Matti Manninen

Hydraulisesti ohjatun servoakselin käyttöliittymän suunnittelu ja toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Kone- ja tuotantotekniikka

Insinöörityö

21.5.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Matti Manninen Hydraulisesti ohjatun servoakselin käyttöliittymän suunnittelu ja toteutus 39 sivua 21.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Kone- ja tuotantotekniikka
Suuntautumisvaihtoehto	Koneautomaatiotekniikka
Ohjaaja	Lehtori Heikki Paavilainen

Tämä opinnäytetyö tehtiin Metropolia Ammattikorkeakoululle. Työn tavoitteena oli luoda hydrauliselle servokytketylle sylinterille graafinen käyttöliittymä, jolla ohjaus mahdollistetaan. Työssä käydään läpi myös kyseisen toimilaitteen viritys hyödyntäen suunniteltua käyttöliittymää.

Logiikan ohjelmointiin on käytetty Beckhoff Automationin TwinCAT-ohjelmistopakettia. Tällä ohjelmistolla on laadittu toimilohkokaaviot ja koodi, jolla toimilaitteen ohjaaminen on mahdollista.

Opinnäytetyössä käydään ensin läpi työhön liittyvät keskeiset aihealueet tarkemmin, minkä jälkeen liikeohjauksen lainalaisuuksia sovelletaan askel askeleelta työn toteutuksen merkeissä.

Insinöörityön lopputuloksena syntyi vaivattomasti laajennettavissa oleva automaatioratkaisu helposti ymmärrettävällä käyttöliittymällä.

Avainsanat	Liikkeenohjaus, PLC, Beckhoff, servo-ohjaus

Author	Matti Manninen
Title	User Interface and tuning for a hydraulic servo-controlled cylinder
Number of Pages	39 pages
Date	21 May 2016
Degree	Bachelor of Engineering
Degree Programme	Mechanical Engineering
Specialisation option	Machine Automation
Instructor	Heikki Paavilainen, Senior Lecturer

This Bachelor's thesis was made in co-operation with Helsinki Metropolia University of Applied Sciences. The goal of the thesis was to program a logic for a hydraulic servo-controlled cylinder, which allows the cylinder to be moved. Secondly, the objective was to design a graphical user interface that is used to control the movement. The thesis also deals with tuning the actuator using the designed interface.

The program used to create this programmable logic is Beckhoff Automation's TwinCAT software suite. All of the function blocks and the code which enables the use of the actuator have been produced with this program.

The thesis examines the key topics of motion control before the actual implementation.

As a result of this thesis, an easily expendable automation solution with an intelligible user interface was created.

Keywords	Motion control, PLC, Beckhoff, Servo drive

Sisällys

Lyhenteet

PLC	*Programmable logic controller*, ohjelmoitava logiikkaohjain
PTP	*Point-to-Point*, pisteestä pisteeseen tapahtuva ohjaus
NC	*Numerical control*, numeerinen ohjaus
ST	*Structured text*, strukturoitu teksti, ohjelmointitapa
IL	*Instruction list*, käskylista, ohjelmointitapa
LD	*Ladder diagram*, tikapuukaavio, ohjelmointitapa
FBD	*Function block diagram*, toimilohkokaavio, ohjelmointitapa
SFC	*Sequential function chart*, sekvenssikaavio, ohjelmointitapa
CFC	*Continuous function chart*, Beckhoffin oma ohjelmointitapa
VISU	Visualisointi
POU	*Program organization unit*
GVL	*Global variable list*, globaali muuttujalista
ENC	Enkooderi
CTRL	Kontrolleri

1 Johdanto

Tämän insinöörityön aiheena oli suunnitella ja toteuttaa Beckhoff-automaatiojärjestelmään kytketylle toimilaitteelle tietokoneelta käytettävissä oleva ohjaus sekä ohjaukselle käyttöliittymä. Työn toteutus tapahtui kokonaisuudessaan automaatioon tarkoitetussa Beckhoffin TwinCAT-ohjelmointiympäristössä. Toimilaitteena toimi servo-ohjattu hydraulinen sylinteri.

Työ on tehty Metropolia Ammattikorkeakoulun toimeksiannosta ja kaikki tarvittava laitteisto sekä ohjattava toimilaite sijaitsivat Metropolian toimipisteessä. Tästä syystä työ on toteutettu kokonaisuudessaan oppilaitoksen tiloissa.

Tekstissä perehdytään aluksi liikeohjauksen historiaan, periaatteisiin ja teoriaan. Tämän jälkeen tutustutaan käytettyyn laitteistoon ja toimilaitteeseen sekä esitellään näiden ominaisuuksia. Seuraavaksi käydään läpi ohjelmiston suunnitteluprosessin vaiheet ja valotetaan ajatuksia valittujen suunnittelupäätösten takana. Ohjelmiston suunnitteluprosessin jälkeen työssä perehdytään vielä servo-ohjauksen virittämiseen ja tästä saataviin hyötyihin.

Suunnittelun tavoitteena oli toteuttaa selkeä ja helppokäyttöinen käyttöliittymä sylinterin ohjausta varten. Ohjauksen logiikkaa suunnitellessa otettiin huomioon funktionaalisuuden lisäksi laitteiston ohjaamisen turvallisuus sekä viritysmahdollisuudet.

2 Liikeohjaus

2.1 Historia

Ennen ensimmäisiä ohjelmoitavia PLC-logiikkakontrollereita yleinen tapa oli, että ohjaus tapahtui pelkästään ON/OFF-tyyppisillä releillä, jotka ohjasivat moottoreita. ON-asennossa oleva moottori pyörii ja OFF-asennossa taas ei. Useammilla moottoreilla ja ohjausreleiden kahden asennon yhdistelmillä ja yhdistelmien tarkoilla ajastuksilla mahdollistettiin hyvin alkeellinen liikeohjauksen muoto. (Hanssen 2015, 4.)

Yhtä konetta ajamaan tarvittiin useita moottoreita ja vielä useampia releitä. Ratkaisu oli tilaa vievä logistinen painajainen. Releet oli asennettava tietyssä järjestyksessä, ja ongelmatilanteissa, esimerkiksi releen rikkoontuessa, koko kone täytyi kokonaisuudessaan sulkea ja ongelman löytämiseen saattoi mennä hyvin pitkiä aikoja. Myös muutostyöt ja asennukset olivat todella haastavia toteuttaa, sillä koko järjestelmä piti käydä läpi liki alusta alkaen. (Segovia & Theorin 2013, 2 - 3.)

Vuonna 1968 tähän tuli muutos, kun Richard Morley kehitti General Motorsille *Modicon 084*:ksi nimetyn logiikkakontrollerin, jonka ensimmäisen mallin muistiin mahtui 125 sanaa ohjelmoitavaa logiikkaa. (Hendricks 2014.)

2.2 Standardit

Logiikkaohjauksen yleistyttyä tarpeeksi luotiin tätä säännöstelemään kansainvälinen standardi IEC 61131, jonka ensimmäinen painos julkaistiin nimellä IEC 1131 maaliskuussa 1993. Standardi koostuu kymmenestä osasta, joista jokaisessa määritellään tietyn osa-alueen pelisäännöt. (John & Tiegelkamp 2001, 14 - 16.)

Ensimmäisessä osassa on yleistä informaatiota standardista, toisessa kappaleessa annetaan laitteistoille vaatimuksia ja määritellään vaatimusten täyttämiselle testausmetodit. Kolmas kappale käsittelee ohjelmointikieliä, neljässä puolestaan esitellään käyttäjän ohjesääntöjä. Loput osat ovat lyhyitä, ja jokainen niistä sisältää syventävää tietoa logiikkaohjauksesta. (IEC 61131, 2003.)

Kansainvälisistä standardeista myös IEC 61499 käsittelee logiikkaohjausta, tällä kertaa toimilohkojen (*function block*) muodossa. Kyseinen standardi määrittelee ohjenuorat teollisuudessa käytettävien toimilohkojen käyttöä varten.

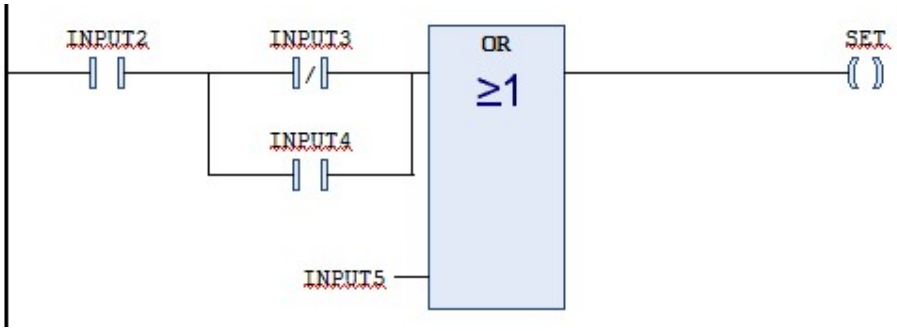
Standardi jakautuu kolmeen osaan, joista ensimmäisessä määritellään toimilohkojen toiminnollisuudet (*architecture*). Toisessa osassa toimilohkoja käsitellään standardia tukevien ohjelmien vaatimukset ja kolmannessa kerrotaan säännöt, joita IEC 61499:ä noudattavan laitteen on täytettävä.

2.3 Ohjelmointikielet

Keskeisin sisältö standardissa IEC 61131 löytyy osasta kolme, jossa määritellään PLC-ohjauksessa käytetyt ohjelmointikielet.

2.3.1 Ladder Diagram (LD)

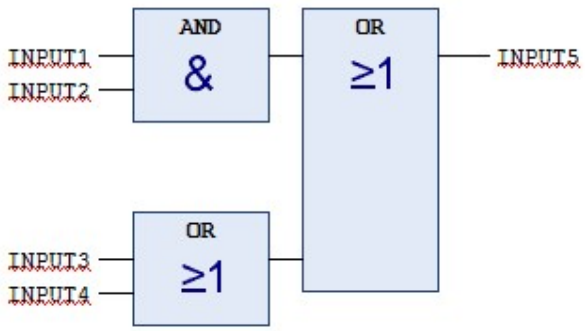
Ladder diagram on tutummalta nimeltään tikapuukaavio. Tämä muistuttaa toteutukseltaan paljon aiemmin mainittua releohjausta, jota käytettiin ennen kuin PLC-järjestelmät yleistyivät. Kuvassa 1 havainnollistettu tikapuukaavio on alkukantainen graafinen ohjelmointikieli, jonka tarkoitus oli olla releiden suunnittelijoille helposti ymmärrettävissä. (Beckhoff… 2016a.)



Kuva 1. Tyypillinen tikapuukaavio

2.3.2 Function Block Diagram (FBD)

Toimilohkokaavio on graafinen ohjelmointitapa, jossa logiikka rakennetaan erilaisista niin sanotuista toimilohkoista, joilla on kaikilla jonkinlainen funktio (kuva 2). Yksinkertaisimmillaan lohkoissa on sisääntuloja ja ulostuloja ja jonkinlainen looginen vaikutus (esim. *AND*- tai *OR*-rakenne). (Beckhoff… 2016a.) Hieman monimutkaisempia lohkoja voidaan käyttää sellaisenaan esimerkiksi liikkeenohjauksessa hyödyntäen TwinCATiin rakennettuja kirjastoja (esim. OpenPLC:n *Tc2_MC2*-toimilohkot).



Kuva 2. Perinteisimpiä toimilohkokaavion lohkoja

2.3.3 Structured Text (ST)

Strukturoitu teksti on ohjelmointikieli, jossa käytetään nykyisiä ohjelmointikieliä hyödyksi. Teksti kirjoitetaan normaalin koodin tapaan. Hyvänä esimerkkinä strukturoidusta tekstistä toimivat perinteiset ohjelmointilauserakenteet (kuva 3). Strukturoitu teksti antaa ohjelmoijalleen hyvin vapaat kädet ja täten helpon tavan luoda ohjelmia.

```
IF value > 1 THEN
    x:=y+a;
ELSE
    x:=y-b;
END_IF
```

Kuva 3. Yksinkertainen *IF*-/*ELSE*-rakenne

2.3.4 Instruction List (IL)

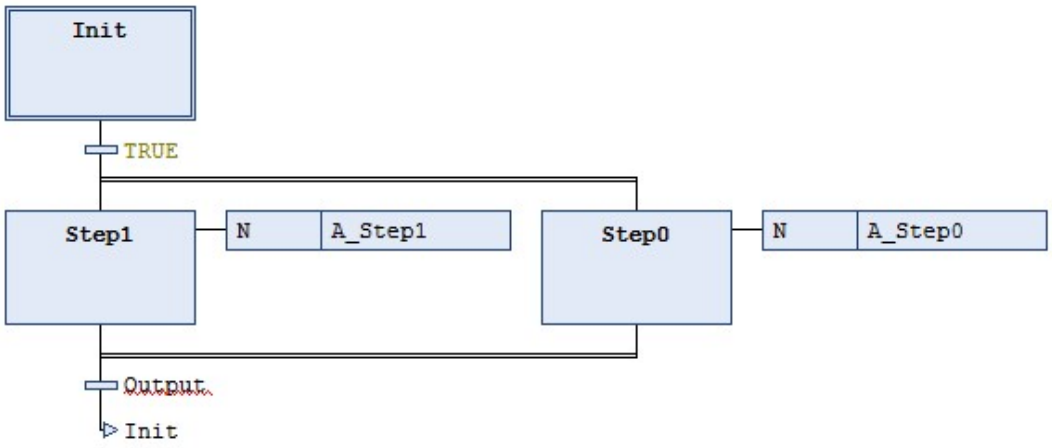
Käskylista muistuttaa hyvin paljon *assembly*-kieltä (kuva 4). Ohjelmointikielenä käskylista on nimensä mukaisesti listaus riveistä, joilla on yleensä käsky, viittaus kohdemuuttujaan tai vaihtoehtoisesti yksinkertainen esimerkiksi *AND*- tai *OR*-rakenne. (Beckhoff… 2016a.)

```
LD      3
MUL     2
ST      instlist
```

Kuva 4. Esimerkki *instruction list* -ohjelmointikielen rakenteesta

2.3.5 Sequential Function Chart (SFC)

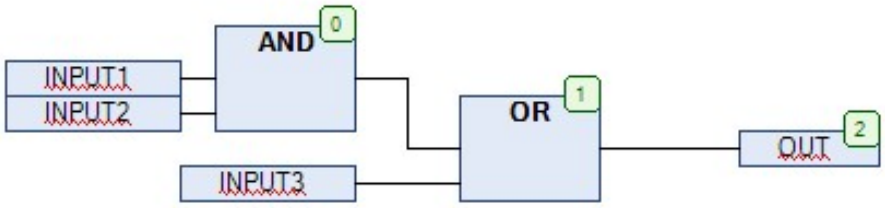
Sekvenssikaavio eli sekvenssiohjaus on ohjaustapa, jossa ohjelmointi tapahtuu askeleittain. Koodi koostuu pääasiassa askeleista ja ehdoista (kuva 5), joilla signaali välitetään seuraavalle askeleelle. Tässä ohjelmointitavassa askeleet vastaavat muiden ohjelmointitapojen toimintoja, joten signaalin pystyy esimerkiksi jakamaan ja useampi toiminto voi täyttyä yhtäaikaisesti. (Beckhoff… 2016a.)



Kuva 5. SFC:n rakenne

2.3.6 Continuous Function Chart (CFC)

Continuous function chart on Beckhoffin oma kieli, joka löytyy TwinCAT-ohjelmointiympäristöstä. Perusteiltaan tämä on hyvin samantapainen kieli kuin toimilohkokaavio, sillä erolla, että CFC:ssä lohkoille pystyy määrittämään numeraalisen suoritusprioriteetin, jonka voi nähdä kuvassa 6. Näin ohjelmoija pääsee käsiksi esimerkiksi komentojen suoritusjärjestykseen. (Beckhoff… 2016a.)



Kuva 6. Sinisten lohkojen yläkulmassa oleva vihreä numero symbolisoi jokaisen lohkon prioriteettia

2.4 PLCopen

PLCopen on itsenäinen organisaatio, jonka tarkoitus on olla johtava järjestö PLC:hen liittyvien pulmien ratkaisussa ja alan kansainvälisten standardien käytön tukemisessa. Järjestö perustettiin vuonna 1992, jolloin sen tarkoitus oli luoda standardisoituja tapoja toteuttaa automaatio-ohjelmointiratkaisuja. (PLCopen 2016.)

Organisaatio toimii yhteistyössä liki kaikkien suurten automaatio-ohjelmointijärjestelmiä tuottavien yhtiöiden kanssa, jotta se pystyisi tarjoamaan kaikkien yritysten ohjelmistoihin vastaavat, globaalisti standardisoidut vapaata lähdekoodia edustavat ohjelmointityökalut. PLCopenilla on valtava katalogi erilaisia komentoja ja kirjastoja liittyen ohjelmoitaviin logiikoihin. (Automation 2014.)

Tässä opinnäytetyössä käytetään useita järjestön tarjoamia PLC-kirjastoja, joista keskeisin työn kannalta on juuri liikeohjauksen komentoja sisältävä *Tc2_MC2*. Kyseinen kirjasto pitää sisällään ohjaukseen tarvittavat toimilohkot, jolloin lopullisesta liikeohjauslogiikasta tulee PLCopenin standardien mukainen ja samalla se noudattaa standardin IEC

61131 asettamia rajoituksia. PLCopenin toimilohkoja käyttämällä säästytään myös turhalta työltä, sillä vapaata lähdekoodia edustavat toimilohkot ovat teollisuudessa standardisoituja ja kaikkien käytettävissä.

2.5 Servo

Nimitystä servo käytetään ohjauspiiristä, jonka tarkoitus on ohjata servo-ohjattu laite tavoiteasemaansa hyödyntäen takaisinkytkentää asema-anturiinsa. Takaisinkytkentä mahdollistaa sen, että piiri pystyy vertaamaan annettua ohjearvoa anturin palauttamaan nykyiseen arvoon ja päättelemään tästä, kuinka suuri ero ohje- ja nykyisellä oloarvolla on. Mitä suurempi erotus on, sitä suuremmalla voimalla servo-ohjattu laite pyrkii kohti annettua ohjearvoa. (Bennett 1986, 1 - 7.)

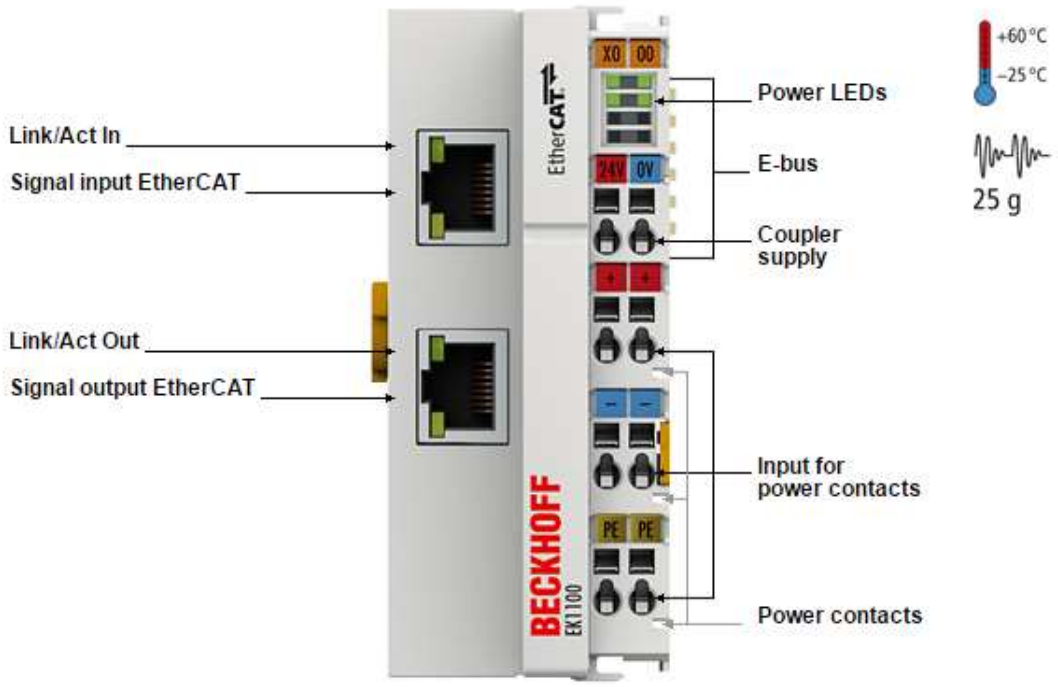
3 Laitteisto

3.1 Yleiskatsaus

Ohjauslaitteistoon kuului kannettava tietokone, virtalähde, pumppu, kytkin, virtalähde, hydraulinen servo-ohjattu toimilaite sekä analogiset sisään- ja ulostuloterminaalit.

3.1.1 Beckhoff EK1100

Ensimmäinen tietokoneeseen liitetty laite on Beckhoffin EK1100 EtherCAT-kytkinterminaali (*coupler*) (kuva 7), jonka käyttötarkoitus on saada kytkettyä muut laitteet EtherCAT-väylään laitteeseen rakennettujen Ethernet-liitäntöjen (*interface*) avulla. Laitteessa sisään- ja ulostulot on toteutettu EtherCATilla. Ulostuloon pystyy liittämään lisää EtherCATilla toimivia laitteita.



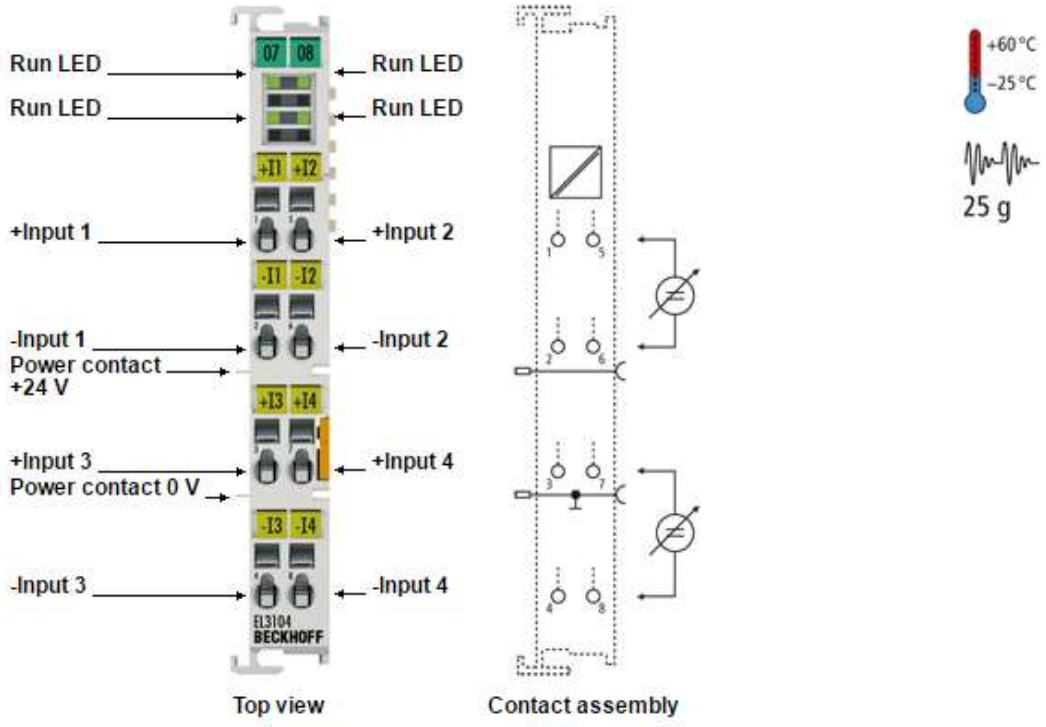
Kuva 7. Beckhoff EK1100:n liitännät (Beckhoff… 2016b)

Tämän lisäksi terminaalin avulla voidaan kytkeä muut toimilaitteet ulkoiseen virtalähteeseen, mikä mahdollistaa laitteiden toiminnan. Terminaali toimii myös konvertterina, joka

kääntää sisään tulevan (*input*) Ethernet-signaalin E-bus-muotoon, jota terminaalin perään kytketyt EtherCAT-laitteet ymmärtävät.

3.1.2 Beckhoff EL3104

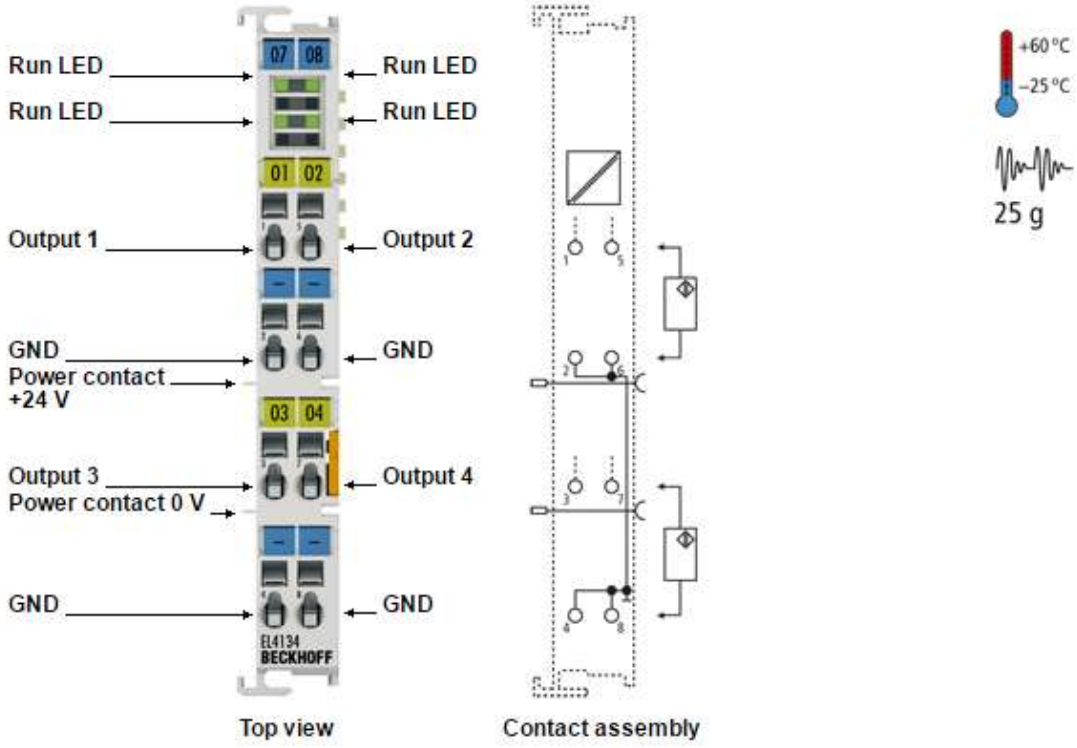
EL3104 on analoginen sisääntuloterminaali (kuva 8), joka pystyy prosessoimaan signaaleja väliltä +/- 10 VDC. Terminaalin resoluutio on 16 bittiä. Resoluutiosta pystytään suoraan päättelemään terminaalin erottelukyky, joka on tässä tapauksessa 2 potenssiin 16.



Kuva 8. Beckhoff EL3104:n liitännät (Beckhoff… 2016c)

3.1.3 Beckhoff EL4134

EL4134 (kuva 9) on vastaavasti analoginen ulostuloterminaali, joka pystyy luomaan signaaleja välillä +/- 10 VDC. Terminaalin resoluutio on 16 bittiä.



Kuva 9. Beckhoff EL4134:n liitännät (Beckhoff… 2016d)

3.2 Toimilaite

Työssä käytetty toimilaite on iskunpituudeltaan 300 mm pitkä servo-ohjattu hydraulinen sylinteri (kuva 10). Toimilaitetta ohjataan Beckhoffin EL4134 -ulostuloterminaalilta tulevalla jännitteellä.

Kuva 10. Insinöörityössä ohjattava toimilaite

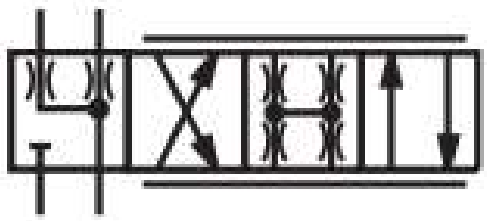
Toimilaitteen anturin toimintamalli vastaa toiminnallisuudeltaan perinteisesti lämmön mittauksessa käytettyä vastusanturia, jonka vastus kasvaa lämpötilan muuttuessa. Toimilaitteessa käytetyssä anturissa männän liike liikuttaa vastusanturin luistia. Männän ollessa sisäasennossa ulostulojännite on 0 VDC ja ulkoasennossa syöttöjännitteen suuruinen. Ulostulojännitteestä saadaan selville männän asema.

Terminaalin (EL4134) kanavasta 1 syötetään toimilaitteelle venttiilin ohjearvo. Terminaalin kanavasta 4 annetaan asema-anturille syöttöjännite muodossa +/- 10V DC. Vastaavasti terminaalin (EL3104) kanavaan 1 tulee venttiililtä karan aseman arvo muodossa +/- 10V DC. Terminaalin kanavaan 4 tulee sylinterin asema-anturin arvo.

3.3 Venttiili

Toimilaitteen ohjauksen venttiilinä toimii Bosch-Rexrothin 4WRPEH 6 C4B12L-10/G24K0/A1M, joka on nopean vasteajan suuntaproportionaaliventtiili ja jossa on integroitu proportionaalivahvistin. Venttiilin karan ohjaus perustuu asematakaisinkytkentään. Proportionaalivahvistin tarvitsee 24 VDC syöttöjännitteen. Venttiilin ohjearvona annetaan +/- 10 VDC ohjausjännite. Venttiilin karan asematieto saadaan +/- 10 VDC jännitteenä. (Rexroth 2009.)

Suuntaproportionaaliventtiilit on tarkoitettu ohjaamaan virtauksen suuntaa ja suuruutta. Tässä käytetty nopean vasteajan venttiili soveltuu pienten peittojen johdosta myös servojärjestelmään. Kuvassa 11 näkyy työssä käytetyn 4/4-suuntaventtiilin eri tilat. Ensimmäinen vasemmalta on niin sanottu *safe*-asema, johon venttiili menee häiriötilanteessa. Kolme oikeanpuoleista ovat normaalitilassa käytössä riippuen annetusta ohjausjännitteestä. Nolla-jännitteellä ollaan niistä keskimmäisessä asemassa (venttiili kiinni). Annetun ohjausjännitteen etumerkki määrää kumpaan suuntaan tästä asemasta liikutaan ja jännitteen suuruus kuinka paljon.



Kuva 11. Työssä käytetyn venttiilin hydrauliikkakaavio (Rexroth 2009)

Venttiilin mallista selviää, että kyseisen venttiilin nimellistilavuusvirta on 12 l/min, kun venttiilin sisällä oleva paine-ero (häviö) on 70 baria. Venttiilin läpi menevä tilavuusvirta riippuu syöttöpaineesta ja sylinterin kuormasta. Tässä sovellutuksessa sylinterin maksiminopeus on n. 250 mm/s (Rexroth 2009.)

3.4 EtherCAT-väylä

EtherCAT on Ethernet-pohjainen kenttäväyläsysteemi, jonka on kehittänyt Beckhoff Automation. Suunnittelun lähtökohtana oli saada aikaiseksi suosittuun Ethernet-pohjaan toteutettu pienen latenssin (<100 µs) ja vähäisen viiveen vaihtelun jaksollisessa signaalissa omaava kenttäväyläsysteemi. (EtherCAT Technology Group 2015.)

EtherCAT eroaa muista kenttäväyläratkaisuista siinä, miten se käsittelee tietoa. Yleensä *master*-yksikön lähettämä informaatiopaketti otetaan *slave*-yksikössä vastaan, tallennetaan ja luetaan, mutta EtherCATissa tiedon lukeminen tapahtuu ”lennosta” ilman, että dataa tarvitsee erikseen käsitellä laitteissa, joiden läpi tieto kulkee. Näin ollen *slave*-yksikkö voi lukea viestin ja kirjoittaa lisää tietoa (esimerkiksi anturin arvon) paketin perään.

Käytyään kaikissa *slave*-yksiköissä tieto palaa takaisin *master*-yksikölle, joka prosessoi saamansa paketit. Lennosta lukeminen madaltaa vasteaikaa huomattavasti, ja mahdollistaa esimerkiksi servon ohjaamisen todella suurilla, jopa 10 kHz:n, taajuuksilla sillä tieto servon takaisinkytkennästä saadaan takaisin vastaavasti todella nopeasti (Jost 2016).

4 TwinCAT

4.1 TwinCAT-ohjelmisto

TwinCAT (lyhenne sanoista *The Windows Control and Automation Technology*) on Beckhoff Automationin luoma automaatiokokonaisuus, johon sisältyy erilaisia PLC-, NC- ja CNC-pohjaisia ohjelmointiympäristöjä. Ohjelmisto on maailman johtava ratkaisu reaaliaikaiseen ohjelmoitavan logiikan luomiseen ja käyttämiseen. TwinCAT:lla on mahdollista tehdä lähes jokaisesta nykyaikaisesta Windows/Linux-pohjaisesta tietokoneesta ohjelmoitavan logiikan ajamiseen pystyvä työpiste. (Beckhoff… 2016.)

Beckhoff/TwinCAT-pohjaisesta automaatiosta tekee luotettavan se, että suoraan ohjelmiston kanssa yhteensopivat Beckhoffin automaatiomoduulit pystyvät tekemään työnsä, vaikka tietokone toimisi epävakaasti. Ohjelmisto pystyy kommunikoimaan tietokoneen rajapinnan kautta moduuleiden kanssa hyvinkin pienellä latenssilla, sillä ohjelmisto osaa priorisoida antamansa käskyt prosessorille ja jopa määrätä moniydinprosessorista dedikoituja ytimiä laskujensa suorittamiselle lisäten täten käyttövarmuutta. (Beckhoff… 2016.)

4.2 TwinCAT-ohjelmointi

Pähkinänkuoressa TwinCATilla ohjelmoiminen tapahtuu kappaleen 2.3 ohjelmointikieliä hyödyntäen. Ohjelmoinnissa voi myös käyttää kappaleessa 2.4 mainittuja PLCopenin kirjastoja.

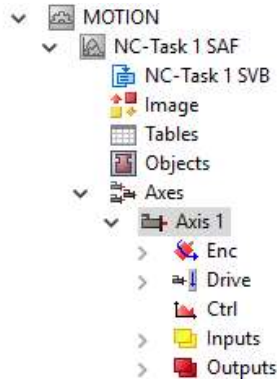
Tämän insinöörityön suunnitteluvaiheessa on käytetty ohjelmointikielistä toimilohkokaaviota (FBD), strukturoitua tekstiä (ST) ja PLCopenin liikkeenohjaustoimilohkoja sisältävää kirjastoa *Tc2_MC2* sekä suurinta osaa TwinCAT XAE 3.1:n mukana tulleista PLC-kirjastoista.

5 Suunnittelu

Logiikka suunniteltiin kohtalaisen virtaviivaiseksi yksinkertaisen suunnittelutavan mahdollistaman modulaarisuuden vuoksi. Ohjelmaa pystytään halutessa laajentamaan vaivattomasti ja siihen voidaan helposti luoda uusia ominaisuuksia.

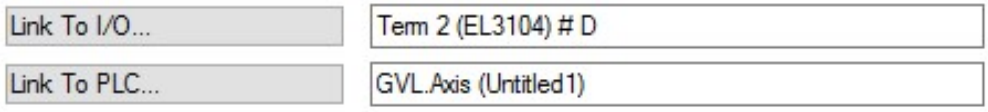
5.1 Liikkeen suunnittelu

Aluksi ohjelmaan täytyi luoda NC/PTP-liike (*motion*). Tämän tarkoituksena on saada ohjelmisto ymmärtämään, millaisia liikkeitä toimilaitteen käyttöön kuuluu. Työssä ohjataan yhtä servo-ohjattua sylinteriä, joten liikkeen alle luodaan projektipuussa yksittäinen kuvassa 12 näkyvä jatkuva akseli (*continuous axis*).



Kuva 12. Akseli projektipuussa

Jos akselille ei anneta minkäänlaisia parametreja, ei akselilla tee mitään. Tästä syystä akselin s*ettings*-välilehdelle on linkitettävä anturitieto toimilaitteelta, tässä tapauksessa terminaalin 2 kanava 1, josta akseli saa toimilaitteeseen asennetun asema-anturin paikkatiedon. Akseli on linkitettävä myös PLC:hen, jotta sitä pystytään ohjaamaan PLC:n puolella olevalla koodilla. Molemmat linkitykset kuvassa 13.



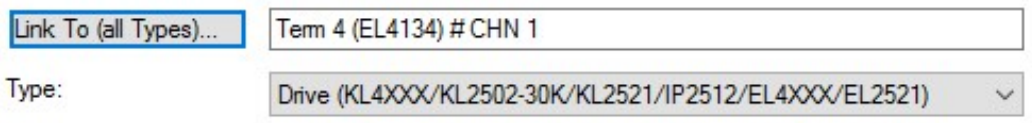
Kuva 13. Akselin I/O ja PLC linkitykset

Tässä vaiheessa ohjelmisto ei vielä tiedä, miten lukea paikkatietoa, jota se saa anturilta. Ohjelmistoon on määriteltävä enkooderi, joka pystyy kääntämään informaation ohjelmiston ymmärtämään muotoon. Tässä tapauksessa valitaan akselin enkooderivalikosta (*enc*) valinta ”KL30XX/KL31XX/EL30XX/EL31XX”, sillä sisääntuloterminaalina 2 toimii Beckhoffin EL3104. Enkooderille täytyy myös määrittää kerroin, jolla skaalata yksittäinen bitti sylinterin karan liikkumaan pituuteen. Kerroin saadaan selville jakamalla syliterin männänvarren kokonaismitta anturin miniminäytteenottotarkkuudella (*resolution*), joka on 15 bittiä. Laskusta 300 mm/2^{15} saadaan yhden bitin mittaamaksi pituudeksi 0,009155 mm (kuva 14). Jotta asema-arvo saadaan, tulee asema-anturille antaa 10 VDC syöttöjännite ulostuloterminaalin EL4134 kanavasta 4.

	Parameter	Offline Value
-	Encoder Evaluation:	
	Invert Encoder Counting Direction	FALSE
	Scaling Factor Numerator	0.009155

Kuva 14. Enkooderin parametreihin asetettava skaalauskerroin

Nyt kun ohjelma käsittää, mihin asemaan toimilaite on ajettu, voidaan akseli säätää myös antamaan toimilaitteelle liikkumakäskyjä. Tämä voidaan toteuttaa akselin ajurivalikosta (*drive*). Tähän valikkoon riittää, että linkkaa ajurin terminaalin 4 porttiin 1, jonka kautta toimilaitteelle syötetään venttiilin ohjausarvoa. Tyypiksi valitaan KL4XXX/KL2502-30K/KL2521/IP2512/EL4XXX/EL2521, koska ulostuloterminaalina toimii EL4134 (kuva 15).

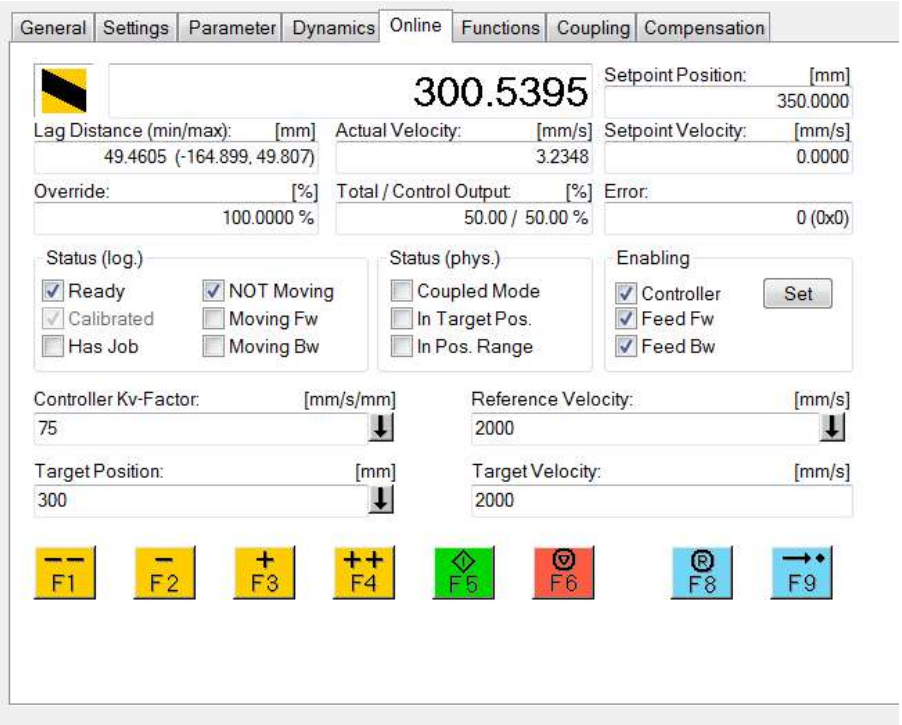


Kuva 15. Valittu ajurin tyyppi

5.2 Testaus

Toimilaitetta on mahdollista ajaa manuaalisesti hyödyntäen edellisessä kappaleessa luotua akselia. Manuaaliohjaus kuitenkin vaatii, että ohjelmaan on laitettu vastaavanlaiset parametrit. Ilman sisään-/ulostuloja tieto ei kulje toimilaitteen ja testilaitteiston välillä ja mitään ei tapahdu.

Käytännössä manuaaliohjaus tapahtuu seuraavasti: valitaan akselivalikon (*axis*) alapuolelta *online*-välilehti (kuva 16). Täältä löytää yksinkertaisimmat käskyt, joilla manuaaliajon voi suorittaa. Jos ohjelma on *config*-moodissa, kaikki napit ovat harmaita. Jotta napit saa käyttöön on järjestelmä käynnistettävä uudelleen *run*-moodiin. Moodin ollessa päällä on mahdollista kirjautua järjestelmään sisään ja ajaa toimilaitetta manuaalisesti.



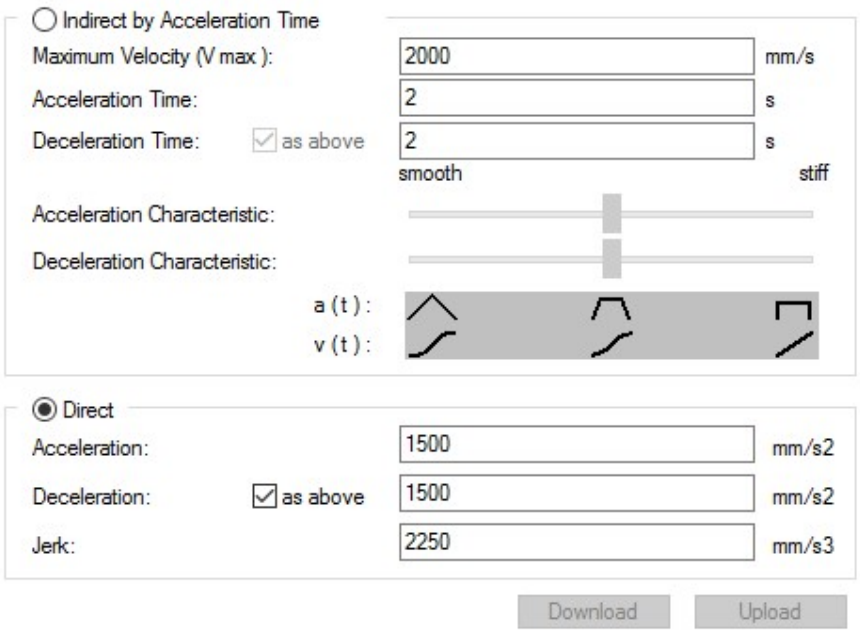
Kuva 16. Akselin *online*-välilehti manuaaliohjauksen yhteydessä. Toimilaitteen kara ajettuna maksimiarvoonsa

Järjestelmän ollessa toiminnassa on hyvä huomioida *enabling*-kohdan valinnat. Kyseisessä kohdassa on määriteltävä, että käyttäjä haluaa komennoillaan toimia toimilaitteen

kontrollerina (*controller*) ja että käyttäjä pystyy liikuttamaan toimilaitetta eteen- ja taaksepäin (*feed fw* ja *feed bw*). Kaikki toiminnot saa päälle painamalla *enabling*-kohdan nappia *set* ja tämän jälkeen avautuvasta ikkunasta nappia *all*.

Eteen- ja taaksepäin liikuttamisen lisäksi välilehdeltä käsin on mahdollista muokata esimerkiksi toimilaitteelle menevän signaalin vahvistusta ja toimilaitteen tekemien liikkeiden nopeutta sekä antaa ohjearvoja toimilaitteen antureille (esim. paikkatiedot).

Online-välilehdellä tapahtuvaa ohjausta on mahdollista hienosäätää myös *parameter*- ja *dynamics*-välilehdiltä. *Parameter*-välilehdellä pystyy vaikuttamaan muun muassa toimilaitteen kiihtyvyyksiin ja erilaisiin nopeusarvoihin. *Dynamics*-välilehti käsittelee nimensä mukaisesti toimilaitteen liikkeiden dynamiikkaa. Kyseiseltä välilehdeltä on mahdollista määrittää toimilaitteen kiihtyvyysarvoja ja näille erilaisia kiihtyvyysprofiileja (kuva 17). Näiden avulla määritetään, mitä profiilia seuraamalla laite kiihtyvyytensä saavuttaa.

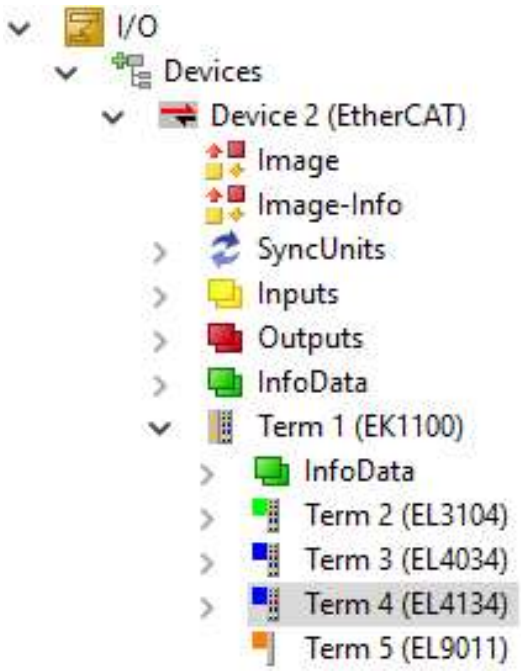


Kuva 17. *Dynamics*-välilehden säädöt

Ohjelmaa pystyy ohjaamaan hieman eri lailla myös *functions*-välilehdeltä, josta löytyy hyvänä esimerkkinä muun muassa absoluuttiseen asemaan ajaminen. Yksi hyödyllisimpiä työvälineitä välilehdellä on karan asematietojen määritys. Tällä voidaan ongelmatilanteessa esimerkiksi säätää toimilaitteen akselin nollakohta uudestaan.

5.3 Laitteet

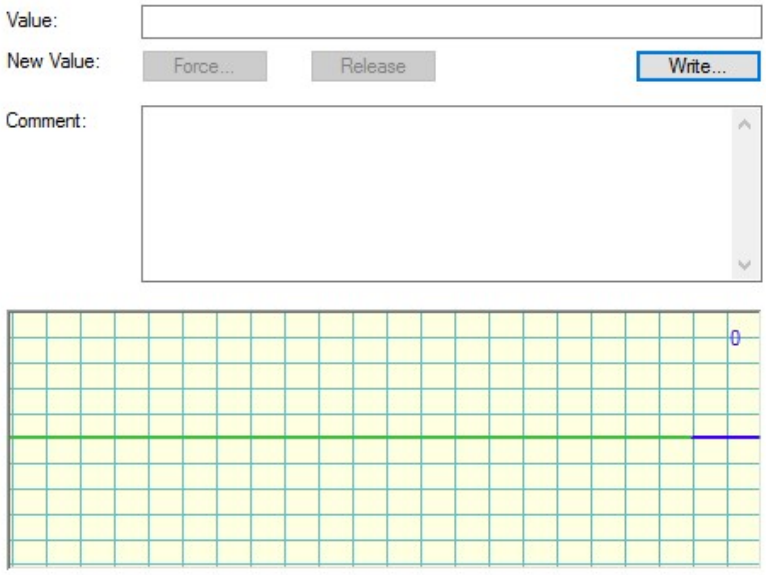
Beckhoff TwinCAT -ohjelmistolla laitteet voidaan ottaa käyttöön lisäämällä ne projektipuun kohdan I/O alle. Tämän alta löytyy valikko *devices*, johon on mahdollista etsiä uusia laitteita klikkaamalla valikkoa hiiren oikealla näppäimellä ja valitsemalla *scan*. Ohjelmiston pitäisi löytää EK1100-kytkinterminaali ja kysyä käyttäjältä, etsitäänkö siihen kytkettyjä terminaaleja (*scan for boxes?*). Kun valintaan on vastattu kyllä, ohjelmisto lisää kytkimen ja terminaalit automaattisesti projektipuuhun (kuva 18). Tänne voidaan linkittää terminaalien kanaville halutut arvot. Työssä käytetyt määritelmät kanaville on käyty läpi kappaleissa 2.2 ja 2.2.1.



Kuva 18. Projektipuuhun lisätyt terminaalit

Kytkentöjen ja toimilaitteen välisen toimivuuden voi testata esimerkiksi seuraavasti: avataan 4 terminaalin 4 kanava ja valitaan avautuneesta ikkunasta *online*-välilehti. Tänne on mahdollista kirjoittaa haluamiaan arvoja toimilaitteelle muodossa +/-10VDC (kuva 19).

Tämä menettelytapa mahdollistaa sylinterin ohjaamisen *config*-moodissa, jos jostain ongelmatilanteesta johtuen järjestelmä ei suostu käynnistämään itseään *run*-moodiin.

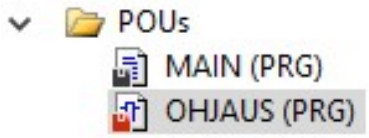


Kuva 19. *Write*-valinnalla toimilaitteelle voi syöttää manuaalisesti ohjausjännitettä

5.4 PLC

PLC on koko tämän insinöörityön keskeisin osio. Tällä mahdollistetaan toimilaitteen ohjaus koneelta käsin logiikan avulla. Ilman PLC:tä tietokoneelta tapahtuva ohjaus pitäisi päätoimisesti hoitaa täysin manuaalisesti eikä tämä yksinkertaisten toimilaitteiden kanssa palvele kenenkään tarkoitusperiä.

Aivan aluksi logiikan suunnittelussa on päätettävä, millä TwinCATin tarjoamilla ohjelmointikielillä suunnittelu tapahtuu. Logiikka tässä insinöörityössä päätettiin toteuttaa pääasiassa toimilohkokaavioilla (*FBD, function block diagram*) ja strukturoidulla tekstillä (*ST, structured text*) (kuva 20).



Kuva 20. Työssä käytetyn logiikan MAIN (ST)- ja toimilohkokaavio-ohjelmat

Mahdollisimman toimivan toimilohkokaaviologiikan mahdollistamiseksi on aluksi ensisijaisen tärkeää tietää, mitä lohkoja tulee käyttämään. Loogisin valinta oli käyttää PLCopenin kehittämää *Tc2_MC2*-liikkeenohjauskirjastoa. Kaikki ohjelmassa käytetyt toimilohkot ovat tästä kirjastosta.

Kirjaston lataamisen jälkeen on hyvä alkaa suunnitella ohjelmaa. Aluksi on syytä luoda kansioon GVLs uusi globaalit muuttujat sisältävä lista (*global variables*). Globaalit muuttujat ovat muuttujia, joita voidaan käyttää kaikkialla ohjelmassa, oli kyseessä sitten pääohjelma tai jokin lukuisista aliohjelmatyypeistä. *Global variablesin* alle on aluksi hyvä syöttää tieto siitä, mitä akselia ohjelmassa käytetään, ja myös esimerkiksi venttiilin ohjearvo tulisi luoda omaksi muuttujakseen (kuva 21). Erillinen ohjearvomuuttuja mahdollistaa kyseisen muuttujan linkittämisen terminaalin 4 kanavaan 4 ilman ylimääräistä työtä.

```
VAR_GLOBAL

Axis : AXIS_REF;
ANTURIN_SYOTTO AT %Q*:INT;

END_VAR
```

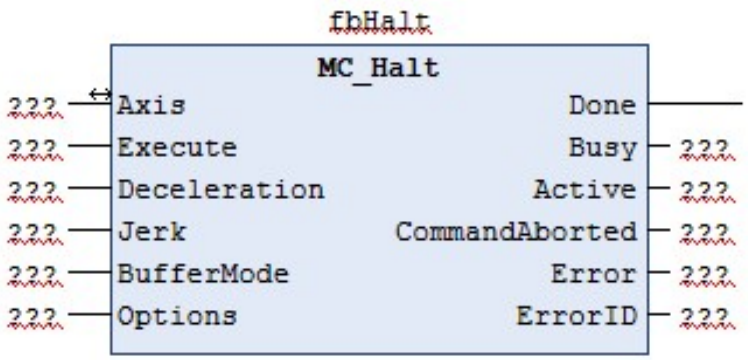
Kuva 21. Referenssiakselin määritys global variables-listaan, sekä anturin syöttöjännitemuuttujan määritys

Tämän insinöörityön pääohjelma (*MAIN*) jätettiin hyvin yksinkertaiseksi modulaarisuutta ajatellen. Ohjelmaa on käytetty ainoastaan akselin tilan reaaliaikaiseen lukemiseen sekä toimilohkokaaviologiikan sisältävän aliohjelman kutsumiseen. Pääohjelmaan pystyy kuitenkin lisäämään helposti lisää aliohjelmakutsuja, jos tällaiselle on tarvetta.

5.5 Toimilohkokaavio

Tämän PLC:n ytimenä toimii toimilohkokaavio, joka koostuu erilaisista funktion sisältävistä lohkoista. PLCopenin *Tc2_MC2* -kirjasto helpottaa suunnitteluprosessia huomattavasti, sillä kyseinen kirjasto tarjoaa juuri käyttötarkoitusta palvelevat toimilohkot.

Tc2_MC2:n toimilohkot toimivat osittain samalla tavalla keskenään. Kaikkiin lohkoihin määritellään akseli, jota kukin lohko ohjaa, ja kaikkia lohkoja ohjataan lohkoon vasemmalta puolelta tulevilla inputeilla (kuva 22). Vastaavasti lohkon oikealta puolelta saadaan output. Yhteisiä valintoja on myös esimerkiksi *BufferMode*-sisääntulo, jonka arvolla pystytään määräämään, kuinka ohjelma reagoi useaan peräkkäin tulevaan komentoon.

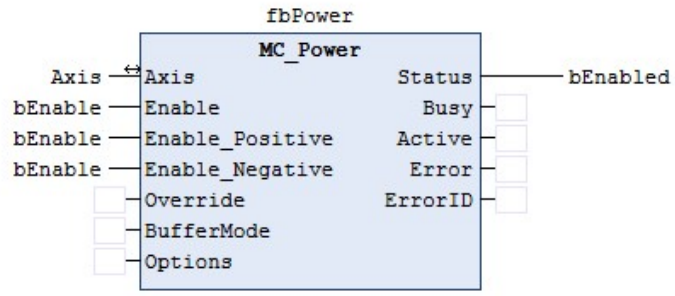


Kuva 22. Havainnollistava kuva toimilohkosta, johon ei ole määritelty sisään- tai ulostuloja

Kunkin sisään- ja ulostulon vaatiman muuttujatyypin saa selville käyttämällä hiirtä kyseisen tulon päällä. Yleisimmät muuttujatyypit *Tc2_MC2*-toimilohkoille ovat *BOOL* (0 tai 1), *LREAL* (8 bitin luku) tai *INT* (kokonaisluku). Sisään- ja ulostuloille voidaan tehdä kaikki tavallisimmat toimenpiteet mitä normaaleille toimilohkoillekin. Näihin voi esimerkiksi määrittää negaation, joka muuttaa *BOOL*-arvon päinvastaiseksi tai vaikka liittää uuden toimilohkon.

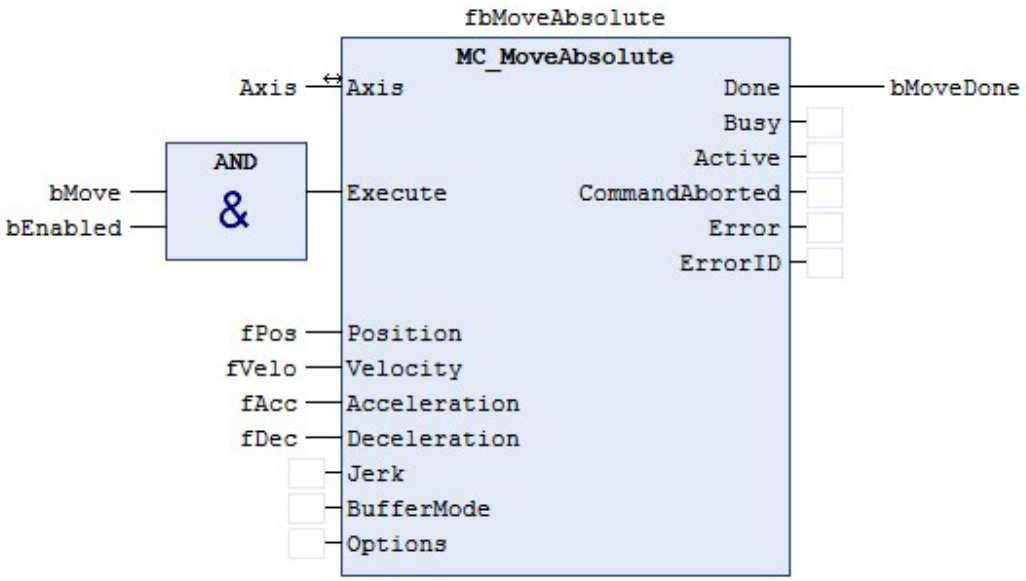
Muuttujat syötetään suoraan toimilohkojen yläpuolelle kohtaan *PROGRAM* tai vaihtoehtoisesti ne voidaan kirjoittaa myös suoraan toimilohkon tuloon/lähtöön, minkä jälkeen ohjelma kysyy muuttujalle tyypin ja hyväksymisen yhteydessä lisää itse muuttujat kohtaan *PROGRAM*.

Logiikka on rakennettu viiden toimilohkon varaan. Ensimmäinen lohkoista *MC_Power* (kuva 23) käynnistää ohjelman saadessaan sisääntuloonsa *enable*-arvon *TRUE* (*BOOL*=1) ja lähettää ulostulostaan *status*-arvon *TRUE*. Päinvastaisesti, jos e*nable* -sisääntuloon ei tule arvoa, ohjelma ei käynnisty.



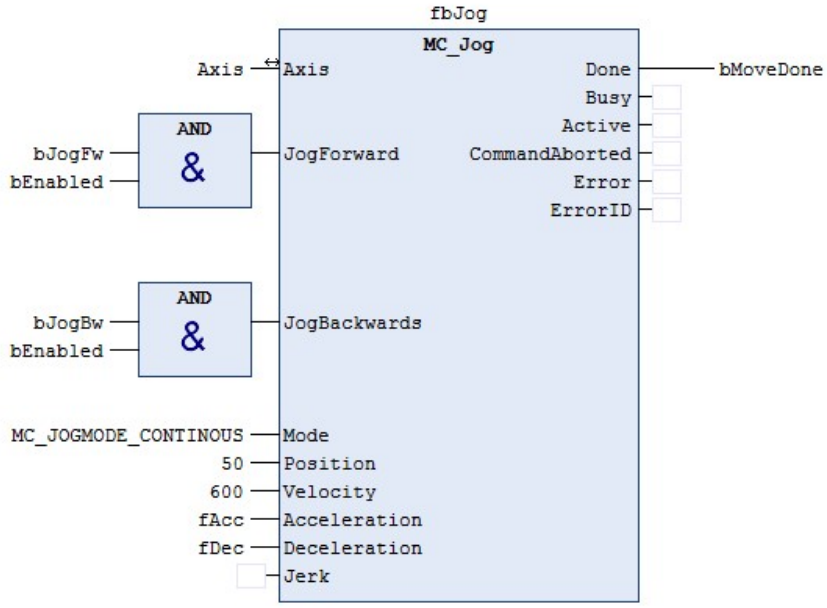
Kuva 23. Käynnistyksestä huolehtiva toimilohko MC_Power

Toinen toimilohkoista on *MC_MoveAbsolute* (kuva 24). Kyseistä toimilohkoa käytetään siten, että lohkolle annetaan paikka- ja nopeusarvot (*position* ja *velocity*). Tämän jälkeen, jos *MC_Powerilta* saadaan tieto siitä, että laite on päällä, voidaan toimilaite ajaa *execute*-sisääntulolla määriteltyyn asemaan määritellyllä nopeudella.



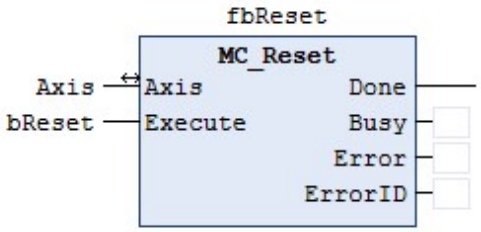
Kuva 24. Toimilaitteen tiettyyn paikkaan tietyllä nopeudella ajava toimilohko *MC_MoveAbsolute*

Seuraava toimilohkoista on *MC_Jog* (kuva 25), joka on myös toimilaitetta liikuttava toimilohko. *MC_MoveAbsolutesta MC_*Jog eroaa siten, että kyseinen lohko liikuttaa toimilaitteen sylinteriä määrätyn pituuden kerrallaan, kun toimilaitteelle annetaan liikkumiskäsky. Tämä voi tarkoittaa esimerkiksi sitä, että kun toimilohkolle tulee tieto, että sen tulisi liikuttaa toimilaitetta, toimilaite liikkuu ennalta määritetyn matkan eteen- tai taaksepäin komennosta riippuen.



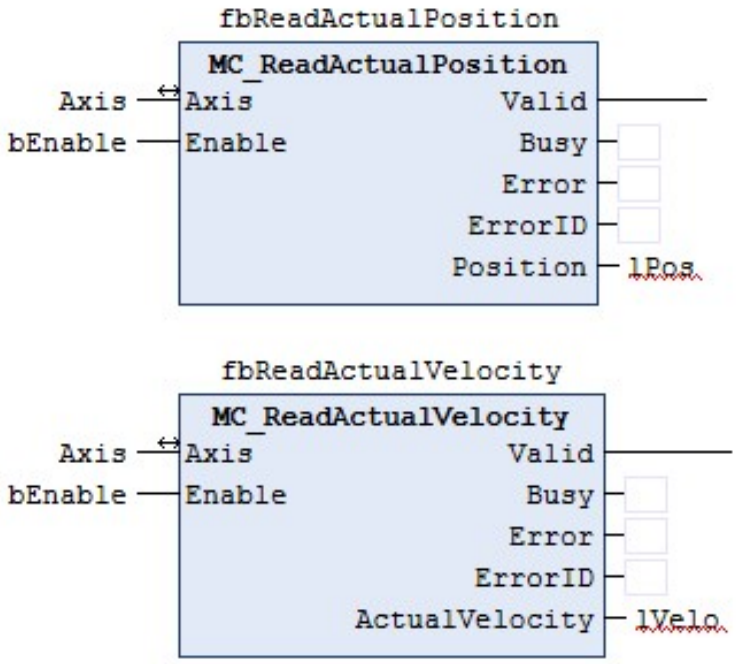
Kuva 25. Tietyn matkan kerrallaan toimilaitetta ajava toimilohko *MC_Jog*

MC_Reset (kuva 26) on nimensä mukaisesti lohko, jonka tarkoitus on sisääntulon aktivoituessa nollata kaikki muut ohjelmalla olevat arvot takaisin lähtöarvoonsa.



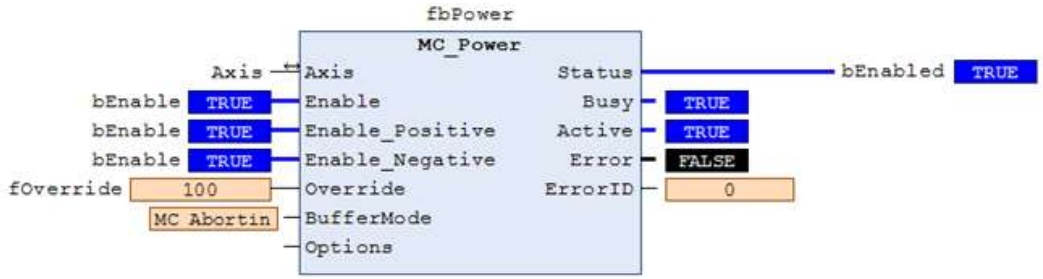
Kuva 26. Ohjelman resetoinnista vastaava toimilohko *MC_Reset*

Viimeiset kaksi toimilohkoa *MC_ReadActualPosition* ja *MC_ReadActualVelocity* (kuva 27) ovat toiminnaltaan hyvin samankaltaisia. Toimilohkojen tarkoitus on yksinkertaisesti ilmoittaa alimmasta ulostulostaan (*position/actualvelocity*) reaaliaikainen paikkatieto, jonka ne saavat toimilaitteelta toiminnassa ollessaan. Tätä käytetään hyödyksi linkkaamalla kyseinen tieto visualisointiin, jotta paikan ja nopeuden reaaliaikainen tarkkailu on mahdollista sieltä käsin.



Kuva 27. Reaaliaikaiset tiedot akselilta ilmoittavat toimilohkot

Ajon aikana kunkin muuttujan tila käy ilmi toimilohkosta (kuva 28). *TRUE*-arvo tarkoittaa, että kyseinen sisään- tai ulostulo on aktiivinen ja *FALSE* taas, että kyseinen liitäntä ei ole käytössä. Myös omavalintaisesti säädettävät arvot, kuten kuvan 28 *fOverride*, jonka arvoa säädetään visualisoinnissa olevasta liukukytkimestä, näkyvät ajon aikana reaaliajassa.

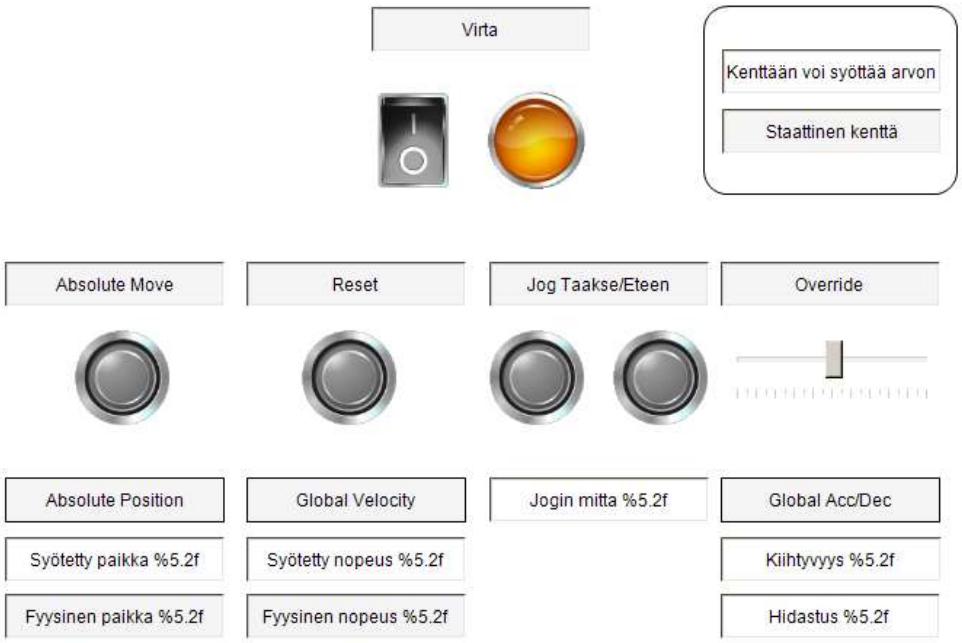


Kuva 28. Toimilohko ajon aikana

5.6 Visualisointi

Visualisoinnin tekeminen on helppo tapa luoda toimilohkoille graafinen käyttöliittymä. Visualisoinnin luominen tapahtuu luomalla kansion VISUs alle ”*Visualization*”-ohjelma. Luotuun aliohjelmaan visualisoinnin luominen tapahtuu yksinkertaisesti raahaamalla hiirellä halutut ohjausvälineet kohdasta *toolbox* visualisoinnin pääruutuun. Tiputettaessa ohjausväline pääruutuun, valittu väline ilmestyy ruudulle. Väline voi olla esimerkiksi nappi, liukusäädin, merkkivalo tai tekstiruutu.

Tässä työssä on käytetty pääasiassa painonappeja, joita painamalla saadaan lähetettyä käsky suoraan logiikkaohjelmaan, ja tekstilaatikoita, joilla voidaan esittää tekstiä tai syöttää tietoa ohjelmalle (kuva 29). Tekstilaatikoilla voidaan syöttää esimerkiksi paikka- ja nopeustiedot toimilohkoille. Laatikoissa esiintyvä %5.2f-merkintä tarkoittaa sitä, että laatikko näyttää itseensä määritellyn muuttujan muodossa *LREAL* ja näyttää kyseisestä luvusta viisi kokonaislukua ja kaksi desimaalia.



Kuva 29. Toimilaitteen ajamista ajatellen suunniteltu käyttöliittymä

Visualisointi on suunniteltu siten, että jokaiselle valkoiselle kentälle voi klikkaamalla syöttää arvon. Syöttämisen jälkeen arvo jää näkyviin kenttään. Harmaat laatikot ovat staattisia, mutta esimerkiksi nopeutta ja paikkaa esittävät laatikot päivittyvät reaaliajassa kuvan 27 toimilohkoilta saamallaan informaatiolla.

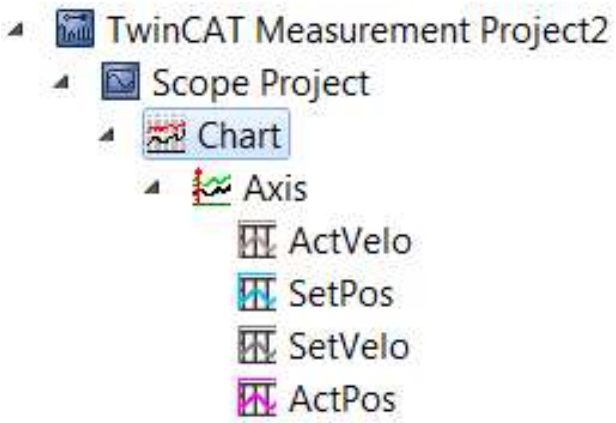
Kuvassa 30 näkyy visualisointi ajon aikana. Kaikki napit ovat näpäytettäviä lukuun ottamatta ylärivin virtanappia, joka jää päälle painettaessa. Oikean reunan liukukytkimellä säädetään ohjauksen *override*-arvoa välillä 0 - 100 %. Jos tämä säätö on suurimmillaan, annetut ohjearvot menevät perille sellaisenaan, muuten niitä rajoitetaan määritellyn prosenttiluvun mukaisesti.

Kuva 30. VISU ajon aikana. Valkoisista laatikoista näkyy niihin syötetty tieto ja alarivin harmaista laatikoista vastaavasti tämänhetkinen oikea tieto

6 Viritys

6.1 Scope

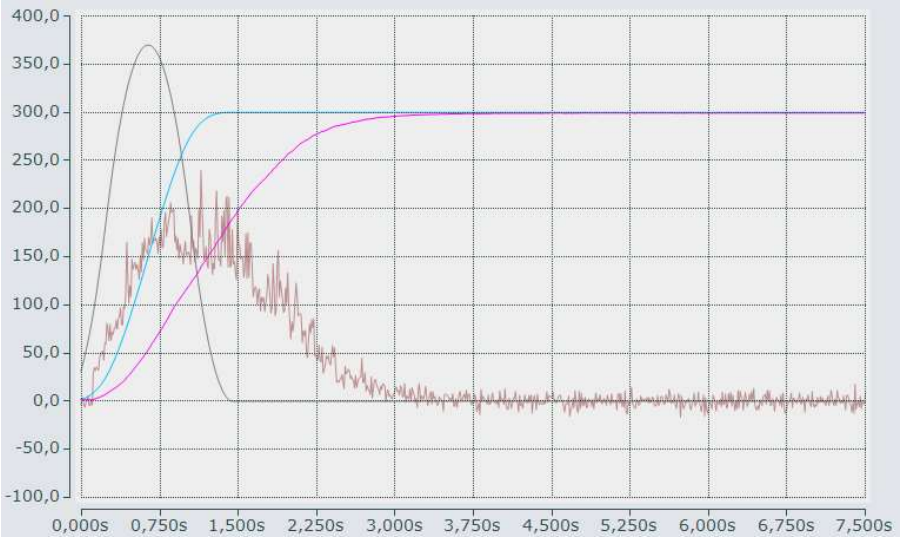
Virityksen tarkastelu tapahtuu TwinCATin *scope*-projektin avulla. Aluksi ohjausta varten tehtyyn projektipuuhun täytyy lisätä uusi projekti (File->Add->New Project->TwinCAT Measurement Project->Scope YT NC Project). Projektin alle pitää luoda halutut akselit (kuva 31) klikkaamalla kohtaa a*xis* oikealla hiiren näppäimellä ja valitsemalla *target browser*. Avautuvasta ikkunasta on mahdollista lisätä halutut arvot projektiin. Valituista arvoista *ActVelo* on toimilaitteen sylinterin oikea nopeus, *SetVelo* ohjenopeus. Vastaavasti *ActPos* on oikea paikkatieto ja *SetPos* tämän ohjearvo.



Kuva 31. Scope ja siihen luodut arvot

Luodut arvot on tarkoitus saada näkymään kuvaajaan. Tämä tapahtuu siten, että saatetaan järjestelmä ensin *run*-tilaan ja laitetaan se käyntiin. Seuraavaksi painetaan *record*-painiketta käyttöliittymästä löytyvästä TwinCAT *measurement*-palkista (vakiona yläreunassa).

Nyt kun järjestelmä on käynnissä, tulostaa *scope* ruudulle halutut arvot graafiseen muotoon (kuva 32). Nauhoituksen voi keskeyttää milloin vain, ja tulokset tämän jälkeen tallentaa kiintolevylle.



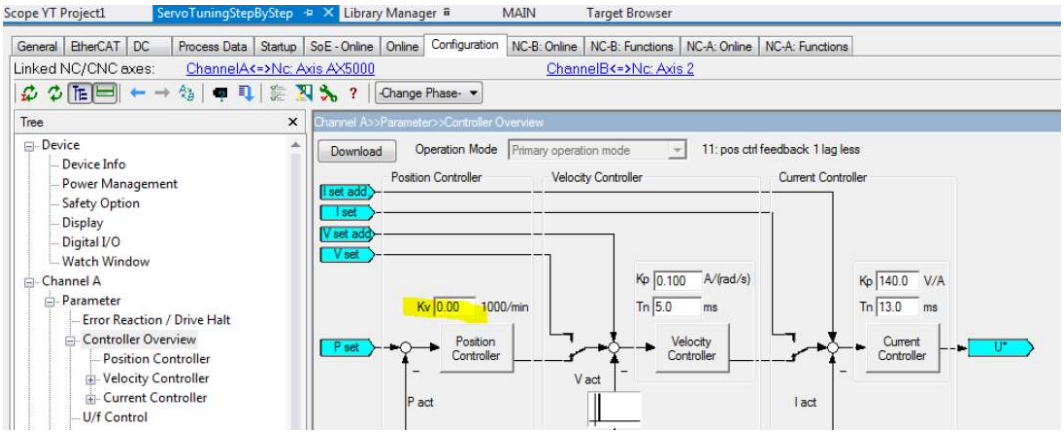
Kuva 32. Kun *scope* luodaan oikein, lopputuloksen pitäisi näyttää vastaavanlaiselta

6.2 Periaatteet

Servon viritys tapahtuu TwinCAT-ohjelmistosta käsin. Virityksen tarkoitus on saada servon paikka-, nopeus- ja kiihtyvyysarvot oikeita arvoja vastaaviksi sekä saada ohjaus toimimaan annettujen arvojen mukaisesti.

Jos TwinCAT tunnistaa servomoottorin, se ilmestyy *devices*-valikkoon sen terminaalin alle, johon se on kytketty. Tämä on yleisin tapaus, vaikka se eroaa hieman tässä opinnäytetyössä käytetyn servon virittämisestä.

Virityksestä suurin osa tapahtuu akselin kontrollerin valikon (*ctrl*) alapuolelta välilehdellä *configuration*. Välilehti ja sen säädöt löytyvät kuvasta 33, josta käyvät ilmi myös erilaiset tavat virittää servoa. Säätöä pystytään tekemään paikka-, nopeus ja virtakontrollerien avulla. Yleensä säätö tehdään näistä kahdella ensimmäisellä, ja virtakontrolleri onkin hyvä jättää rauhaan sen virityksen vaativuuden vuoksi.



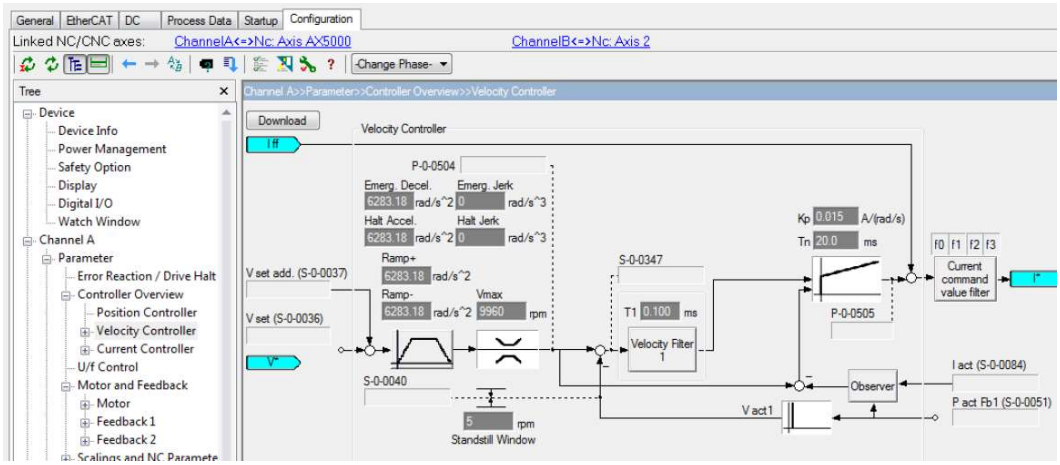
Kuva 33. Servon virityksessä auttava *configuration*-välilehti (Airto 2016)

Ennen säädön aloittamista on hyvä käydä laittamassa akselin kontrollerin kohta p*osition lag monitoring* pois päältä. Kyseinen kohta tarkkailee ohjauksessa olevaa viivettä, ja ennen servon täydellistä viritystä systeemissä on ajoittain viivettä paljonkin. Myös akselin *online*-välilehdeltä on hyvä käydä ottamassa kontrolleri pois päältä, jotta akselia pystytään ajamaan muualta käsin.

Paikkakontrollerin viritys tapahtuu pääasiassa *Kv*-arvoa muutamalla. Kyseinen arvo vastaa siitä, kuinka paljon vahvistusta paikan ohjesignaaliin annetaan. Kyseinen arvo on vakiona 1, mutta sitä muuttamalla saadaan servo ajamaan paikkaansa nopeammin.

Servon nopeuden viritys tapahtuu klikkaamalla ensin kuvan 33 ikkunassa kohtaa *velocity controller*. Seuraavassa ikkunassa (kuva 34) saadaan käyttöön useita parametreja, joilla nopeuden viritystä voidaan säätää. Tärkeimmät arvot, joita ikkunassa virityksen yhteydessä yleensä säädetään, ovat *Kp*-, *Tn*- ja *T1*-arvot.

Kp-arvo vastaa paikkaohjauksen *Kv*-arvoa eli tätä säätämällä päästään käsiksi nopeussäädön vahvistuskertoimeen. *Tn*-arvo on puolestaan käyrän integrointiaika mikä tarkoittaa, että tätä arvoa säätämällä saadaan kuvaajassa olevia kulmia hienosäädettyä. Arvo vaikuttaa myös siihen, kuinka hanakasti akseli alkaa hakeutumaan ohjearvoonsa, jos akseli ajaa esimerkiksi liiallisen nopeuden takia ohi ohjearvostaan. *T1*-arvolla suodatetaan servon säädölle annetut ohjausarvot ja tällä servo saadaan arvosta riippuen toimimaan sulavammin.

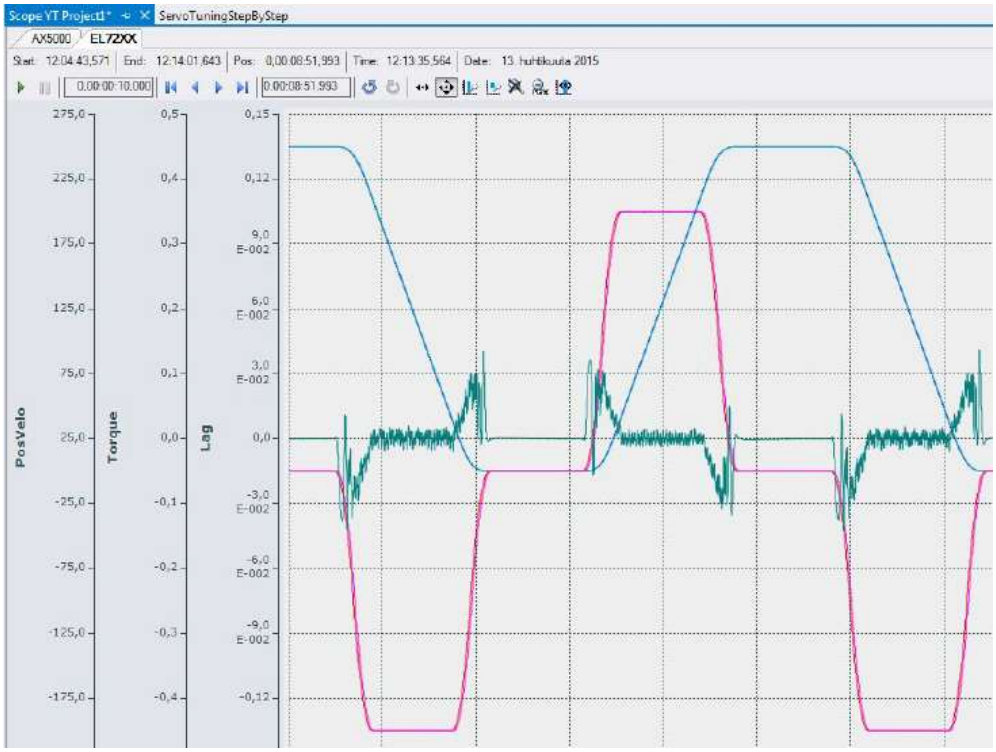


Kuva 34. Nopeuden virityksestä vastaava ikkuna (Airto 2016)

Paikan vahvistuksen (*Kv*) arvoa ei kannata mielivaltaisesti muuttaa suuremmaksi, vaan viritys tapahtuu helpoiten seuraavasti: Nostetaan nopeuskontrollerin *Kp*-vahvistusarvoa pikkuhiljaa, samalla pitäen *Tn*- ja *T1*-arvot nollassa. Tätä jatketaan kunnes ohjattava akseli saavuttaa halutun nopeuden, kunnes akseli alkaa oskilloimaan (nähdään jos *scopeen* laitetaan myös toimilaitteessa tapahtuva vääntö [*torque*]) tai siinä tapauksessa, kun tulokset eivät enää parane. Jos akseli alkaa oskilloimaan, on *Kp*-arvoa hyvä laskea vähintään 20 %. (Airto 2016.)

Kp-arvoa säätämällä ohjaus pitäisi optimaalisesti saada ajamaan ohjearvosta hieman ohi (10 - 20 %). *Tn*-arvoa muuttamalla pystytään vaikuttamaan siihen, kuinka voimakkaasti tämän tapahtuessa arvo alkaa ajaa akselia kohti ohjearvoa. Kun *Tn*-arvo säädetään tarpeeksi suureksi, ei ohjaus aja akselia yli ohjearvon, vaan *Tn*-arvo vain ajaa ak-

selin ohjearvoon mahdollisimman nopeasti. Jos systeemissä on nähtävissä paljon häiriöitä (*noise*), voidaan nämä hävittää nostamalla *T1*-arvoa. Lopullisen kuvaajan pitäisi näyttää vastaavalta kuin kuvassa 35. (Airto 2016.)

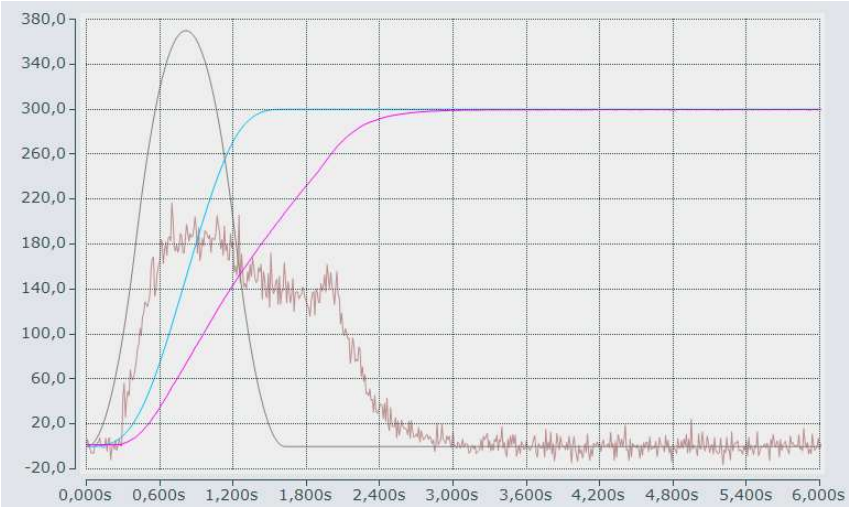


Kuva 35. Kuvaajan pitäisi näyttää tältä, kun viritys on tehty oikein ja ohjaus tottelee ohjearvoa hyvin (Airto 2016)

6.3 Virittäminen

Työssä käytetty toimilaite ei ole TwinCATin yhteensopiva servomoottori, vaan sitä ohjataan erillisellä proportionaalivahvistimella. Tästä syystä kyseinen laite ei ilmesty laitteiden skannauksen yhteydessä *devices*-kansioon projektipuuhun. Tämä tekee käytännön virittämisestä jonkin verran haasteellisempaa, sillä esimerkiksi edellisessä kappaleessa esimerkin omaisesti läpikäytyä *controller overview* -valikkoa ei toimilaitteelle löydy.

Viritys päätettiin toteuttaa pelkästään paikkakontrollerin vahvistuskerrointa, nopeutta ja kiihtyvyyttä säätäen. Kuvassa 36 näkyvä ajo on tapahtunut kuvan 16 *dynamics*-välilehden kiihtyvyysarvoilla (*acceleration/deceleration* 2s).

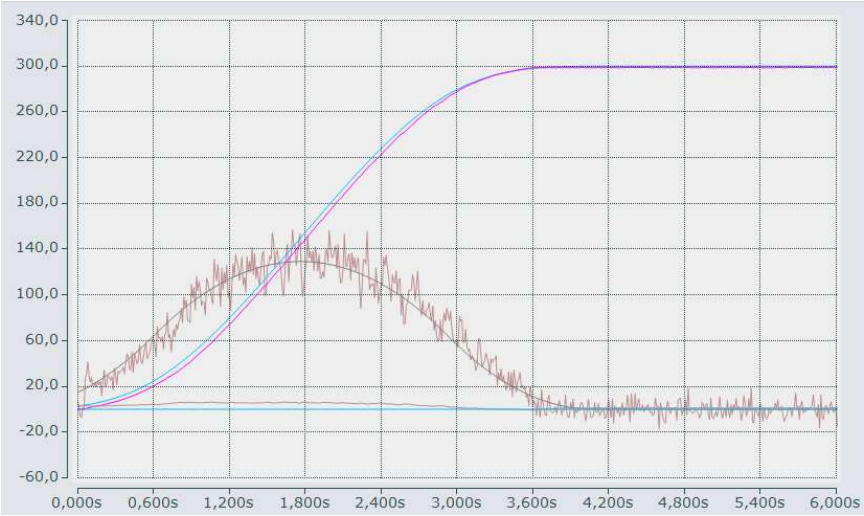


Kuva 36. *Kv*-arvo 10, kiihtyvyydet vakiot

Edellisen kuvan 35 ajosta huomattiin, että ohjaus tapahtuu aivan liian hitaasti. Tämän jälkeen *Kv*-arvoa nostettiin huomattavasti ja tarkasteltiin mitä ohjaukselle tapahtuu (kuva 37).

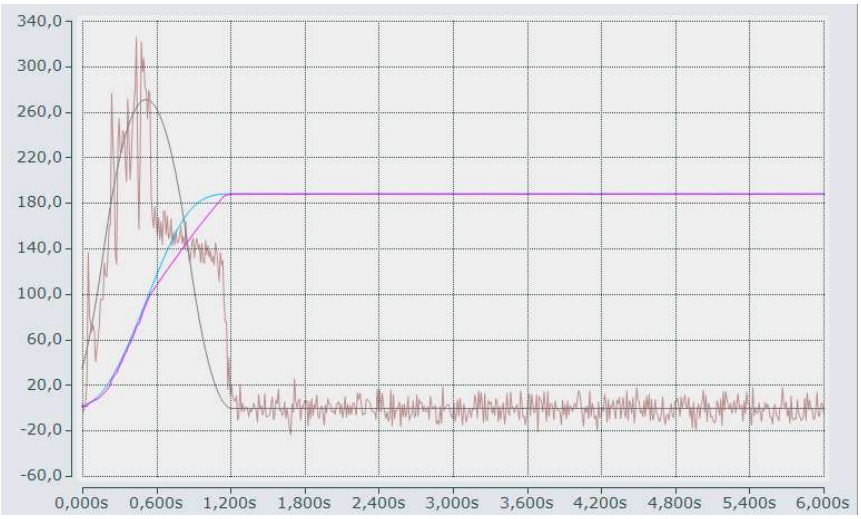
Kuva 37. *Kv*-arvo 75, kiihtyvyydet vakiot

Ohjaus suoriutuu suuremmalla *Kv*-arvolla jo paljon paremmin. Ohjaus ei tästä paljon muutu, vaikka *Kv*-arvoa kasvattaisi lisää. Seuraavaksi kiihtyvyyttä säädettiin hieman pienemmäksi, ja katsottiin uudelleen sen vaikutusta ohjaukseen (kuva 38).



Kuva 38. *Kv*-arvo 75, kaksi kertaa hitaammilla kiihtyvyysarvoilla

Nyt antureilta saatu tieto vastaa hyvin annettua ohjearvoa. Ohjaus on hieman hitaampi, mutta paljon tarkempi. Kuvassa 39 kokeillaan testiä uudestaan uusilla kiihtyvyysarvoilla ja isommalla *Kv*-arvolla ja seurataan miten muutos vaikuttaa ohjaukseen. Samalla testataan myös, onnistuuko toimilaite pysähtymään kesken ajon. Aiemmissa esimerkeissä toimilaite on ajettu aina maksimiarvoonsa.



Kuva 39. Ajo asemaan 150 mm, *Kv* 250, nopeudet samat kuin edellä

Esimerkistä voi huomata, että suurempi *Kv*-arvo aiheuttaa vain poikkeamaa ohjearvosta, joten näin suurista *Kv*-luvuista ei sinänsä ole mitään hyötyä. Toimilaite saatiin parhaiten viritettyä ohjearvoa seuraavaksi antamalla sille hieman maltillisemmat kiihtyvyysarvot ja hillitymmät *Kv*-arvot.

7 Päätelmät

Työn tavoitteena oli luoda yksinkertainen, helposti laajennettavissa oleva logiikkapohja. Tässä onnistuttiin kohtuullisen vaivattomasti TwinCAT -ohjelmiston helppokäyttöisyyden ansiosta. Lopputulos vastaa annettuja tavoitteita enemmän kuin hyvin.

Toimilaitteen ohjaaminen on helppoa ja ohjelmaan pystyy kuka tahansa asiaan perehtynyt henkilö lisäämään halutunlaisia ominaisuuksia yksinkertaisesti. Helppo laajennettavuus on tärkeää, jos laitteistoa halutaan myöhemmin muuttaa tai ohjaukseen halutaan lisäominaisuuksia.

Työssä käytetty ohjelmisto vaati ylimääräistä opiskelua Hyvinkäällä Beckhoff Automation Oy:n tiloissa. Yrityksen järjestämällä kurssilla opiskeltiin pääasiassa TwinCATin liikeohjausta, josta oli hyvin paljon hyötyä opinnäytetyöhön liittyen. Tälle kurssille ilmoittauduttiin alun perin siitä yksinkertaisesta syystä, että liikeohjauksen suunnittelu oli tarpeellista suorittaa mahdollisimman ammattitaitoisesti.

Vaikka toimilaite ei ollut pienten vikojensa takia helpoin laite työskentelyä varten, saatiin sen ohjauksesta tarkka ja toimiva. Lopullinen tuote täytti asetetut tavoitteet täysin, ja lopullista ohjausta on mahdollista käyttää useisiin eri sovelluksiin.

Lähteet

Airto Antti 2016: Servo Comissioning. [kurssimateriaali] [Viittauspäivä 25.4.2016.]

Automation 2014. Tietoa PLCopenista: [verkkodokumentti]. Saatavissa: <http://www.automation.com/automation-news/industry/plcopen-and-opc-foundation-define-function-blocks-for-iec-61131-3>. [Viittauspäivä 20.3.2016.]

Beckhoff Automation Oy 2016a. Ohjelmointikielet. [verkkodokumentti]. Saatavissa: <https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/tcplcctrl_languages%20ld.htm&id=>. [Viittauspäivä 10.3.2016.]

Beckhoff Automation Oy 2016b: Kuva EK1100:sta. Saatavissa: <https://www.beckhoff.com/english.asp?ethercat/ek1100.htm> [Viittauspäivä 22.4.2016.]

Beckhoff Automation Oy 2016c: Kuva EL3104:stä. Saatavissa: <https://www.beckhoff.com/english.asp?ethercat/el3104.htm> [Viittauspäivä 22.4.2016.]

Beckhoff Automation Oy 2016d: Kuva EL4134:stä. Saatavissa: <https://www.beckhoff.com/english.asp?ethercat/el4134.htm> [Viittauspäivä 22.4.2016.]

Bennet Stuart 1986: A History of Control Engineering. Peter Peregrinus LTD. United Kingdom.

EtherCat Technology Group 2015: Tietoa EtherCatista. [verkkodokumentti]. Saatavissa: <https://www.ethercat.org/en/technology.html> [Viittauspäivä 15.4.2016.]

Hanssen Dag H. 2015: Programmable Logic Controllers. John Wiley & Sons, Ltd. United Kingdom

Hendricks Howard 2014: The History of the PLC [verkkodokumentti]. Saatavissa: <http://www.barn.org/FILES/historyofplc.html>. [Viittauspäivä 10.3.2016.]

IEC 61131 2007: Programmable controllers. Saatavissa: <https://webstore.iec.ch/publication/4550>. [Viittauspäivä 17.5.2016.]

John Karl-Heinz & Tiegelkamp Michael 2001: Programming Industrial Automation Systems. Springer-Verlag. Germany.

Jost Michael 2016: Tietoa Ethercatista. [presentaatio]. Saatavissa: <http://www.beckhoff.com/english/highlights/ethercat/presentation.htm > [Viittauspäivä 15.4.2016.]

PLCopen 2016. Tietoa PLCopenista: [verkkodokumentti]. Saatavissa: <http://www.plcopen.org/>. [Viittauspäivä 20.3.2016.]

Rexroth 2009: 4/4-way servo solenoid directional control valves, directly operated, with electrical position feedback and on-board electronics (OBE) [verkkodokumentti]. Saatavissa: <http://www.airlinehyd.com/Images/Hydraulic/RexrothBosch/PDF/Industrial_Hydraulics_0910/IndustrialProducts/PropValves/High-Response/re29035_2009-02.pdf > [Viittauspäivä: 5.5.2016]

Segovia Vanessa Romero & Theorin Alfred 2013: History of Control, History of PLC and DCS [verkkodokumentti]. Saatavissa: <http://www.control.lth.se/media/Education/DoctorateProgram/2012/HistoryOfControl/Vanessa_Alfred_report.pdf>. [Viittauspäivä 10.3.2016.]