Teuvo Ruotsalainen

Hydraulisylinterin asennonohjauksen mallinnus ja toteutus PWM-ohjauksella CODESYS 3.5 -ympäristössä

Opinnäytetyö

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

2022

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Teuvo Ruotsalainen
Työn nimi	Hydraulisylinterin asennonohjauksen mallinnus ja toteutus PWM-ohjauksella CODESYS 3.5 -ympäristössä
Vuosi	2022
Sivut	37 sivua, liitteitä 9 sivua
Työn ohjaaja(t)	Teemu Manninen

TIIVISTELMÄ

Tutkimustyön tavoitteena oli saada aikaan dokumentoitu ohjelmamoduuli CODESYS 3.5 -ympäristöön, jota voisi käyttää pohjana erilaisten ohjainlaitteiden säätökäyrien muokkaamiseen. Tutkimustyön aikana käytettiin testiympäristössä nk. peukalopyörää lineaarisen toimilaitteen ohjaamisessa. Lineaarinen toimilaite mallinsi tässä tilanteessa hydraulisylinteriä, jolle pyrittiin löytämään peukalopyörältä hyvä ohjaustuntuma logiikan sallimin ohjelmoinnin keinoin.

Suuri osa tutkimustyöstä kohdistui matemaattisten ratkaisujen suunnitteluun. Tavoitetta lähestyttiin ensin teoreettisesti pyrkien löytämään sopivat kuvaajat (matemaattiset funktiot) ja yhdistelmäkuvaaja tavoiteltua tarkoitusta varten.

Perustutkimustyön jälkeen kehitetty malli sovitettiin MATLAB-ohjelman avulla toimiviksi yhtälöiksi, jotka mahdollistivat keskeisten säätökäyrään vaikuttavien parametrien muuttamisen halutuiksi. Lopuksi kehitettyjen matemaattisten yhtälöiden pohjalta toteutettiin tarkoitukseen sopiva ohjelma CODESYS 3.5 -ympäristössä ST-ohjelmointikielellä. Lopputuotoksena rakennettu ohjelma testattiin käyttämällä hyväksi olemassa olevia testiympäristön laitteita.

Opinnäytetyössä kuvataan myös pääpiirteittäin tutkimustyössä käytettyjen tekniikoiden ja laitteiden perusteita. Siinä kuvataan pulssinleveysmodulointi-tekniikkaa, H-silta-mekanismin periaate, peukalopyörän signaalin ominaisuuksia ja PWM-venttiilin tyypillinen ominaiskäyrä.

Opinnäytetyön tuloksiin voi olla erittäin tyytyväinen, koska tutkimustyön aikana syntynyttä matemaattista ideaa, ja sen pohjalta ST-ohjelmointikielellä toteutettua ohjelmaa on mahdollista soveltaa käytäntöön tulevina kuukausina työtehtäviin liittyvissä kehityshankkeissa.

Asiasanat: automaatio, logiikkaohjelmointi, CODESYS 3.5

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Engineering
Author (authors)	Teuvo Ruotsalainen
Thesis title	Modelling of hydraulic cylinder position control and implementation with PWM control in CODESYS 3.5 environment
Time	2022
Pages	37 pages, 9 pages of appendices
Supervisor	Teemu Manninen

ABSTRACT

The objective of this thesis was to build in CODESYS 3.5 environment a documented program module, which could be used as a basis for modifying the control curves of various control devices. During the research, a so-called thumbwheel was used in the test environment to control the linear actuator. In this situation, the linear actuator modelled a hydraulic cylinder for where the aim was to find a good steering feel on the thumbwheel by means of programming allowed by logic.

Much of the research focused on the design of mathematical solutions. The goal was first approached theoretically with the aim of finding suitable graphs (mathematical functions) and a composite graph for the intended purpose.

The model developed after the basic research was adapted into equations that work with the help of the MATLAB program, which made it possible to change the key parameters influencing the control curve as desired. Finally, on the basis of the developed mathematical equations, a suitable program was implemented in the CODESYS 3.5 environment in the ST programming language. The final program was tested using existing test environment equipment.

The thesis also outlines the basics of the techniques and equipment used in the research. It describes the pulse width modulation technique, the principle of the H-bridge mechanism, the characteristics of the thumbwheel signal, and the typical characteristic curve of a PWM valve.

We can be very satisfied with the results of my thesis, because it is possible to apply the mathematical idea generated during the research and the program implemented in the ST programming language to development projects related to my work in the coming months.

Keywords: Automation, PLC programming, CODESYS 3.5

South-Eastern Finland University of Applied Sciences

KÄSITTEET

A/D-muunnin	(Engl. Analog-to-Digital converter) Analogia-Digitaalimuunnin on elektroninen laite tai piiri, joka muuntaa jatkuvan analogisen signaalin arvoja digitaaliseksi binäärisignaaliksi, ja sitä kautta binääriluvuiksi.
CODESYS	(Engl. COntroller DEvelopment SYStem) Ohjelmoitavan logiikan ja sulautettujen järjestelmien ohjelmointiin tarkoitettu kansainvälisen IEC 61131-3 -standardin mukainen ohjelmointiympäristö, joka sisältää viisi ko. standardin mukaista ohjelmointitapaa.
FB	(Engl. Function Block) Toimintolohko (ohjelmointiin liittyvä), joka voi koostua tulomuuttujista, lähtömuuttujista, läpimuuttujista, sisäisistä muuttujista ja toimintolohkon sisäisen toiminnon kuvauksesta.
FET	(Engl. Field Effect Transistor) Kanavatransistori on transistori, jossa varauksenkuljettajien liikettä ohjataan puolijohdekanavassa vaikuttavan sähkökentän avulla. Kanavatransisitorin päätyypit ovat metallioksidi-puolijohdekanavatransistori (MOSFET) ja liitoskanavatransistori (JFET).
PLC	(Engl. Programmable Logic Controller) Ohjelmoitava logiikka on yleensä automatisoiduissa kohteissa käytetty tietokone, joka sisältää sähköisiä tuloja ja lähtöjä. Ohjelmoitavan logiikan tehtävänä on hallita automaatiojärjestelmää, lukea antureita ja ohjata siihen liitettyjen laitteiden toimintaa. Ohjelmoitava logiikka sisältää ohjelman tai ohjelmiston, jonka perusteella se suorittaa määriteltyjä tehtäviä.
PWM	(Engl. Pulse Width Modulation) Pulssinleveysmodulaatio on modulointitapa, jossa jännitteen vakiotaajuisen ja kanttiaaltoisen pulssin suhdetta muutetaan sähköistä kuormaa vasten niin, että vaikuttavan sähkövirran tehollisarvo muuttuu pulssin leveyden kanssa samassa suhteessa.
ST	(Engl. Structured Text) Rakenteinen teksti on yksi IEC 63131-3 -standardissa määritellyistä ohjelmointikielistä. ST-ohjelmointikieltä ja sen murteita käytetään nykyään laajasti ohjelmoitavien logiikoiden ohjelmien kehittämisessä. ST-ohjelmointikieli muistuttaa jonkin verran Pascal- ja C-ohjelmointikieliä.

SISÄLLYS

LIITTEET

1 JOHDANTO

Suorat hydrauliset ohjaukset ovat tänä päivänä vielä hyvin yleisiä liikkuvissa työkoneissa, kuten esim. kaivinkoneissa ja niihin rinnastettavissa alustakoneissa. Turvallisuusvaatimukset, käytännöt ja lakiin perustuvat vaatimukset ovat kuitenkin lisänneet automaation tarvetta liikkuvissa työkoneissa, ja samalla tilanne on johtanut siihen, että niiden ohjauksia on alettu toteuttaa lisääntyvissä määrin myös tietokoneen (esim. ohjelmoitavan logiikan) välittämäksi.

Liikkuvan työkoneen ohjaukseen ohjelmoitavan logiikan avulla liittyy paljon osakokonaisuuksia, joista turvallisuus on yksi merkittävimmistä, ellei merkittävin. Toinen liikkuvan työkoneen ohjaukseen liittyvä osa-alue on käyttömukavuus ja ohjauksen vasteen toteutuminen niin, että käyttäjällä on hyvä tuntuma ohjattavaan laitteeseen hänen sitä operoidessa. Tarkoitukseni on syventyä tässä tutkimuksessa juuri liikkuvien työkoneiden käytettävyyden ja hallittavuuden parantamiseen käytäntöön sovellettavan esimerkin avulla.

Työssä käytetään niin kutsuttua peukalopyörää lineaarisen toimilaitteen ohjaamisessa. Lineaarinen toimilaite mallintaa tässä tilanteessa hydraulisylinteriä, jolle on pyritty tuomaan peukalopyörältä hyvä ohjaustuntuma logiikan sallimin ohjelmoinnin keinoin. Työssä huomioidaan myös yleisen nk. PWM-venttiilin ominaiskäyrän vaikutus.

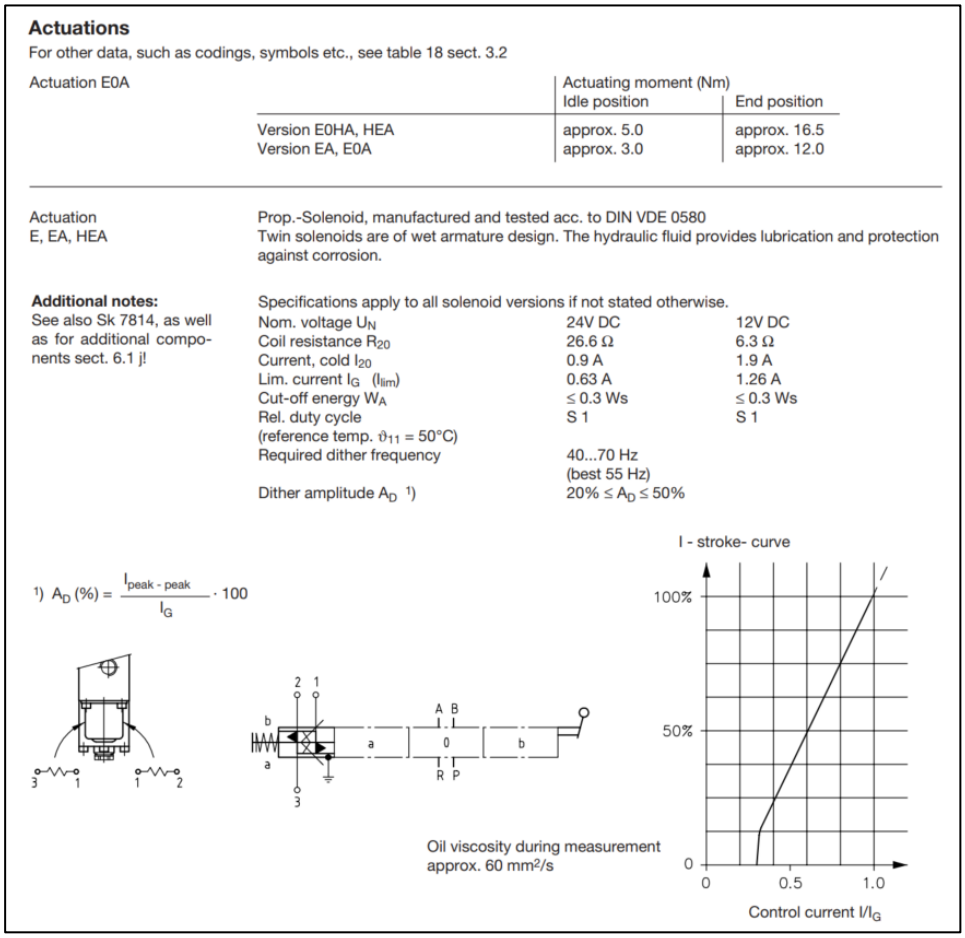
Tavoitetta lähestytään ensin teoreettisesti pyrkien löytämään sopivat kuvaajat (matemaattiset funktiot) haluttua tarkoitusta varten. Sen jälkeen sitä mallinnetaan MATLAB -ohjelmassa, jossa ohjaukseen vaikuttavia tekijöitä on pyritty ottamaan huomioon lisäämällä niiden vaikutustekijöitä yhtälöihin. Tämä jälkeen saatu teoreettinen pohja on toteutettu ohjelmallisesti ST-ohjelmointikielellä CODESYS 3.5 ympäristössä. Lopuksi ohjelman toimivuus testataan laboratorioympäristön laitteilla. Tekstissä kuvataan myös H-siltaominaisuuden toimintaperiaate, jota tarvittiin testilaitteiston yhteydessä lineaaritoimilaitteen tasavirtamoottorin napaisuuden kääntämiseen.

2 KOHDELAITTEISIIN LIITTYVÄ TEKNIIKKA

Tässä luvussa kuvataan kohdelaitteiden ominaisuuksia ja toimintaperiaatteita, sekä havainnollistetaan pulssinleveysmodulaatiotekniikan perusteita.

2.1 PWM-venttiilin sähköiset parametrit

PWM-venttiilillä tarkoitetaan yleensä proportionaalista hydrauliventtiiliä, Tällaista proportionaaliventtiiliä ohjataan sähköisesti pulssinleveysmodulaatio-signaalilla (PWM). Kuvassa 1 on ote erään PWM-venttiilin sähköisistä parametreista (Proportional directional spool valve type PSVF 1998, 14).



Kuva 1. PWM-venttiilin sähköiset parametrit (Proportional directional spool valve type PSVF 1998, 14)

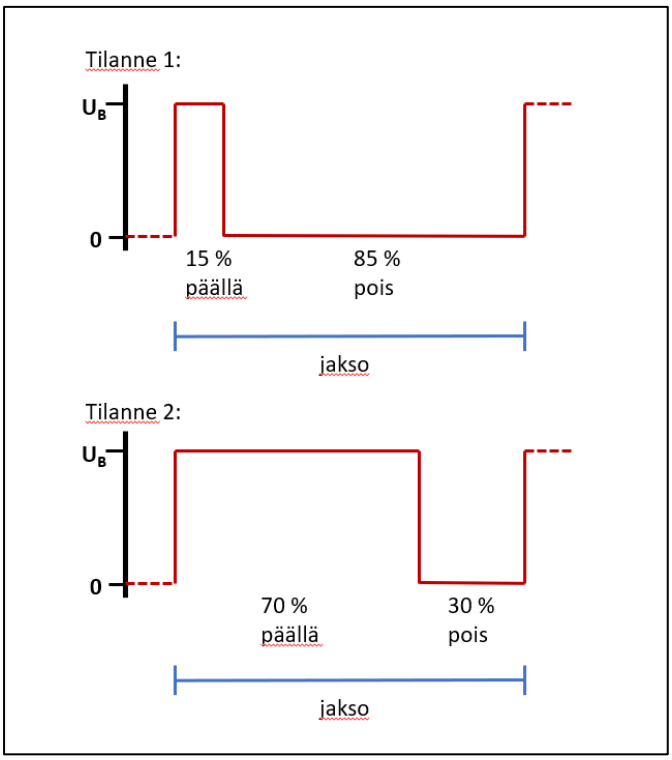
Kuvasta 1 voidaan huomata, että proportionaaliventtiilin ohjaukseen liittyy useita sähköisiä parametreja, esimerkiksi minimivirta, maksimivirta ja dithertaajuus. Tässä opinnäytetyössä ei käsitellä varsinaisesti PWM-venttiilin mekaanista rakennetta ja toimintaa, vaan keskitytään pääasiassa proportio-naali-

venttiiliä säätävään pulssinleveysmodulaatiosignaaliin ja ohjelmoitavan logiikan lähdöstä ulostulevaan säätökäyrään, jolla pulssinleveyssignaalin tasoa ohjataan.

2.2 Pulssinleveysmodulaatiosäädön periaate

Lyhenne PWM tulee englannin kielen sanoista "pulse width modulation", ja se on suomennettu sanaksi pulssinleveysmodulaatio. Pulssinleveysmodulaatiota käytetään usein PWM-venttiileiden säätöön liikkuvien koneiden ohjauksessa. PWM-signaali on myös mahdollista muuntaa sähköisten lisälaitteiden avulla esimerkiksi analogiseksi lähtöjännitteeksi, jos se on säädettävän kohteen kannalta tarpeen (Know-How ecomatmobile 2018, 96).

PWM-signaali vaihtelee maapotentiaalin ja syöttöjännitteen välillä tietyn etukäteen määritellyn perustaajuuden tahdissa. Yhden jakson sisällä olevaa pulssisuhdetta muutetaan sen mukaan, millaista säädön tasoa halutaan saada aikaan. Kuvassa 2 on esitetty pulssisuhteiden muutos tilanteesta 1 (päällä/pois=15/85) tilanteeseen 2 (päällä/pois=70/30). UB tarkoittaa syöttöjännitteen tasoa (esim. 24 VDC).

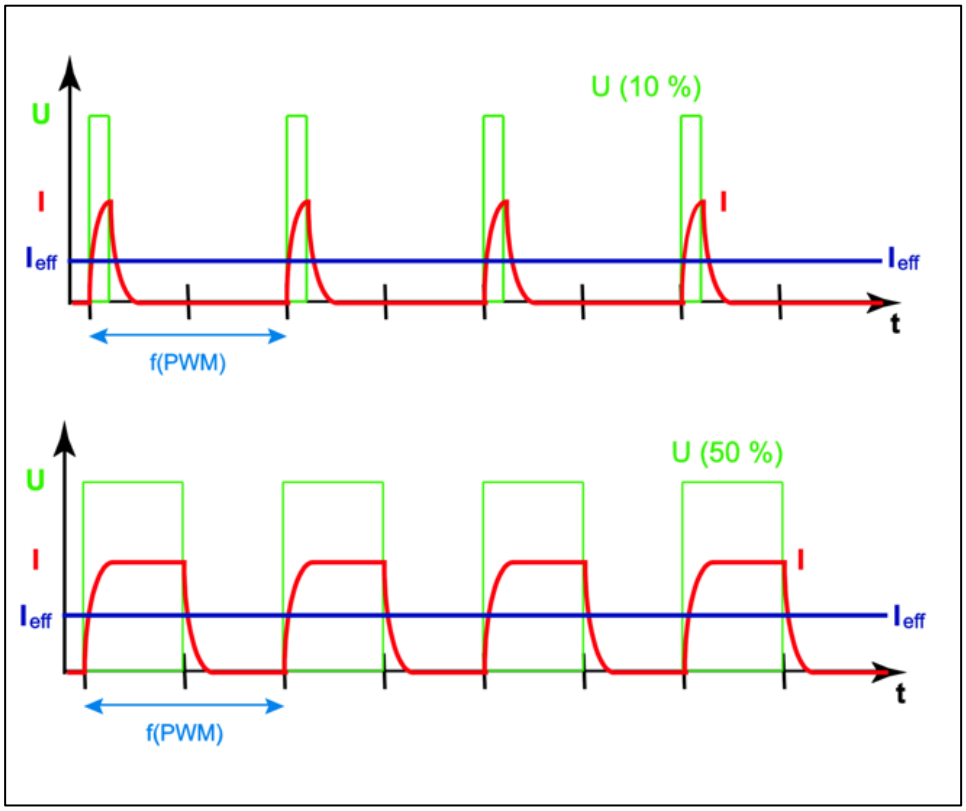


Kuva 2. PWM-pulssisuhteen muutos tilanteesta 1 tilanteeseen 2 (mukaillen Know-How ecomatmobile 2018, 96)

Näiden pulssien suhteesta riippuen, kullekin ajan hetkelle muodostuu laskennallisesti tehollisarvon suuruinen jännite. Tämän jännitteen tason vaikuttaessa säädettävän systeemin kuormaan (impedanssiin), saa se aikaan sitä vastaavan tehollisarvon sähkövirralle. Tämä sähkövirran arvo saa säädettävän kohteen asettumaan ohjauksen määrittämään tasoon.

PWM-signaalissa syöttöjännitteen arvo ei siis muutu. Voidaan ajatella niin, että syöttöjännite katkaistaan aina jakson aikana sen mukaan, mikä jännitteen laskennallinen tehollisarvo kulloinkin halutaan voimaan. Pulssin pituus voi vaihdella 0 %–100 %. Näin syntyy PWM-neliöaaltopulssia, jonka vaikutus on sähköisen tehollisarvon tasoa.

Kuvassa 3 on PWM-signaalin teoreettinen esitys, ja sen vaikutus sähköisen kuorman aikaan saamaan tehollisen sähkövirran tasoon kahdessa eri tilanteessa.



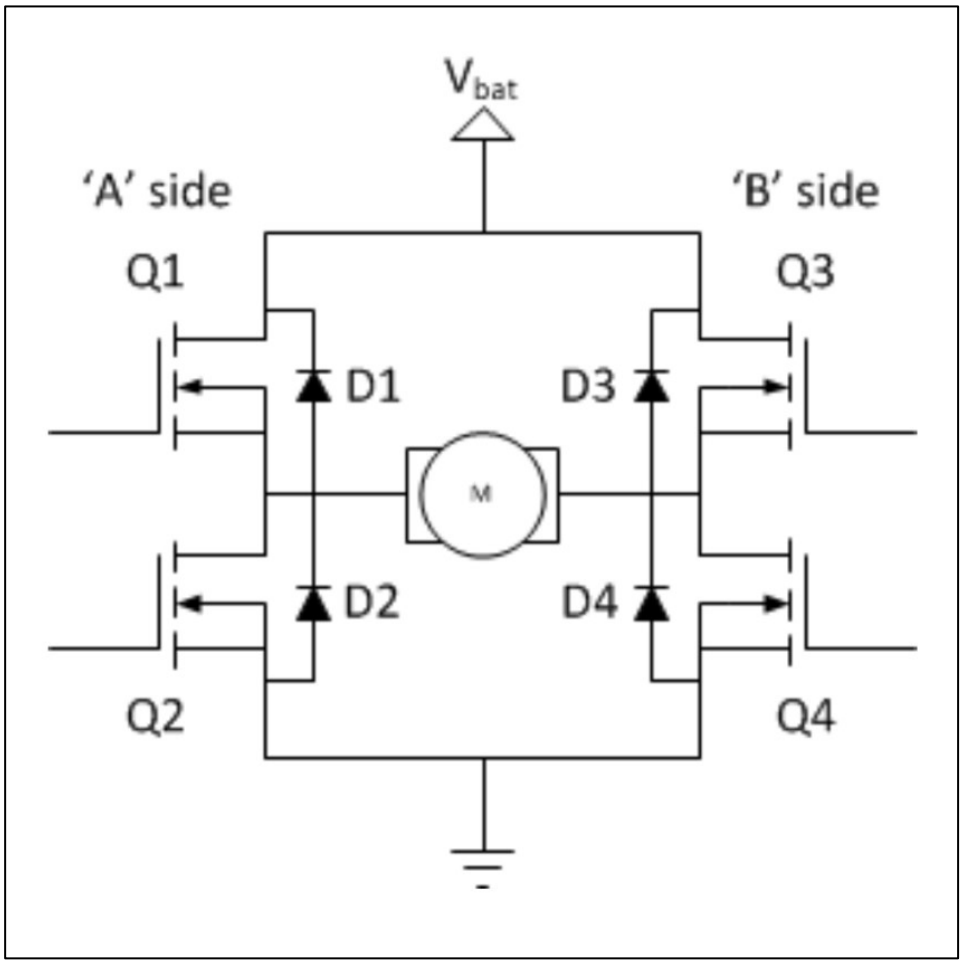
Kuva 3. PWM-signaalin ja sähköisen kuorman aikaan saama tehollinen sähkövirta kahdessa eri tilanteessa (Know-How ecomatmobile 2018, 98)

Kuvan 3 perusteella huomataan, että laskennallinen tehollisvirta muuttuu säädetyn pulssisuhteen mukaan. Se, mikä on PWM-signaalin perustaajuus, riippuu parametreista, jotka on annettu tapauskohtaisesti kullekin säädettävälle laitteelle. Käytännössä PWM-perustaajuus voi olla 50 Hz–2000 Hz.

2.3 H-siltakytkennän periaate

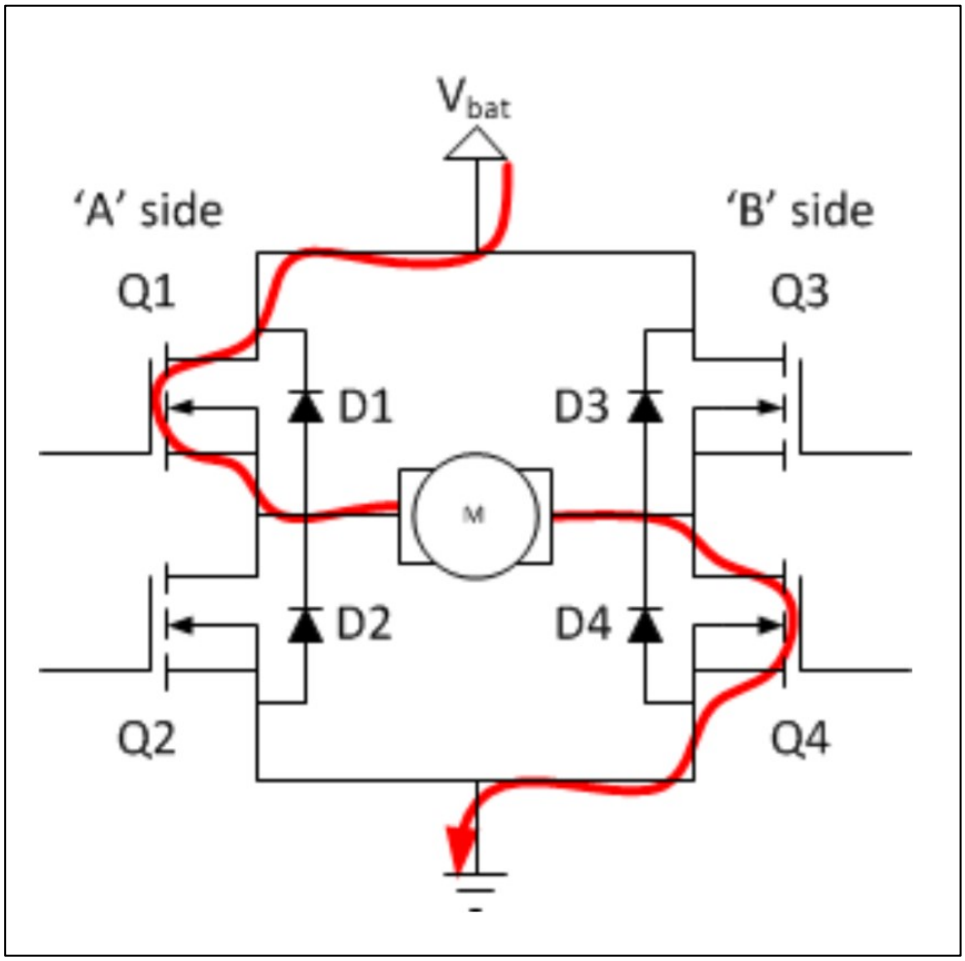
H-silta esitellään tässä yhteydessä sen vuoksi että laboratorioympäristössä käyttämäni lineaarinen toimilaite vaatii sen, mutta käytännössä PWM-venttiilejä ohjattaessa sitä ei tarvita, koska työliikkeen sunnanvaihto tapahtuu yleensä kahden eri sähköisen kelan ohjaamana.

H-sillan perusperiaate on yksinkertainen, ja siihen käytetyt tekniikat eivät ole kovin monimutkaisia. H-sillan tarkoitus on ohjatusti kääntää syöttöjännitteen napaisuus halutuksi. Tätä tarvitaan esimerkiksi tasavirtamoottoreissa suunnan vaihtamiseksi. Kuvassa 4 on esitelty eräs malli H-sillasta (H-Bridges – the Basics s.a.).



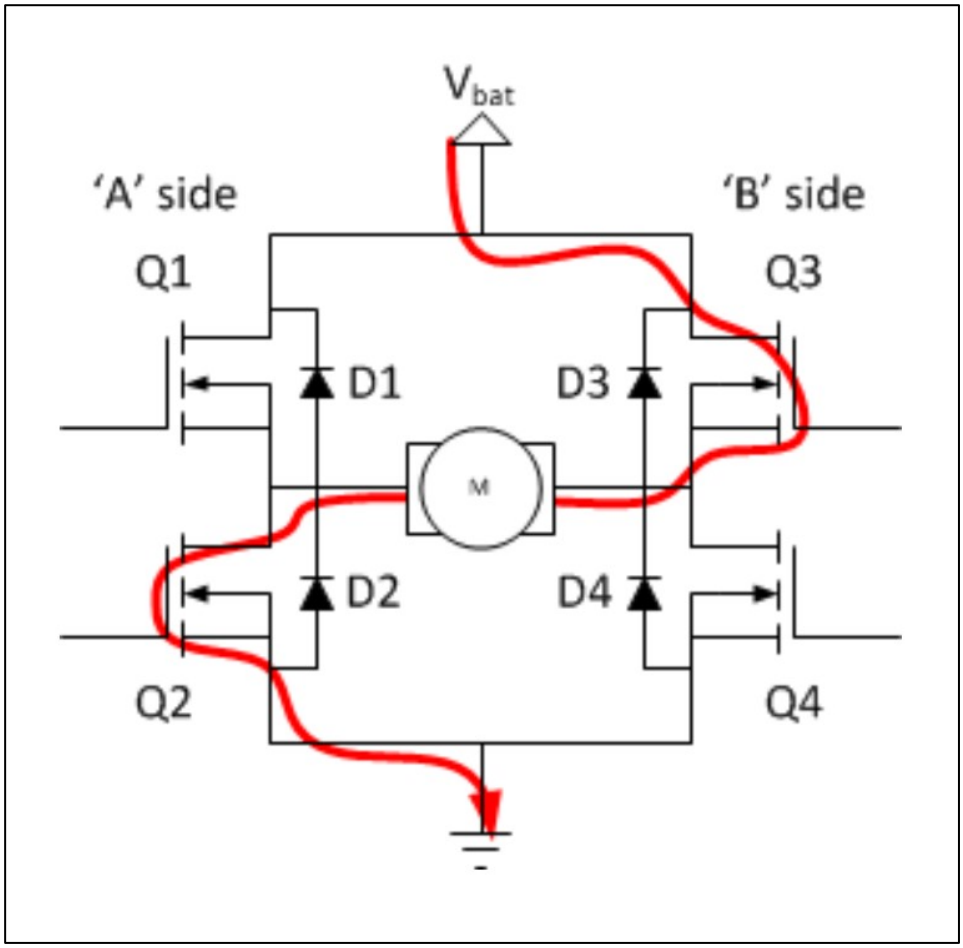
Kuva 4. Eräs malli H-sillasta (H-Bridges – the Basics s.a.)

Kun syöttöjännite on kytketty toiseen päähän siltaa ja maapotentiaali toiseen päähän ja kun FET-transistorit Q1 ja Q4 aktivoidaan, niin syöttöjännitteelle ja maapotentiaalille syntyy sähköinen reitti, jossa syöttöjännite kytkeytyy moottorin vasemman puoleiseen napaan ja maapotentiaali moottorin oikeanpuoleiseen napaan. Kuvassa 5 tilanteen sähköinen reitti on osoitettu mutkittelevalla viivalla.



Kuva 5. H-sillan sähköinen reitti tilanteessa, kun FET-transistorit Q1 ja Q4 ovat aktivoituina (H-Bridges – the Basics s.a.)

Kun tilannetta muutetaan siten, että FET-transistorit Q2 ja Q3 aktivoidaan (ja Q1 ja Q4 ovat sähköttömässä tilassa), muuttuu moottorille tuleva sähköinen kytkentä päinvastaiseksi. Syöttöjännite kytkeytyy moottorin oikeanpuoleiseen napaan ja maapotentiaali moottorin vasemmanpuoleiseen napaan. Kytkennän sähköinen reitti on esitetty kuvassa 6 siinä tilanteessa, kun FET-transistorit Q2 ja Q3 ovat aktivoituneena ja ja Q1 ja Q4 ovat sähköttömässä tilassa.



Kuva 6. H-sillan sähköinen reitti tilanteessa, kun FET-transistorit Q2 ja Q3 ovat aktivoituina (H-Bridges – the Basics s.a.)

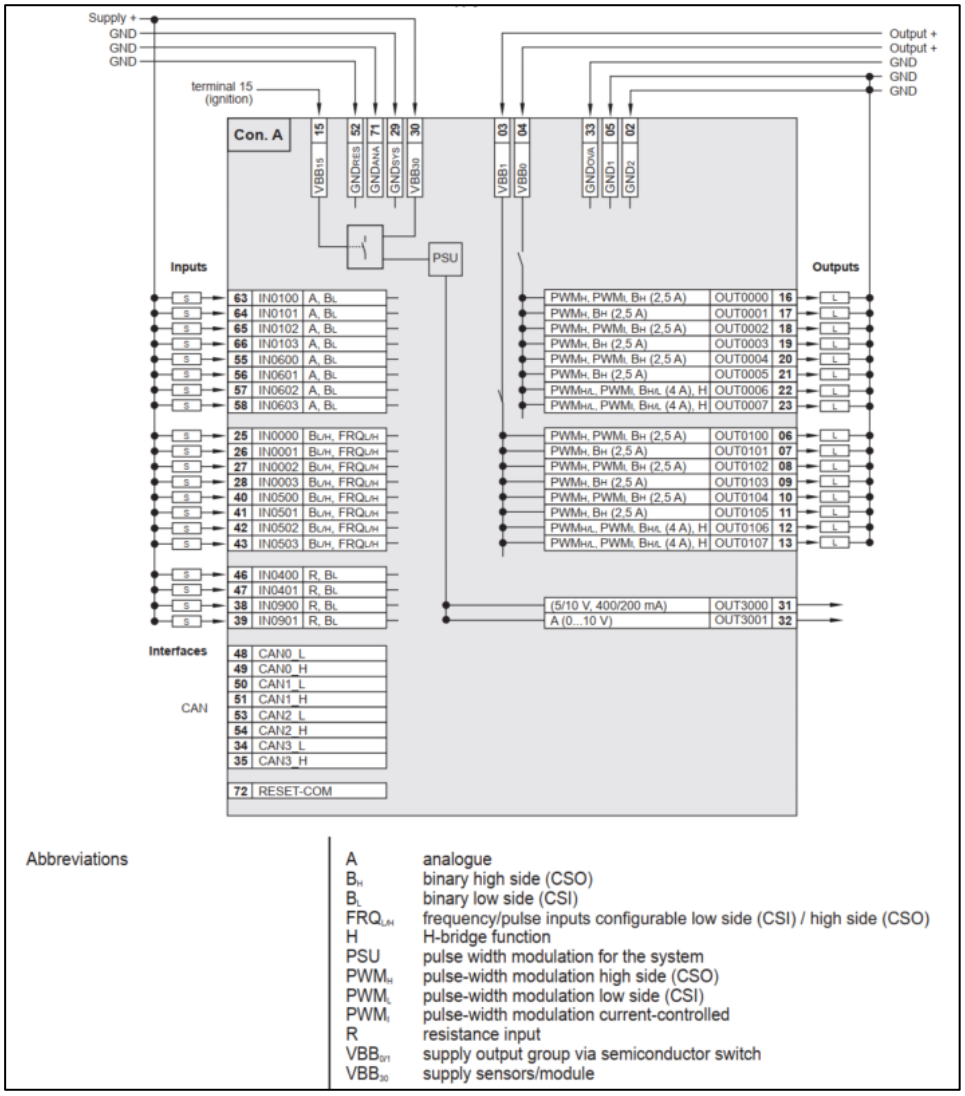
Tutkimustyössä käytössä oleva IFM:n logiikkakontrolleri sisältää PWM-signaalilla varustetun H-siltalähdön. Sen käyttöä käsitellään myöhemmin tässä dokumentissa.

3 TUTKIMUSTYÖSSÄ KÄYTETTÄVÄ LAITTEISTO

Tässä luvussa käydään tarkemmin läpi tutkimustyössä käytettävää laitteistoa, ja kuvataan laitteiden välisiä rajapintoja niiden ollessa yhteydessä toisiinsa.

3.1 Ohjelmoitava logiikka ecomatController CR710S

Tutkimustyötä varten oli käytössä IFM:n ecomatController CR710S. CR710S on monipuolinen ohjelmoitava logiikka, jonka ohjelmointiympäristö on IEC 61131-3-standardia mahdollisimman tarkasti noudattava CODESYS 3.5. Se on tarkoitettu liikkuviin koneisiin asennettavaksi, ja sen suojausluokka on IP 67. Kuvasta 7 voi huomata, että CR710S-logiikassa on monipuolisesti tuloja ja lähtöjä, joita voi myös parametroida tarkoitukseen sopiviksi.



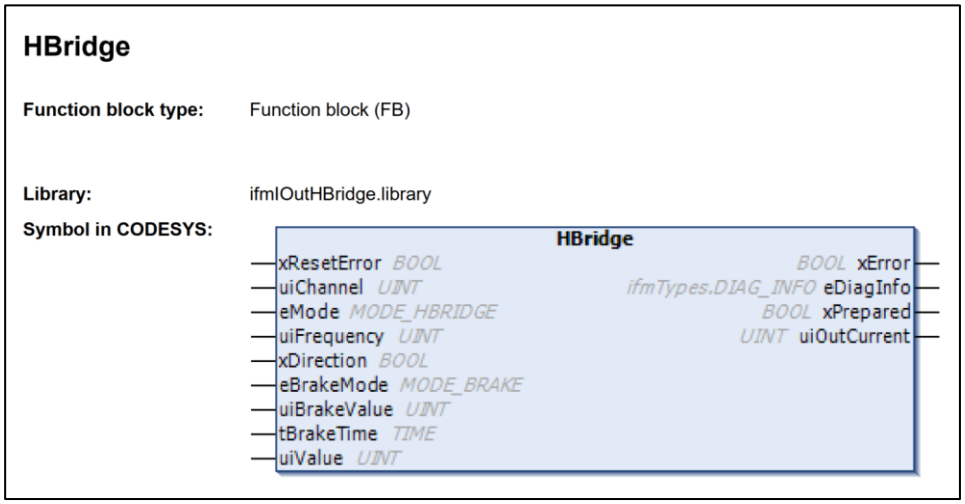
Kuva 7. ecomatController CR710S:n tulot ja lähdöt (Installation instructions ecomatController CR710S 2017, 30)

Esimerkiksi kuvassa 7 esitettyä tuloa numero 63 voidaan käyttää peukalopyörän ohjausjännitteen lukemiseen logiikalle, kun se on parametroitu analogiseksi tuloksi alueelle 0 V–10 V. Peukalopyörän koko ulostulojännitealue on noin 0,5 V – 4,5 V, jolloin tulon mittausalue 0 V–10 V sopii tähän tarkoitukseen hyvin. CR710S:n tuloissa ei ole A/D-muunninta, vaan tulon arvot tuodaan tässä tapauksessa logiikan ohjelmarajapintaan suoraan kokonaisina millivoltteina (mV). 1 millivoltin erottelutarkkuus peukalopyörän osalta on tämän tutkimustyön tarkoituksiin riittävä.

Kuvassa 7 näkyy numeroilla 12, 13, 22 ja 23 PWM-signaalilla varustetut H-silta-lähdöt (kirjain H on kuvausrivin lopussa). Ohjelmoitavan logiikan laitteistokäsikirjassa oleva kaaviokuva on näiden lähtöjen osalta hieman harhaanjohtava, koska siinä on esitetty kytkentä tilanteessa, jossa näitä lähtöjä

käytetään tavallisina PWM-lähtöinä ilman H-silta-ominaisuutta. Kun H-silta-ominaisuutta käytetään esimerkiksi lähdöissä 22 ja 23, täytyy huomioida, että lähtö 22 edustaa PWM-syötön toista napaa ja lähtö 23 edustaa PWM-syötön toista napaa (sähköinen kuorma sijoitetaan siis näiden napojen väliin). Lähtöjen sähköinen polariteetti määräytyy H-siltaa ohjaavan ohjelmallisen toimintolohkon (function block) tilan perusteella.

ecomatController CR710S:n ohjelmiston mukana tulee sen oma monipuolinen H-silta-toimintolohko CODESYS-ohjelmointiympäristöä varten (Kuva 8).

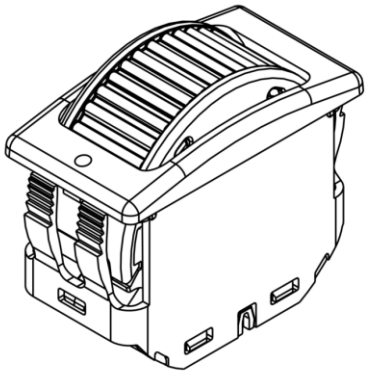


Kuva 8. ecomatController CR710S:n H-silta-toimintolohko CODESYS-ohjelmointiympäristöä varten (Programming manual CR710S 2021, 305)

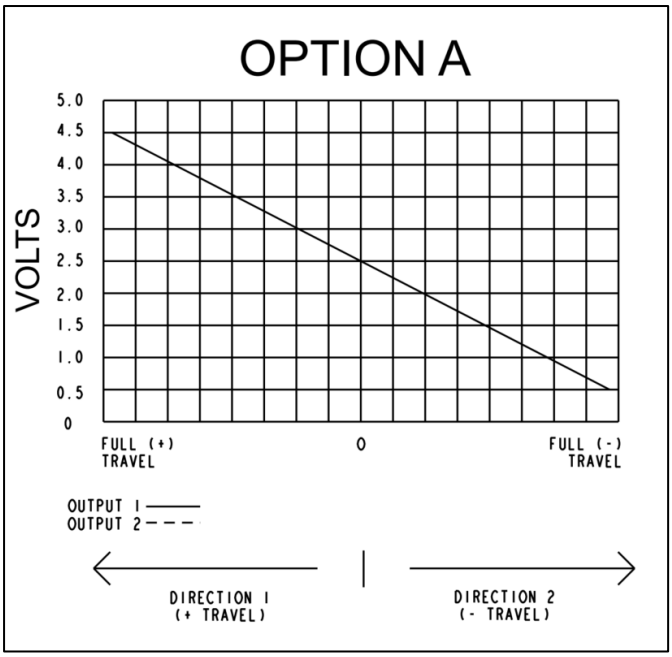
Tätä H-silta-toimilohkoa varten tulee ohjelmassa määritellä useita parametreja, joiden merkitys on kerrottu CR710S:n ohjelmointikäsikirjassa (Programming manual CR710S 2021, 305).

3.2 Peukalopyörä OTTO 1148

Manuaaliseksi säätölaitteeksi tutkimustyöhön oli valittu peukalopyörä OTTO 1148 (Mini proportional output thumbwheel s.a.). Käytännön tilanteissa peukalopyörä on usein asennettu ohjainsauvaan, ja sitä voidaan toki käyttää tilanteen mukaan muillakin sormilla kuin peukalolla, ja erityisesti silloin, kun samaan ohjaussauvaan on asennettu esimerkiksi kolme peukalopyörää (laitteen toinen yleisesti käytetty nimi on proporulla). Kuvassa 9 on esitetty OTTO 1148 -mallinen peukalopyörä, ja kuvassa 9 kyseisen peukalopyörän ulostulosignaalin kuvaaja (Mini proportional output thumbwheel s.a., 2).



Kuva 9. OTTO 1148 -mallinen peukalopyörä (Mini proportional output thumbwheel s.a., 2)



Kuva 10. OTTO 1148 -peukalopyörän ulostulosignaalin kuvaaja (Mini proportional output thumbwheel s.a., 3)

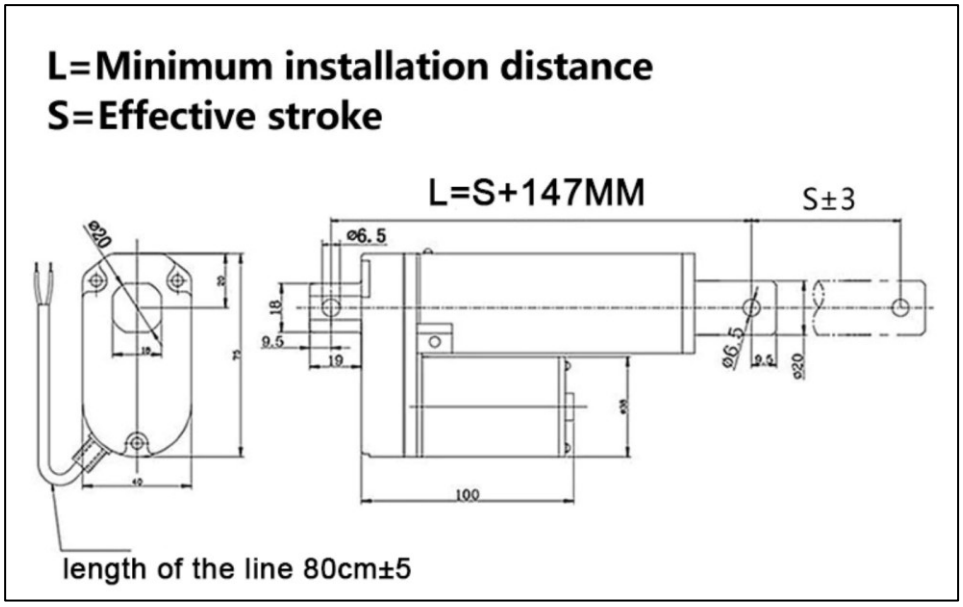
Kuten signaalin kuvaajasta ilmenee (kuva 10), signaalin jännite on 0,5 V–4,5 V, ja keskikohta eli lepotila on arvossa 2,5 V. Tällöin peukalopyörän molempien säätösuuntien säätöalueet ovat 0 mV–2000 mV (500 mV–2500 mV ja 2500 mV–4500 mV).

Käytännössä useita samanmallisia peukalopyöriä testattaessa kävi ilmi, etteivät laitteiden minimikohta, keskikohta ja maksimikohta olleet aina yhdenmukaisia. Ne saattoivat poiketa jopa 0,2 V ilmoitetuista arvoista. Tarkoissa sovellutuksissa nämä poikkeamat on tarpeellista ottaa huomioon, jotta kohteen oh-

jauksesta saadaan luotettava. Peukalopyörän asennuksessa on myös huomioitava sen vaatima + 5 V:n syöttöjännite (DC), jota ei aina ole niin helposti saatavilla ohjelmoitavien logiikkojen toimiessa 12 V:n ja 24 V:n jännitteillä. Usein tuo + 5 V:n jännite (DC) pitää järjestää peukalopyörälle erillisen jännitemuuntimen kautta.

3.3 Lineaarinen toimilaite

PWM-venttiilillä ohjattavan hydraulisylinterin simulointia varten hankin verkkokaupasta lineaarisen toimilaitteen (24V DC 750N Linear Actuator HB300 s.a.), joka toimii 24 V:n tasajännitteellä (Kuva 11).

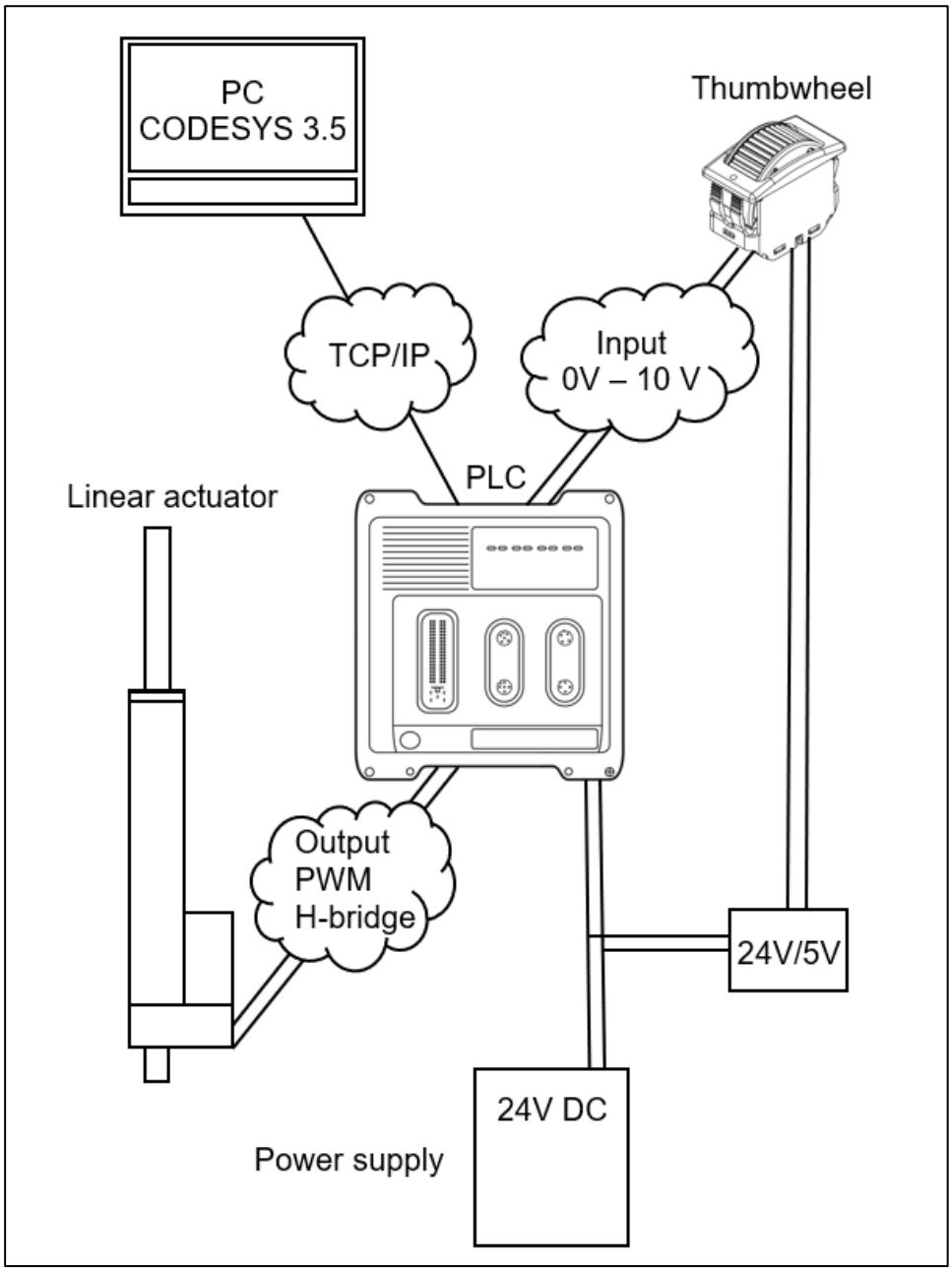


Kuva 11. Perspektiivikuvat lineaariaarisesta toimilaitteesta (Electric Linear Actuator 12V/24V DC s.a.)

Toimilaitteessa piti olla myös toimiva pulssilaskin, jonka avulla olisi voinut saada toimilaitteen liikkeen nopeustieto suoraan ohjelmoitavalle logiikalle. Kun laitetta testattiin suuremmilla nopeuksilla, ei ohjelmoitavan logiikan tuloon kuitenkaan muodostunut järkeviä taajuuslukemia. Tästä johtuen liikkeen nopeuden mittaus täytyi tehdä manuaalisesti 10 s mittausjaksoissa eri nopeuksilla, käyttäen hyödyksi perinteistä työntömittaa. Edullinen toimilaite osoittautui muutoinkin hieman ongelmalliseksi, kun ilmeni, että liikkeen toistettavuus toimilaitteessa samoilla parametreillä ei ollut kovin luotettava. Tutkimuksen lopputuloksen kannalta laitteen toiminta oli kuitenkin riittävää tasoa.

3.4 Testilaitteet liitettyinä toisiinsa

Peukalopyörä ja lineaarinen toimilaite tuli liittää sopivilla rajapinnoilla ohjelmoitavaan logiikkaan. Peukalopyörää varten käytettiin tuloa, jossa jännitetaso oli 0 mV–10000 mV. Lineaarinen toimilaite liitettiin PWM-lähtöihin, joissa oli H-silta-ominaisuus. Kuvassa 12 on esitetty kaavio testilaitteistoista ja laitteiden välisistä rajapinnoista.



Kuva 12. Kaavio testilaitteistoista ja laitteiden välisistä rajapinnoista. Thupwheel (Mini proportional output thumbwheel s.a., 2), PLC (Installation instructions ecomatController CR710S 2017, 1)

4 SÄÄTÖKÄYRÄN MATEMAATTINEN MALLI

Tässä luvussa esitetään matemaattiset kuvaajat, joiden pohjalta voidaan suunnitella lopullisen ohjelman sisältämät yhtälöt säätökäyrää varten.

4.1 Lähtötietojen määrittely

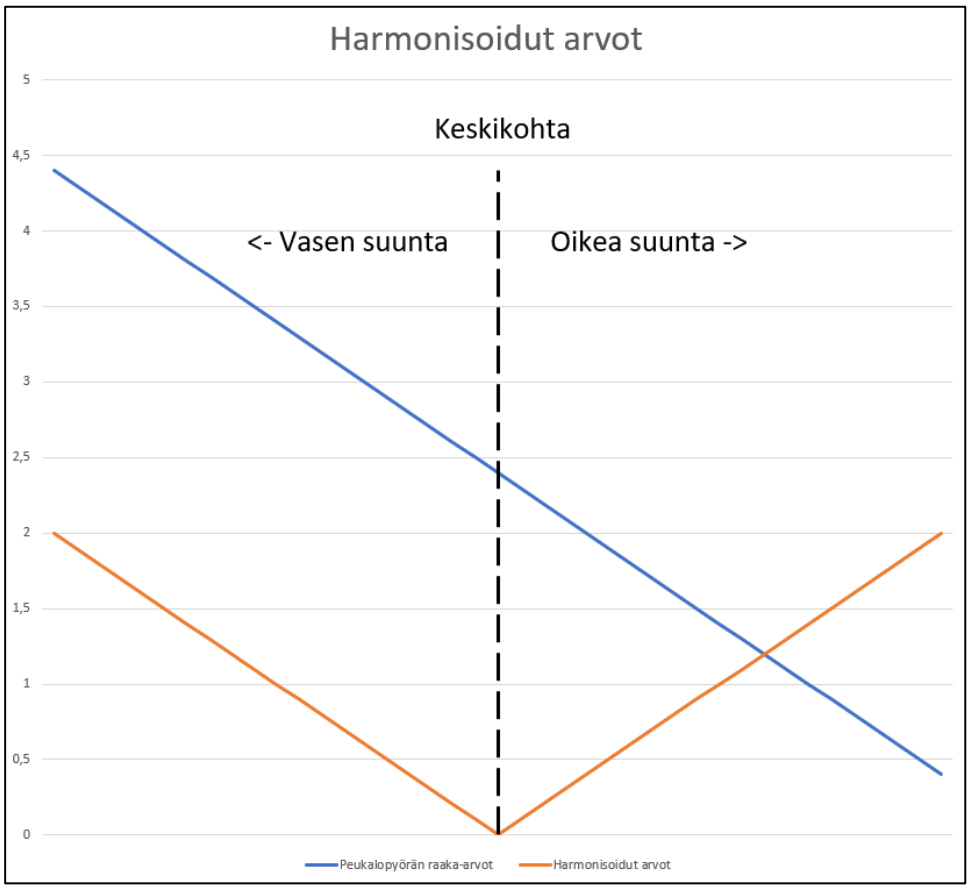
Säätimissä ja toimilaitteissa on usein poikkeamia (esim. hystereesi), jotka vaikuttavat säädön koko ketjuun ohjaussignaalilta viimeiselle toimilaitteelle saakka. Siksi tarkoissa säädöissä systeemin rakenteet on hyvä jakaa pienempiin osiin, edetä vaihe vaiheelta ottaen huomioon kunkin osakokonaisuuden ongelmat ja ratkaista ne. Tässä mallissa oletetaan, että peukalopyörän ja PWM-venttiilin välillä ei ole mitään säätökäyrää muuttavia tekijöitä. Toisin sanoen lähtötietoina ovat peukalopyörältä tulevat nk. raaka-arvot ja PWM-venttiilin vastekäyrä.

Toinen tärkeä tekijä ohjauksessa on käyttömukavuus ja ohjauksen vasteen toteutuminen niin, että käyttäjällä on hyvä tuntuma ohjattavaan laitteeseen hänen sitä operoidessa. Tämä tarkoittaa usein sitä, että säätökäyrään on sovitettava nk. ramppiosuuksia, jotta edellä mainittu toteutuisi ja laitteen käyttäminen olisi juohevaa.

Peukalopyörän valmistajan mukaan laitteen tuottama signaalikäyrä on lineaarinen (Mini proportional output thumbwheel s.a., 2). Tässä yhteydessä en kyseenalaista tätä valmistajan antamaa tietoa, vaan hyväksyn sen sellaisenaan yhdeksi lähtötiedoista. Käytännössä jonkin ohjauslaitteen antamien arvojen kuvaaja voisi olla myös ei-lineaarinen, jolle täytyisi iteroida jokin matemaattinen malli, jotta sen pohjalta voisi suunnitella tarkasti hallittavaa kokonaisuutta. Aiemmin totesimme kokeellisesti, että peukalopyörän miniarvot, keskikohtaarvot ja maksimiarvot saattavat poiketa jopa 0,2 V (= 200 mV) ilmoitetuista arvoista, joten nämä arvot on hyvä määritellä muutettaviksi parametreiksi käytettäviin kaavoihin.

4.2 Peukalopyörän raaka-arvojen harmonisointi

Ohjelman kannalta on kätevää, kun peukalopyörältä saatava raaka-arvoskaala harmonisoidaan aina samaan yhdenmukaiseen skaalaan jatkotoimenpiteitä varten. Esimerkiksi jos alustavissa mittauksissa ilmenee, että peukalopyörän mini-, keski- ja maksimikohdat olisivat 400 mV, 2400 mV ja 4400 mV, on raaka-arvojen lineaariset kuvaajat harmonisoitava ohjelmassa niin, että välille 2400 mV–400 mV (peukalopyörän vasen suunta) saadaan vastaava lineaarinen kuvaaja 0 mV–2000 mV, ja välille 2400 mV–4400 mV (peukalopyörän oikea suunta) saadaan myös vastaava lineaarinen kuvaaja 0 mV–2000 mV. Kuvassa 13 on esitetty peukalopyörän raaka-arvojen harmonisointi, jossa ohjelmalle jatkokäsittelyyn annettavat arvot sisältävät aina saman skaalavälin (esim. 0 mV–2000 mV). Kaava tähän mallin osaan (kuten muihinkin) esitetään myöhemmin Matlab-ohjelman kaavoissa ja kuvaajissa.



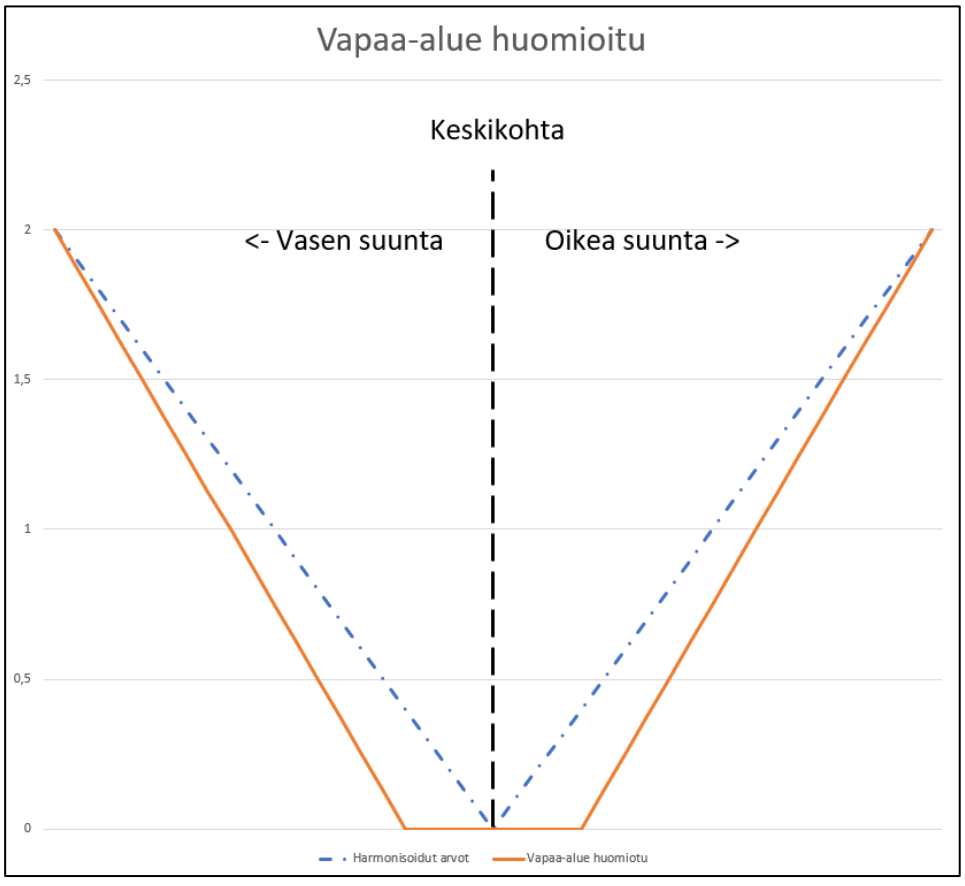
Kuva 13. Peukalopyörän raaka-arvojen harmonisointi

On huomattava, että käytännössä peukalopyörän vasen suunta ohjaisi toista PWM-venttiiliä ja peukalopyörän oikea suunta toista PWM-venttiiliä. Tämä me-

kanismi on rakennettava ohjelmaan. Testilaitteiston yhteydessä ohjelma välittää H-silta-toimintolohkolle tiedon siitä, kummasta säätösuunnasta on kysymys, ja H-silta-toimintolohko asettaa PWM-lähtöjen napaisuuden tuon suuntatiedon mukaiseksi. Säätöarvoissa on myös huomioitava mahdolliset turvallisuuteen liittyvät ehdot. Esimerkiksi jos raaka-arvot peukalopyörän vikaantuessa antavat jonkin muun kuin sen normaaliin skaalaan sisältyvän arvon, niin sitä ei saa hyväksyä sellaisenaan, vaan vaatii erillisen käsittelyn.

4.3 Vapaa-aluekuvaaja

Kun pyrimme ajattelemaan peukalopyörän toimintaa laitteen käyttäjän kannalta, niin on hyvä, jos peukalopyörän keskikohdan molemmin puolin olisi jonkin verran nk. vapaata aluetta. Toisin sanoen, kun peukalopyörällä aloitetaan laitteen ohjaus, ohjelmassa sallittaisiin käyttäjän kannalta sopiva toleranssi (esim. n. 15 % yhden puolikkaan säätöalueesta), kun peukalopyörän rullaa aletaan kääntämään. Kuvassa 14 on esitetty tämän toleranssin huomioiminen säätökäyrässä.

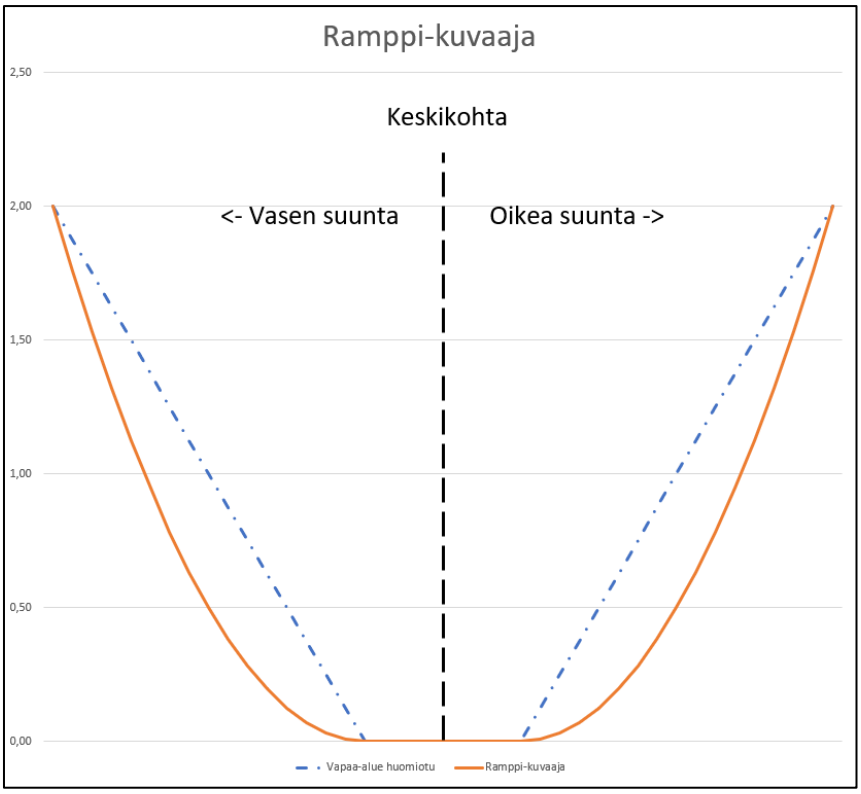


Kuva 14. Vapaa-alueen (toleranssin) huomioiminen säätökäyrässä

Matemaattisessa kaavassa vapaa-alueen huomioiminen edellyttää sitä, että lineaarisen osan kulmakerrointa muutetaan sopivasti ja maksimikohdan leikkauspiste säilytetään samana. Myös jos kuvaajan arvo olisi pienempi kuin 0 (nolla), niin arvoksi annetaan 0 (nolla).

4.4 Ramppi-kuvaaja

Liikkuvissa työkoneissa hallintalaitteiden antamat alkusignaalit tulisi säätää siten, ettei liikkeeseen tulisi tarpeettomia ja ylimääräisiä nykäyksiä. Näin ollen säätöliikkeen alkuun tulisi saada säätökäyrään nk. ramppi. Ramppi-kuvaaja sisältää yleensä toisen asteen yhtälön kaavan. Ympyrän XY-koordinaattiesitys olisi ollut mieluinen tutkittava kuvaaja, mutta sen osalta laskenta osoittautui niin haasteelliseksi, ettei Casion CAS-laskimeni pystynyt suoriutumaan siitä (Matlab-ohjelma kyllä suoriutui). Koska tuo raskas laskenta ympyrän kaavaa käytettäessä olisi pitänyt loppujen lopuksi suorittaa ohjelmoitavassa logiikassa (aina yhden kerran sen käynnistyessä), oli varmempaa käyttää laskennallisesti yksinkertaisempaa kuvaajan kaavaa. Näin ollen päädyin käyttämään paraabelia (tai sen puolikasta). Kuvassa 15 on esitetty ramppi-kuvaaja, joka perustuu paraabeliyhtälöön. Ramppi-kuvaajan leikkauspisteet ovat tässä vaiheessa samat kuin vapaa-aluekuvaajalla.

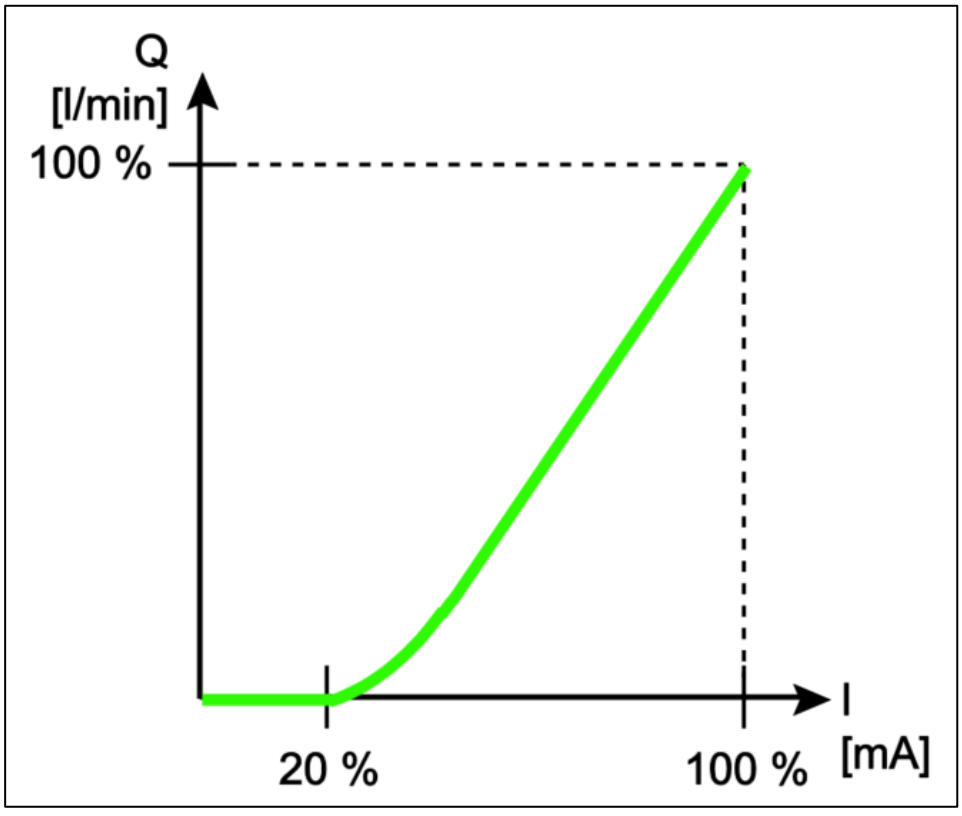


Kuva 15. Ramppi-kuvaaja, jolla yhteiset leikkauspisteet vapaa-aluekuvaajan kanssa

4.5 Kynnysarvo ja yhdistelmäkuvaaja

Seuraavaksi täytyy ottaa huomioon PWM-venttiilin ominaiskäyrä, jonka vaikutus säätökäyrään voi olla merkittävä tilanteesta riippuen. Tässä yhteydessä tarkastelemme melko yleistä tapausta ja jatkamme säätömallin suunnittelemista sen pohjalta.

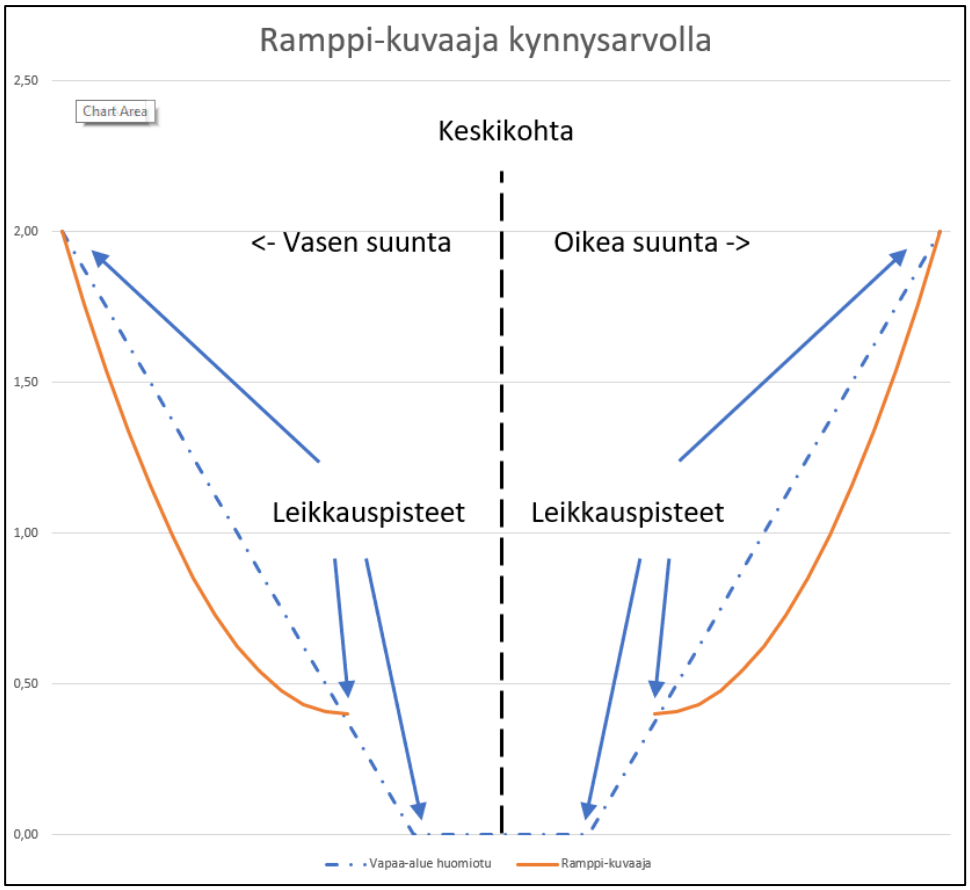
Kuvassa 16 on esitetty melko tyypillinen hydrauliventtiilin ominaiskäyrä. Voimme huomata, että öljyn virtaus alkaa vasta noin 20 % kohdalla maksimisäätöarvosta (Know-How ecomatmobile 2018, 103). Tämä katvealue (kynnystekijä) on otettava huomioon ohjelman säätökäyrää rakennettaessa, ja se on yksi parametroitava muuttuja, joka pitäisi voida asettaa halutuksi riippuen käyttöön otettavan PWM-venttiilin ominaiskäyrästä. Tässä yhteydessä voidaan katvealueen käsitteestä käyttää termiä ”kynnysarvo”.



Kuva 16. Tyypillinen hydrauliventtiilin ominaiskäyrä (Know-How ecomatmobile 2018, 103)

Kynnysarvo voidaan huomioida säätökäyrässä esimerkiksi siten, että paraabelin puolikkaan alkamiskohtaa nostetaan halutun verran vapaa-aluesuoralla.

Jos PWM-venttiilin ominaiskäyrässä tuo kynnysarvo on 20 %, voidaan paraabelin puolikkaan alkupiste kohdistaa maksimiarvoon verrattuna esimerkiksi 18 % kohdalle vapaa-alue suoralla. Kuvassa 17 on esitetty malli tästä tilanteesta.



Kuva 17. Ramppi-kuvaaja kynnysarvolla esitettynä. Parametrien mukaiset leikkauspisteet vapaa-aluekuvaajan kanssa säilyvät.

Näiden kahden kuvaajan (vapaa-aluekuvaaja ja ramppi-kuvaaja) avulla voidaan saada aikaan sopiva parametroitavissa oleva säätökäyrä, joka antaa säätöarvon PWM-venttiilille sen perusteella, mikä peukalopyörältä tuleva raaka-arvo on milloinkin voimassa. Tässä vaiheessa voimme myös todeta, että säätökäyrä on kahden kuvaajan yhdistelmä, jotka voidaan ohjelmassa saattaa voimaan sopivilla ehtolausekkeilla. Vapaa-alueen loppuessa säätökäyrä alkaa nousta lineaarisesti. Kun vapaa-aluekuvaaja kohtaa leikkauspisteen tulee voimaan ramppi-kuvaajan (paraabelin puolikas) arvot.

Jos vertaamme vielä peukalopyörältä saatavaa ominaiskäyrää (raaka-arvot) kokonaiseen säätökäyrään (joka on tavoitteena saada aikaan ohjelmallisesti), on niiden eroavaisuus nähtävissä kuvassa 18 (Huom. y-akselin skaala on eri

edellisiin kuvaajaesityksiin verrattuna, koska peukalopyörän raaka-arvot alkavat tässä yhteydessä 4400 mV:sta).

Kuva 18. Ohjelmallisesti muodostettava yhdistelmäkuvaaja vs. peukalopyörän raaka-arvokuvaaja

Aiemmista kuvaajista ja kuvan 18 yhdistelmäkuvaajasta voimme päätellä, että molempien suuntien (vasen ja oikea) matemaattinen jatkokäsittely on harmonisoidusta kuvaajasta eteenpäin identtistä, joten voimme jatkossa keskittyä vain toisen suunnan kaavoihin ja hyödyntää ohjelmassa samoja kaavoja myös toisen suunnan käsittelyssä.

5 MATEMAATTISET KAAVAT JA KUVAAJAT MATLAB-OHJELMASSA

Tämä luku käsittelee aiemmin esitettyihin matemaattisiin malleihin perustuvien kaavojen toimivuutta käytännössä. Matematiikan perustietojeni avulla ja Matlab-ohjelmaa hyödyntäen suunnittelin luvussa esitetyt yhtälöt käytettäviksi lopullista ohjelmoitavan logiikan ohjelmaa varten.

5.1 Matlab-ohjelman käyttö opinnäytetyössä

Matlab on yleisesti yliopistoissa ja korkeakouluissa käytetty numeeriseen laskentaan tarkoitettu tietokoneohjelmisto (MathWorks 2022). Sillä voidaan käsitellä laajasti eri matematiikan alueita, ja se sisältää myös oman ohjelmointikielensä. Opinnäytetyössä oli tarkoitus hyödyntää Matlab-ohjelmiston Simulink-laajennosta, mutta Matlab:in Live Script -ominaisuus osoittautui tässä yhteydessä sopivammaksi vaihtoehdoksi työstää kaavat ja osoittaa niiden toimivuus.

Kaikki seuraavassa esitetyt matemaattiset kaavat ja niiden väliset riippuvuudet kuuluvat perusmatematiikan osaamisen piirin (Calculus - A Complete Course 7th ed 2010, 12-19). Matlab-ohjelmiston käyttöön löytyy runsaasti oppimateriaalia, jota on myös tässä opinnäytetyössä hyödynnetty (Self-Paced Online Courses). Opinnäytetyötä tehdessä käytössä oli Matlab-ohjelman versio R2022a.

5.2 Peukalopyörän raaka-arvojen harmonisointi

Kaavoja ja niiden toimintaa mallinnettaessa määritellään aluksi, että tarkastelemme pääasiassa peukalopyörän ominaiskäyrän raaka-arvoja välillä 2500 mV–4500 mV ja että harmonisoidussa kuvaajassa arvot kiinnitetään skaalaan 0 mV–2000 mV.

Harmonisointikaavoja varten määritellään seuraavat symbolit (muuttujat):

x_u= harmonisoidun kuvaajan arvot

n = säätöalue per suunta (2000 yksikköä)

r_1= peukalopyörän raaka-arvot keskikohdasta maksimiin

h = peukalopyörän maksimiarvo 4500 mV

m = peukalopyörän keskikohdan arvo 2500 mV

o = peukalopyörän minimiarvo 500 mV

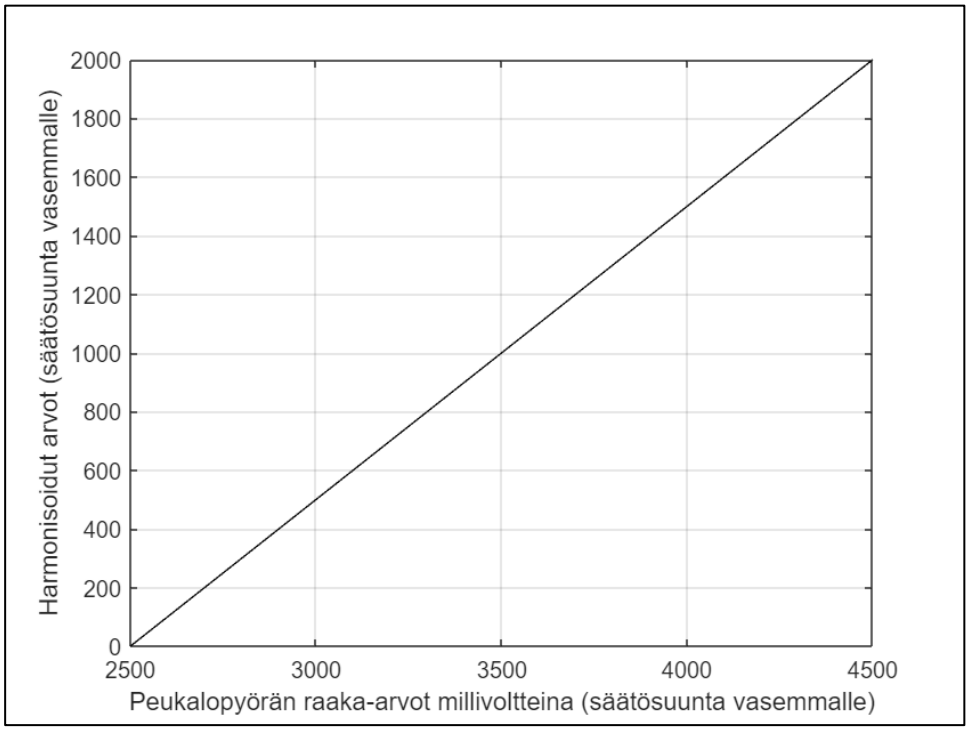
Kaava, jolla peukalopyörän raaka-arvojoukko harmonisoidaan yhdenmukaiseksi arvojoukoksi, esitellään seuraavaksi. Suunta on keskikohdasta (2500 mV) raaka-arvojen maksimiin (2400 mV):

$$x_u = n \frac{r_1 - m}{h - m} \quad (1)$$

Vastaava kaava toiselle suunnalle olisi seuraava:

$$x_u = n \frac{m - r_1}{m - o} \quad (2)$$

Jälkimmäistä kaavaa hyödynnetään vasta varsinaisessa ohjelmoitavan logiikan ohjelmassa siinä vaiheessa, kun toisen suunnan (keskikohdasta minimiin) raaka-arvot harmonisoidaan skaalaan 0–2000. Matlabin kautta saatu kuvaaja harmonisoiduista arvoista on esitetty kuvassa 19. Kuvaajissa ei selvyyden vuoksi esitetä mittayksiköille mitään laatuja. Kaavojen ja kuvaajien Matlab-lähdekoodit ovat koottuina liitteessä 1.



Kuva 19: Matlab-kuvaaja harmonisoiduista arvoista (Huom. Näkyvissä on vain toinen ohjaussuunta)

Kuvassa 19 x-akselilla on esitetty peukalopyörän antamat raaka-arvot, jotka samalla edustavat peukalopyörän fyysistä säätöasentoa sen säätöpyörää vasemmalle liikutettaessa (peukalopyörän ominaiskäyrä on lineaarinen). y-akselilla on harmonisoidut arvot (0–2000).

5.3 Vapaa-aluekuvaaja

Kun säädön vapaa-aluetta muutetaan esimerkiksi 15 % (0.15), niin lineaarisen osan säätökäyrän kulmakerroin muuttuu. Vapaa-aluekuvaajan kaava on seuraava:

$$x_i = \frac{x_u}{1-f} - \frac{f\,n}{1-f} \qquad (3)$$

jossa

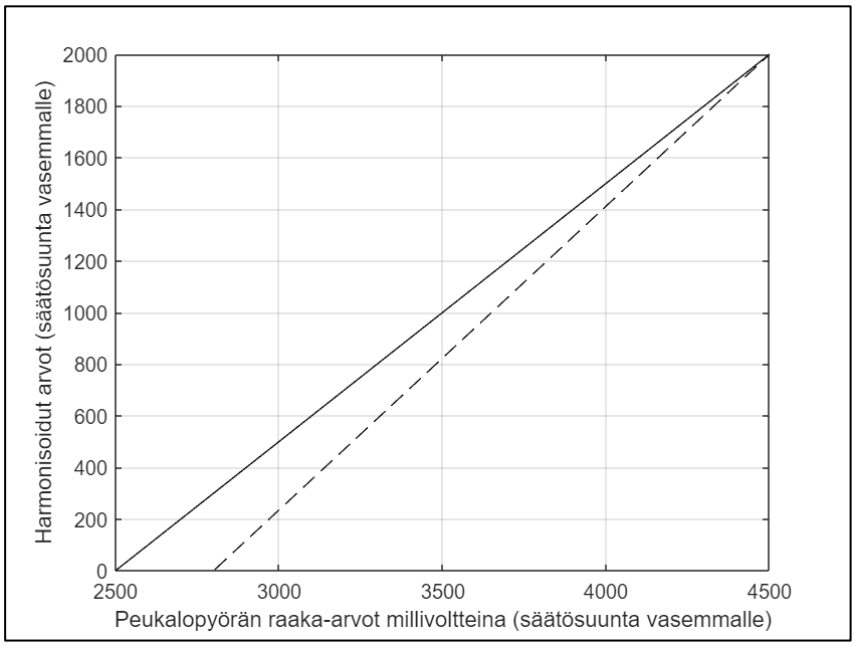
x_i= vapaa-alueen sisältävän kuvaajan arvot

x_u= harmonisoidun kuvaajan arvot

n = säätöalue per suunta (2000 yksikköä)

f = peukalopyörän suhteellinen vapaa-alue per suunta

Kuvassa 20 esitetään, kuinka muodostettu vapaa-aluekuvaaja leikkaa x-akselin 15 %:n kohdalla (arvossa 2800), kun vapaa-alueeksi on asetettu 15 % (f = 0.15).



Kuva 20. Matlab-kuvaaja vapaa-aluearvoista (katkoviivalla esitetty)

5.4 Ramppi-kuvaaja, kynnysarvo ja yhdistelmäkuvaaja

Seuraavaksi vapaa-aluekuvaajan pariksi tulisi saada paraabeli (ramppi-kuvaaja), joka huomioi kynnysarvon mukaisen leikkauspisteen vapaa-aluekuvaajan kanssa ja maksimiarvon leikkauspisteen. Jos PWM-venttiilin ominaiskäyrässä venttiilin avautumiskohta olisi 22 %:n kohdalla, niin kynnysarvo voitaisiin laittaa esimerkiksi 20 %:iin (kerroin 0.2).

Seuraavassa esitetään matemaattiset toisiinsa liittyvät yhtälöt, jotka toteuttavat edellä mainitut ehdot.
Kuvaajien ensimmäisen leikkauspisteen y-koordinaatti:

$$v = e\,n \qquad (4)$$

Raaka-arvojen tarkasteluväli (x-akselilla):

$$p = h - m \qquad (5)$$

Kuvaajien ensimmäisen leikkauspisteen x-koordinaatti:

$$u = \frac{p\,v + f\,n\,p - f\,p\,v}{n} \qquad (6)$$

Paraabelin kuvaajan yhtälö kiinnitettynä leikkauspisteisiin:

$$x_e = e\,n + \frac{\left(u - \frac{p\,x_u}{n}\right)^2 (n - e\,n)}{(p-u)^2} \qquad (7)$$

Yhtälöissä on huomioitu seuraavat arvot ja muuttujat:

v = kuvaajien ensimmäisen leikkauspisteen y-koordinaatti

e = kynnyskerroin per suunta

n = säätöalue per suunta (2000 yksikköä)

u = kuvaajien ensimmäisen leikkauspisteen x-koordinaatti

p = raaka-arvojen tarkasteluväli (x-akselilla)

m = peukalopyörän keskikohdan arvo 2500 mV

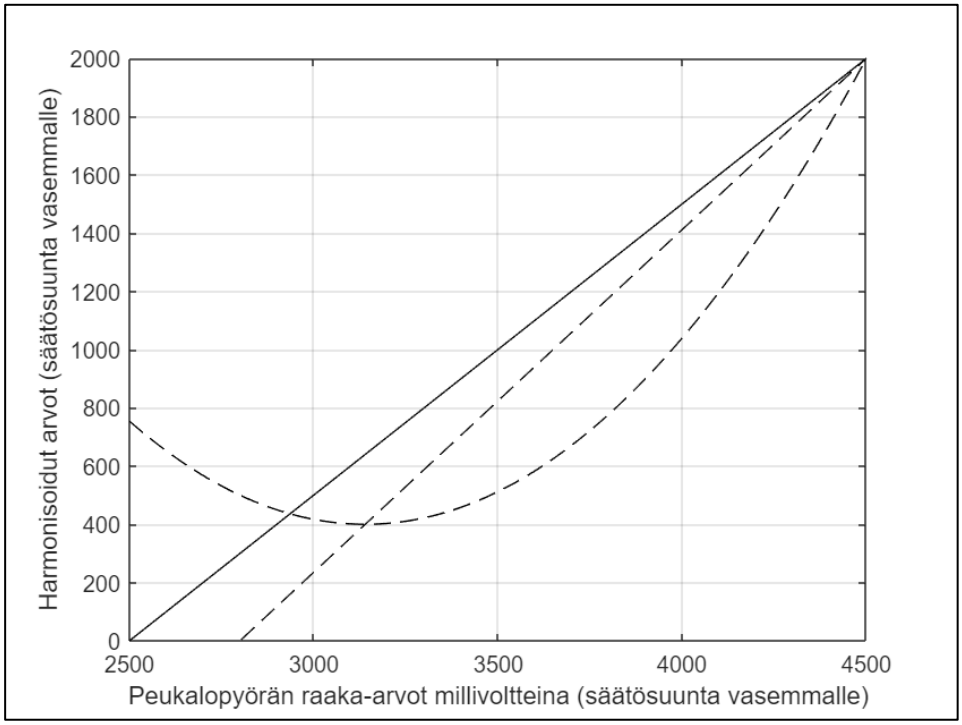
h = peukalopyörän maksimiarvo 4500 mV

f = peukalopyörän suhteellinen vapaa-alue per suunta

x_u= harmonisoidun kuvaajan arvot

x_e= leikkauspisteisiin kiinnitetyn paraabelin arvot

Kuvassa 21 näkyy edellä esitettyjen yhtälöiden mukainen ramppi-kuvaaja (paraabeli) suhteessa vapaa-aluekuvaajaan ja harmonisoituihin arvoihin.



Kuva 21.Ramppi-kuvaaja (paraabeli) ja sen leikkauspisteet vapaa-aluekuvaajan kanssa

Yhtälöt on rakennettu niin, että paraabelin minimikohta leikkaa vapaa-aluekuvaajan kanssa aina kynnysarvon määrittelemässä paikassa. Näillä arvoilla nähdään, että leikkauspisteen y-koordinaatti on 400, joka on 20 % (kerroin 0.2) y-akselin maksimiarvosta 2000. Kuvaajien kaavoissa voidaan siis muuttaa useita muuttujien arvoja, ja silti kuvaajien suhteet ja leikkauspisteet toteuttavat halutun lopputuloksen. Kuvaa 22 varten on kokeeksi muutettu huomattavasti seuraavien muuttujien arvoja:

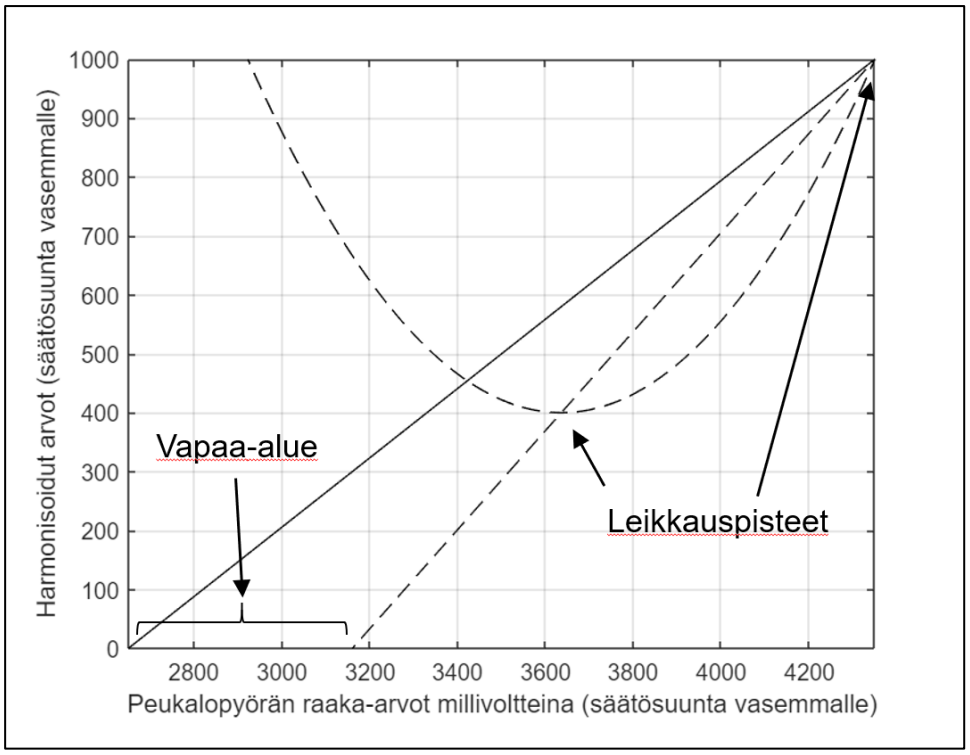
n = säätöalue per suunta (2000 → 1000 yksikköä)

m = peukalopyörän keskikohdan arvo (2500 mV → 2650 mV)

h = peukalopyörän maksimiarvo (4500 mV → 4350 mV)

f = peukalopyörän suhteellinen vapaa-alue per suunta 0.15 → 0.3

e = kynnyskerroin per suunta 0.2 → 0.4



Kuva 22. Kuvaajien suhteiden ja leikkauspisteiden säilyminen parametreja muutettaessa

Kuvassa 22 nähdään että laskentakaavat toimivat keskenään niin, että parametrien muuttaminen onnistuu hyvin. Vapaa-aluekuvaajan ja ramppi-kuvaajan ensimmäisen leikkauspisteen y-koordinaatti on 400, joka on 40 % (kerroin 0.4) y-akselin maksimiarvosta 1000. Voimme todeta, että näillä yhtälöillä vapaa-aluekuvaajan ja ramppi-kuvaajan yhdistelmä on toimiva, ja kuvaajien keskinäiset suhteet säilyvät ja leikkauspisteet asettuvat oikeisiin kohtiin.

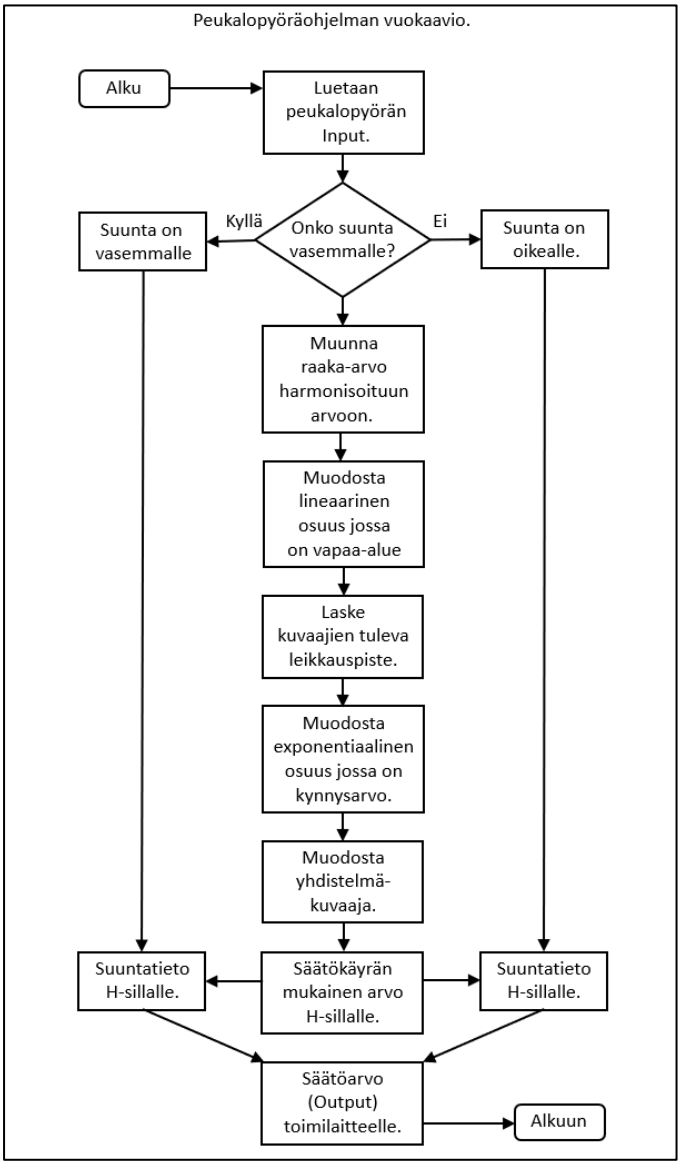
Ohjelmoitavan logiikan ohjelmassa kuvaajien yhdistelmä tullaan toteuttamaan siten, että säätökäyrä seuraa 0-arvosta ylöspäin ensin lineaarista vapaa-aluekuvaajaa ensimmäiseen leikkauspisteeseen saakka, ja sen jälkeen säätökäyrä seuraa ramppi-kuvaajaa maksimiin saakka. Sama säätökäyrä toimii myös toiseen suuntaan, jolloin maksimisuunnasta tullaan kohti pienempiä arvoja.

6 OHJELMAN TOTEUTUS CODESYS 3.5 -YMPÄRISTÖSSÄ

Tässä luvussa kerrotaan, kuinka lopullinen ohjelma ohjelmoitavaan logiikkaan toteutettiin CODESYS 3.5 -ympäristössä ja miten ohjelman toimivuutta testattiin eri menetelmin.

6.1 Ohjelman rakenne ja vuokaavio

Ohjelman rakentaminen CODESYS 3.5 -ohjelmointiympäristössä IFM:n CR710S-logiikalla oli opinnäytetyön seuraava vaihe. Ohjelmoinnissa käytettiin Structured Text -ohjelmointikieltä (CODESYS Online Help s.a.).



Kuva 23. Vuokaavio toteutetusta ST-kielisestä peukalopyörä-ohjelmasta

Kuvassa 23 esitetään vuokaavio toteutetusta ST-kielisestä ohjelmasta, jossa peukalopyörällä ohjataan PWM-venttiiliä (testissä lineaarista toimilaitetta) niin, että ohjattavan laitteen käyttömukavuus ja ohjauksen hyvä vaste toteutuvat.

Ohjelman lähdekoodi on esitetty kokonaisuudessaan liitteessä 2. Ensin on esitelty ohjelmassa käytetyt globaalit muuttujat ja sen jälkeen varsinaiset ohjelmamoduulit. Jotta ohjelmaa tai sen osia voisi käyttää jossakin toisessa ympäristössä, täytyy ymmärtää, kuinka esimerkiksi globaalit muuttujat sijoitellaan omille paikoilleen kehitysympäristössä.

Vaikka yleinen ajatus on se, että CODESYS-ohjelmointiympäristö takaisi lähdekoodin siirrettävyyden eri laitteistoalustalta toiselle, niin siinä on kuitenkin tilanteesta riippuen joitakin rajoituksia, jotka olisi otettava huomioon jo, ennen kuin ohjelmaa aletaan rakentamaan. Esimerkiksi IFM:n laitteissa on monia laitteistorajapintaan peilaavia, nk. natiiveja toimintolohkoja, jotka toimivat vain IFM:n laitteissa (esimerkkinä HBridge-toimintolohko). Joten laajemmissa kokonaisuuksissa on järkevää erottaa natiiveja toimilohkoja käyttävät osuudet omiksi moduuleikseen ja säilyttää yhteensopivat osuudet erillään edellisistä, jos se vain suinkin on mahdollista.

Toinen hyvä tapa on dokumentoida lähdekoodiin eri osakokonaisuudet ja niiden keskinäiset riippuvuudet selkeästi, jolloin valmiin aiemmin tehdyn ohjelman tai sen osan käyttäminen toisessa ympäristössä voisi olla kohtuullisella vaivalla mahdollista.

6.2 Ohjelman toimivuuden testaaminen

Aluksi ohjelman toimivuus todettiin siten, että tarkastettiin kaavojen oikeellisuus ohjelmassa. Ohjelmaa muutettiin testausta varten siten, että oli mahdollista syöttää raaka-arvo ohjelmaan manuaalisesti ja verrata, vastasiko säätöarvo Matlab-ohjelman antamia tuloksia. Molemmissa ohjelmissa parametrien arvot tuli olla tarkasteluhetkellä samat. Tämä testaus suoritettiin muutamilla erilaisilla parametriasetuksella, ja lopputulokset osoittivat, että kaavat oli ohjelmoitu toimimaan logiikan ohjelmassa oikein.

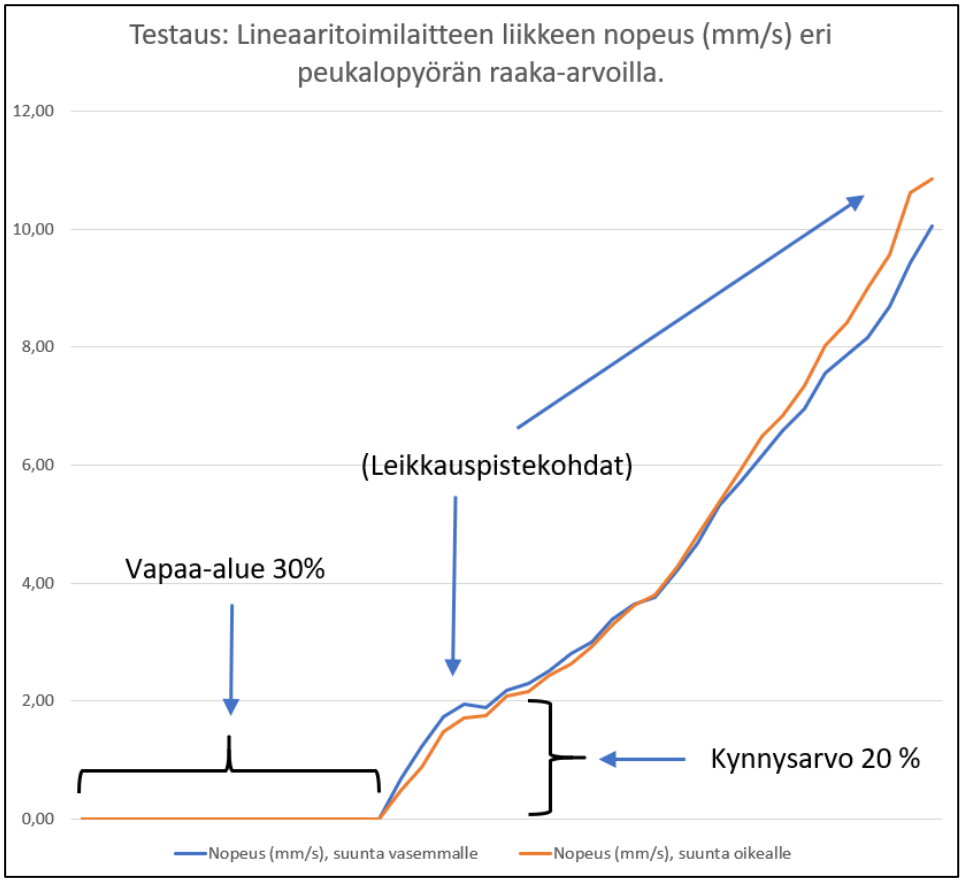
Toinen testausmenettely oli se, että haluttiin nähdä, miten peukalopyörältä annettu säätöarvo vaikutti lineaaritoimilaitteen liikkeen nopeuteen. Tämän testin tuloksia oli alkuasetteluissa tarkoitus käsitellä ohjelmallisesti, mutta kun lineaaritoimilaitteen enkooderin pulssin luotettavaa tietoa ei saatu luettua logiikalle, piti tämä testaus suorittaa perinteisin menetelmin.

Ohjemaa muutettiin tätä testausta varten siten, että manuaalisesti annettua säätöarvoa syötettiin ohjelmalle tasan 10 s ajan. Ajastus oli tehty ohjelmallisesti, mutta lineaaritoimilaitteen tässä ajassa kulkema matka piti mitata työntömitalla. Testikierroksia kertyi kaiken kaikkiaan 41 kertaa molempiin suuntiin. Vapaa-alue oli parametroitu 30 %:n arvoon (0.3) ja kynnysarvo 20 % (0.2), jotta testin tuloksiin perustuvasta kuvaajasta voitaisiin havaita säädön vaikutus toimilaitteelle saakka.

Testin aikana ilmeni, että lineaaritoimilaitteen toiminnan herkkyys oli erilainen riippuen siitä, kumpaan suuntaan toimilaitteen karaa liikutettiin. Tällainen tilanne voi olla hyvin mahdollinen myös käytännössä, kun puhutaan PWM-venttiilin ja hydraulisylinterin yhdistelmästä (öljyn virtaus ja kitkatekijät voivat olla toiseen liikesuuntaan erilaiset).

Testin tulokset kirjattiin Excel-ohjelmaan, ja saatujen arvojen perusteella muodostettiin kuvaaja, joka on esitetty kuvassa 24.

Kuvan 24 sisältämistä testituloksista nähdään, että rakennettu ohjelma toteuttaa hyvin sen säätökäyrän, mihin on pyrittykin. Parametroitu vapaa-alue toteutuu (30 %), ja kynnysarvo (20 %) osuu juuri yhdistelmäkuvaajan taitekohtaan. Myös lineaari- ja ramppiosuus näkyvät kuvaajassa selvästi.



Kuva 24. Lineaaritoimilatteen liikkeen nopeus peukalopyörän eri raaka-arvoilla

Käytännössä kuvassa 24 esitetty tilanne vaikuttaisi hydraulisylinterin liikkeeseen vasta siinä vaiheessa, kun PWM-venttiilin ominaiskäyrän kynnysarvo (esim. 20 %) olisi saavutettu. Tällöin liikkeen ohjaus alkaisi ramppi-kuvaaja-osuuden alkupisteestä (tämä toivottu tavoite voidaan saavuttaa parametroinnin avulla), joka mahdollistaisi hyvän ohjaustuntuman jo liikkeen alkuun, ja laitteen käyttäjä voisi hallita laitteen liikkeitä tarkemmin ennakoiden ja turvallisesti.

7 TUTKIMUSTYÖN TULOSTEN POHDINTA

Opinnäytetyön tavoitteena oli saada aikaan käytäntöön sovellettavan esimerkin avulla osamenetelmä, jolla voitaisi parantaa ohjelmoitavalla logiikalla ohjattavan liikkuvan työkoneen ohjattavuutta ja käyttömukavuutta.

Koko opinnäytetyön kannalta matemaattisen mallin suunnittelu oli eniten aikaresursseja vievä osuus. Aluksi tarkastelin ympyrän yhtälöä koordinaatistossa. Lukuisten ruutupapereille tehtyjen suunnitelmien jälkeen ja CAS-laskimeni suorituskyvyn loppumisen jälkeen päädyin käyttämään ramppi-kuvaajana

paraabelin puolikasta. Tämä oli toteutuksen kannalta hyvä asia, koska paraabelin yhtälön käsittely nykyisillä suorituskyvyn puolesta keveilläkin ohjelmoitavilla logiikoilla on mahdollista.

Myös yhdistelmäkaavion yhdistäminen matemaattisesti oli itselleni haasteellinen tehtävä. Erityisesti vapaa-aluekuvaajan ja ramppikuvaajan leikkauspisteiden kohdistaminen yhtälön avulla oikeaan paikkaan - annettujen parametriarvojen vaikuttaessa - oli suuren työn takana. Loppujen lopuksi sain ratkaistua halutut matemaattiset yhtälöt ja niiden keskinäiset riippuvuudet, ja itse olen erittäin tyytyväinen lopputulokseen yhtälöiden osalta.

Ohjelman rakentaminen ja testaus sujuivat kohtuullisen hyvin huolimatta siitä, että lineaaritoimilaitteen enkooderin pulssia ei saatu kunnolla välittymään ohjelmoitavalle logiikalle. Enkooderilta saatiin 5 V -tasoinen pulssi, jonka tasoa nostettiin logiikkaa varten elektroniikkapiirillä lähelle syöttöjännitteen arvoa (24 V). Suuremmilla lineaarianturin tasavirtamoottorin kierrosnopeuksilla pulssin taajuus näytti kuitenkin epäkelpoja arvoja. Ongelman ratkaisemiseen en käyttänyt enempää aikaresursseja, vaan päätin, että suunnittelen tämän osan testausta uudelleen ja suoritan sen perinteisin menetelmin.

Lopputulokseen kokonaisuutena olen myös erittäin tyytyväinen. Tutkimustyön aikana syntynyttä matemaattista ideaa ja sen pohjalta toteutettua Structured Text -ohjelmaa voin soveltaa käytäntöön tulevina kuukausina työhöni liittyvissä kehityshankkeissa.

.

LÄHTEET

24V DC 750N Linear Actuator HB300. Ebay. s.a. WWW-dokumentti. Saatavissa: https://www.ebay.com/itm/313558300789?hash=item490187f875:g:Yt0AA-OSw0~pgvs1A&var=612384375327 [viitattu 12.1.2022].

Calculus - A Complete Course 7th ed 2010. Adams, R. A. & Essex, C. 2010. PDF-tiedosto. Saatavissa: http://eng.usc.ac.ir/files/1508848447251.pdf [viitattu 11.1.2022].

CODESYS Online Help. CODESYS. s.a. WWW-dokumentti. Saatavissa: https://help.codesys.com/ [viitattu 17.3.2022].

Electric Linear Actuator 12V/24V DC. Aliexpress. s.a. WWW-dokumentti. Saatavissa: https://www.aliexpress.com/item/32824701062.html [viitattu 12.1.2022].

H-Bridges – the Basics. Modular Circuits. s.a. WWW-dokumentti. Saatavissa: https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/ [viitattu 16.2.2022].

Installation instructions ecomatController CR710S. IFM. 2017. PDF-tiedosto. Saatavissa: https://www.ifm.com/fi/fi/product/CR710S?tab=documents [viitattu 11.1.2022].

Know-How ecomatmobile. IFM. 2018. PDF-tiedosto. Saatavissa: https://www.ifm.com/mounting/7391020UK.pdf [viitattu 12.1.2022].

MathWorks. MathWorks. 2022. WWW-dokumentti. Saatavissa: https://se.mathworks.com/ [viitattu 16.2.2022].

Mini proportional output thumbwheel. Mouser. s.a. PDF-tiedosto. Saatavissa: https://www.mouser.com/datasheet/2/603/Otto_HTWM-1215034.pdf [viitattu 12.1.2022].

Proportional directional spool valve type PSVF. HAWE. 1998. PDF-tiedosto. Saatavissa: https://downloads.hawe.com/7/7/D77007F-en.pdf [viitattu 13.1.2022].

Programming manual CR710S. IFM. 2021. PDF-tiedosto. Saatavissa: https://www.ifm.com/fi/fi/product/CR710S?tab=documents [viitattu 11.1.2022].

Self-Paced Online Courses. MathWorks. 2022. WWW-dokumentti. Saatavissa: https://matlabacademy.mathworks.com/ [viitattu 17.2.2022].

MATLAB-ohjelmakoodi

```
% 3. Paraabelin liittäminen leikkauspisteisiin suoran ja loppupisteen
% kanssa.
% Kun säätöön halutaan nk. vapaata aluetta esim. 15 % saakka niin kerroin
% f on 0.15.
% Kun säätöön halutaan lineaarista osuutta esim. 20 % saakka niin kerroin
% e on 0.2
syms m h n r_1 f e v u p x_u x_i x_e
disp("Kuvaajien ensimmäisen leikkauspisteen y-koordinaatti:");
displayFormula("v=e.*n");
disp("Raaka-arvojen tarkasteluväli (x-akselilla):");
displayFormula("p=h-m");
disp("Kuvaajien ensimmäisen leikkauspisteen x-koordinaatti:");
displayFormula("u=(p.*v+f.*n.*p-f.*p.*v)/n");
disp("Paraabelin kuvaajan yhtälö kiinnitettynä leikkauspisteisiin:");
displayFormula("x_e=e.*n+(((u-p.*x_u/n).^2).*(n-e.*n))/((p-u).^2)");
m=(2500);               % peukalopyörän keskikohdan arvo (raaka)
h=(4500);               % peukalopyörän maksimiarvo (raaka)
n=(2000);               % säätöalue per suunta
r_1=m:h;                % peukalopyörän raaka-arvot keskeltä ylöspäin
f=0.15;                 % peukalopyörän vapaa-alue per suunta (0-0.15 eli 0-15%)
p=h-m;                  % raaka-arvojen tarkasteluväli (x-akselilla)
e=0.2;                  % kynnyskerroin per suunta (0-0.2 eli 0-20%)
v=e.*n;                     % ensimmäisen leikkauspisteen x-koordinaatti
u=(p.*v+f.*n.*p-f.*p.*v)/n; % ensimmäisen leikkauspisteen y-koordinaatti
fprintf('Kaavoissa on huomioitu seuraavat arvot ja muuttujat:')
fprintf('v = kuvaajien ensimmäisen leikkauspisteen y-koordinaatti')
fprintf('e = kynnyskerroin per suunta (kertoimella %0.1f)',e)
fprintf('n = säätöalue per suunta (%u yksikköä)',n)
fprintf('u = kuvaajien ensimmäisen leikkauspisteen x-koordinaatti')
fprintf('p = raaka-arvojen tarkasteluväli (x-akselilla)')
fprintf('m = peukalopyörän keskikohdan arvo (%u mV)',m)
fprintf('h = peukalopyörän maksimiarvo (%u mV)',h)
fprintf('f = peukalopyörän suhteellinen vapaa-alue per suunta (kertoimella %0.1f)',f)
fprintf('\x0078\x1D64= harmonisoidun kuvaajan arvot')
fprintf('\x0078\x2091= leikkauspisteisiin kiinnitetyn paraabelin arvot')
%fprintf('\x0072\x2081= peukalopyörän tulon arvojoukko keskikohdasta maksimiin')
x_u=(r_1-m)/(h-m).*n;   % lasketaan muuntokuvaaja skaalaan 0-2000
x_i=(f.*n-x_u)/(f-1);   % lasketaan lineaarisen osan kuvaaja
x_e=e.*n+(((u-p.*x_u/n).^2).*(n-e.*n))/((p-u).^2); % lasketaan exponentiaalisen osan
kuvaaja
axis([m h 0 n])         % määritellään x- ja y-akseleiden min ja max
plot(r_1,x_u,'black',r_1,x_i,'black--',r_1,x_e,'black--')   % tulostetaan lähtöarvojen
kuvaajat
grid on
xlabel('Peukalopyörän raaka-arvot millivoltteina (säätösuunta vasemmalle)')
ylabel('Harmonisoidut arvot (säätösuunta vasemmalle)')
xlim([m h])
ylim([0 n])
```

CODESYS 3.5 ST-ohjelmakoodi

```
// VAR_INPUTS-muuttujat:
VAR_GLOBAL
            uintProporoll_01_AnalogInput01 : UINT;
            // con2 pin2=B IN01        3 (0...10 000 mV)          Wheel_01 / <-> / Saadaan ulos 0...2000 mV suuntaansa
END_VAR

// VAR_JOYSTICK-muuttujatS:
VAR_GLOBAL
// Yhteiset muuttujat
            reaProporoll_Center_Point_Tolerance_Raw : REAL := 20;              // Keskipiste-arvon toleranssi real-tyyppinä, huojunta
tarkistettava
                                                                               // proporullalohtaisesti

            reaProporoll_Common_Min_Value_Point_Raw : REAL := 400;             // Yhteinen minimi-arvon real-tyyppinä
                                                                               // Huom! Tällä estetään haamuarvon syntyminen jos kyt-
kentä irtoaa                                                                                  // (jolloin Raw-arvo = 0)

            reaProporoll_Controlling_Area_Value_2000 : REAL := 2000;           // Säätöalue jota käytetään (0-2000)

            // Peukalopyörän arvot (minimi, keskikohta ja maksimi)
            reaProporoll_Left_01_Left_Min_Value_Raw : REAL := 520;             // Min-arvo real-tyyppinä
            reaProporoll_Left_01_Center_Point_Raw : REAL := 2590;              // Keskipiste real-tyyppinä
            reaProporoll_Left_01_Right_Max_Value_Raw : REAL := 4590;           // Max-arvo real-tyyppinä

            // Peukalopyörän vasen suunta
            reaProporoll_Left_01_Left_Controlling_Area : REAL := 2000;         // Säätöalue n
            reaProporoll_Left_01_Left_Position_Area : REAL := 1;               // Proporullan positioalue p
            reaProporoll_Left_01_Left_Coefficient_For_Free_Area : REAL := 0.3; // Vapaa-alue-kerroin f
            reaProporoll_Left_01_Left_Coefficient_For_Threshold : REAL := 0.2; // Kynnysalue-kerroin e
            reaProporoll_Left_01_Left_Raw_2000 : REAL;                         // raaka-arvo 0-2000 real-tyyppinä
            reaProporoll_Left_01_Left_Get_Output_1000 : REAL := 0.5;           // Kertoimella saadaan PWM-ohjaussignaaliksi arvo 0-1000
            reaProporoll_Left_01_Left_Line01 : REAL;                           // Lineaarinen osuus
```

```
            lreaProporoll_Left_01_Left_Expline01 : LREAL;                    // Exponentiaalinen osuus

            // Left suunnan Line01-suoran ja Expline01-paraabelin leikkauspisteet
            reaProporoll_Left_01_Left_Curves_Intersection_Point_X : REAL;    // Leikkauspisteen X-koordinaatti
            reaProporoll_Left_01_Left_Curves_Intersection_Point_Y : REAL;    // Leikkauspisteen Y-koordinaatti
            lreaProporoll_Left_01_Left_Curve_Out : LREAL;                                  // Line01-suoran ja Expline01-paraabelin
yhdistelmä

            // Peukalopyörän oikea suunta
            reaProporoll_Left_01_Right_Controlling_Area : REAL := 2000;      // Säätöalue
            reaProporoll_Left_01_Right_Position_Area : REAL := 1;            // Proporullan positioalue
            reaProporoll_Left_01_Right_Coefficient_For_Free_Area : REAL := 0.3; // Vapaa-alue-kerroin
            reaProporoll_Left_01_Right_Coefficient_For_Threshold : REAL := 0.2; // Kynnysalue-kerroin
            reaProporoll_Left_01_Right_Raw_2000 : REAL;                                    // raaka-arvo 0-2000 real-
tyyppinä
            reaProporoll_Left_01_Right_Get_Output_1000 : REAL := 0.5;        // Kertoimella saadaan PWM-ohjaussignaaliksi arvo 0-1000
            reaProporoll_Left_01_Right_Line01 : REAL;                        // Lineaarinen osuus
            lreaProporoll_Left_01_Right_Expline01 : LREAL;                   // Exponentiaalinen osuus
            // Right suunnan Line01-suoran ja Expline01-paraabelin leikkauspisteet
            reaProporoll_Left_01_Right_Curves_Intersection_Point_X : REAL;   // Leikkauspisteen X-koordinaatti
            reaProporoll_Left_01_Right_Curves_Intersection_Point_Y : REAL;   // Leikkauspisteen Y-koordinaatti
            lreaProporoll_Left_01_Right_Curve_Out : LREAL;                   // Line01-suoran ja Expline01-paraabelin yhdistelmä

            bProporoll_Left_01_Left_Direction : BOOL;                        // Vasen suunta aktiivinen
            bProporoll_Left_01_Right_Direction : BOOL;                       // Oikea suunta aktiivinen
END_VAR

// GVL_OUTPUTS-muuttujat:
VAR_GLOBAL
            uintLeft_01_Left_PwmOutput : UINT;                  // (PWM output) / Ohjaus uint-tyyppinä skaalalla 0-1000
            uintLeft_01_Right_PwmOutput : UINT;                 // (PWM output) / Ohjaus uint-tyyppinä skaalalla 0-1000
            uintLeft_01_Common_PwmOutput : UINT;                // (PWM output) / Ohjaus uint-tyyppinä skaalalla 0-1000 // H-siltaa
varten
END_VAR
```

```
===========
PROGRAM PLC_PRG
VAR
END_VAR
------------
CR710S_Config();                        // IO Configuration
IOWrapper.READ_INPUTS();                // Call IOWrapper Main Program and Read Data from Inputs
JOYSTICKS_CONTROLS();                   // JOYSTICKS_CONTROLS aliohjelma
HBRIDGE_CONTROL();                      // HBRIDGE_CONTROL aliohjelma
IOWrapper.WRITE_OUTPUTS();              // Call IOWrapper Main Program and Write Data to Outputs

===========
PROGRAM JOYSTICKS_CONTROLS
VAR
END_VAR
------------
// Peukalopyörän raaka-arvon muuntaminen kaavojen kautta säätöarvoksi

// Luetaan peukalopyörän jännitearvo (skaala esim. 500 - 4500 mV)
GVL_INPUTS.uintProporoll_01_AnalogInput01 := IN0100.VAR_OUT.uiValueAnalogue;

// Left ja Right suunnat muunnetaan harmonisoituun skaalaan arvoihin 0-2000. Myös suuntatieto asetetaan H-silta-toimintolohkolle
// Kaavoissa on huomioitu raaka-arvon minimi, keskikohta ja maksimi
IF GVL_INPUTS.uintProporoll_01_AnalogInput01 < (GVL_JOYSTICKS.reaProporoll_Left_01_Center_Point_Raw -
             GVL_JOYSTICKS.reaProporoll_Center_Point_Tolerance_Raw)
             AND GVL_INPUTS.uintProporoll_01_AnalogInput01 > GVL_JOYSTICKS.reaProporoll_Common_Min_Value_Point_Raw THEN
             GVL_JOYSTICKS.reaProporoll_Left_01_Left_Raw_2000 := ((GVL_JOYSTICKS.reaProporoll_Left_01_Center_Point_Raw -
             UINT_TO_REAL(GVL_INPUTS.uintProporoll_01_AnalogInput01)) / (GVL_JOYSTICKS.reaProporoll_Left_01_Center_Point_Raw -
             GVL_JOYSTICKS.reaProporoll_Left_01_Left_Min_Value_Raw)) * GVL_JOYSTICKS.reaProporoll_Controlling_Area_Value_2000;

             GVL_JOYSTICKS.bProporoll_Left_01_Left_Direction := TRUE;
             GVL_JOYSTICKS.bProporoll_Left_01_Right_Direction := FALSE;
ELSE
             GVL_JOYSTICKS.bProporoll_Left_01_Left_Direction := FALSE;
```

```
            GVL_JOYSTICKS.bProporoll_Left_01_Right_Direction := TRUE;
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Raw_2000 := 0;
END_IF

IF GVL_INPUTS.uintProporoll_01_AnalogInput01 > (GVL_JOYSTICKS.reaProporoll_Left_01_Center_Point_Raw +
            GVL_JOYSTICKS.reaProporoll_Center_Point_Tolerance_Raw) THEN
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Raw_2000 := ((UINT_TO_REAL(GVL_INPUTS.uintProporoll_01_AnalogInput01) -
            GVL_JOYSTICKS.reaProporoll_Left_01_Center_Point_Raw) / (GVL_JOYSTICKS.reaProporoll_Left_01_Right_Max_Value_Raw -
            GVL_JOYSTICKS.reaProporoll_Left_01_Center_Point_Raw)) * GVL_JOYSTICKS.reaProporoll_Controlling_Area_Value_2000;

            GVL_JOYSTICKS.bProporoll_Left_01_Left_Direction := FALSE;
            GVL_JOYSTICKS.bProporoll_Left_01_Right_Direction := TRUE;
ELSE
            GVL_JOYSTICKS.bProporoll_Left_01_Left_Direction := TRUE;
            GVL_JOYSTICKS.bProporoll_Left_01_Right_Direction := FALSE;
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Raw_2000 := 0;
END_IF

// Left ja Right suuntien lineaariset säätökäyrät joissa on huomioitu säätöalue n ja proporullan vapaa-aluekerroin f // Line01=(1/(1-
f))*(Raw_2000-n*f)
// Myös negatiiviset arvot "leikataan" pois
IF GVL_JOYSTICKS.reaProporoll_Left_01_Left_Raw_2000 > 0 THEN
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Line01 := (1/ (1 - GVL_JOYSTICKS.reaProporoll_Left_01_Left_Coeffi-
cient_For_Free_Area)) *
             (GVL_JOYSTICKS.reaProporoll_Left_01_Left_Raw_2000 - GVL_JOYSTICKS.reaProporoll_Controlling_Area_Value_2000 *
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Coefficient_For_Free_Area);
ELSE
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Line01 := 0;
END_IF

IF GVL_JOYSTICKS.reaProporoll_Left_01_Right_Raw_2000 > 0 THEN
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Line01 := (1/ (1 - GVL_JOYSTICKS.reaProporoll_Left_01_Right_Coeffi-
cient_For_Free_Area)) *
```

```
            (GVL_JOYSTICKS.reaProporoll_Left_01_Right_Raw_2000 - GVL_JOYSTICKS.reaProporoll_Controlling_Area_Value_2000 *
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Coefficient_For_Free_Area);
ELSE
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Line01 := 0;
END_IF

// Left ja Right suuntien Line01-suoran ja Expline01-paraabelin leikkauspisteet
// Y-piste = e*n jossa e=kynnyskerroin ja n=säätöalue, ja X-piste = =(v-v*f)/n+f jossa v=Y-piste ja f=vapaa-alue-kerroin
GVL_JOYSTICKS.reaProporoll_Left_01_Left_Curves_Intersection_Point_Y := GVL_JOYSTICKS.reaProporoll_Left_01_Left_Coefficient_For_Threshold *
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Controlling_Area;

GVL_JOYSTICKS.reaProporoll_Left_01_Left_Curves_Intersection_Point_X := (GVL_JOYSTICKS.reaProporoll_Left_01_Left_Curves_Intersec-
tion_Point_Y -
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Curves_Intersection_Point_Y * GVL_JOYSTICKS.reaProporoll_Left_01_Left_Coeffi-
cient_For_Free_Area) /
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Controlling_Area + GVL_JOYSTICKS.reaProporoll_Left_01_Left_Coeffi-
cient_For_Free_Area;

GVL_JOYSTICKS.reaProporoll_Left_01_Right_Curves_Intersection_Point_Y := GVL_JOYSTICKS.reaProporoll_Left_01_Right_Coefficient_For_Threshold
*
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Controlling_Area;

GVL_JOYSTICKS.reaProporoll_Left_01_Right_Curves_Intersection_Point_X := (GVL_JOYSTICKS.reaProporoll_Left_01_Right_Curves_Intersec-
tion_Point_Y -
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Curves_Intersection_Point_Y *
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Coefficient_For_Free_Area) /
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Controlling_Area + GVL_JOYSTICKS.reaProporoll_Left_01_Right_Coeffi-
cient_For_Free_Area;

// Left ja Right suuntien exponentiaaliset säätökäyrät  // Expo01=(n-n*e)/((p-u)^2)*((Raw_2000/n*p-u)^2)+n*e
GVL_JOYSTICKS.lreaProporoll_Left_01_Left_Expline01 := (GVL_JOYSTICKS.reaProporoll_Left_01_Left_Controlling_Area -
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Controlling_Area *
```

```
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Coefficient_For_Threshold) / EXPT((GVL_JOYSTICKS.reaProporoll_Left_01_Left_Posi-
tion_Area -
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Curves_Intersection_Point_X),2) * EXPT((GVL_JOYSTICKS.reaPro-
poroll_Left_01_Left_Raw_2000 /
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Controlling_Area * GVL_JOYSTICKS.reaProporoll_Left_01_Left_Position_Area -
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Curves_Intersection_Point_X),2) + GVL_JOYSTICKS.reaProporoll_Left_01_Left_Control-
ling_Area *
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Coefficient_For_Threshold;

GVL_JOYSTICKS.lreaProporoll_Left_01_Right_Expline01 := (GVL_JOYSTICKS.reaProporoll_Left_01_Right_Controlling_Area - GVL_JOYSTICKS.reaPro-
poroll_Left_01_Right_Controlling_Area *
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Coefficient_For_Threshold) / EXPT((GVL_JOYSTICKS.reaProporoll_Left_01_Right_Posi-
tion_Area -
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Curves_Intersection_Point_X),2) * EXPT((GVL_JOYSTICKS.reaPro-
poroll_Left_01_Right_Raw_2000 /
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Controlling_Area * GVL_JOYSTICKS.reaProporoll_Left_01_Right_Position_Area -
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Curves_Intersection_Point_X),2) + GVL_JOYSTICKS.reaProporoll_Left_01_Right_Control-
ling_Area *
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Coefficient_For_Threshold;

// Left ja Right suuntien yhdistelmä säätökäyrät (Line01 ja Expline01)
IF GVL_JOYSTICKS.reaProporoll_Left_01_Left_Line01 > 0 THEN
            IF GVL_JOYSTICKS.reaProporoll_Left_01_Left_Line01 < GVL_JOYSTICKS.lreaProporoll_Left_01_Left_Expline01 THEN
                        GVL_JOYSTICKS.lreaProporoll_Left_01_Left_Curve_Out := GVL_JOYSTICKS.reaProporoll_Left_01_Left_Line01;
            ELSE
                        GVL_JOYSTICKS.lreaProporoll_Left_01_Left_Curve_Out := GVL_JOYSTICKS.lreaProporoll_Left_01_Left_Expline01;
            END_IF
ELSE
            GVL_JOYSTICKS.lreaProporoll_Left_01_Left_Curve_Out := 0;
END_IF


IF GVL_JOYSTICKS.reaProporoll_Left_01_Right_Line01 > 0 THEN
```

```
            IF GVL_JOYSTICKS.reaProporoll_Left_01_Right_Line01 < GVL_JOYSTICKS.lreaProporoll_Left_01_Right_Expline01 THEN
                        GVL_JOYSTICKS.lreaProporoll_Left_01_Right_Curve_Out := GVL_JOYSTICKS.reaProporoll_Left_01_Right_Line01;
            ELSE
                        GVL_JOYSTICKS.lreaProporoll_Left_01_Right_Curve_Out := GVL_JOYSTICKS.lreaProporoll_Left_01_Right_Expline01;
            END_IF
ELSE
            GVL_JOYSTICKS.lreaProporoll_Left_01_Right_Curve_Out := 0;
END_IF

===========
PROGRAM HBRIDGE_CONTROL
VAR
            FB_ifmOutHBridge : ifmOutHBridge.HBridge;
            OUT_H_BRIDGE: ifmOutHBridge.MODE_HBRIDGE;
            BRAKE_OFF: ifmOutHBridge.MODE_BRAKE;
END_VAR
------------
// Peukalopyörän säätöarvo H-sillan kautta PWM-ulostuloon
GVL_OUTPUTS.uintLeft_01_Left_PwmOutput := LREAL_TO_UINT(GVL_JOYSTICKS.lreaProporoll_Left_01_Left_Curve_Out *
            GVL_JOYSTICKS.reaProporoll_Left_01_Left_Get_Output_1000);

GVL_OUTPUTS.uintLeft_01_Right_PwmOutput := LREAL_TO_UINT(GVL_JOYSTICKS.lreaProporoll_Left_01_Right_Curve_Out *
            GVL_JOYSTICKS.reaProporoll_Left_01_Right_Get_Output_1000);

IF GVL_JOYSTICKS.bProporoll_Left_01_Left_Direction THEN
            GVL_OUTPUTS.uintLeft_01_Common_PwmOutput := GVL_OUTPUTS.uintLeft_01_Left_PwmOutput;
ELSIF GVL_JOYSTICKS.bProporoll_Left_01_Right_Direction THEN
            GVL_OUTPUTS.uintLeft_01_Common_PwmOutput := GVL_OUTPUTS.uintLeft_01_Right_PwmOutput;
END_IF

FB_ifmOutHBridge(
            // Output parameters
            xResetError := FALSE,
```

```
            uiChannel := 6,
            eMode := OUT_H_BRIDGE,
            uiFrequency := 2000, // PWM-venttiilissä käytetty taajuus on esim. >= 110 Hz (tässä lineaaritoimilaitetta varten se on 2000
Hz)
            xDirection := GVL_JOYSTICKS.bProporoll_Left_01_Left_Direction, // Suunta
            eBrakeMode := BRAKE_OFF,
            uiBrakeValue := 0,
            tBrakeTime := T#0MS,
            uiValue := GVL_OUTPUTS.uintLeft_01_Common_PwmOutput,
            // Output parameters
            xError =>,
            eDiagInfo =>,
            xPrepared =>,
            uiOutCurrent =>);

// ===========
```