lähetetään ilmoitus notifikaation muodossa viennin onnistuneen. Notifikaatio sisältää myös aseman numeron, jonne säteilylähde vietiin. Jos vapaata asemaa ei ole, jätetään säteilylähde odottamaan säteilyntuottoasemaan, kunnes asema vapautuu. Jos säteilylähteen testin status ei ole hyväksytty, tarkastetaan, onko ulostulohihnalla tilaa. Säteilylähde viedään ulostulohihnalle, jos hihnalla on tilaa. Jos säteilylähteen vienti jostain syystä epäonnistuu, lähetetään testisekvenssille notifikaatio, jossa käsketään testisekvenssin terminoida itsensä. Kun säteilyntuottoasemat on käyty läpi, siirrytään tarkastelemaan kestotestiasemia.

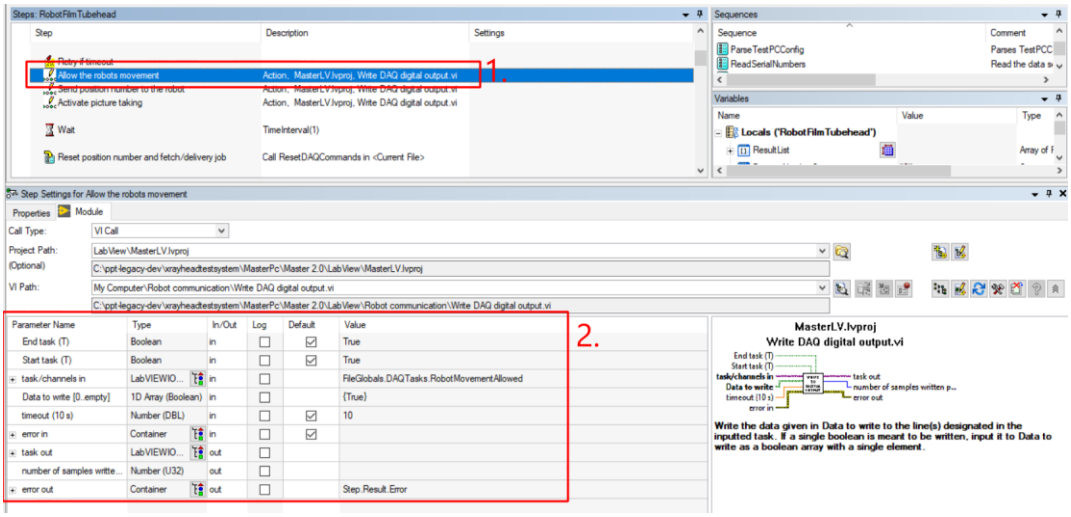
Kestotestiasemien tilan tarkastamiseen tuo monimutkaisuutta asemien asemointi lokeroiden sisällä. Jokainen kestotestilokero sisältää kolme asemaa kuvan 12 mukaisesti. Jotta päästään käsiksi sisempiin asemiin, täytyy ensin tyhjentää ulommat asemat. Tämä tosin ratkaistiin melko yksinkertaisesti liitteessä kaksi olevan hakulogiikan mukaan. Muuten kestotestistä hakeminen tapahtuu samaan tyyliin kuin säteilyntuotosta hakeminen.



Kuva 12. Sivuprofiili kestotestilokerosta, jossa lokeron ovi kuvan oikeassa reunassa.

Pääsekvenssin main-askelryhmän viimeinen vaihe on säteilylähteen noutaminen sisääntulohihnalta. Sekvenssiin sisältyy sisääntulohihnan valokennon tilan seuranta, robotin käskyttäminen aloittamaan säteilylähteen tunnistus, säteilylähteen nouto hihnalta, säteilylähteen sarjanumeroiden lukeminen, tarkastaminen onko kestotestiasemissa vapaata tilaa sekä testisekvenssin käynnistäminen tulohihnalta haetulle säteilylähteelle. Jos hihnan valokenno on aktiivisena ja kestotestissä on vapaa asema, robotille lähetetään käsky aloittaa säteilylähteen tunnistus.

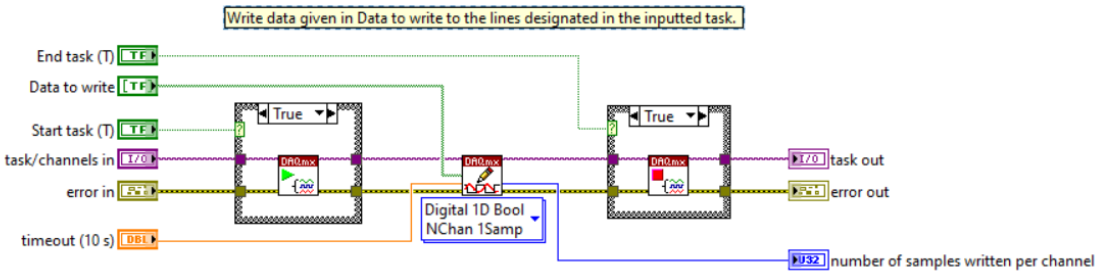
Käytännössä robottia käskytetään kutsumalla TestStandistä LabVIEW-koodia, joka kirjoittaa masterin I/O-kortin ulostulot tiettyyn tilaan. Robotti tulkitsee masterin asettamat ulostulot, jotka on kytketty robotin I/O:n sisääntuloihin ja suorittaa masterin haluamat toiminnot oman ohjelmointinsa mukaan, kuvassa 13 LabVIEW-koodin kutsu TestStandistä. Kuvaan on merkattu numerolla yksi itse TestStandin sekvenssin kohta, jossa LabVIEW-koodia kutsutaan action askeleella. LabVIEW-koodiin syötettävät parametrit on merkitty kohdassa kaksi.



Kuva 13. LabVIEW-koodin kutsu TestStandistä.

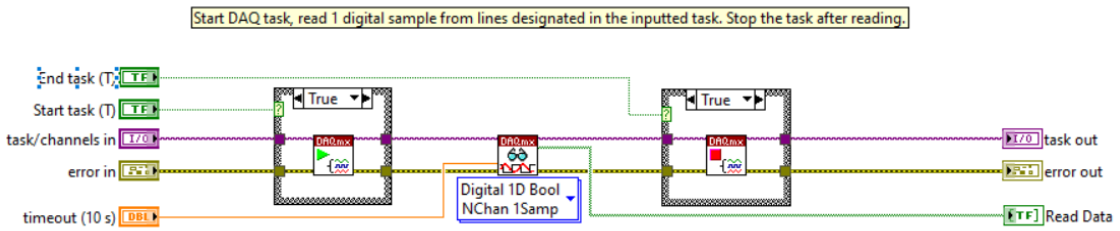
LabVIEW:n puolella ulostulot asetetaan käyttäen NI-DAQmx VI:tä kuvan 14 mukaisesti. DAQmx on ajuriohjelma, jonka avulla voidaan kommunikoida NI:n DAQ-laitteiden kanssa [NI-DAQmx Overview 2023]. Tässä tapauksessa kyseisillä VI:llä ohjataan NI-9476 digitaalista ulostulomoduulia. TestStandistä annetut

parametrit syötetään kuvan vasemmassa reunassa oleviin ohjausterminaaleihin, minkä jälkeen DAQmx taski käynnistetään, asetetaan ulostulot haluttuihin tiloihin ja pysäytetään taski. Parametrisoinnilla voidaan olla käynnistämättä tai pysäyttämättä taskia ja suorittaa ainoastaan kirjoitus.



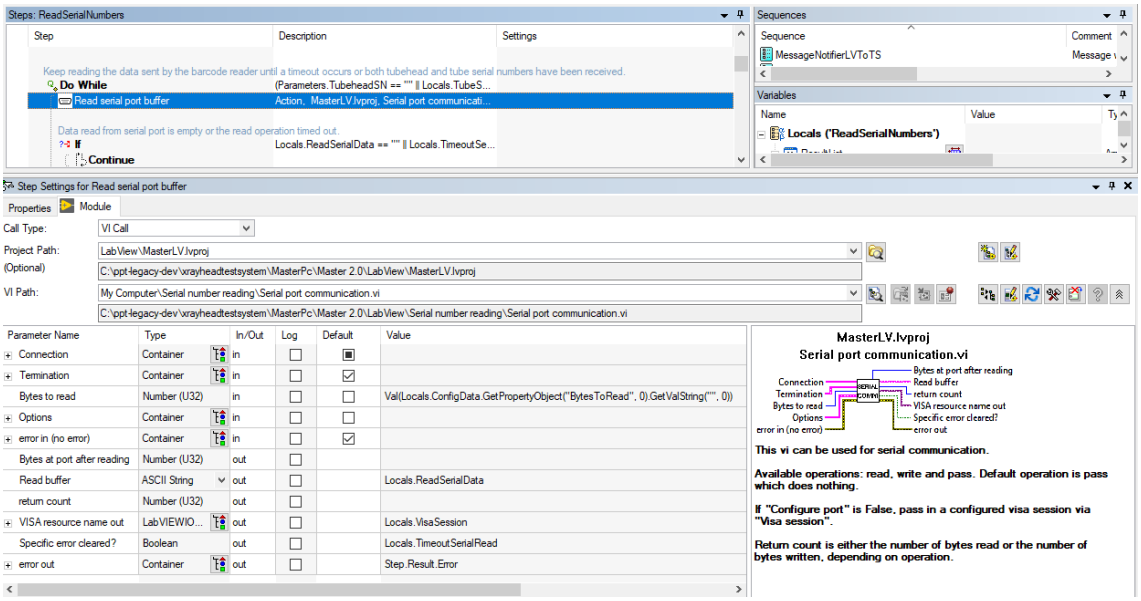
Kuva 14. Masterin ulostulojen laitto LabVIEW-koodissa.

Tiedon lukeminen robotilta hoidetaan samaan tapaan, kuin datan kirjoittaminen. Kuvassa 15 on datan lukemisen LabVIEW-koodi.



Kuva 15. DAQmx-sisääntulojen lukeminen.

Robotin kuvattua ja tunnistettua säteilylähteen tyypin robotti noutaa säteilylähteen ja aloittaa sarjanumeron lukuliikkeet. Lukuliikkeiden aikana säteilylähde viedään hitaasti viivakoodinlukijan edestä, minkä aikana master lukee säteilylähteen sarjanumerot. Viivakoodinlukija on ohjelmoitu aloittamaan lukeminen, kun se havaitsee liikettä, joten erillistä liipaisukäskyä ei tarvita. Lukija lähettää luetun sarjanumeron masterille USB:n välityksellä käyttäen sarjaliikennettä. TestStandissä lähetetyn sarjanumeron lukeminen tapahtuu kuvan 16 mukaan kutsumalla LabVIEW:llä ohjelmoitua VI:tä, jonka avulla luetaan sarjaliikenne bufferi, jonne viivakoodi on lähetetty.



Kuva 16. Sarjaliikenne VI:n kutsuminen TestStandistä.

Liitteessä neljä on esitelty itse sarjaliikenne VI. VI:ssä on käytetty NI-VISA-palikoita, jotka tarjoavat ohjelmointirajapinnan eli APIn muun muassa USB ja sarjaliikenteen ohjelmointiin LabVIEW:llä [NI-VISA Overview 2022]. VI voidaan parametrisoida käyttötarkoituksen mukaan joko lukemaan, kirjoittamaan tai olemaan tekemättä mitään, sekä näiden lisäksi konfiguroimaan sarjaliikenne portti ja sulkemaan portti.

Kun sarjanumero on luettu onnistuneesti, säteilylähde toimitetaan vapaaseen kestotestiasemaan ja säteilylähteelle käynnistetään oma testisekvenssi pyörimään pääsekvenssin rinnalle.

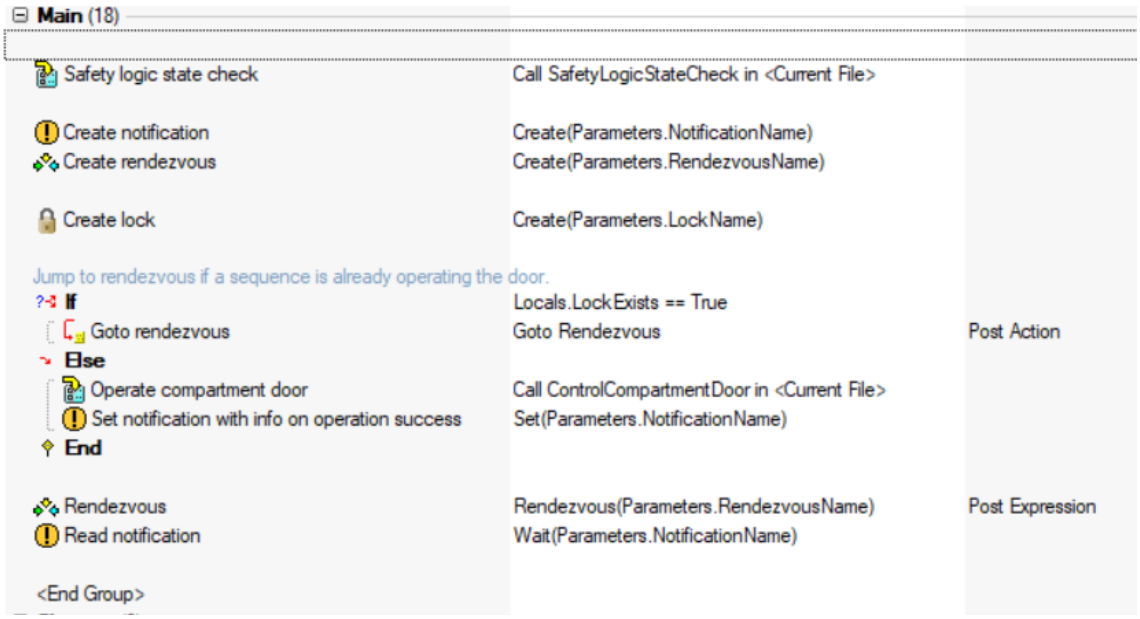
5.6 Testisekvenssi

Testisekvenssin päätehtävänä on toimia rajapintana pääsekvenssin ja säteilylähteen testiohjelmistojen välissä. Käytännössä testisekvenssi valvoo säteilylähteen testin tilaa ja ilmoittaa statusmuutoksista pääsekvenssille.

Sekvenssin setupissa tarkastetaan säteilylähteen tyyppi, jota tarvitaan myöhemmin, jotta osataan kommunikoida oikean testiohjelman kanssa. Tarkastetaan

myös, mihin kestotestilokeroon säteilylähde on viety, koska kestotesti voidaan aloittaa vasta, kun lokeron kaikissa kolmessa asemassa on säteilylähde.

Kun lokeron asemat on saatu täyteen, testisekvenssi käskee lokeron ohjelmoitavaa logiikkaa sulkemaan lokeron oven. Kestotestilokeron oven sulkemissekvenssissä käytetään kuvan 17 mukaisesti lukkoa, jotta ainoastaan yhden lokerossa olevan säteilylähteen testisekvenssi käskyttää logiikkaa kahden muun sekvenssin odottaessa. Samassa sekvenssissä käytetään myös notifikaatiota ja rendezvousta. Notifikaation tarkoituksena on ilmoittaa kahdelle odottavalle sekvenssille, onnistuiko lokeron sulkeminen. Rendezvousin idea on käskeä kahden sekvenssin odottaa kolmatta, oven sulkevaa sekvenssiä, kunnes kaikki kolme jatkavat koodin suoritusta.



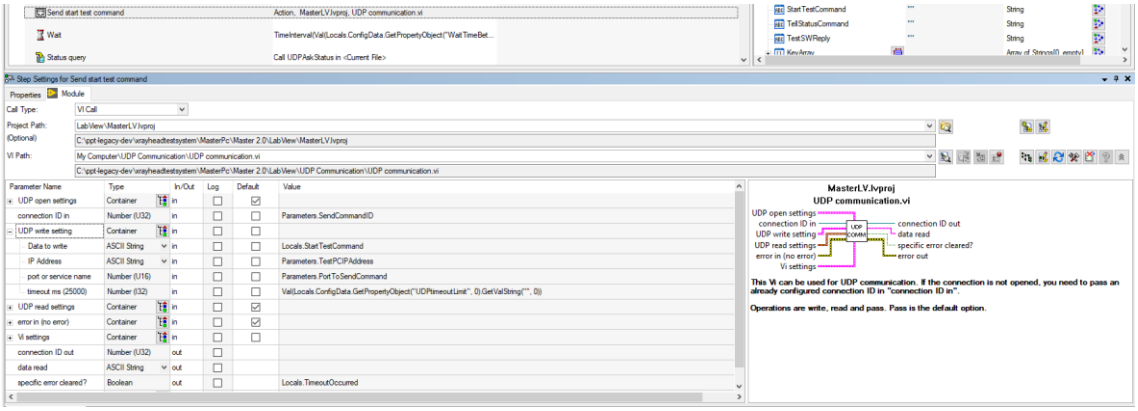
Kuva 17. Kestotestilokeron oven sulkeminen.

Kun ovi on saatu suljettua, testisekvenssi käskee aseman, jossa säteilylähde on, testausohjelmaa käynnistämään testin. Testin käynnistykseen sisältyy sarjanumeron sekä testauskomennon lähetys.

Kun säteilylähteen testi on saatu onnistuneesti käynnistettyä, testisekvenssi aloittaa testin statuksen kyselyn testiohjelmalta. Testiohjelma ei vastaa status kyselyihin niin kauan, kun testi on vielä kesken. Testin päätyttyä testiohjelma

vastaa testin tuloksella eli joko hyväksytty tai hylätty. Statuskyselyn yhteyteen on rakennettu ajastin, joka pitää kirjaa, kuinka kauan testi on ollut käynnissä, jolloin huomataan, jos testiohjelma jumiutuu. Jumiutumisesta ilmoitetaan operaattorille käyttöliittymän kautta, jolloin operaattori voi ryhtyä tarvittaviin toimenpiteisiin, vikatilanteen korjaamiseksi.

UDP-viestien lähetys tehdään kutsumalla TestStandistä LabVIEW VI:tä, joka on ohjelmoitu viestien lähettämiseen kuvan 18 mukaisesti. VI:lle annetaan parametrina lähetettävä viesti sekä muut olennaiset lähettämiseen liittyvät tiedot.



Kuva 18. UDP-viestin lähettäminen testiohjelmalle.

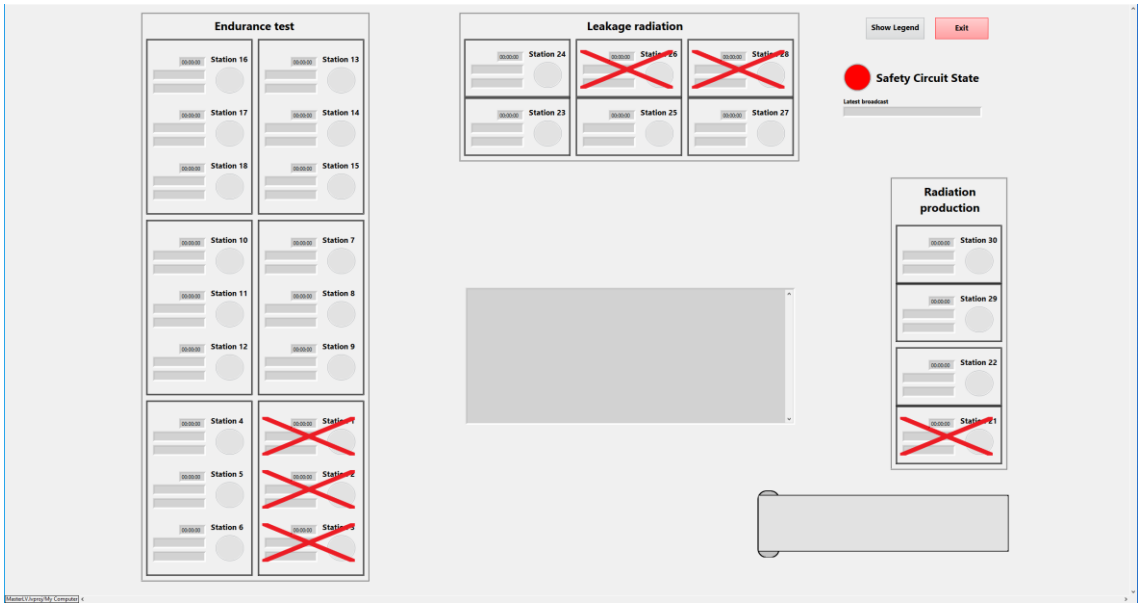
Liitteessä kolme on esiteltynä itse lähettämisestä vastaava VI. UDP-viestin lähettäminen LabVIEW:llä tehdään ensin valmistelemalla lähetys UDP Open -toiminnolla, minkä jälkeen itse viesti lähetetään UDP Write -toiminnolla [Using LabVIEW with TCP/IP and UDP 2023].

Testin statustiedon vastaanottamisen jälkeen testisekvenssi jää odottamaan, kunnes kaikki kestotestilokeron säteilylähteet ovat saaneet statuksen. Kun statukset on saatu, yksi sekvensseistä avaa lokeron oven, minkä jälkeen sekvenssit lähettävät statukset pääsekvenssille. Jos säteilylähteen testi hylättiin, sekvenssi tuhoaa itsensä statuksen lähetyksen jälkeen. Läpi menneen testin jälkeen sekvenssi jää odottamaan notifikaatiota pääsekvenssiltä. Pääsekvenssi lähettää notifikaation yhteydessä testisekvenssille tiedon, mihin säteilyntuottoasemaan säteilylähde on viety, minkä perusteella testisekvenssi sulkee aseman

lokeron oven ja lähettää testiohjelmalle aloituskäskyn. Aloituskäskyn jälkeen sekvenssi aloittaa kestotestin tapaan statuskyselyn. Hyväksytyn säteilyntuotto-testin jälkeen sekvenssi lähettää statuksen pääsekvenssille ja jää taas odottamaan notifikaatiota. Notifikaation saatuaan sekvenssi suorittaa hajasäteilytestin säteilyntuottotestin tapaan.

5.7 Käyttöliittymä

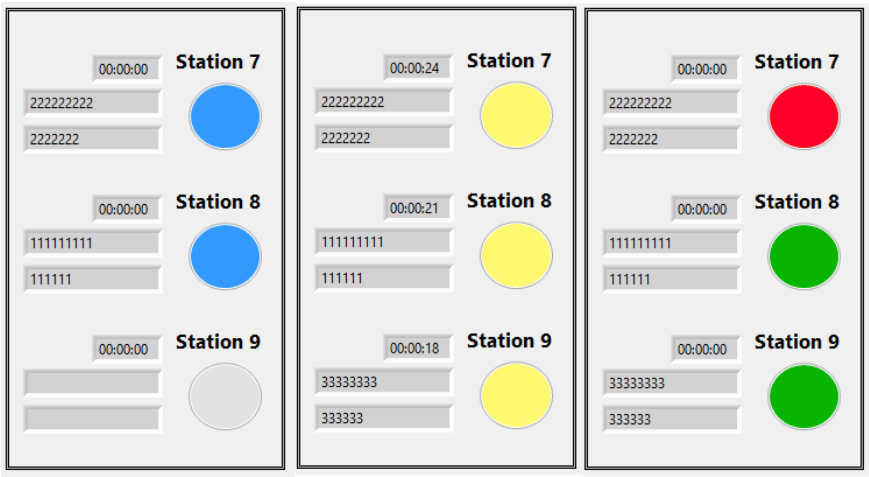
Käyttöliittymän visuaalisen suunnittelun tavoitteena oli luoda mahdollisimman yksinkertainen ja helposti ymmärrettävä ulkoasu. Tarkoituksena oli, että operaattori näkee yhdellä vilkaisulla kaiken oleellisen testaussolun tilasta. Elementtien sijoittelu käyttöliittymässä mukailee testaussolun totuudenmukaista pohjapiirustusta, mikä helpottaa operaattoria yhdistämään käyttöliittymän tapahtumat itse soluun. Käyttöliittymä on esitelty kuvassa 19.



Kuva 19. Master-ohjelman käyttöliittymä.

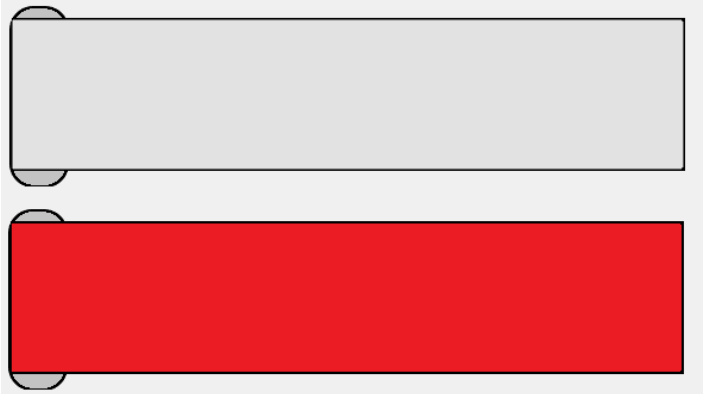
Käyttöliittymässä jokaiselle testiasemalle on oma elementtinsä, joka sisältää indikaattorit säteilylähteen sarjanumeroille ja ajastimen, kuinka kauan testi on ollut käynnissä, ja pyöreän indikaattorin, joka ilmaisee aseman tilan. Pois käytöstä olevat asemat on ilmaistu aseman päällä olevalla punaisella raksilla.

Aseman pyöreä indikaattori ilmaisee kuvan 20 mukaisesti eri värein aseman tämänhetkisen tilan. Kun aseman säteilylähde odottaa testin alkamista, indikaattorin väri on sininen. Testin ollessa käynnissä indikaattori on keltainen ja testin loputtua indikaattori on joko vihreä, jos testi on mennyt hyväksytysti läpi, tai punainen, jos testi ei ole mennyt läpi. Tyhjän aseman indikaattori on harmaa.



Kuva 20. Testiaseman pyöreän indikaattorin eri tilat.

Käyttöliittymän oikeassa alakulmassa on indikaattori, joka ilmaisee ulostulohihnan tilaa. On tärkeää, että operaattori huomaa mahdollisimman nopeasti, jos ulostulohihna täyttyy, sillä täynnä oleva hihna estää valmistuneiden säteilylähteiden tyhjentämisen testiasemista. Ilmoittaminen tapahtuu kuvan 21 mukaisesti vilkuttamalla hihnan indikaattoria punaisena, kunnes hihnan sensorit eivät enää havaitse säteilylähteitä.



Kuva 21. Ulostulohihnan indikaattori hihnan ollessa tyhjä ja täysi.

Testaussolun virhetilanteet ilmoitetaan operaattorille käyttöliittymän keskellä olevan viesti-ikkunan kautta. Solun toimiessa normaalisti ilman virheitä virheikkuna on tyhjä ja virheiden mukana näytettävät painonappulat ovat piilossa. Virhetilanteen tapahtuessa virheviesti näytetään operaattorille viesti-ikkunassa ja huomionherättävyyden kasvattamiseksi viesti-ikkunan yläpuolelle ilmestyy vilkkuva varoituskolmio, kuten kuvassa 22 näkyy. Jokaisen virheviestin mukana näkyy myös virheelle yksilöity nappi tai useampi nappi, jos virheen voi kuitata usealla eri tavalla.

Kuva 22. Virheviestin näyttäminen operaattorille.

6 Yhteenveto

Insinöörityössä tutustuttiin Planmecan säteilylähdetuotannossa sijaitsevaan testaussoluun ja testaussolua ohjaavaan master-ohjelmaan. Insinöörityön tavoitteena oli ohjelmoida vanhan master-ohjelman pohjalta uusi ohjelma, joka on toimintavarmuudeltaan ja päivitettävyydeltään parempi. Haluttiin myös, että ohjelman virhetilanteet ovat helpommin havaittavissa ja korjattavissa.

Insinöörityön aikana saatiin ohjelmoitua kaikki tarvittavat toiminnallisuudet ja parannettua testaussolun käyttäjäystävällisyyttä niin operaattoreiden kuin ylläpitäjienkin kannalta. Lopputuloksena saatiin valmis ohjelma, joka pystytään ottamaan vanhan master-ohjelman tilalle tuotantoon käyttöön. Kaikki kehityksen aikana ilmenneet ongelmatilanteet ja ohjelman virheet saatiin korjattua.

Tulevaisuuden kannalta ohjelman päivitettävyys parani huomattavasti uuden masterin-koodin modulaarisuuden takia. Käytön aikana ilmenevät ongelmat tulevat myös olemaan helpommin korjattavissa.

Ohjelmaa ei keretty saada tuotantoon käyttöön insinöörityön puitteissa pitkän validointiprosessin takia. Planmeca toimii terveysteknologia-alalla ja yrityksen laitteet tuottavat säteilyä, mikä johtaa vahvaan regulointiin. Reguloinnin takia kaikki laitteiden testaamiseen liittyvät ohjelmistot ja apuvälineet joutuvat käymään kattavan validointiprosessin läpi, mikä saattaa kestää useita kuukausia. Ohjelman toimintavarmuudesta ei voida olla täysin varmoja ennen kuin ohjelma otetaan tuotannossa täysin käyttöön. Kehityksen aikana tehtyjen testien perusteella voidaan kuitenkin olla melko varmoja, että myös toimintavarmuuden parantamistavoitteeseen päästiin.

Lähteet

Getting Started with TestStand. 2014. Verkkoaineisto. National Instruments. <https://docs-be.ni.com/bundle/373436h/raw/resource/enus/373436h.pdf>. Luettu 17.3.2024.

Hampel, Jorg. 2022. S7NetCom. Verkkoaineisto. Hampel Software Engineering. <https://dokuwiki.hampel-soft.com/code/open-source/s7netcom>. Luettu 21.1.2024.

Larsen, Ronald. 2010. LabVIEW for Engineers. Pearson.

NI-DAQmx Overview. 2023. Verkkoaineisto. NI. <https://www.ni.com/docs/en-US/bundle/daqhelp/page/ni-daqmx-overview.html>. Luettu 2.3.2024.

NI TestStand Reference Manual. 2008. Verkkoaineisto. NI. <https://download.ni.com/support/manuals/373435c.pdf>. Luettu 15.10.2023.

NI-VISA Overview. 2022. Verkkoaineisto. NI. <https://www.ni.com/en/support/documentation/supplemental/06/ni-visa-overview.html>. Luettu 9.3.2024.

Planmeca Better care through innovation. Verkkoaineisto. <https://www.planmeca.com/fi/yritys/>. Luettu 17.3.2024.

Planmecan 2D-kuvantamisratkaisut. Verkkoaineisto. Planmeca. <https://www.planmeca.com/fi/kuvantaminen/2d-kuvantaminen/tarkeimmat-hyodyt/>. Luettu 23.3.2024.

Planmecan KKTT-laitteet. Verkkoaineisto. Planmeca. <https://www.planmeca.com/fi/kuvantaminen/kktt-kuvaus/tarkeimmat-hyodyt/>. Luettu 23.3.2024.

Planmed Verity Brochure. Verkkoaineisto. Planmed. <https://materialbank.planmed.com/catalog/Planmed%20brochures/r/701/viewmode=previewview>. Luettu 23.3.2024.

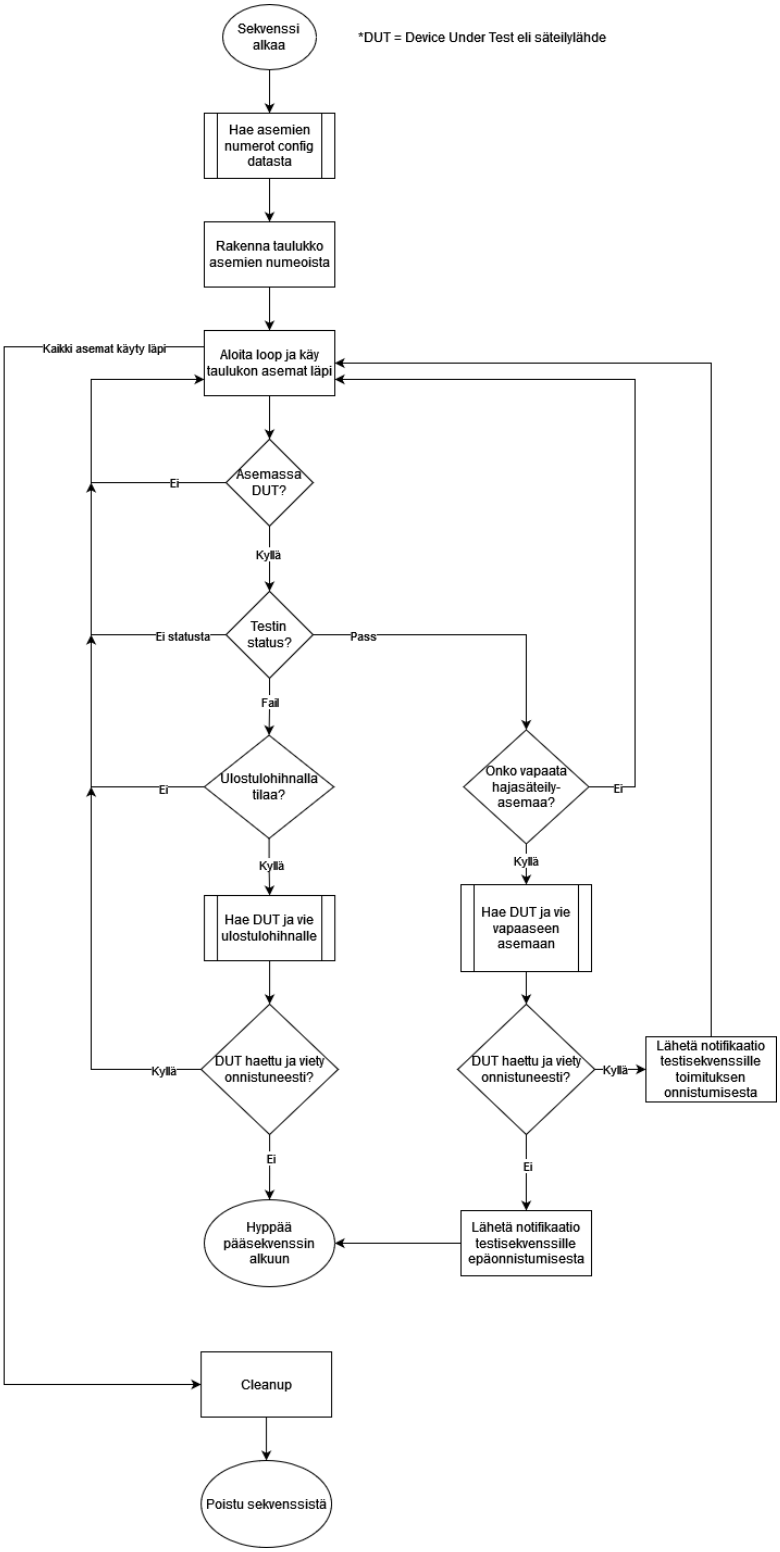
Reynders, Deon & Wright, Edwin. 2003. Practical TCP/IP and Ethernet Networking. Iso-Britannia: Newnes.

TestStand API Reference. 2023. Verkkoaineisto. NI. <https://www.ni.com/docs/en-US/bundle/teststand-api-reference/page/teststand-api-ref/operators_function_expr.htm>. Päivitetty 13.10.2023 Luettu 29.1.2024.

Test Executive Software – Build or Buy? A Financial Comparison Using NI TestStand. 2023. Verkkoaineisto. NI. <https://www.ni.com/en/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-teststand/test-executive-software---build-or-buy--a-financial-comparison-u.html>. Päivitetty 26.1.2023. Luettu 15.10.2023.

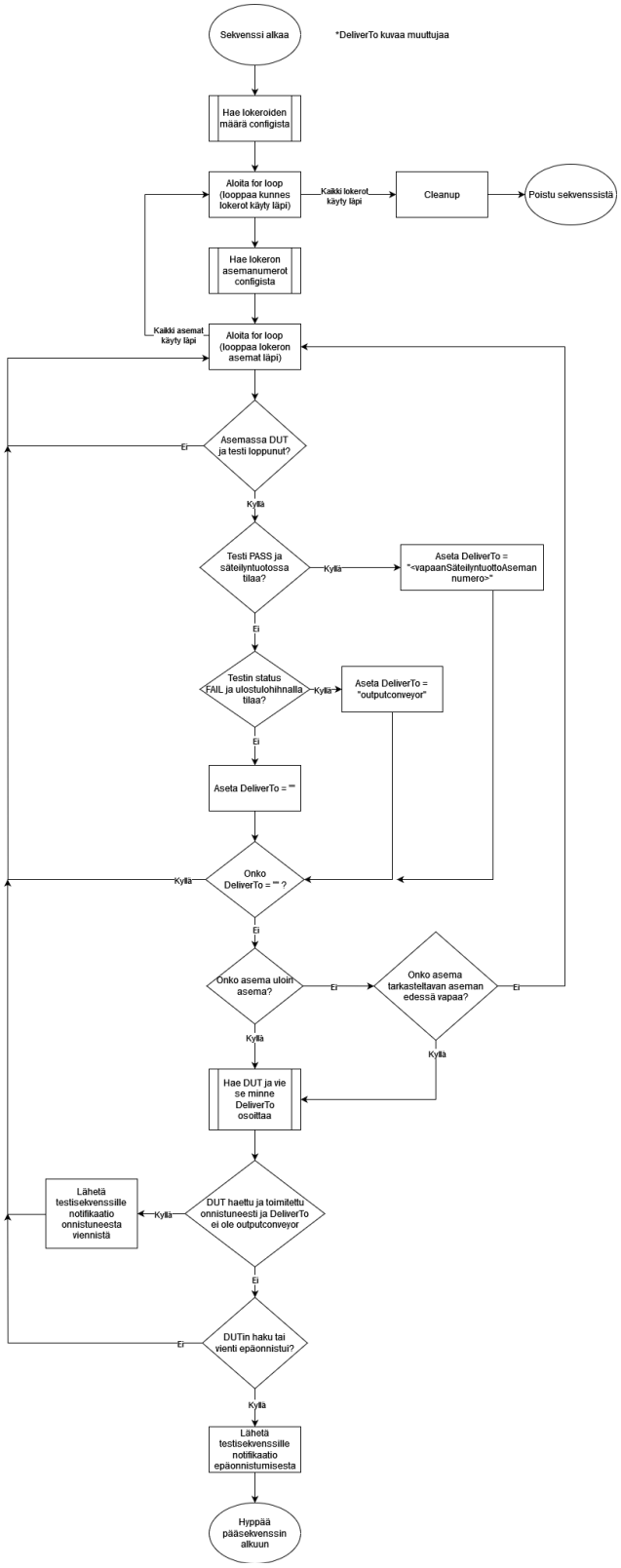
Using LabVIEW with TCP/IP and UDP. 2023. Verkkoaineisto. NI. <https://www.ni.com/docs/en-US/bundle/labview/page/using-labview-with-tcpip-and-udp.html>. Luettu 9.3.2024.

Vuokaavio säteilyntuottotestin hakulogiikasta



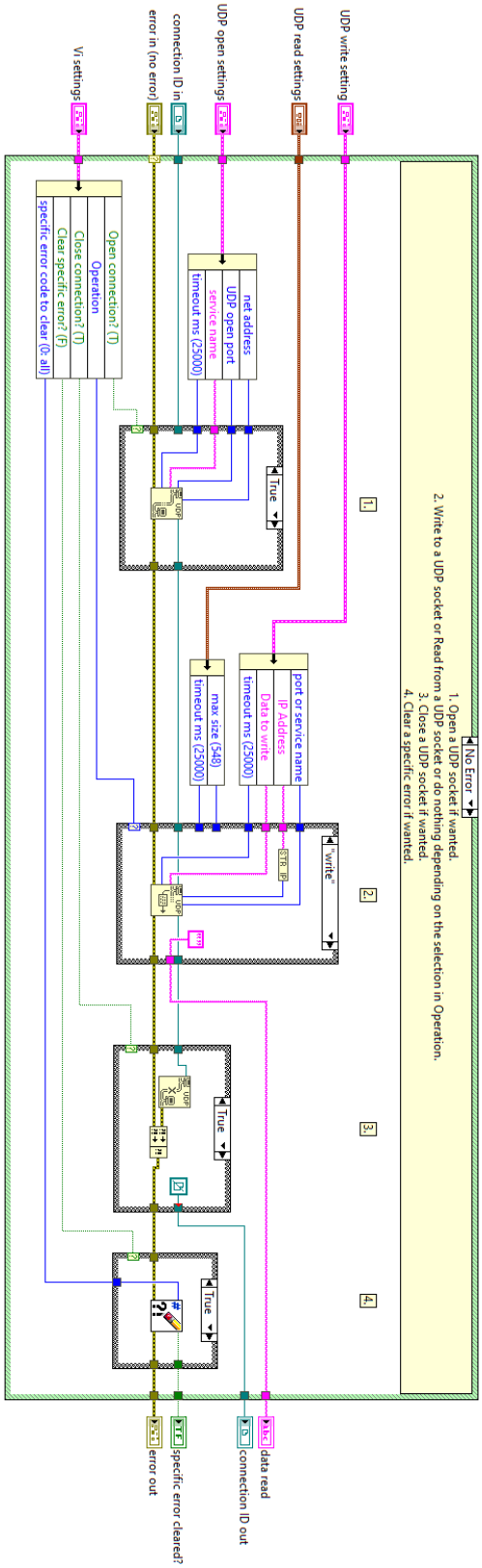
Kuva 1. Säteilyntuottotestin hakulogiikka.

Vuokaavio kestotestin hakulogiikasta



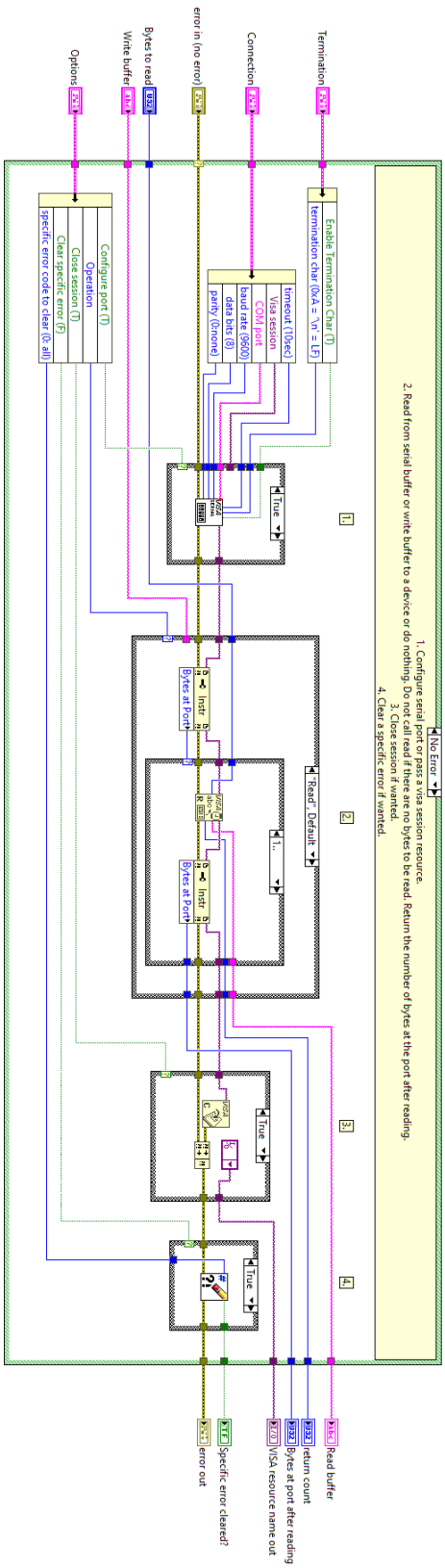
Kuva 1. Vuokaavio kestotestilokeroiden hakemisen logiikasta.

UDP kommunikaation VI



Kuva 1. UDP kommunikaatioon käytetty VI.

Sarjaliikenteen hallintaan käytetty VI



Kuva 1. Sarjaliikenteen hallintaan käytetty VI.