

Toni Telin

# KÄRKIOHJAUSJÄRJESTELMÄN KÄYTTÖÖNOTTOPROSESSIN KEHITYS

Diplomityö  
Informaatioteknologian ja viestinnän tiedekunta  
Lokakuu 2023

# TIIVISTELMÄ

Toni Telin: Kärkiohjausjärjestelmän käyttöönottoprosessin kehitys  
Diplomityö  
Tampereen yliopisto  
Sulautetut järjestelmät  
Syyskuu 2023

---

Tässä työssä tutkittiin mahdollisuuksia Technion Oy:n kehittämän xCrane-ohjausjärjestelmän käyttöönottoprosessin kehittämiseen. Tämänhetkinen käyttöönottoprosessi vaatii tietokoneen yhdistämistä järjestelmään joko etänä tai läsnä. Käyttöönottoprosessi myös vaatii CODESYS kehitysympäristön käytön hallitsevan henkilön. Käyttöönottoprosessissa yhteyttä tietokoneeseen vaativat osat ovat antureiden poikkeaman määrittäminen sekä liikenopeuksien mittaaminen. Työ on toteutettu yhteistyössä Technion Oy:n kanssa.

Työn teoriaosuudessa tutustuttiin metsäkoneisiin ja niiden puomien ohjaukseen. Näiden lisäksi tutustuttiin metsätyökoneiden työn kannalta olennaisten sulautettujen järjestelmien toimintaan. Viimeisenä perehdyttiin xCrane-järjestelmään ja sen komponentteihin.

Työssä kehitettiin järjestelmän käyttöliittymää niin, että järjestelmän antureiden poikkeamat voidaan määrittää käyttöliittymän avulla. Käyttöjärjestelmää kehitettiin myös niin, että järjestelmän liikenopeuksia mitattaessa voidaan tulokset näyttää järjestelmän näytöllä. Näytöllä voidaan myös kuvaajan avulla vertailla mitattuja arvoja venttiilin vastekäyrän arvoihin.

Työssä päästiin tavoitteisiin simulaattoriympäristössä. Kehitetty järjestelmä on valmis testattavaksi oikean nosturin kanssa.

Avainsanat: Käyttöönottoprosessi, kärkiohjaus, metsätyökone, CAN, CANopen

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

# ABSTRACT

Toni Telin: Development of the Boom Control System Implementation Process  
Master of Science Thesis  
Tampere University  
Embedded Systems  
September 2023

---

This work investigated possibilities for improving the implementation process of the xCrane control system developed by Technion Ltd. The current implementation process requires connecting a computer to the system either remotely or in person. Additionally, the implementation process necessitates the use of the CODESYS development environment by a knowledgeable individual. The parts of the implementation process that require a connection to the computer involve defining sensor offsets and measuring motion speeds. This work was done in collaboration with Technion Ltd.

The theoretical part of the work introduced forestry machines and their boom control. In addition to this, we familiarized ourselves with the operation of embedded systems that are used in forestry machinery and are essential for this thesis. Finally, we delved into the xCrane system and its components.

In this work, the system's user interface was developed so that sensor offsets can be defined using the interface. The operating system was also enhanced to display the results when measuring the system's motion speeds on the system's screen. The screen can also be used, with the help of a graph, to compare the measured values to the valve response curve values.

The objectives were achieved in a simulator environment. The developed system is ready to be tested with an actual crane.

Keywords: Implementation Process, Boom Control, Forestry Machine, CAN, CANopen

The originality of this thesis has been checked using the Turnitin Originality Check service.

# ALKUSANAT

Tämä työ on tehty yhteistyössä Technion Oy:n kanssa. Iso kiitos Tommi Sairolle, joka oli työni ohjaaja Technionin puolelta. Tommilta sain apua ja ohjeistusta aina pyydettyä ja mikä tärkeintä, hän auttoi aina mielellään. Kynnys avun pyytämiseen oli pieni, mikä mahdollisti työn valmistumisen kohtuullisen tiukalla aikataululla. Haluan lisäksi kiittää esihenkilöäni Maria Kivistä, jonka antama henkinen tuki oli arvokasta prosessin aikana. Iso kiitos myös Antti Pasaselle, joka palkkasi minut 2021: olit ensimmäinen esimieheni tällä alalla ja merkityksesi on äärimmäisen suuri siihen, missä olen tällä hetkellä perheeni kanssa. Onnea ja menestystä uusiin tehtäviisi muualla, ehkä vielä törmätään!

Kiitos Pilpalan alakoulun, Lopen yläkoulun ja lukion opettajat. En ollut hyvä oppilas, osa teistä kutsui minua ihan syystä luokan pelleksi, mutta siitä huolimatta onnistuitte takomaan jotain umpiluuhun. Se mahdollisti sen, että olen tässä pisteessä. Kiitos myös kaikille professoreille Tampereen Yliopistosta, erityismaininta tämän työn tarkastajalle Jukka Vanhalalle. Erinomaisten kurssien pitämisen lisäksi olit minulle varmasti paras mahdollinen työn tarkastaja.

Iso kiitos äidille, isälle, siskolle ja veljelle. Kotona, tai nykyään mummolassa, on ollut aina mukava käydä ja saada mieli pois töistä ja opiskeluista. Kiitos myös mummolle. Seurasit opintojen läpi uteliaana, miten se poika pärjää ja kai tässä jotenkin pärjättiin. Kiitän myös vaimoni Lotan perhettä, ilman mummin lapsenvahtiapua en valmistuisi vielä. Iso osa työstä on kirjoitettu niin, että mummi vahti taaperoa mahdollistaen minulle kirjoitusrauhan.

Valtava kiitos kotiin vaimolle Lotalle. Olet jo vuosia seurannut sivusta tätä minun touhuani ja olet nostonut itsetuntoani aina tarvittaessa. Kiitos isän prinsessat Saaga ja Selja, ette osaa puhua, ettekä varsinkaan osaa auttaa käyttöliittymän ohjelmoinnissa. Mutta te osaatte parhaiten tuoda minut maan pinnalle ja muistutatte, mikä on oikeasti tärkeää elämässä.

Viimeisenä vielä todella iso kiitos Papalle, joka oli vuosia elämäni tukipilari. Noin 12 vuoden ajan kävin lähes päivittäin juoden sinun ja mummon kanssa päiväkahvit. Päivieni paras hetki. Ehdit tutustumaan vaimooni ja ilmeisesti ihan piditkin hänestä. Ehdit näkemään Saagan ja rakastit häntä välittömästi. Et ehtinyt tapaamaan Seljaa etkä näkemään valmistumistani, mutta sinut tuntien tiedän mitä ajattelisit näistä hetkistä. Kuten tapasit aina sanoa, kiitos käynnistä.

Lopella 5.10.2023

Toni Telin

# SISÄLLYSLUETTELO

1. JOHDANTO .....	1
1.1 Technion Oy .....	2
2. METSÄTYÖKONE .....	3
2.1 Hydraulitekniikka ja metsätyökoneen rakenne .....	4
2.2 Puomin sähköinen ohjaus .....	6
2.3 CAN .....	7
2.3.1 CAN-väylän toiminta .....	9
2.3.2 Viestikehykset .....	10
2.4 CANopen-protokolla .....	11
2.5 Anturit .....	13
2.5.1 IMU .....	13
3. KÄRKIOHJAUS .....	15
3.1 Perinteinen ohjaus .....	15
3.2 Kärkiohjauksen toiminta .....	18
3.3 Kärkiohjauksella saavutettavat hyödyt .....	19
4. XCRANE-JÄRJESTELMÄ .....	21
4.1 Järjestelmän rakenne .....	21
4.2 Käyttöönottoprosessi .....	24
5. KEHITYSYMPÄRISTÖ .....	27
5.1 Kehitysympäristö .....	27
5.1.1 IEC-61131-3 Standardi .....	27
5.2 Testiympäristö .....	28
6. KÄYTTÖÖNOTTOPROSESSIN KEHITYS .....	31
6.1 Tutustuminen järjestelmään .....	31
6.2 CANopen xCrane PRO-järjestelmässä .....	33
6.3 Antureiden poikkeaman määrittäminen .....	34
6.3.1 Kontrollerin ohjelmointi .....	34
6.3.2 Käyttöliittymän ohjelmointi .....	39
6.4 Liikenopeuksien kalibrointi .....	41
6.4.1 Näytön ohjelmointi .....	41
6.4.2 Käyttöliittymän ohjelmointi .....	49
7. TULOSTEN ANALYSOINTI .....	51
7.1 Testaaminen simulaattorilla .....	51
7.2 Johtopäätökset .....	54
7.3 Tulevaisuuden jatkokehitys .....	54
LÄHTEET .....	56

## LYHENTEET JA MERKINNÄT

CAN	engl. Controller Area Network, sarjaliikenneväylä
CiA	engl. CAN in Automation, CAN väyläteknologiaa kehittävä organisaatio
COB-ID	engl. Communication Object Identifier, viestintäobjektin tunniste
DOF	engl. Degrees of freedom, puomin liikesuunnat
eds	engl. electronic data sheet, sähköinen tietolomake
GVL	engl. Global variable list, globaali muuttujalista
IMU	engl. Inertial Measurement Unit, inertian mittaussyksikkö
NRZ	engl. Non Return to Zero, digitaalisen viestinnän koodausmenetelmä
OSI	engl. Open Systems Interconnection Reference Model, tiedonsiirtoprotokollien yhdistelmä seitsemässä kerroksessa
PDO	engl. Process Data Object, prosessidatan objekti
PLC	engl. Programmable Logic Controller, ohjelmoitava logiikkakontrolleri
RPDO	engl. Receive PDO, vastaanotettava PDO-viesti
SAE	engl. Society of Automotive Engineers, autoalan standardisointijärjestö
SDO	engl. Service Data Object, palveludatan objekti
TPDO	engl. Transfer PDO, siirrettävä PDO-viesti

# 1. JOHDANTO

Metsäteollisuus on pitkään ollut merkittävässä osassa monien maiden taloutta, tarjoten sekä raaka-aineita, että työllistäen ihmisiä. Teknologian kehityksen myötä myös metsäteollisuuden työvälineet ja koneet ovat kehittyneet. Nykyiset metsäkoneet, joita käsitellään luvussa 2, ovat hyvin kehittyneitä ja työskentelyolosuhteet ovat verrattuna menneisyyteen huomattavasti mukavammat. Työskentelyolosuhteiden lisäksi myös koneiden tehokkuus on kasvanut huomattavasti ja yhdellä koneella voidaan suorittaa useampia työvaiheita.

Moderneissa metsäkoneissa koneen puomin ohjausta on kehitetty ja uuden ohjaustavan nimi on kärkiohjaus. Kärkiohjauksella on pyritty tekemään metsäkoneen puomin ohjauksesta helpompaa ja yksinkertaisempaa, mikä lisää työn tehokkuutta, sekä tekee uusien kuljettajien kouluttamisesta helpompaa ja nopeampaa.

Tämän diplomityön tarkoituksena on tutkia ja kehittää Technion Oy:n kehittämän *xCrane*-ohjausjärjestelmän käyttöönottoprosessia. Työn tavoitteena on ensisijaisesti tunnistaa käyttöönottoprosessin nykyiset haasteet sekä tarjota ratkaisut näihin haasteisiin. Tavoitteena on vähentää käyttöönottoprosessin vaatimia työtunteja, tehden käyttöönottoprosessista taloudellisempaa. Tavoitteena on käyttää hyödyksi järjestelmän käyttöliittymää ja luoda käyttöönottoprosessista selkeä. Työssä tutkitaan, onko käyttöönottoprosessi mahdollista toteuttaa käyttäen järjestelmän käyttöliittymää, niin että käyttöönottoprosessi voidaan suorittaa nopealla koulutuksella.

Tällä hetkellä käyttöönottoprosessi vaatii lähtökohtaisesti kahden henkilön työpanoksen. Käyttöönottoprosessi onnistuu myös yhdeltä henkilöltä, mutta tällöin vaaditaan kehitysympäristön käytön hallitsevaa henkilöä, joka suorittaa käyttöönoton koneen luona. Työn tavoitteena on vähentää tarvetta erityisosaamiselle ja ylipäätään vähentää tarvetta yhdistää järjestelmää tietokoneeseen. Tavoitteena on mahdollistaa käyttöönottoprosessin suorittaminen järjestelmän oman käyttöliittymän avulla.

Kehittämisen kohteina ovat antureiden poikkeaman määrittäminen sekä liikenopeuksien määrittämisen mahdollistaminen käyttöliittymällä. Käyttäjän tulee myös pystyä muokkaamaan liikenopeuksien kalibroinnissa mitattavien pisteiden ohjausarvoja. Liikenopeuksien kalibroinnissa tulee luoda mahdollisimman tarkka kuvaaja, mistä

käyttäjä voi seurata saavutetaanko konfiguraatiodostossa olevat arvot tarpeeksi tarkasti.

Tavoitteena on luoda käyttöönottoprosessin käyttöliittymästä mahdollisimman selkeä, jotta prosessi olisi mahdollista suorittaa mahdollisimman vähällä opastuksella. Käyttöliittymän tulee varmistaa antureiden poikkeamia määrittäessä, että kyseinen liike on ajettu ääriasentoon.

Työssä tutustutaan metsätyökoneisiin, niiden hydraulikkaan sekä sähköiseen ohjaukseen. Sen jälkeen syvennyttään metsätyökoneista löytyvään teknologiaan, kuten CAN-väylään (engl. Controller Area Network), CANopen-protokollaan sekä antureihin, joilla voidaan määrittää puomin asento ja liikenopeudet.

Viimeisenä teoriaosuutena, ennen kuin tutustutaan xCrane-ohjausjärjestelmään, on perehtyminen kärkiohjaukseen, sen toimintaperiaatteeseen ja sillä saavutettaviin hyötyihin. xCrane-järjestelmässä käydään läpi käytettävät komponentit sekä järjestelmän rakenne, minkä jälkeen perehdyttään nykyiseen käyttöönottoprosessiin.

Tämän jälkeen tutustutaan tarkemmin kehitysympäristöön, jonka avulla tutkimus suoritetaan. Samalla tutustutaan myös käytössä olevaan simulaattoriin, jonka avulla tutkimus suoritetaan. Viimeisenä käydään läpi käyttöönottoprosessin kehitystä, minkä jälkeen analysoidaan tuloksia, luodaan johtopäätökset ja pohditaan tulevaisuuden jatkokehityskohteita.

## **1.1 Technion Oy**

Tämä työ suoritetaan yhteistyössä Technion Oy:n kanssa. Technion Oy suunnittelee ja valmistaa ohjausjärjestelmäratkaisuja liikkuvia työkoneita valmistaville yrityksille. Koska Technion Oy on osa saksalaista HYDAC-ryhmää, Technion Oy vastaa HYDAC-ohjausjärjestelmistä Suomessa ja Ruotsissa, metsäkonejärjestelmistä maailmanlaajuisesti sekä johtosarjojen valmistuksesta ja kokoonpanosta. [1]

xCrane PRO on Technion Oy:n suunnittelema ja valmistama kärkiohjausjärjestelmä, joka helpottaa metsätyökoneen puomin ohjausta. Järjestelmä sopii tehdasasennettavaksi useisiin metsätyökoneisiin. [2]

## 2. METSÄTYÖKONE

Liikkuva työkone on moottorikäyttöinen kone, joka liikkuu pyörillä tai telaketjuilla. Koneita ohjataan ohjaamosta tai kauko-ohjauksella. Työkoneessa on toimilaitteita, joilla haluttu työ tehdään. Toimilaitte voi olla esimerkiksi kauha tai tartuntakoura.

Metsätyökoneella tarkoitetaan liikkuvaa työkoneita, joka suorittaa metsän käsittelyyn liittyviä työtehtäviä, kuten hakkuut ja puiden korjuu. Metsätyökoneita on monenlaisia, kuten hakkuukone, kuormatraktori ja yhdistelmäkone. Metsätyökoneena voi toimia myös esimerkiksi traktori, johon on kiinnitetty esimerkiksi nosturilla varustettu perävaunu. Kuvassa 1 on esitetty Kronos Kargo 121 metsävaunu kiinnitettynä traktoriin.



**Kuva 1.** Kronos Kargo 121 metsävaunu [1]

Puunkorjuun koneellistaminen on lähtenyt yksittäisten työvaiheiden koneellistamisesta. Tästä koneet ovat kehittyneet niin, että yksi metsätyökone voi suorittaa useamman työvaiheen, kuten puiden kaadon ja kasauksen. [2]

Nykyaikaisissa työkoneissa on lukuisia sulautettuja järjestelmiä, mutta monet niistä eivät ole tämän työn kannalta olennaisia. Tässä työssä keskitytään puomin ohjaukseen liittyviin sulautettuihin järjestelmiin.

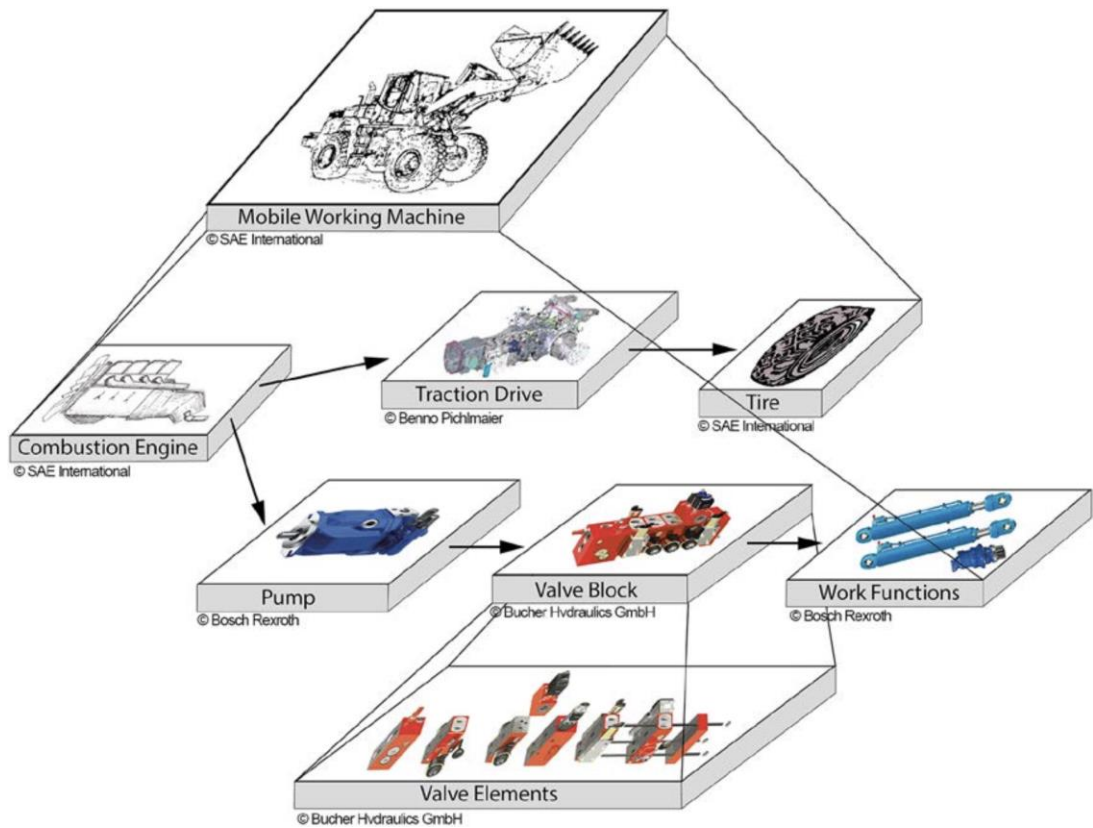
Tässä luvussa keskitytään työn kannalta olennaisiin osiin metsätyökoneesta, kuten hydrauliteknikkaan ja puomin rakenteeseen. Tämän lisäksi käsittelemme työn kannalta olennaisia metsätyökoneissa olevaa teknologiaa. Työssä suuressa osassa on tiedonsiirto, joka suoritetaan käyttämällä CAN-väylää. Tarkemmin tiedonsiirrossa

käytetään CANopen protokollaa, mihin perehdytään luvussa 2.4. Nykyaikaisessa metsäkoneessa on antureita, jotka tuottavat dataa sekä itse järjestelmälle, että kuljettajalle. Työn kannalta tärkeitä antureita ja niiden toimintaa käsitellään luvussa 2.5.

## **2.1 Hydraulitekniikka ja metsätyökoneen rakenne**

Metsätyökoneet käyttävät hydraulikkaa voiman tuottamiseen, tällaisia koneita kutsutaan mobilekalustoksi tai liikkuviksi työkoneiksi. Hydraulijärjestelmä on tehonsiirtoketju, joka muuttaa mekaanisen tehon hydrauliseksi. Hydrauliset järjestelmät ovat yleisiä, koska ne tarjoavat hyvän tehopainosuhteen. Hydraulijärjestelmät eivät ole myöskään sidoksissa ennalta määrättyyn tehonsiirtorataan, koska teho siirretään letkujen ja putkien kautta. Tämä tekee helpoksi tehon siirtämisen sinne, missä sitä tarvitaan. Hyvä tehopainosuhte mahdollistaa komponenttien pienen koon verrattuna muihin tehonsiirtojärjestelmiin. Hydraulijärjestelmän huonot ominaisuudet ovat kohtalainen hyötysuhde ja tehonsiirrossa väliaineena käytettävän nesteen huonot ominaisuudet. [3]

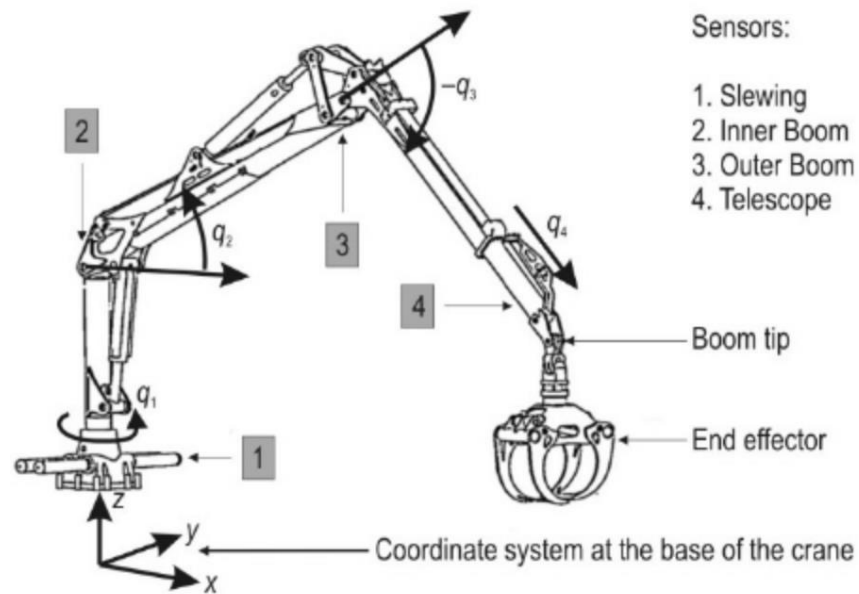
Liikkuvissa työkoneissa mekaanista tehoa tuottaa yleensä poltto- tai sähkömoottori. Moottorin tuottama pyörimisenergia muutetaan hydraulipumpulla tilavuusvirraksi ja paineeksi. Tilavuusvirtaa ohjataan venttiileiden avulla haluttuihin toimilaitteisiin, kuten sylintereihin tai moottoreihin. Huomioitavaa on se, että samasta mekaanista energiaa tuottavasta moottorista kyetään käyttämään energiaa myös työkoneen liikuttamiseksi, kuten kuvasta 2 voimme nähdä. [4]



**Kuva 2.** Liikkuvan työkoneen rakenne

Metsätyökone voidaan jakaa kolmeen hydraulijärjestelmään: ajovoimansiirtoon, ohjausjärjestelmään ja työhydrauliikkaan. Ohjausjärjestelmä vastaa toimilaitteiden ohjauksesta ja hallinnasta. Työhydrauliikan ja ajovoimansiirron tehtävä on muuntaa mekaaninen energia hydrauliseksi energiaksi. Hydraulinen energia välitetään putkien ja letkujen avulla toimilaitteille ohjausjärjestelmän kuljettamien käskyjen mukaisesti. [5]

Metsätyökoneen puomia voidaan ohjata neljään eri suuntaan, eli kyseessä on 4-DOF (engl. degrees of freedom) puomi. Puomin rakenne on esillä kuvassa 3. Mahdolliset liikesunnat on kuvattu nuolilla.



**Kuva 3.** Metsätyökoneen puomin rakenne [6]

Puomin päässä oleva toimilaitte vaihtelee metsäkoneissa työtehtävän mukaan, kuvassa 3 olevassa puomissa on kiinni tartuntakuora, jota kutsutaan myös kahmariksi. Toinen yleinen vaihtoehto on harvesteripää, jolla voidaan kaataa puita pinoamisen lisäksi.

## 2.2 Puomin sähköinen ohjaus

Puomin sähköinen ohjausjärjestelmä koostuu ohjainkahvoista, ohjausyksiköstä ja sähköisesti ohjatuista venttiileistä. Tämän lisäksi puomissa voi olla antureita mittaamassa puomin asentoa ja liikenopeuksia, jos on tarve esimerkiksi säätää ohjausta takaisinkytkentää käyttäen.

Ohjainkahvoista tuleva ohjaussignaali voi olla joko analoginen tai digitaalinen, joka välitetään ohjausyksikölle. Ohjausyksikkö suorittaa ohjausalgoritmin annetun signaalin ja mahdollisen anturitiedon perusteella. Lopputuloksena ohjausyksiköstä lähtee ohjaus venttiileille, mikä päästää tilavuusvirtaa halutulle toimilaitteelle, metsäkoneen puomin ohjauksen tapauksessa sylinterille. [7]

Nykyään järjestelmissä on usein liitettynä näyttö, jonka käyttöliittymä mahdollistaa kommunikoinnin käyttäjän ja järjestelmän välillä. Kuvassa 4 on esiteltyä *xCrane PRO*-ohjausjärjestelmän käyttöliittymä.



**Kuva 4.** xCrane PRO-ohjausjärjestelmän käyttöliittymä [8]

Puomin sähköinen ohjaus varustettuna antureilla mahdollistaa muitakin ohjaustapoja kuin perinteisen ohjauksen. Perinteisessä ohjauksessa koneen käyttäjä ohjaa puomin jokaista neljää liikettä erikseen ohjainkahvoilla. Metsätyökoneen puomin ohjauksessa käytetään kahta ohjainkahvaa, joista molemmilla voidaan ohjata kolmea toimilaitetta. Kuljettaja voi siis ohjainkahvoilla ohjata puomin lisäksi kahta muuta toimilaitetta. Jos puomiin on kiinnitetty kuvan kolme mukainen tartuntakoura, nämä kaksi muuta toimilaitetta avaa ja sulkee kouran sekä pyörittää kouraa myötä- ja vastapäivään.

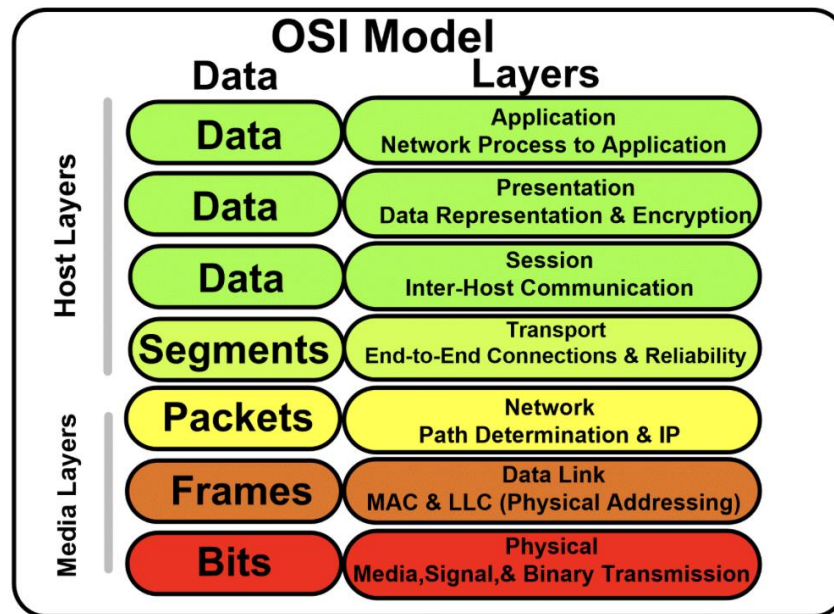
Puomien ohjausjärjestelmien kehitys on mahdollistanut sen, että perinteisen ohjauksen sijaan kuljettaja voikin ohjata puomin kärkeä haluttuun suuntaan [7]. Tällöin, jos kuljettaja esimerkiksi haluaa ohjata puomin kärkeä kauemmas koneesta, ohjausyksikkö määrittää tarvittavat ohjaukset toimilaitteille. Tätä ohjaustapaa kutsutaan kärkiohjaukseksi. Perinteisen- ja kärkiohjauksen toimintaan, sekä puomin rakenteeseen perehdytään lisää luvussa 3.

## 2.3 CAN

CAN on automaatiassa laajasti käytetty sarjaliikenneväylä. CAN-väylä on hierarkkisesti järjestetty hajautettu sarjaliikenneväylä reaaliaikaisiin sovelluksiin [9]. Vuonna 2007 arvioitiin, että vuosittaisella tasolla valmistettiin 400 miljoonaa mikrokontrolleria, jotka käyttivät CAN-väylää. [10]

CAN esiteltiin 1986 Detroitissa SAE:n (engl. Society of Automotive Engineers) kongressissa. CAN kehitettiin alun perin käytettäväksi henkilöautoissa, mutta nopeasti käyttökohteet levisivät myös muualle. [11] CAN on yleisesti käytössä työkoneissa.

Tiedonsiirtoprotokollia voidaan kuvata OSI-mallia (engl. Open Systems Interconnection Reference Model) käyttämällä. OSI-malli on ollut käytössä 80-luvulta asti ja on edelleen yleisessä käytössä oleva tietoliikenteen tarkasteluvitekehys. OSI-mallin rakenne on esitetty kuvassa 5. [12]



Kuva 5. OSI-mallin kerrokset [13]

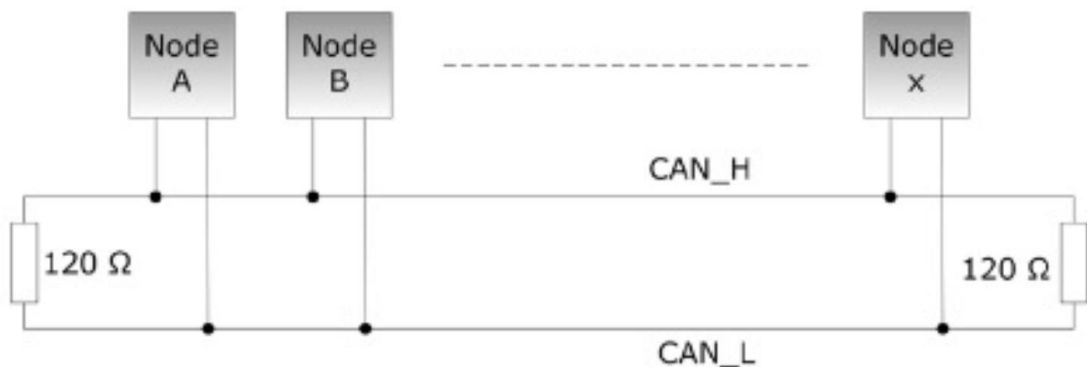
CAN-väylä käyttää kolmea kerrosta OSI-mallista, jotka ovat kuvan 5 kerrokset *physical*, *data link* ja *application*. Suomeksi puhutaan fyysisestä, siirtoyhteys- ja sovelluskerroksesta. Fyysinen kerros välittää bitit laitteiden välillä ja siirtoyhteyskerros kehystää datan. CAN-viestin kehys esitetään luvussa 2.3.2. Sovelluskerroksella lähetettyä dataa muokataan ja hallitaan yhteyttä, samalla tasolla on myös sisältöä ja formaattia määrittelevä protokolla. CANopen -protokollan toimintaa esitetään luvussa 2.4. [12] [14]

CAN-väylän tärkeimmät ominaisuudet ovat luotettavuus, saatavuus ja kestävyys. Tämän takia CAN-väylän käyttö on yleistä muun muassa työkoneissa. Myös mahdolliset virheelliset viestit havaitaan helposti. Jos ympäristössä on paljon kohinaa, järjestelmä on helposti eristettävissä, mikä lisää luotettavuutta entisestään. [9]

Kun CAN-väylät yleistyivät autoteollisuudessa, *high-speed CAN* (suomeksi *korkean nopeuden CAN*) esiteltiin kansainvälisesti ISO 11898:na. Tämä mahdollisti 1 Mbit/s nopeuden. Myöhemmin esiteltiin myös pienen nopeuden CAN ja yhden kaapelin CAN. [15]

### 2.3.1 CAN-väylän toiminta

CAN-väylä koostuu sitä käyttävistä laitteista, joita kutsutaan myös solmuiksi (engl. node). CAN-väylässä on vain kaksi kaapelia, CAN\_H ja CAN\_L, jotka tulee suojata häiriöiltä esimerkiksi kiertämällä. Teoriassa väylään voidaan liittää jopa 2032 laitetta, mutta käytännössä lähetinvastaanottimien hallitsevista kontrollereista johtuvat rajoitukset laskevat liitettävien moduulien määrää. Väylän topologia on esitetty kuvassa 6. CAN-väylässä kaikki laitteet ovat yhdenvertaisia ja väylän rakenne mahdollistaa sen, että siihen voidaan lisätä tai siitä voidaan poistaa moduuleja häiritsemättä viestiliikennettä väylällä. Kuten kuvasta 6 voidaan myös havaita, jokainen laite näkee jokaisen viestin väylällä. [9]



**Kuva 6.** CAN-väylän topologia [16]

Kuvassa 6 väylän molemmissa päissä on 120 ohmin terminointivastukset. Niiden tehtävänä on estää lähetettyjen signaalien heijastuminen, jota voi tapahtua viestejä kuljettavan kaapelin epätäydellisyyksien vuoksi. Epätäydellisyydet aiheuttavat sen, ettei kaapelin toiminta ole signaalin näkökulmasta lineaarista ja kaapelissa voi olla impedanssin muutoksia. Terminoimalla molemmat päät väylästä saavutetaan luotettavuuden lisäksi suurin mahdollinen tiedonsiirtonopeus ja väylän fyysinen pituus. Tällöin väylän rakenne on mahdollisimman lähellä yksikaapelista rakennetta. [17]

CAN-signaali on digitaalinen NRZ-enkoodattu (engl. Non Return to Zero) signaali, mikä tarkoittaa sitä, kun kaksi peräkkäistä lähettyvää bittiä ovat molemmat "1" ei signaalia käytetä nollassa bittien välillä. Tämä vähentää tarvittavien transitioiden määrää ja tekee väylästä vähemmän herkän ulkoisille häiriöille. [17]

CAN-väylään liitettävä laite sisältää controllerin viestiliikenteen hallintaan tarkoitetun controllerin sekä lähetinvastaanottimen. Vastaanottimena toimiessaan se muuntaa väylältä luetun sähköisen signaalin digitaalseksi controllerin luettavaksi ja päinvastoin. Tämän lisäksi lähetinvastaanotin kykenee havaitsemaan mahdolliset viat väylässä, kuten oikosulun tai katkenneen kaapelin. [17]

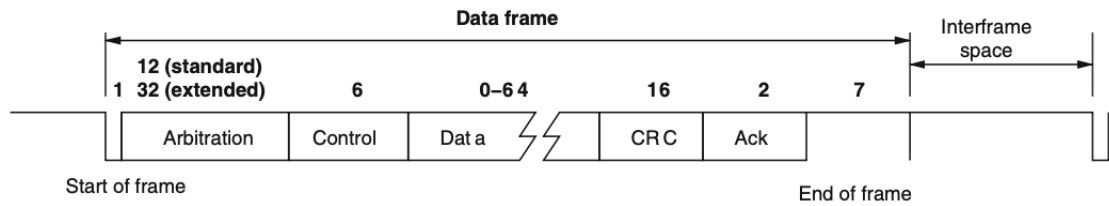
Solmuille ei anneta osoitteita CAN-väylässä, mutta viestikehyksissä on osoite, josta voidaan päätellä, tarvitseeko laitteen välittää viestin sisällöstä vai ei. Viestit eivät itsessään sisällä lähettävän tai vastaanottavan laitteen osoitetta. Esimerkiksi samassa väylässä voi olla liitettyä useita eri antureita, jotka näkevät toistensa lähettämän datan, mutta todennäköisesti eivät käytä tietoa mihinkään. Vastaavasti väylään voi olla liitettyä kontrolleri, joka kyselee yksittäiseltä tai useammalta anturilta dataa samanaikaisesti. Viestien tunnisteessa on määritelty viestille prioriteetti, mitä pienempi on tunnisteiden binääriluvun arvo, sitä korkeampi on viestin prioriteetti. [9]

CAN-arbitraatioprotokolla on prioriteettiin perustuva sekä ei-pre-emptiivinen. Tarkoittaen sitä, että jo lähetyksessä olevaa viestiä ei voida keskeyttää korkeamman prioriteetin viesteillä. Kun solmu haluaa lähettää viestin, se tarkastelee, onko väylä vapaana. Tämän jälkeen alkaa niin sanottu kilpailuvaihe. Solmu, joka haluaa lähettää viestin, aloittaa tunnisteiden lähettämisen, jos jokin toinen solmu lähettää tunnistetta myös, eriävät bitit ratkaistaan loogisella ja-ehdolla. Jos solmu havaitsee, ettei sen lähettämä tunniste muutu väylällä, se tietää voittaneensa kilpailuvaiheen ja pystyy aloittamaan lähettämisen väylälle. Tämä tarkoittaa myös sitä, ettei kahdella laitteella voi olla samaa tunnistetta. [9] [17]

### 2.3.2 Viestikehykset

CAN-väylällä voidaan lähettää neljää erilaista viestikehystä, jotka ovat *data-*, *remote-*, *error-* ja *overload frame*. Suomennettuna frame tarkoittaa kehystä ja yleisesti puhutaan viestikehyksistä. Datakehyksellä lähetään dataa vastaanottajalle, joita voi olla yhdelle viestille useita. Remote-kehyksellä pyydetään dataa toiselta laitteelta. Error-kehys lähetetään aina, kun jokin laite havaitsee virheellistä toimintaa väylällä ja overload-kehyksellä pyydetään pientä viivästystä ennen seuraavan data- tai remote-kehysten lähettämistä. [9] [17]

Datakehyksellä on lähettäjä ja yksi tai useampi vastaanottaja. Kehys ei sisällä tietoa mille solmulle tietoa lähetetään, vaan jokainen solmu tietää itse millä tunnisteella olevista viesteistä sen tarvitsee välittää. Riippuen protokollasta tunniste on joko 11-bittiä tai 29-bittiä pitkä. Datakehysten formaatti on esitetty kuvassa 7. [17]



**Kuva 7. Datakehysten formaatti CAN-viestinnässä [17]**

Kuten kuvasta 7 huomataan, arbitraatioon käytettävien bittien määrä vaihtelee sen mukaan, käytetäänkö pidempää vai lyhyempää tunnustetta. Muuten viestin formaatti on sama. *Control*-bitit kertovat viestin tyypin, *data*-bitit sisältävät lähetettävän datan. Loppuosa kehyksestä sisältää tarkastussumman ja *acknowledge*-bitit, joilla todennetaan, että viesti on otettu onnistuneesti vastaan. Remote-kehyksellä on sama formaatti kuin datakehyksellä, sisältäen vain muutamia eroja. [17]

Error-kehys lähetetään, kun jokin solmuista havaitsee virheellistä toimintaa väylällä. Kehyksessä on osa varattu virhelipulle, jonka bitit määrittävät errorin, ja virheen erotinalueelle. Erotinalue koostuu kahdeksasta resessiivisestä bitistä ja se mahdollistaa viestiliikenteen käynnistämisen virheen poistumisen jälkeen. Overload-kehyksellä on sama formaatti kuin error-kehyksellä. Erona on se, että overload-kehysten lähetyks voidaan aloittaa kahden kehyksen välissä, kun taas error-kehys lähetetään viestin siirron aikana. Overload-kehys lähetetään, jos solmu havaitsee väylällä virheellisen arvon kehysten välillä tai jos solmu ei ole valmis aloittamaan seuraavan viestin vastaanottoa.

## 2.4 CANopen-protokolla

CANopen on CAN-väylää käyttävä viestintäprotokolla. CANopen kehitettiin sulautetuksi verkoksi, joka on konfiguroitavissa monenlaiseen käyttöön. Tämä saavutetaan korkeamman tason protokollien ja profiilimäärittelyjen avulla. Aluksi CANopenin käyttökohteeksi oli tarkoitettu liikkuvien työkoneiden ohjausjärjestelmät, mutta nykyään sitä käytetään monissa eri käyttökohteissa. Esimerkkejä CANopenin käyttökohteista on lääketieteelliset laitteet ja maastoajoneuvot. [18]

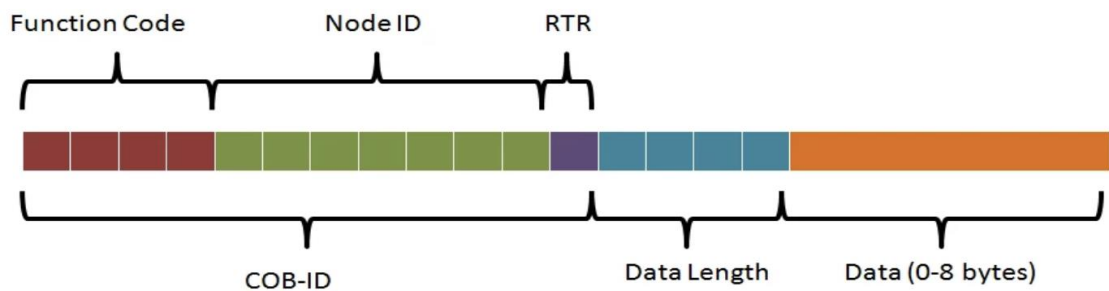
CANopen-protokollaa käyttävät laitteet ovat helppoja integroida CANopen-järjestelmään mikä helpottaa valmiiden laitteiden käyttöä. Standardisoidut rajapinnat mahdollistavat sen, että samaan järjestelmään voidaan vaivatta liittää eri laitevalmistajien tuotteita. Tästä syystä CANopen-protokollaa hyödyntäviä laitteita on laajalti saatavilla kohtuulliseen hintaan. [18]

ISO 11898 sisältää standardin CANopen lähetyksien vastaanottimille ja ohjaimille. CiA:n (engl. CAN in Automation) spesifikaatiot muodostavat aiemmin mainitut

profiilimäärittelyt. Spesifikaatio DS-301 määrittelee sovellustason ja kommunikaatioprofiilin, DSP-302 viestikehyksen ja kommunikaatioprofiilille ja DRP-303-1 suositukset kaapeloinnille ja liittimelle. [9]

CANopenissa alemmat kerrokset tarjoavat viestintäinfrastruktuurin korkeammille tasoille sekä objektit, joita käytetään kommunikointiin väylällä. Objektit määritellään objektisanakirjassa, jossa kerrotaan kaikkien objektien datatyypit. Sanakirjan avulla luodaan CANopen rajapinta sovellukselle. Jokainen laite ylläpitää omaa objektisanakirjaansa. [17] CANopen standardin mukaan, objektisanakirjassa tulee olla 16-bitin indeksointi, minkä lisäksi on vielä 8-bitin ali-indeksointi. Standardi myös määrittelee tietyille indekseille tiettyä dataa, esimerkiksi indeksillä 1008h ali-indeksillä 00h täytyy olla laitteen nimi tallennettuna. [19]

CANopenin viestiformaatti perustuu luvussa 2.3.2 esitettyyn datakehyksen formaattiin. Viestiformaatti on esitetty kuvassa 8. Databittejä lukuun ottamatta kuvassa 8 viestin eri osien bitit ovat eroteltuina. [19]



**Kuva 8.** CANopen viestiformaatti [19]

Kuten kuvasta 8 näemme, node id:lle on varattu 7 bittiä. Jokaisella väylään kytketyllä laitteella tulee uniikki node id. Käytännössä se tarkoittaa, että CANopen verkkoon voidaan liittää korkeintaan 127 solmua. COB-ID (engl. Communication Object Identifier) muodostuu kuvan 8 mukaisesti funktiokoodista ja node id:stä. Verkossa ei voi olla kahta laitetta samalla COB-ID:llä. [19]

Objektit voidaan jakaa kommunikaatio- ja sovellusobjekteiksi. Yleisesti käytetään lyhenteitä SDO (engl. Service Data Object) ja PDO (engl. Process Data Object). SDO- ja PDO-kommunikaatiossa käsitellään objektisanakirjassa olevaa dataa. SDO-kommunikaatiossa on *client* ja *server*. Client aloittaa jokaisen tiedonsiirron ja serveri pitää hallussaan objektisanakirjaa. Jokainen SDO-kommunikointi vaatii kahden CANopen viestin lähettämistä, ensimmäisessä viestissä client esittää pyynnön, johon server vastaa. Clientin lähettämässä pyynnössä on sen serverin CAN-ID, jolta tietoa halutaan. [17] [19]

PDO-kommunikaatiossa välitetään dataa, joka voi muuttua milloin vain. Tämän tyyppistä dataa ovat esimerkiksi sisääntulot, kuten anturitieto. Kuten SDO-kommunikaatiossa myös PDO-kommunikaatiossa data tallennetaan objektisanakirjaan, erona on ettei PDO-kommunikaatiossa tarvitse pyytää solmua lähettämään dataa, vaan solmu voi lähettää dataa milloin vain. PDO:ita on kahdenlaisia: TPDO:ita (engl. Transfer PDO) ja RPDO:ita (engl. Receive PDO). [19] Koska dataa voidaan lähettää milloin tahansa, vaikka jatkuvasti, puhutaan tuottaja/kuluttaja viestinnästä. Esimerkiksi anturi tuottaa jatkuvasti dataa, jota kontrolleri kuluttaa. jos tuottaja aloittaa kommunikoinnin, puhutaan TPDO:sta ja kuluttajan aloittaessa kommunikoinnin puhutaan RPDO:sta. [17]

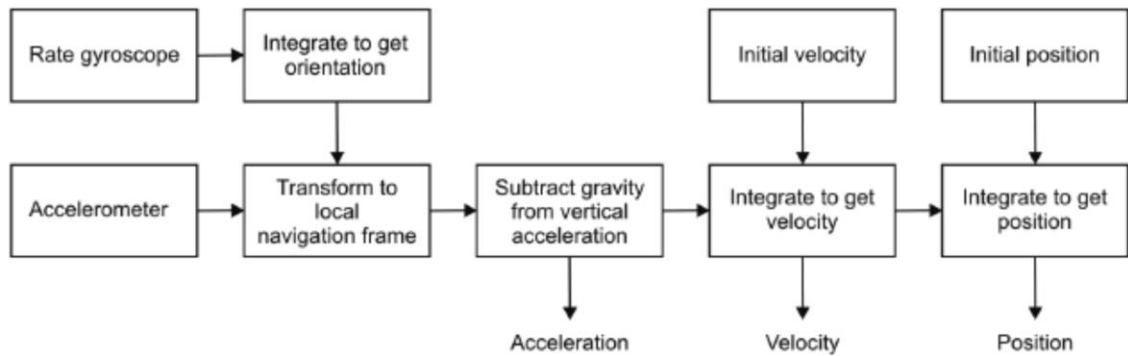
SDO- ja PDO-protokollien lisäksi CANopen sisältää erityisiä objektiprotokollia, kuten synkronisaatio- (SYNC), hätätila- (EMCY) ja aikaleimaprotokollat. Näiden lisäksi löytyy protokollia verkon hallintaan. [17]

## 2.5 Anturit

Nykyaikaisessa metsäkoneessa on lukuisia antureita. Antureilla voidaan mitata muun muassa järjestelmässä hydraulioöljyn painetta ja lämpöä, puomin asentoa tai kuljettajan läsnäoloa. Antureiden tuottamalla tiedolla voi olla useita käyttökohteita järjestelmässä. Tietoa voidaan käyttää vianetsintään ja toiminnan oikeellisuuden tarkkailuun, sekä toiminnan optimointiin. Tässä luvussa käsitellään työn kannalta oleellisia anturityyppejä.

### 2.5.1 IMU

IMU (engl. Inertial Measurement Unit) on gyroskooppeja ja kiihtyvyyssantureita käyttävä laite, joka mittaa suhteellista sijaintia, nopeutta ja kiihtyvyyttä. IMU mittaa kaikkia kuutta vapausastetta, eli  $x$ -,  $y$ - ja  $z$ -akselien suuntaisten liikkeiden lisäksi vaaka-, pysty- ja sivusuuntaista pyörimistä. IMU:n toimintaa kuvaava lohkokaavio on esillä kuvassa 9. [20]



**Kuva 9.** IMU:n toiminnan lohkokkaavio [20]

Kuten kuvasta 9 huomataan, IMU:lla voidaan mitata kuuden vapausasteen lisäksi nopeutta ja kiihtyvyyttä. Kuvasta voidaan huomata myös, jotta saadaan halutut suureet mitattua, tulee toteuttaa lukuisia laskutoimituksia. [21]

Gyroskooppien avulla voidaan suorittaa vaaka-, pysty- ja sivusuuntaisten pyörimisliikkeiden mittaaminen. Tämä voidaan toteuttaa mittaamalla kulmanopeutta. Gyroskoopin toiminta perustuu kulmaliikkeen momentin säilymisen lakiin. Gyroskooppeja on toteutettu eri tavoilla, esimerkiksi mekaniikkaa ja optiikkaa hyödyntämällä. [21]

Kiihtyvyyssantureilla mitataan laitteeseen kohdistuvia ulkoisia voimia. Täytyy huomioida, että kiihtyvyyssanturit voivat mitata voimia, vaikkeivat ne aiheuta liikettä, esimerkiksi painovoima on tällainen ulkoinen voima. Kiihtyvyyssantureita käytetään IMU:ssa mittamaan x-, y- ja z-suuntaisia liikkeitä. Kiihtyvyyssanturit ovat joko mekaanisesti toimivia tai piezoelektronisesti toimivia. [21]

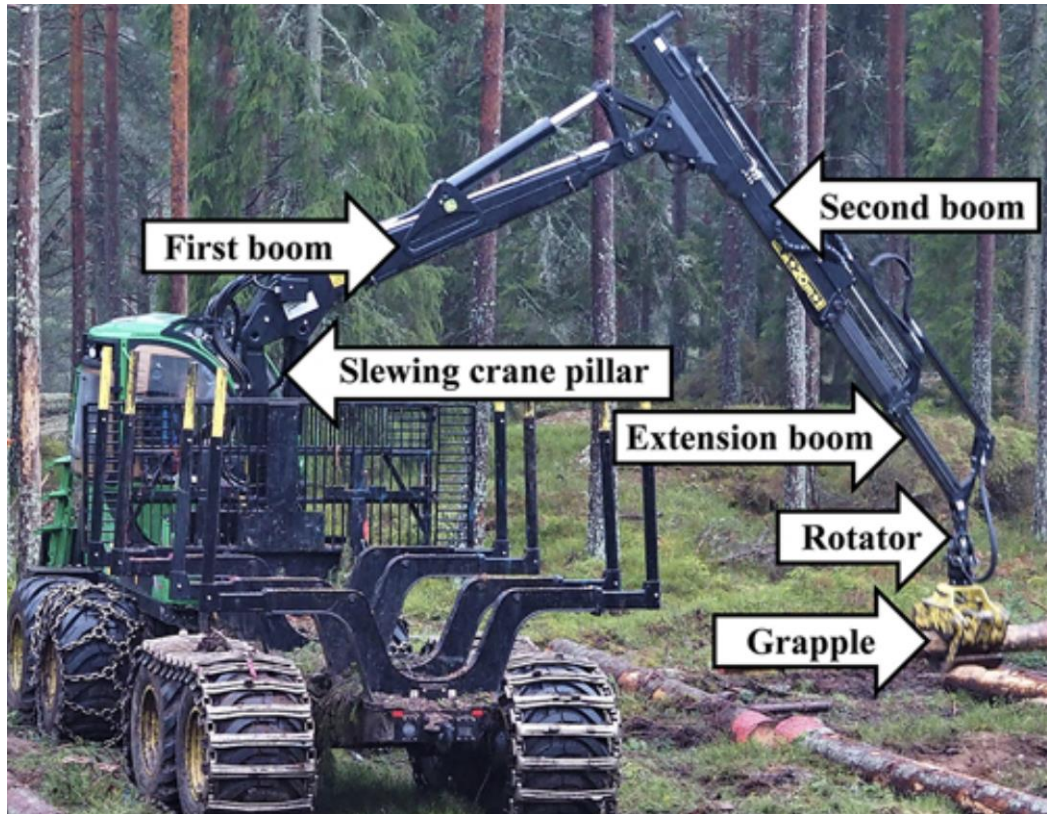
IMU:ja käytettäessä tulee huomioida, että ne ovat hyvin herkkiä mittausvirheille. Esimerkiksi gyroskooppien mittauservo alkaa väistämättä vaeltamaan, mistä seuraa virheellinen painovoimavektorin kumoaminen. Tämän takia IMU:jen lisäksi tulisi käyttää ulkoista mittausta, minkä avulla saadaan referenssitulos. [20]

## 3. KÄRKIOHJAUS

Metsäkoneen puomia voidaan ohjata eri tavoilla. Järjestelmissä on lähtökohtaisesti aina kaksi ohjainsauvaa, joilla puomin ohjaus toteutetaan. Riippumatta ohjaustavasta puomin liikkeen nopeutta hallitaan sillä, kuinka paljon ohjainsauvaa liikutetaan keskipisteestä. Ohjaustavoista tunnetuin on perinteinen ohjaus, jossa jokaista puomin liikettä ohjataan erikseen. Markkinoilla on kuitenkin yleistynyt ohjausjärjestelmät, joissa toimilaitteiden ohjausta on osittain automatisoitu. Yksi tällaisista ohjauksista tunnetaan nimellä kärkiohjaus. Tässä luvussa esitellään perinteinen ohjaus ja kärkiohjaus, sekä syyt miksi kärkiohjaus on kehitetty ja myöhemmin yleistynyt.

### 3.1 Perinteinen ohjaus

Metsäkoneen puomia ohjataan kahdella ohjainsauvalla, joissa on keinukytkimet. Perinteisessä ohjauksessa puomin jokaista toimilaitetta ohjataan erikseen. Tämä tapahtuu liikuttamalla ohjainsauvaa eteen, taakse, oikealle, vasemmalle tai painamalla keinukytkimestä jompaankumpaan suuntaan. Useampaa liikettä voidaan kuitenkin ohjata kerrallaan, esimerkiksi liikuttamalla toista ohjainsauvaa etuviistoon ja painamalla keinukytkintä. Puomin liikkeet, joita sauvoilla tulee ohjata, on esitetty kuvassa 10.



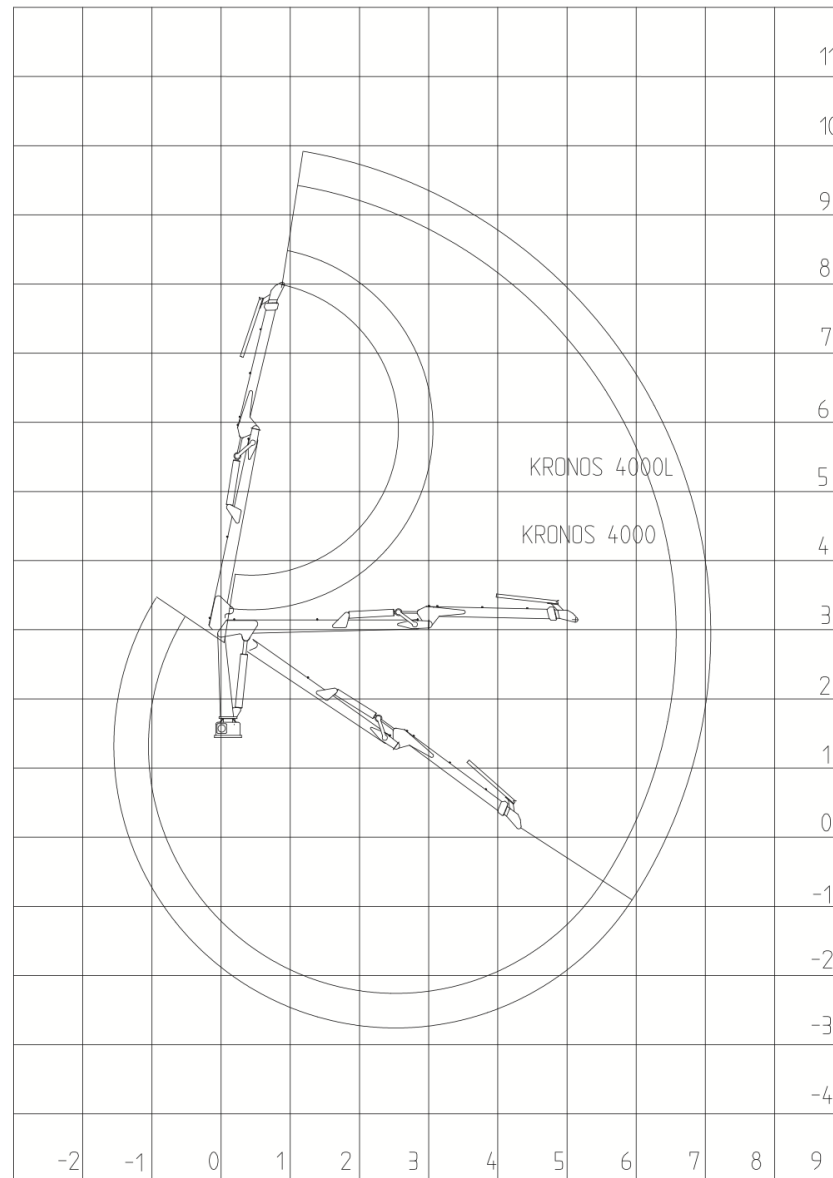
**Kuva 10.** Metsäkoneen puomin ohjattavat osat [22]

Kuvassa 10 esitetty *slewing crane pillar* on suomeksi puomin kääntö, *first boom* on tunnettu nostopuomina ja *second boom* taittopuomina. *Extension boom* on jatke. *Rotator* ja *grapple* ovat pyöritys ja koura. Kouran tilalla voisi olla myös mahdollisesti jokin toinen toimilaite, kuten hakkuupää.

Puomia käännetään vasemman puolen ohjainsauvan oikea-vasen-akselin liikkeellä ja oikean ohjainsauvan keinukytkin sulkee ja avaa kouraa sekä oikean sauvan oikea-vasen-akseli ohjaa pyöritystä. Edellä mainittujen kolmen liikkeen ohjaus on sama riippumatta siitä, onko kyseessä perinteinen- vai kärkiohjaus. [22]

Erot perinteisen- ja kärkiohjauksen välillä tulee muiden liikkeiden ohjauksesta. Perinteisessä ohjauksessa oikean ohjainsauvan eteen-taakse liike ohjaa nostopuomia ja vasemman ohjainsauvan eteen-taakse liike ohjaa taittopuomia. Vasemman ohjainsauvan keinukytkin ohjaa jatketta. [22] Kuudesta liikkeestä neljä siirtää puomin kärjen sijaintia.

Perinteistä ohjausta käytettäessä nosto- tai taittopuomin ohjaus pelkästään liikuttaa puomin kärkeä kaarella. Puomin liikeradat ovat esillä kuvassa 11. Tämä siis tarkoittaa sitä, jotta saadaan puomin kärkeä kuljetettua esimerkiksi vaakasuoraan kohti konetta, tulee kuljettajan ohjata mahdollisesti jopa kolmea eri liikettä yhtäaikaisesti.



**Kuva 11. Puomin liikeradat [23]**

Kuten luvussa 2 todettiin, nykyiset metsäkoneet kykenevät suorittamaan useita työtehtäviä. Harvesteri suorittaa puun kaadon, kasaa puut, kuorii oksat ja leikkaa määrätyn mittaisiksi. Nykyaikaiset monitoimikoneet ovat työskentelyolosuhteiltaan erittäin mukavat. Verrattuna entisaikaan, nykyään metsätyö on hyvin erilaista. Ennen työssä oltiin alttiita säälle, koska työt jouduttiin suorittamaan ulkona ilman koneita. Nykyaikaisissa työkoneissa työstä on tullut yksitoikkoisempaa, mutta kuitenkin algoritmisesti haastavaa. Kuten aikaisemmin todettiin ja kuvan 11 liikeradoista voidaan päätellä, vaaka- tai pystysuoran liikkeen tuottaminen vaatii kolmen eri liikkeen tarkkaa hallintaa. [24]

Edellä mainittujen syiden takia tuli tarve lisätä automaation osuutta metsäkoneiden puomin ohjauksessa. Tiettyjen toistuvien toimintojen automatisointi vähentää käyttäjän

työkuormaa ja täten edesauttaa jaksamisessa. Automaation lisääminen voi myös lisätä työn tehokkuutta. Kärkiohjauksen avulla saavutettavia hyötyjä käsitellään lisää luvussa 3.3.

## 3.2 Kärkiohjauksen toiminta

Kärkiohjauksessa koneen ohjaaja ohjaa puomin kärkeä. Ohjainsauvoilla annetaan ainoastaan tieto mihin suuntaan kärkeä halutaan ohjata ja kuinka nopeasti. Tämä on mahdollista, kun anturitiedon avulla voidaan tietää puomin kärjen sijainti ja hydraulisyliinterien toiminta on lineaarisesti optimoitu. Tällöin voidaan laskea vaaditut ohjausarvot jokaiselle sylinterille, jotta puomin kärjelle voidaan saavuttaa haluttu liike. [25]

Kärkiohjauksen toimintaa voidaan verrata siihen, kuinka ihminen liikuttaa kättään. Sormenpäättä on helppo liikuttaa vaakasuoraan, vaikka se vaatii useamman nivelen samanaikaista liikettä, joita ohjataan aivoilla. [25] Metsäkoneessa näkö ja aivot on korvattu antureilla ja kontrollerilla ja lihasten tilalla on hydraulisyliinterit ja venttiilit. Tämän mahdollistamiseen käytetään käänteistä kinematiikkaa.

Puhuttaessa kinematiikasta, tarkoitetaan liikkeen tutkimista välittämättä sitä, mikä aiheuttaa liikkeen. Käänteinen kinematiikka on termi sille, kun käytetään kinematiikan yhtälöitä määrittäessä mitä liikkeitä tarvitaan, jotta saavutetaan haluttu piste. Käänteisen kinematiikan laskemiseen tulee tietää puomin nivelten kulmat sekä millaisella ohjauksella on mikäkin vaikutus liikuteltavan pisteen sijaintiin. Tämän jälkeen luodaan Jacobian matriisi. Jacobian matriisilla voidaan määrittellä ohjausten välisiä suhteita sekä halutun pisteen nopeutta. [26]

Kuten luvussa 3.1 todettiin, kärkiohjauksessakin vasemman ohjainsauvan vasen-oikea-akselin ohjauksella liikutetaan puomin kääntöä, oikean ohjainsauvan keinukytkin hallitsee kouraa ja vasen-oikea-akselin liikkeellä pyöritetään kouraa. Muuten puomin ohjaus poikkeaa perinteisestä ohjauksesta. Kärkiohjauksessa vasemman ohjainsauvan eteen-taakse-akselin liikuttamisella liikutetaan puomin kärkeä joko kohti tai poispäin koneesta vaakasuoraan ja oikean ohjainsauvan vastaavalla liikkeellä nostetaan puomin kärkeä joko pystysuoraan ylös tai lasketaan alas. Vasemman ohjainsauvan keinukytkintä ei tarvita, mutta koneen ohjaaja voi käyttää sitä halutessaan yhä jatkeen ohjaamiseen. [22]

### 3.3 Kärkiohjauksella saavutettavat hyödyt

Kärkiohjauksella voidaan saavuttaa monia hyötyjä. Aiemmin todettiin, että metsäkoneen ohjaaminen voi olla algoritmisesti haastavaa. Puomin ohjaamisen lisäksi metsäkoneen ohjaaja joutuu tekemään päivässä lukuisia päätöksiä liittyen työn toteuttamiseen ja suunnitteluun. Perinteisellä ohjauksella vaaditaan käyttäjältä jatkuvaa tarkkailua ja ohjainsauvojen tarkkaa hallintaa. Käsillä tehtävä tarkkuuta vaativa työ vaatii suhteellisen paljon ajatustyötä, mikä rasittaa ohjaajaa, milloin virheiden mahdollisuudet lisääntyvät. Täten kärkiohjauksella voidaan edesauttaa työssä jaksamista ja mahdollisesti jopa estää vahinkoja sekä virheistä johtuvia potentiaalisia vaaratilanteita ja konerikkoja. [27]

Kärkiohjauksen avulla voidaan saavuttaa myös pidempi elinikä metsäkoneen puomille. Tämä korostuu etenkin kokemattomien kuljettajien kanssa. Kokenut kuljettaja voi saavuttaa perinteiselläkin ohjauksella jouhevan ja sulavan käytön. Mutta etenkin kokemattomien kuljettajien kanssa ohjaus voi olla hyvinkin epätasaista, mikä aiheuttaa puomille ja hydraulikalle ylimääräistä räsitusta. [27]

Metsätyössä yleisesti puomin ohjaus on eniten aikaa vievä työvaihe. Tästä syystä on tärkeää, että uudet kuljettajat oppivat ohjaamaan puomia tehokkaasti. Puhuttaessa oppimiskäyrästä tarkoitetaan kokemuksen ja tehokkuuden välistä suhdetta. Oppimiskäyrä siis kuvaa tehokkuuden kasvamista ajan myötä. Metsäkoneen puomi on hyvin monimutkainen ohjattava. Kuvassa 11 esitettyyn puomin liikeratojen sisällä on todella useita pisteitä, jotka on mahdollista saavuttaa lukuisilla eri puomin asennoilla. Yksinkertaistettu, automatisoitu ohjaus tekee ohjauksesta helpompaa ja edesauttaa nopeaa ohjauksen oppimista, mistä voi seurata jyrkempi oppimiskäyrä. [27] [28]

Manner, Gelin, Mörk ja Englund olivat tutkineet oppimiskäyrän kehitystä aloittelevilla metsäkoneen käyttäjillä. Tutkimuksessa 17 nuorta, joilla ei ollut aikaisempaa kokemusta metsäkoneen ohjaamisesta laitettiin tekemään ennalta määrättyjä tehtäviä. Yhdeksän nuorta käytti perinteistä ohjausta ja kahdeksan kärkiohjausta. Tutkimuksessa mitattiin muun muassa ennalta määrättyjen tehtävien suorittamiseen kuluva aika ja puomin kärjen kulkeva matka. Tutkittavat toistivat ennalta määrättyt tehtävät neljä kertaa. [28]

Tutkimus ei suoranaisesti todistanut sitä, että oppimiskäyrä olisi jyrkempi kärkiohjauksella kuin perinteisellä ohjauksella. Itseasiassa perinteisellä ohjauksella saavutettiin jyrkempi oppimiskäyrä, eli tutkittavien työn tehokkuus kasvoi kokemuksen myötä nopeammin. Myös kärkiohjausta käyttävillä tulokset paranivat, muttei yhtä merkittävästi. Tässä tulee huomata myös se, että kärkiohjauksella tutkittavat suorittivat tehtävät heti alusta huomattavasti nopeammin. Voidaan siis todeta, että etenkin uusien ohjaajien kohdalla kärkiohjaus on erittäin kätevä työkalu. [28]

Vaikka kärkiohjauksessa on lukuisia etuja ja se on todettu monissa suhteissa perinteistä ohjausta paremmaksi, todennäköisesti kärkiohjaus ei tule koskaan syrjäyttämään kokonaan perinteistä ohjausta. Perinteinen ohjaus vaatii vähemmän komponentteja ja siinä on vähemmän mahdollisia kohtia, jotka voivat hajota tai aiheuttaa vikatiloja. Tästä syystä kärkiohjausjärjestelmissä on aina mahdollisuus käyttää myös perinteistä ohjausta, jos vaikka yhdestä puolesta hajoaa anturi, ei voida laskea käänteistä kinematiikkaa eikä täten toteuttaa kärkiohjausta.

## 4. XCRANE-JÄRJESTELMÄ

xCrane on Technion Oy:n suunnittelema ja valmistama puomin ohjausjärjestelmä metsätyökoneisiin ja nostureihin. Kuvassa 12 on esitetty xCrane-järjestelmän käyttökohteita. Järjestelmästä on saatavilla kolme eri lisenssiversiota, joista tässä työssä keskitymme xCrane PRO-ohjausjärjestelmään, koska kyseisessä versiossa on kärkehjäus käytettävissä.



Harvesterit ja muut metsäkoneet



Perävaunun nosturin ohjaus



Auto- ja teollisuusnostureiden ohjaus



Traktoriin liitetyn nosturin ohjaus

**Kuva 12.** xCrane PRO-ohjausjärjestelmän käyttökohteita [29]

Kuten kuvasta 12 nähdään, samalla järjestelmällä voidaan ohjata useita erilaisia metsätyökoneita ja nostureita. Tämä on mahdollista sillä, että kaikkien nostureiden toiminta ja puomin rakenne on pääpiirteittäin samanlainen.

Tässä luvussa käydään läpi xCrane PRO-järjestelmän rakennetta ja käyttöönottoprosessia. Luvussa esitettyä käyttöönottoprosessia ei tarvitse suorittaa jokaiselle metsäkoneelle. Lähtökohtaisesti käyttöönottoprosessi tehdään kuvatun mukaisesti aina uudelle nosturityypille. Tämän jälkeen käyttöönottoprosessissa tallennetut arvot ladataan muihin vastaaviin nostureihin. Käyttöönottoprosessi voidaan suorittaa myös osana vianetsintää tai korjauksessa olevan laitteen toiminnan tarkastamiseksi, esimerkiksi anturin vaihtamisen jälkeen.

### 4.1 Järjestelmän rakenne

xCrane-järjestelmä sisältää näytön, ohjausyksikön, ohjauslaitteet ja antureita sekä nämä kaikki yhdistävän johtosarjan. Puomin ohjauksessa käytössä olevat anturityypit ovat

IMU- ja induktiivianturit. IMU-antureiden toimintaa käsiteltiin luvussa 2.5.1. IMU-antureilla määritetään puomin asentoa. Induktiivianturia käytetään referenssinä IMU-antureille, millä määritetään kääntöliikkeen keskikohta. xCrane PRO-järjestelmässä käytössä oleva IMU-anturi on esitetty kuvassa 13.



**Kuva 13.** xMove-anturi [29]

Kuvassa 13 esitetty xMove anturi mittaa kuuden eri vapausasteen liikettä. Anturin kotelo on alumiinia ja sen suojaustaso on IP X9. Laitteen molemmissa päissä on M12-liittimet. Toisessa päässä on naaras- ja toisessa urosliitin. Anturit kytketään järjestelmässä sarjaan ja viimeisen anturin päähän tulee kytkeä terminointivastus. [30]

Näiden lisäksi järjestelmään voidaan liittää muun muassa sekä ulko-, että hydraulioiljyn lämpötila-anturi, hydraulijärjestelmän paineanturi ja kuorman paineanturi. Käyttäjän läsnäoloa voidaan tarkkailla penkki- sekä ovikytkimellä ja kuolleen miehen kytkimellä. [31] Järjestelmän rakenne on esitetty kuvassa 14.



**Kuva 14.** xCrane PRO-järjestelmän rakenne

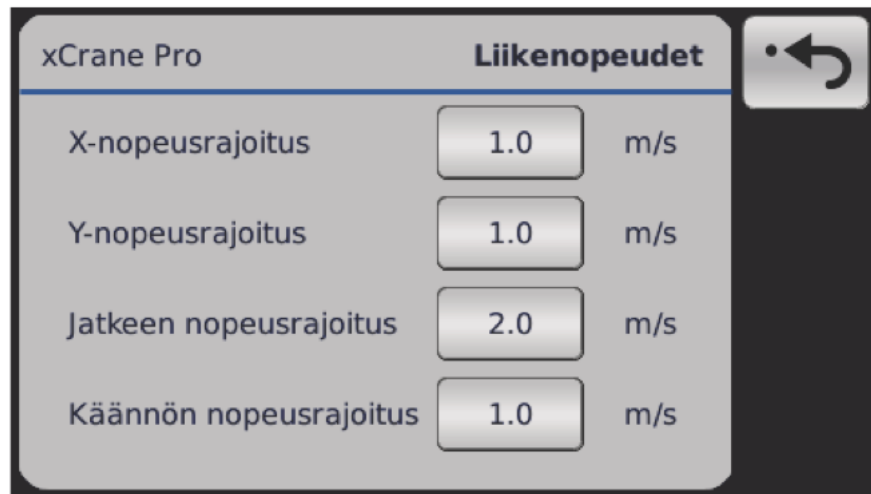
Kuvassa 14 siniset viivat kuvastavat johtosarjaa, joka kytkee komponentit yhteen. Oikealla alhaalla on venttiilit, joiden yläpuolella on TEC152 kontrolleri. Kuvassa vasemmassa reunassa on näyttö. Näytön ja kontrollerin välisen viivan yläpuolella on antureita ja alapuolella ohjauslaitteita.

Näyttönä järjestelmässä toimii *Opus A3 Eco Basic Plus*. Kyseessä on 4,3 tuuman halkaisijalla oleva kosketusnäyttö. Näytössä on Freescale I.MX35 prosessori, jonka kellotaajuus on 532 MHz, mikä pyörittää käyttöliittymää. Näytön avulla kuljettaja voi esimerkiksi vaihtaa ohjaustapaa perinteisen- ja kärkeiohjauksen välillä, säätää asetuksia ja kalibroida ohjauslaitteita. Näytön ja ohjausyksikön välillä tietoa siirretään CAN-väylää pitkin noudattaen CANopen protokollaa. Näytön käyttöliittymä on ohjelmoitu Codesys V3:lla. Järjestelmää voidaan käyttää myös ilman näyttöä. [31] [32]

Ohjausyksikkönä toimii TEC152, jossa on MPC5674F prosessori. TEC152 on suojattu tasolla IP67, joka tarkoittaa laitteen olevan vesi- ja pölytiivis. Ohjausyksikössä on 4 CAN-liitäntää ja yksi RS-232 liitäntä. Laitteen kotelo on alumiinista. Kuten xCrane-järjestelmän näyttö, myös TEC152 ohjelmoidaan Codesys V3:lla. [33]

Ohjauslaitteina järjestelmässä toimii yleensä kaksi ohjainsauvaa ja kytkimiä. Ohjainsauvat voivat olla joko analogisia tai digitaalisia. Analoginen ohjainsauva antaa ohjausarvon jännitteenä ja digitaalinen ohjainsauva välittää digitaalista dataa CANopen protokollaa käyttäen. Ohjauslaite voi myös olla radioyhteydellä liitetty, milloin käyttäjä voi olla muualla kuin koneessa konetta käytettäessä. [29]

Järjestelmä mahdollistaa myös liikenopeuksien rajoittamisen. Liikenopeuksien rajoittaminen tapahtuu käyttöliittymän kautta. Kuvassa 14 on esitettyä käyttöliittymä liikenopeuksia rajoittaessa. [31]



**Kuva 15.** xCrane PRO-käyttöliittymän liikenopeuksien säätö

Järjestelmä kykenee siis laskemaan IMU:ista saatavasta datasta kärjen nopeuden. Täten järjestelmä kykenee myös rajoittamaan nopeuksia kuljettajan mieleiseksi. Järjestelmään voidaan tallentaa kuudelle eri kuljettajalle mieluisat arvot. Nopeusrajoitukset ovat myös perusta kärkiohjaukselle. Nopeusrajoituksen avulla järjestelmä tietää, mikä tulee olla nopeus maksiminopeudella, jonka jälkeen järjestelmä skaalaa muille ohjausarvoilla nopeudet. Kuljettaja voi säätää myös muun muassa aloitus- ja lopetusrampit. Rampeilla säädetään sitä, kuinka nopeasti järjestelmä muuttaa ohjausarvoja. Näillä mahdollistetaan se, että puomin käyttö on sulavampaa ja työskentely mukavampaa. [31]

## 4.2 Käyttöönottoprosessi

Ohjausjärjestelmän käyttöönotto alkaa sillä, että tarkistetaan ovatko ennakkovaatimukset täyttyneet. Ensimmäisenä ennakkovaatimuksena on, että meillä on tiedossa puomin geometria, käytössä olevien sylinterien mitat ja anturien asettelu. Tämän lisäksi meidän tulee tietää, millainen anturi on asennettu mittaamaan puomin jatketta. Jotta käyttöönotto voidaan tehdä, tulee hydraulikan toimia normaalisti. Käyttöönotossa täytyy luoda yhteys tietokoneella järjestelmää ohjaavaan kontrolleriin. Yhteys luodaan CODESYS V3.5 kehitysympäristöön, joko läsnä olevaan tietokoneeseen tai internetin yli etänä.

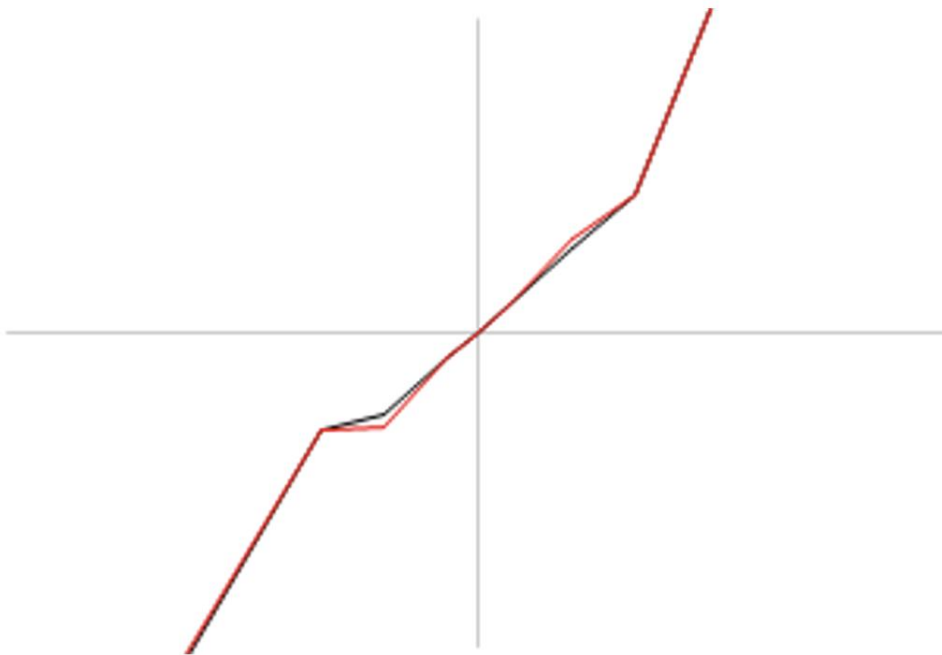
Ilman yhteyttä kehitysympäristöön onnistuu käyttöönoton alkuvaiheessa tehtävät minivirtojen säätö ja antureiden konfigurointi. Minivirta on hydraulikassa käytetty termi,

jolla tarkoitetaan pienintä ohjausarvoa venttiilille, mikä ei vielä aiheuta liikettä toimilaitteessa. Toimilaite tässä tapauksessa on sylinteri, joka liikuttaa puomia. Minivirtojen määrittelemisellä mahdollistetaan mukavampi ja herkempi ohjaus. Antureiden konfiguroinnissa määritetään, mikä kolmesta xMove anturista mittaa mitäkin puomin liikettä. Yksi anturi on nosturin pystypilarissa, jonka avulla määritetään laitteen kulma. Yksi anturi mittaa nostopuomin liikettä ja toinen anturi taittopuomia. Huomioitavaa on, että jatkeen anturit vaihtelevat kohdesovelluksen ja koneenrakentajan tarpeiden mukaan. Tällöin järjestelmästä tehdään yhteensopiva kyseisen anturin kanssa.

Yhteyden avulla ensin asetetaan antureiden *offsetit* eli poikkeamat puomiin asennetun anturin asennon ja todellisen puomikulman välillä. Käytännössä tämä tarkoittaa sitä, että puomista ajetaan yksi liike kerrallaan maksimiin ja anturin sen hetkinen arvo tallennetaan järjestelmään. Ensimmäisenä määritetään nostoliikkeen anturin poikkeama ja viimeisenä asetetaan jatkeanturin poikkeama. Tämän jälkeen puomin asento tiedetään järjestelmässä riittävällä tarkkuudella ja asentotietoa käytetään karkiohjauksessa laskemaan ohjausarvoja.

Käänteisen kinematiikan laskentaa tarvitaan tieto, millaisella ohjausarvolla saavutetaan mikäkin liike. Tämän takia suoritetaan seuraavaksi liikenopeuksien kalibrointi. Liikenopeudet mitataan 11 eri pisteestä, joista yksi piste on nollaohjaus. Jos puomi liikkuu nollaohjauksella, on järjestelmän hydraulikassa vika.

Järjestelmään tallennetaan konfiguraatitiedosto, mikä myös pitää sisällään oletusarvoiset liikenopeudet sylinterien geometriatietojen ja venttiilin arvojen perusteella. Liikenopeuksien kalibroinnissa laskennallisesti määritetään sylintereihin menevä tilavuusvirta puomianturoinnista saatavan asentotiedon muutoksen perusteella. Kun tiedetään, kuinka paljon puomin asento muuttuu, voidaan laskea sylinterien geometriatietojen mukaan tilavuusvirrat. Mitatuista arvoista piirretään kuvaaja, jota verrataan konfiguraatitiedoston arvoihin. Jos arvot ovat tarpeeksi lähellä, ei vaadita toimenpiteitä. Jos arvoissa on eroja, voidaan muuttaa pisteen ohjausarvoa, minkä avulla voidaan saavuttaa enemmän konfiguraatitiedoston mukainen kuvaaja. Kuvaaja piirretty CODESYS V3.5 kehitysympäristöön, mitä käsitellään enemmän luvussa 5.1. Kehitysympäristöön piirtyvä kuva on esitetty kuvassa 16.



**Kuva 16.** CODESYS V3 kehitysympäristöön piirtyvä kuvaaja liikenopeuksista

Kuvassa 16 esitettyssä kuvaajassa vaaka-akselilla on ohjausarvot ja pystyakselilla virtausarvot. Punainen kuvaaja esittää konfiguraatitiedostoon tallentuvia arvoja ja musta mitattuja. Kuvaaja on luotu simulaatiolla ja virhe on luotu keinotekoisesti, jotta kuvaajan toimintaa voidaan havainnoida.

Kun liikenopeuksien kalibrointi on suoritettu, voidaan suorittaa järjestelmän testaus. Testauksessa tarkastellaan kärkiohjauksen toimintaa. Jos kaikki toimii moitteettomasti, voidaan todeta käyttöönoton olevan valmis.

## 5. KEHITYSYMPÄRISTÖ

Tässä luvussa käsitellään luvussa 4 esitetyn käyttöönottoprosessin kehitystä ja testausta. Ensimmäisenä perehdytään kehitysympäristöön ja IEC-61131-3 standardiin. Tämän jälkeen esitellään välineet ja ympäristö, jossa käyttöönottoprosessin kehitystä voidaan testata.

### 5.1 Kehitysympäristö

Työ suoritetaan CODESYS V3 kehitysympäristössä. CODESYS on ohjelmistoalusta, joka on kehitetty teollisen automaation teknologiaan ja sitä käytetään erityisesti PLC-ohjelmoinnissa (engl. programmable logic controller). Alustan pohjana on IEC-61131-3 standardiin pohjautuva ohjelmointityökalu *CODESYS Development System*. IEC-61131-3 standardia käsitellään omassa aliluvussaan 5.1.1. [34]

CODESYS:tä käyttävät monet automaation laitevalmistajat omien ohjelmitavien tuotteiden implementoinnissa. CODESYS sisältää valmiita funktioita käytössä oleviin teknologioihin, kuten CANopeniin. Perinteisen PLC-ohjelmoinnin lisäksi CODESYS mahdollistaa myös objektorientoituneen ohjelmoinnin. [35]

CODESYS sisältää myös visualisointityökalun CODESYS VISUALIZATIONin. Kyseistä työkalua käytetään käyttöliittymän ohjelmoinnissa. CODESYS VISUALIZATION mahdollistaa käyttöliittymän ohjelmoinnin ja sen näyttämisen joko kohdelaitteella tai tietokoneen näytöllä. [36]

#### 5.1.1 IEC-61131-3 Standardi

IEC-61131-3 standardi antaa vaatimukset PLC-systeemeille. Vaatimukset käsittelevät käytössä olevaa teknologiaa ja ohjelmointiympäristöä. IEC-61131-3 standardia voidaan pitää myös tietynlaisena oppaana PLC-ohjelmointiin. Standardi luotiin, koska PLC-ohjelmointia vaativien laitteiden monimutkaisuuden kasvaessa ohjelmoijien kouluttamisen, ohjelmiston luomisen sekä implementaation kulut kasvavat tasaisesti. [37]

IEC-61131-3 standardin mukaisessa ohjelmoinnissa ohjelmointi tapahtuu lohkoissa, joita kutsutaan lyhenteellä POU (engl. Program Organization Unit). POU:ita on kolmea eri tyyppiä: funktio (engl. function), funktiolohko (engl. function block) ja ohjelma (engl. program) tyyppisiä. Funktio POU tuottaa aina saman lopputuloksen samoilla sisääntuloilla, eikä täten omaa muistia. Funktiolohko puolestaan kykenee muistamaan

statustietoa kutsujen välillä, funktiolohkot kykenevät myös instantiointiin, täten niitä käytetään olio-ohjelmoinnin tapaan. Ohjelma on POU, joka tuottaa yhden tai useamman arvon suorituksen aikana. Ohjelman suorituksen jälkeen kaikki arvot säilytetään seuraavaan suoritukseen asti. [37]

Saman projektin sisällä voidaan käyttää PLC-ohjelmoinnissa eri kieliä. IEC-61131-3 sisältää viisi eri kieltä. *Instruction list* on matalan tason kieli, jota käytetään laiteorientoituneessa ohjelmoinnissa ja *structured text* on korkeamman tason kieli, jolla voidaan suorittaa monimutkaisempia matemaattisia laskuja. *Function block diagramilla* ohjelmointi tapahtuu graafisesti aritmeettisten operaatioiden avulla. *Ladder diagram* on myös graafinen ohjelmointikieli, jonka avulla ohjaustoimintoja kuvataan loogisina kontakteina. *Sequential function chart* on toimintakaavio, jolla ohjaustehtävä pilkotaan osiin. Osia voidaan suorittaa samanaikaisesti tai peräkkäin.

## 5.2 Testiympäristö

CODESYS kehitysympäristöön luodussa xCrane projektissa sijaitsee sekä näytön, että kontrollerin ohjelmakoodi. Tästä seuraa hallittavuus komponenttien yhteen toimimiselle sekä mahdollisten ongelmien korjaamiselle. Tämä tarkoittaa sitä, että vain toisen ohjelmakoodia voidaan suorittaa tai molempia samaan aikaan. Tämä on hyödyllistä testaamisen kannalta, sillä esimerkiksi havaittaessa pieni virhe näytön koodissa päivittäminen tapahtuu nopeasti.

Testiympäristö jakautuu kahteen eri tasoon. Korkeimpana tasona on PLC simulaatio, jossa ohjelmistokoodia suorittaa tietokone. Käyttöliittymä piirtyy tietokoneen näytölle ja projektin TEC152 puolelle voidaan asettaa keinotekoisesti arvoja, esimerkiksi ohjainsauvan ohjainarvoja. PLC-tason testauksessa voidaan varmistaa tiettyjen ohjelmistokoodirakenteiden toiminta.

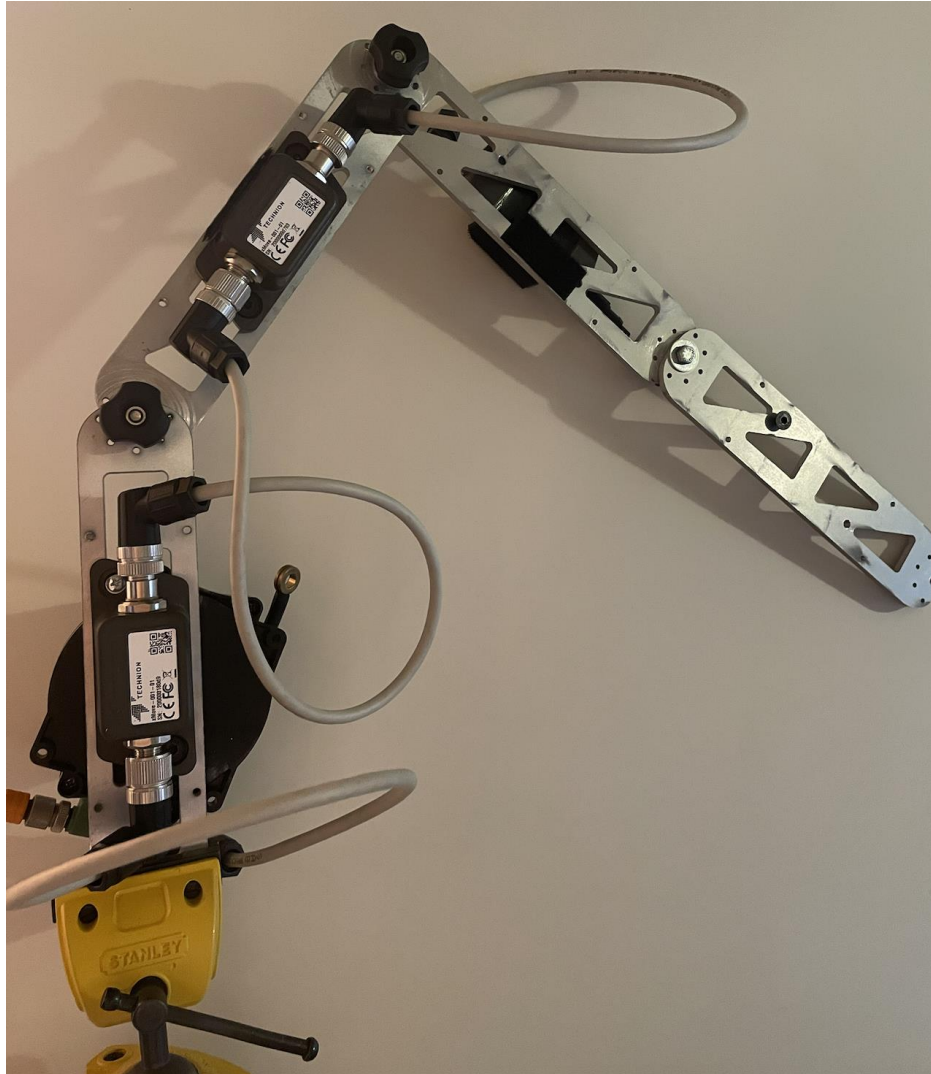
Alemman tason testauksessa käytetään TEC152 kontrolleria sekä OPUS A3 näyttöä. Testauksessa käytetään niin sanottua *simulaattorilaatikkoo*, joka on esitetty kuvassa 17.



**Kuva 17. Simulaattorilaatikko**

Kuten kuvasta 17 voidaan nähdä, simulaattorilaatikossa on kytkettynä myös kaksi ohjainsauvaa. Teoriassa simulaattorilaatikon voisi kytkeä oikeaan metsäkoneeseen ja se toimisi, kunhan järjestelmään ladataan oikea konfiguraatitiedosto sekä puomiin asennettaisiin anturit. CODESYS kehitysympäristöön simulaattorilaatikko kytketään *PEAK PCAN*-adapterin avulla.

Simulaattorilaatikko voidaan yhdistää *Farming Simulator 17* videopeliin, jossa sillä voidaan ohjata puutavaranoستوريا. Simulaattorin avulla voidaan ohjata nosturia sekä perinteisellä-, että kärkeohjauksella. *Farming Simulator 17* videopelin lisäksi testauksessa voidaan simuloida puomia oikeilla antureilla, jotka ovat kiinni metsäkoneen puomin tyyppisessä rakenteessa. Kyseinen rakenne on esillä kuvassa 18.



**Kuva 18.** Metsäkoneen puomia imitoiva rakenne

Kuvan 18 puomia imitoivassa rakenteessa on kolme xMove-anturia, kuten oikeassa järjestelmässä, sekä jatkeen anturi. Rakenteen nivelet ovat liikuteltavissa, joten tämän yhdistämällä simulaattorilaatikkoon voidaan suorittaa testejä oikealla anturidatalla. Samanaikaisesti ei voi suorittaa testejä Farming Simulator 17:lla.

## 6. KÄYTTÖÖNOTTOPROSESSIN KEHITYS

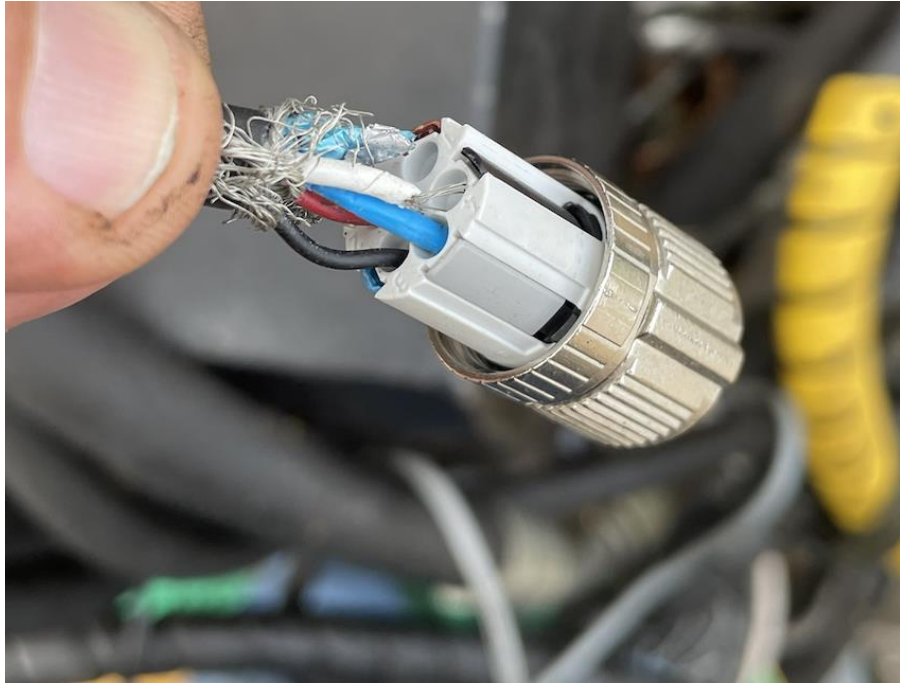
Tässä luvussa käsitellään käyttöönottoprosessin kehittämistä johdannossa esitettyjen tavoitteiden mukaiseksi. Ensimmäisenä tutustuttiin järjestelmään ja käyttöönottoprosessiin käytännössä. Käyttöönottoprosessin kehittäminen aloitettiin antureiden poikkeamien määrittämisen mahdollistamiseksi käyttöliittymän avulla. Ensimmäisenä tuli kuitenkin tutustua siihen, miten CANopen protokolla on toteutettu xCrane PRO-järjestelmässä.

### 6.1 Tutustuminen järjestelmään

Tutustuminen järjestelmään alkoi testiympäristöön tutustumisella ja tutkimalla järjestelmän toimintaa. Järjestelmään tutustuesssa pyrittiin havainnoimaan muun muassa maksiminopeuksien säätöä sekä yleistä järjestelmän toimintaa. Tämän jälkeen harjoiteltiin käyttöönottoprosessin suorittamista simulaattorin avulla.

Työn alkuvaiheessa tarjoutui mahdollisuus päästä tutustumaan järjestelmään käytännössä. Tietoomme tuli nosturi, josta oli anturidatan kulkeminen katkennut. Tarkoituksena oli mennä etsimään vika, korjata se ja suorittaa käyttöönottoprosessi tutkimustarkoituksessa, sekä varmistaa järjestelmän toiminta.

Väylästä löydettiin vikaantunut M12-liitin pystypilarin anturista. Vikaantunut liitin on esillä kuvassa 19. Kuten kuvassa näkyy, valkoinen kaapeli on irronnut lähes täysin. Todennäköistä on, että kaapeliin oli kohdistunut odottamatonta vetoa, mikä aiheutti hajoamisen.



**Kuva 19.** Järjestelmän vian aiheuttanut liitin

Liitin onnistuttiin vaihtamaan paikan päällä uuteen. Kaapeleissa riitti pituus, vaikka niitä jouduttiin lyhentämään. Korjattu ja uudelleen kytketty kaapeli, sekä pystypilarin anturi on esillä kuvassa 20.



**Kuva 20.** Pystypilarin anturi liittimen vaihtamisen jälkeen, ylempi liitin vaihdettu

Kuvassa 20 huomioitavaa on se, vaikka anturin kotelo on kestävää alumiinia, tulee se suojata teräslevyllä metsäkoneessa. Kuvassa näkyy kehykset, joihin kiinnitetään suojaava teräslevy.

Oletusarvo oli, että kyseisen korjauksen jälkeen järjestelmä toimisi ja päästäisiin toteuttamaan käyttöönotto. Kuitenkin, kun järjestelmä käynnistettiin ja siihen muodostettiin yhteys, havaittiin ettei järjestelmä löydä antureita. Kuten luvussa 2.3.1 kerrottiin, CAN-väylään voidaan kytkeä laitteita häiritsemättä väylän toimintaa. Yhteys luotiin M12-T-haaralla, johon kytkettiin PEAK PCAN lukija. Kyseisellä lukijalla voitiin todeta, ettei väylällä kulkenut dataa.

Tämän jälkeen vikaa aloitettiin paikantamaan kytkemällä anturi kerrallaan järjestelmään. Tämä tapahtui käytännössä niin, kun anturit ovat kytkettynä sarjaan, irrotettiin anturista jatkava kaapeli ja kytkettiin sen tilalle terminointivastus. Jos väylällä kulki dataa, tiedettiin vian olevan pidemmällä.

Näin toimimalla onnistuttiin paikantamaan vika lopulta taittopuomin anturiin. Ensimmäisenä tarkastettiin nosto- ja taittopuomin antureiden välinen kaapeli kiinnittämällä kaapelin päähän terminointivastus. Kun väylällä kulki silloinkin dataa, tiedettiin kaapelin olevan ehjä. Todettiin anturi rikkinäiseksi, joten se vaihdettiin. Tämän jälkeen järjestelmä toimi ja pystyttiin aloittamaan käyttöönottoprosessi.

Kokonaisuutena käyttöönottoprosessin testaaminen oli hyvin opettavainen kokemus, mistä havaittiin myös tarve tälle tutkimukselle. Etenkin liikenopeuksien kalibrointi tuntui turhan aikaa vievältä, koska siihen vaadittiin kahden henkilön työpanos. Myös käytännön tutustuminen järjestelmään yleisesti koettiin hyödylliseksi.

## **6.2 CANopen xCrane PRO-järjestelmässä**

xCrane PRO-järjestelmässä on käytetty laitteiden välisessä viestinnässä CANopen protokollaa, jota käsiteltiin luvussa 2.4. Järjestelmässä projektin xCrane-osuus toimii serverinä, eli osuus, jota TEC152 suorittaa. Näyttö, OPUS A3, toimii niin sanottuna clienttinä.

Käytännön tasolla tämä tarkoittaa sitä, että kaikki tieto järjestelmästä sijaitsee kontrollerissa ja näyttö pyytää arvoja sekä pyytää muuttamaan niitä. Esimerkiksi, jos näytöllä halutaan muuttaa maksiminopeutta, näyttö pyytää kontrolleria muuttamaan kyseistä arvoa.

Objektisanakirja sisältää abstraktilla tasolla tiedon siitä, millaista tietoa serverillä on. Objektisanakirjassa on määritelty muun muassa jokaisen objektin tyyppi, indeksi ja ali-

indeksi. Molemmat sovellukset, eli näyttö ja kontrolleri, sitovat omilla mekanismeillaan omia muuttujiaan objektisanakirjan indekseihin ja ali-indekseihin. Objektisanakirjaan luodaan lisäyksiä EDS-editorin (engl. electronic data sheet) avulla, joka luo eds-tiedoston. Kyseisestä tiedostosta voidaan kirjoittaa objektisanakirja suoraan Python-scriptillä. Vector CANeds sovelluksella voidaan suoraan luoda indeksejä ja niille ali-indeksejä, sekä määrittää muuttujien tyyppi ja kuinka niitä voidaan käsitellä.

xCrane PRO-järjestelmässä serverin ja clientin eroa vähentää tapa, jolla näyttösovellus on toteutettu. Periaatteessa näyttösovellus toteuttaa protokollatasolla operaatiota niin, että se pyytää lukemaan tietyn indeksin ja ali-indeksin tiettyyn muuttujaan. Kirjoittaessa toiminta tapahtuu muuttujasta oikeaan indeksiin ja ali-indeksiin. Yksittäiset operaatiot kestävät ohjelmiston mittakaavassa kohtuullisen kauan ja yksi operaatio tulee saada loppuun ennen kuin seuraava voidaan aloittaa. Tämän takia ei olisi järkevää, että jokaisessa ohjelmiston osassa missä tietoa tarvitaan, sitä pyydetäisiin omia aikojaan. Koska ohjelmiston eri osat saattavat olla myös rinnakkaisesti käynnissä, olisi vaikeaa varmistaa, ettei kaksi kohtaa yrittäisi lukea tai kirjoittaa samaan aikaan. Näytön tekemisessä SDO-operaatioissa objekteihin on oleellista niiden ajoittaminen vuorotteluperiaatteella syklistä. Tämä ongelma on ratkaistu niin, että kaikki operaatiot ovat sijoitettu yhteen GVL:ään (engl. Global variable list).

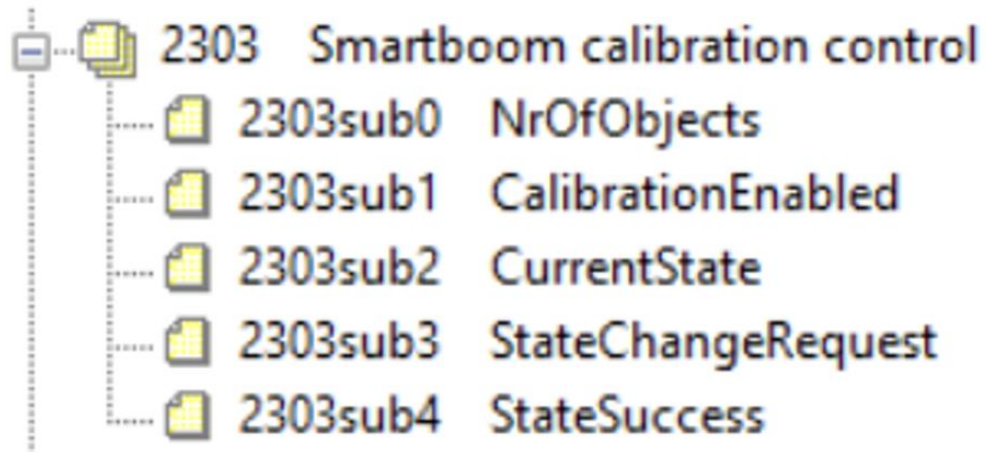
### **6.3 Antureiden poikkeaman määrittäminen**

Antureiden poikkeaman määrittämisen käyttöjärjestelmän avulla kehitys aloitettiin suunnittelemalla toiminta näytön ja kontrollerin välillä. Ensimmäisenä linjattiin, että mahdollisimman paljon logiikasta tulisi tapahtua kontrollerilla ja näyttö vain informoisi kuljettajaa. Kehittäminen koostui kahdesta osasta, ensimmäisenä luotiin kontrolleriin logiikka, jolla poikkeamat määritellään. Sen jälkeen, kehitettiin käyttöliittymää.

Vaatimuksia antureiden poikkeaman määrittämiselle oli, että toiminta olisi selkeää ja käyttöliittymä helposti ymmärrettävissä sekä käytettävissä. Vaatimuksena on myös se, että järjestelmä kertoo käyttäjälle, milloin arvo on validi. Tämä voidaan toteuttaa seuraamalla käyttäjän aiheuttamaa ohjausta ja sitä, ettei puomin asento muutu enää ohjauksesta huolimatta.

#### **6.3.1 Kontrollerin ohjelmointi**

Vaikka päätettiin, että logiikasta suoritettaisiin mahdollisimman paljon kontrollerilla, tulee dataa välittää kontrollerin ja näytön välillä. Kuvassa 21 on esitetty Vector CANeds:in avulla luodut objektit, jotka kuljettavat tarvittavaa dataa näytön ja kontrollerin välillä.



**Kuva 21.** Antureiden poikkeaman määrittämiseen luodut objektit

Huomioitavaa kuvassa 21 on se, että indeksin ensimmäinen ali-indeksi kertoo aina objektien lukumäärän. Näyttö kirjoittaa objekteihin *CalibrationEnabled* ja *StateChangeRequest*. Näyttö lukee objekteja *CurrentState*, *StateChengeRequest* ja *StateSuccess*.

Antureiden poikkeamien määrittäminen päätettiin tehdä tilakoneena. Näytön ohjelmakoodiin luotiin ohjelma tyyppinen POU nimeltä *BoomCalibration*. BoomCalibrationin alle luotiin kolme metodia, yksi jokaisen liikkeen anturin poikkeaman määrittämiseen. Ohjelmalle luotu rajapinta on esillä ohjelmassa 1. Tilakoneen mahdolliset tilat ovat *IDLE*, *LIFT\_MAX\_POS*, *TILT\_MAX\_POS* ja *EXT\_MAX\_POS*.

```

1 PROGRAM BoomCalibartion
2 VAR_IN_OUT
3 // To check if the correct movement is driven
4 JoystickDirectionBits : ARRAY[0..1] OF WORD;
5 END_VAR
6 VAR
7 // If FALSE, do nothing
8 enabled: BOOL;
9 // Used to control the state machine
10 stateChanged : BOOL;
11 // Request to change state in state machine
12 stateChangeRequest : SINT;
13 // Indicate for UI that which state we are currently in
14 currentState : BoomCalibrationState_t;
15 // Timer to check how long the joystick has been active
16 JoystickActiveTimer : TON;
17 // Boolean to save if joystick has been active for a second
18 TimerBool : BOOL;
19 // Boolean to start and stop timer
20 TimerOnOff : BOOL;
21 // Indicate UI that the current state is successfully finished
22 stateSuccess : BOOL;
23 END_VAR

```

**Ohjelma 1.** BoomCalibration ohjelman rajapinta

Ohjelman 1 rivillä 8 luotu muuttuja *enabled* on linkattu kuvassa 21 esitettyyn objektiin *CalibrationEnabled*. Muuttuja *stateChangeRequest* linkittyy vastaavan nimiseen objektiin, kuten myös muuttujat *currentState* ja *stateSuccess*. Huomioitavaa ohjelmassa 1 on se, että kyseiselle ohjelmalle tulee sisään- ja ulostulona muuttuja *JoystickDirectionBits*, jonka avulla voidaan olla varmoja siitä mitä liikettä ohjataan. *BoomCalibrationState\_t* on muuttujatyyppi, joka on luotu enumeraationa. Se pitää sisällään kaikkien tilojen nimet ja niille annetut numeeriset arvot,

Kuvassa 21 esitetyt objektit on määritelty niin, että ainoastaan näyttö, eli client, kykenee kirjoittamaan objektiin *CalibrationEnabled*. Kun *CalibrationEnabled* on näytön toimesta asetettu todeksi, päivittyy muiden objektien arvot jokaisella syklillä. Näyttö kykenee kirjoittamaan myös objektin *stateChangeRequestin* arvoa.

Kun aloitetaan antureiden poikkeamien määrittely, asetetaan *enabled* muuttujan arvo todeksi, jonka jälkeen siirrytään tilakoneessa tasoon *LiftMaxPos*. Ennen tilakoneeseen siirtymistä tehdään muutamia alustustoimenpiteitä, jotka ovat esitetty ohjelmassa 2.

```

1 // Initial action if boom calibration is started
2 IF NOT enabled THEN
3   currentState := BoomCalibrationState_t.IDLE;
4   RETURN;
5 ELSIF enabled = TRUE AND currentState = BoomCalibrationState_t.IDLE THEN
6   currentState := BoomCalibrationState_t.LIFT_MAX_POS;
7 END_IF
8 // Activate/Deactivate timer
9 JoystickActiveTimer(IN:=TimerOnOff, PT := T#1S, Q => TimerBool);
10 // Resetting stateChanged
11 stateChanged := FALSE;
```

### **Ohjelma 2.** Ennen tilakoneeseen siirtymistä tehtävät alustustoimenpiteet

Kuten ohjelmasta 2 voidaan huomata riveiltä 2–4, jos *enabled* on epätosi, pysytään tilassa *IDLE*, eikä tehdä muita toimenpiteitä. Tämä on tarpeen, koska huolimatta siitä onko käyttöönottoprosessi käynnissä vai ei, *BoomCalibration* suoritetaan jokaisella syklillä. Näin varmistutaan, ettei kyseisessä ohjelmassa käytetä turhaa aikaa. Rivillä 9 jokaisella syklillä, kun *enabled* on tosi, tarkastetaan, tuleeko ajastin käynnistää. *JoystickActiveTimer* asettaa muuttujan *TimerBool* todeksi, kun ohjainsauvaa on ajettu sekunti haluttuun suuntaan. Muuttuja *TimerOnOff* arvo asetetaan metodeissa.

Kun käyttöliittymän valikosta siirrytään käyttöönottoprosessia varten luodun näppäimen kautta käyttöönottoprosessin suorittamiseen, asettuu *enabled* arvo todeksi.

Tilakone on toteutettu kahtena case rakenteena. Ensimmäinen rakenteista, joka huolehtii tilojen vaihtamisesta, on esitetty ohjelmassa 3.

```

1 // State machine
2 // Transition to new state and giving information to display
3 CASE currentState OF
4 BoomCalibrationState_t.LIFT_MAX_POS:
5 IF stateChangeRequest>0 THEN
6   currentState := BoomCalibrationState_t.TILT_MAX_POS;
7   stateChanged := TRUE;
8 ELSIF stateChangeRequest<0 THEN
9   currentState := BoomCalibrationState_t.IDLE;
10  stateChanged:= TRUE;
11 END_IF
12 BoomCalibrationState_t.TILT_MAX_POS:
13 IF stateChangeRequest>0 THEN
14   currentState := BoomCalibrationState_t.EXT_MAX_POS;
15   stateChanged := TRUE;
16 ELSIF stateChangeRequest<0 THEN
17   currentState := BoomCalibrationState_t.LIFT;
18   stateChanged := TRUE;
19 END_IF
20 BoomCalibrationState_t.EXT_MAX_POS:
21 IF stateChangeRequest < 0 THEN
22   currentState := BoomCalibrationState_t.TILT_MAX_POS;
23   stateChanged := TRUE;
24 END_IF
25 END_CASE

```

### **Ohjelma 3.** *BoomCalibration tilakoneen tilojen vaihto*

Kuten ohjelmasta 3 voidaan lukea, ei jokaisesta tilasta ole mahdollista siirtyä jokaiseen tilaan. Kun käyttöliittymä havaitsee, että ollaan tilassa *EXT\_MAX\_POS* ja halutaan siirtyä seuraavaan tilaan, aloitetaan liikenopeuksien kalibrointi. Siirtymä on toteutettu visualisoinnin sisällä olevassa näytön ohjelmakoodissa.

Tilakoneen toinen osa ainoastaan kutsuu kyseisen tilan metodia, antaa metodille sisääntulot ja kirjoittaa metodin antaman totuusarvon muuttujaan *stateSuccess*. Tilakoneen kahden *case* rakenteen välissä asetetaan aina *stateSuccess* arvo epätodeksi, jos *stateChanged* totuusarvo on tosi. Metodeissa muutetaan *stateChanged* arvo epätodeksi.

Kaikki kolme metodia ovat lähes identtisiä keskenään. Ainoastaan niiden nimet ja tarkasteltava ohjainsauvan ohjaus, sekä luettava anturi vaihtuvat. Metodeja varten tarvitaan lisää muuttujia, jotka ovat esiteltyinä ohjelmassa 4.

```

1 METHOD TiltMax : BOOL
2 VAR_STAT
3   StateReady : BOOL;
4   LastSensorValue : REAL; // To save sensor value when control started
5 END_VAR
6 VAR
7   MaxAngleChange : REAL := 0.5; // How much movement is accepted.
8   AngleChange : REAL; // Variable used to calculate angle change
9 END_VAR
10 VAR_INPUT
11   stateChanged : REFERENCE TO BOOL;
12 END_VAR

```

#### **Ohjelma 4. Metodin TiltMax esittely**

Kuten ohjelmasta 4 voidaan lukea, täytyi luoda kaksi staattista muuttujaa. Tämä oli ratkaisu, kun havaittiin ettei metodi omaa muistia samalla tavalla kuin ohjelma. Staattisilla muuttujilla pystyttiin tallentamaan muuttujien arvo kutsujen välillä. Muuttuja *MaxAngleChange* on luotu jo tässä vaiheessa, vaikkei sitä varsinaisesti tarvita simulaatioissa. Farming Simulator 17 pelissä olosuhteet ovat ideaaliset, eikä järjestelmässä ole kohinaa tai nosturi heilu esimerkiksi tuulen vaikutuksesta. Antureiden poikkeaman määrittelyä testattiin aluksi myös kuvan 18 mukaisella rakenteella, jossa on kiinni xMove anturit, mutta sen avulla ei kyetty määrittelemään muuttujalle *MaxAngleChange* realistista arvoa.

Metodin ensimmäisellä kutsulla asetetaan *stateChanged* muuttuja epätodeksi ja muuttuja *changeRequest* nolnaan. Ensimmäisen kutsun jälkeen anturin poikkeaman määrittämiseen käytetään kolmea *IF*-ehtoa, joista kaksi on ensimmäisen sisällä. Metodien toiminta on esitetty ohjelmassa 5.

```

1 // Things to do on first cycle
2 IF stateChanged THEN
3   stateChanged := FALSE;
4   changeRequest := 0;
5 ELSE
6   // Calibrate sensor offset
7   IF JoystickDirectionBits[0] = 4 THEN
8     IF NOT BoomCalibration.TimerOnOff THEN
9       BoomCalibration.TimerOnOff := TRUE;
10    END_IF
11    IF BoomCalibration.TimerBool THEN
12      // Calculate angle change
13
14      AngleChange:=ABS(BoomKinematics.boom_k.CurrentJointPosition.Joint2)-
15      LastSensorValue);
16      IF AngleChange <= MaxAngleChange THEN
17        StateReady := TRUE;
18        BoomCalibration.TimerOnOff := FALSE;
19      ELSE
20        StateReady := FALSE;
21      END_IF
22    END_IF
23  ELSE
24    BoomCalibration.TimerOnOff := FALSE; // Reset the timer
25    LastSensorValue := BoomKinematics.boom_k.CurrentJointPosition.Joint2;
26  END_IF
27  END_IF
28  TiltMax := StateReady;

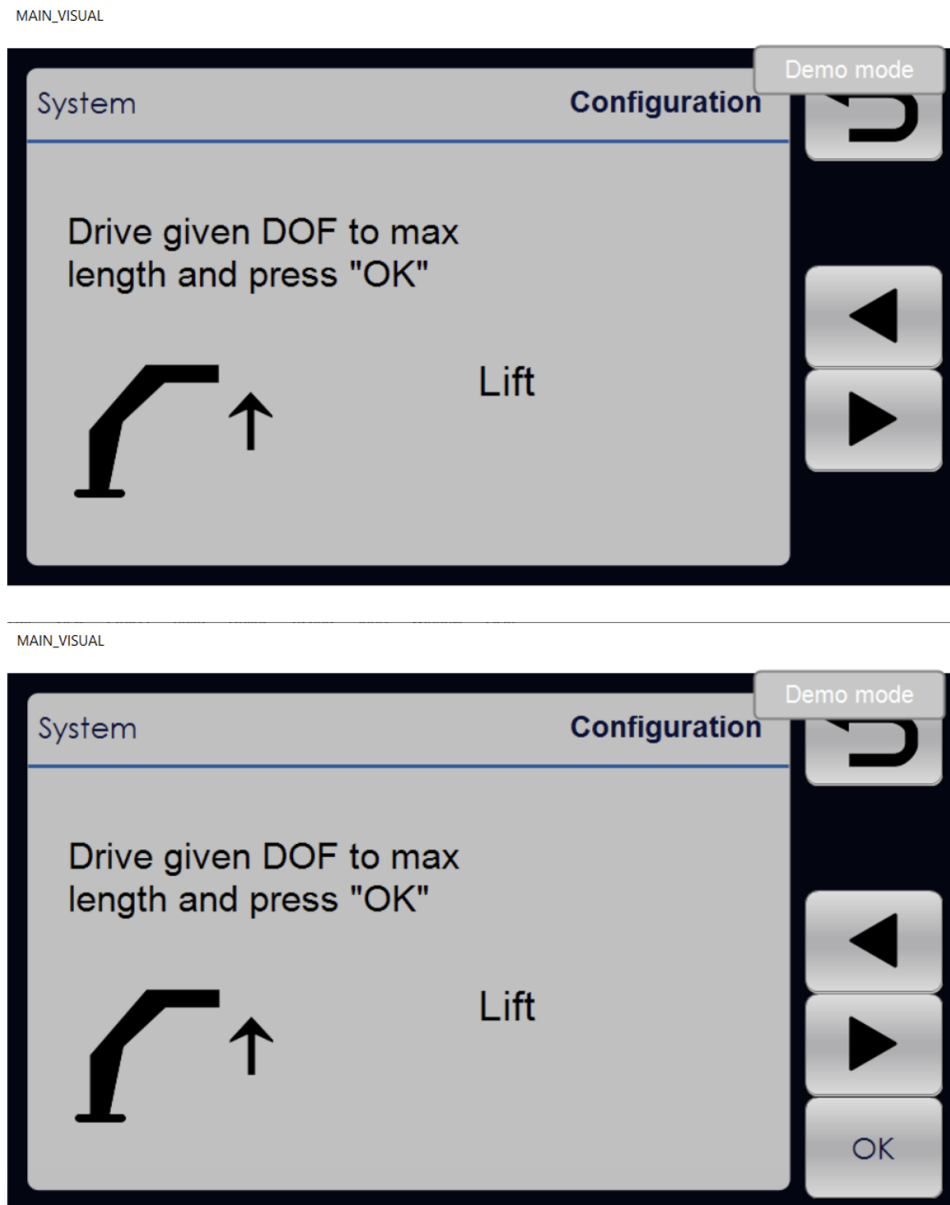
```

### ***Ohjelma 5. Metodin TiltMax ohjelmakoodi***

Ohjelmassa 5 ei ole toteutettuna rakennetta, joka tallentaisi itse anturin poikkeaman arvon. Tässä tutkimuksessa pyrittiin tekemään vain rakenne, joka kertoo koska puomin eri liikkeet on ajettu ääriasentoihin. Arvojen tallennuksen tulisi tapahtua käyttöliittymän avulla, jota käsitellään luvussa 6.3.2. Jotta metodi saisi arvon tosi, tulee haluttu liike ajaa ensiksi ääriasentoon ja ohjaus lopettaa. Tämän jälkeen ohjaus tulee aloittaa uudestaan ja järjestelmä laskee, pysyykö puomi sallituissa rajoissa yhden sekunnin ohjauksen jälkeen.

### **6.3.2 Käyttöliittymän ohjelmointi**

Näytön ohjelmakoodiin ei tarvinnut lisätä yhtään POU:ta. Käyttöliittymään lisättiin visualisointi, jossa antureiden poikkeamat määritellään. Luotu visualisointi on esillä kuvassa 22. Johdannossa esitettyjen tavoitteiden mukaan käyttöliittymän tulee ilmoittaa selkeästi, kun on todettu liikkeen olevan maksimissa.



**Kuva 22.** Luotu visualisointi antureiden poikkeaman määrittämiseen

Kuvassa 22 on kaksi PLC-simulaatiosta otettua kuvaa, ylemmässä kuvassa metodi *LiftMax* palauttaa arvon epätosi ja alemmassa arvon tosi. Kuten kuvasta voidaan havaita, käyttöliittymään ilmestyy näppäin *OK*, kun on todettu puomin olevan ääriasennossa. *OK*-näppäimeen ei ole lisätty toiminnallisuutta, vaan tällä hetkellä se toimii ainoastaan indikaattorina. Oikeassa reunassa olevilla nuolinäppäimillä voidaan liikkua edellisessä luvussa esitettyjen tilojen välillä.

Käyttöliittymän lisättiin tekstin lisäksi kuva havainnoimaan, mitä liikettä tulee ajaa ääriasentoon. Jos oltaisiin tilassa *TILT\_MAX\_POS*, olisi puomin kuvan oikealla puolella oleva nuoli kaareva, mutta osoittaisi silti ylöspäin. Tilassa *EXT\_MAX\_POS* nuoli osoittaisi oikealle.

## 6.4 Liikenopeuksien kalibrointi

Liikenopeudet kalibroidaan xCrane-järjestelmässä niin, että nosturia ohjataan ennalta määrätyllä ohjausarvolla. Tämän jälkeen IMU:n välittämän datan perusteella puomin asennon muutoksesta voidaan laskea hydraulisylinteriin virtaava tilavuusvirta. Huomioitavaa on se, että vaikka puhutaan liikenopeuksista, on mitattava arvo kuitenkin tilavuusvirtaa litroina minuutissa. Tämä johtuu siitä, kun pohjimmiltaan arvoja vertaillaan venttiilin vastekäyrää vasten.

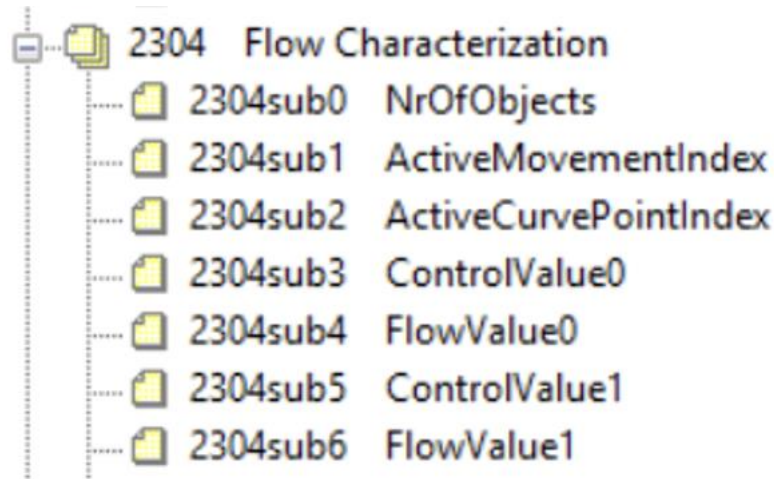
Liikenopeuksien kalibrointi aloitettiin hahmottelemalla, miten toiminta tulisi toteuttaa käyttöliittymässä. Lopulta päätettiin käyttää valmista visualisointia pohjana, joka piirtää kuvaajan näytölle. Kuvaajaan piirretään konfiguraatitiedostosta luettavat arvot, sekä sen jälkeen mitatut arvot. Kalibrointi suoritetaan maksimissaan 11 pisteestä, joista yksi on niin sanottu nollapiste. Nollapisteessä sekä ohjausarvo ja sylinteriin virtaava tilavuusvirta on nolla.

Antureiden poikkeaman määrittelyssä ohjelmistoa kehitettiin kontrollerin puolelle ja näyttöön tehtiin ainoastaan käyttöliittymään liittyvät lisäykset. Liikenopeuksien kalibrointiin luotiin yksi ohjelma, jolla on kaksi metodia, mutta lisäykset tehtiin täysin näytön ohjelmakoodiin. Tämä johtuu siitä, että controllerissa oli jo valmiina toiminnallisuus liikenopeuksien mittaamiseen ja muokkaamiseen, mutta toiminnon visualisointia ei ollut vielä luotu käyttöliittymään.

### 6.4.1 Näytön ohjelmointi

Näytön ohjelmakoodia kirjoittaessa objektien arvojen lukeminen ja kirjoittaminen tapahtuu *Display* nimisen GVL:n kautta. Objekteja käsitellessä ne haetaan kirjoittamalla eteen *Display*. ja loppuun *.value*, jos halutaan käsitellä objektin arvoa tai *.needs\_update*, kun halutaan päivittää arvo lukemista tai kirjoittamista varten.

Mekanismi liikenopeuksien mittaamiseen oli jo valmiina controllerissa, joten kontrollerin ohjelmakoodiin ei tarvinnut tehdä lisäyksiä. Kuitenkaan liikenopeuksia ei ollut ennen käsitelty käyttöliittymässä, joten liikenopeuksien kalibrointiin tuli lisätä objekteja objektisanakirjaan. Luoduista objekteista osa on esitetty kuvassa 23. Kuten aiemmin mainittiin, mitattavia pisteitä on korkeintaan 11. Tämä tarkoittaa sitä, että jokaiselle pisteelle on oma ohjausarvo- ja tilavuusvirtaobjekti.



**Kuva 23.** Liikenopeusten kalibrointia varten luodut objektit

Kuvasta 23 puuttuu siis 18 objektia. Kuvasta puuttuvat objektit ovat kaikki *ControlValue* ja *FlowValue* objekteja, viimeiset objektit ovat *ControlValue10* ja *FlowValue10*.

Liikenopeuksia kalibroidessa tulee lukea ja kirjoittaa kaikkiin muihin objekteihin, paitsi *FlowValue*-objekteihin, joita vain luetaan. Kun objekteja linkitetään muuttujiin, *ControlValue* ja *ActiveMovementIndex* objektit linkitettiin erikseen muuttujiin lukemista ja kirjoittamista varten. Tämän avulla voidaan tarkkailla, ovatko objektien arvot päivittyneet ja voidaanko uudet arvot kirjoittaa. Muuttujien nimissä on lopussa joku *W*- tai *R*-kirjain kuvastamassa sitä, käytetäänkö muuttujaa lukemiseen vai kirjoittamiseen. *W*-kirjain kuvastaa kirjoittamista. Kyseisiin muuttujiin kirjoitetut arvot päivitetään objektisanakirjaan *erikseen* pyytämällä näytön ohjelmakoodissa, kirjoittamalla esimerkiksi *ActiveMovementIndexR.needs\_update*.

Kuten antureiden poikkeaman määrittelyssä, myös liikenopeuksien kalibroinnissa käytetään *CASE*-rakenteella luotua tilakonetta. Luotu ohjelma on nimeltään *Movementspeed\_Curve*. Luodun ohjelman rajapinta on esitelty ohjelmassa 6.

```

1 PROGRAM Movementspeed_Curve
2 VAR
3   i, j : UINT; // For loops
4   CurrentState : MovementspeedState_t; // Used in state machine control
5   Points : ARRAY [0..10] OF UINT :=
6     [5,4,6,3,7,2,8,1,9,0,10]; // Order to calibrate points
7   InitialReadReady : BOOL; // Tells if system is ready to measure points
8   // Next variables are controlled by UI
9   Movement : SINT; // Tells what movement is under calibration
10  CurrentPoint : UINT; // Tells what point is under calibration
11  AdjustControl : SINT := 0; // Tells if we increase/decrease control
12  // Scale variables are used to calculate correct position of points
13  X_scale : REAL;
14  Y_SCALE : REAL;
15  CurveParameters : CurveParameters_t; // Variable linked to visualization
16 END_VAR
17 VAR CONSTANT
18   // Values to axels in visualization
19   X_AXL : ARRAY[0..10] OF INT :=
20     [-1000,-800,-600,-400,-200,0,200,400,600,800,1000];
21   Y_AXL : ARRAY[0..10] OF INT :=
22     [-125,-100,-75,-50,-25,0,25,50,75,100,125];
23   ORIGO : ARRAY[0..1] OF UINT := [200,120]; // Pixel location of origo
24   X_MAX_VALUE : UINT := 1000; // Max control value
25   Y_MAX_VALUE : UINT := 125; // Max flow value
26   // Furthest points from origo in pixels
27   X_FP : UINT := 351;
28   Y_FP : UINT := 40;
29   // Variables used instead of magic values
30   X_INDEX : BYTE := 0;
31   Y_INDEX : BYTE := 1;
32   MAX_POINTS : UINT := 11;
33 END_VAR

```

### **Ohjelma 6. *Movementspeed\_Curve* ohjelman rajapinta**

Ohjelmassa 6 esitettyssä rajapinnassa mainitaan *pixel location* ja *furthest point from origo in pixels* kommentteissa riveillä 23 ja 26. Tämä johtuu siitä, että käytetyssä simulaatiossa, joka piirtää kuvaajan käyttöliittymään, pisteiden sijainti annetaan pikseleinä. Näytön vasemmassa yläreunassa on pikseli, jonka sijainti on [0, 0]. Tämän takia myös ohjelmassa 6 rivillä 23 määritelty origo on erisuuri kuin [0, 0], koska kyseessä on sijainti näytöllä. Programissa *Movementspeed\_Curve* lasketaan siis kalibroittavien pisteiden sijainnit pikseleinä. Kyseinen ohjelman on esitetty ohjelmassa 7. Rivillä 4 esitelty muuttuja *CurrentState* on tyyppiä *MovementspeedState\_t*, joka on vastaava enumeraatio kuin luotiin antureiden poikkeaman määrittelyyn tilakoneen hallintaa varten.

```

1 // Setting the active point
2 Display.ActiveCurvePointIndex.value := TO_BYTE(Points[CurrentPoint]);
3 IF AdjustControl <> 0 THEN // Adjusting the control value. UI calls.
4   Display.ControlValueW[Points[CurrentPoint]].value :=
5     Display.ControlValueR[Points[CurrentPoint]].value+(25*AdjustControl);
6   Display.ControlValueW[Points[CurrentPoint]].needs_update := TRUE;
7   AdjustControl := 0;
8 END_IF
9 // State machine: changing states
10 CASE CurrentState OF
11 MovementspeedState_t.IDLE:
12   IF Movement = 1 THEN // Setting initial values for visualization
13     CurveParameters.X_IsReal := FALSE; // We use UINT
14     CurveParameters.Y_IsReal := FALSE;
15     CurveParameters.Hide_X_label := FALSE;
16     CurveParameters.Hide_Y_label := FALSE;
17     // Calculate scale values
18     X_scale :=
19       (TO_REAL(X_FP)-TO_REAL(ORIGO[X_INDEX]))/TO_REAL(X_MAX_VALUE);
20     Y_scale :=
21       (TO_REAL(ORIGO[Y_INDEX])-TO_REAL(Y_FP))/TO_REAL(Y_MAX_VALUE);
22     FOR i := 0 TO MAX_POINTS-1 DO
23       CurveParameters.HideInitialLine[i] := FALSE;
24       CurveParameters.HideLine[i] := FALSE;
25     END_FOR
26     CurveParameters.Y_INT_values := Y_AXL; // Set axel values
27     CurveParameters.X_INT_values := X_AXL;
28     // State change
29     CurrentState := Display.ActiveMovementIndexW.value :=
30       MovementspeedState_t.LIFT;
31     Display.ActiveMovementIndexW.needs_update := TRUE;
32   END_IF
33 MovementspeedState_t.LIFT:
34   // Initial read
35   IF Movement = 1 AND Display.ActiveMovementIndexR.value = 1 AND
36     InitialReadReady = FALSE THEN
37     InitialReadReady := ReadInitialParameters();
38   ELSIF Movement = 2 THEN // Change state
39     CurrentState := Display.ActiveMovementIndexW.value :=
40       MovementspeedState_t.TILT;
41     Display.ActiveMovementIndexW.needs_update := TRUE;
42     InitialReadReady := FALSE;
43   ELSIF Movement = 3 THEN
44     CurrentState := Display.ActiveMovementIndexW.value :=
45       MovementspeedState_t.EXT;
46     Display.ActiveMovementIndexW.needs_update := TRUE;
47     InitialReadReady := FALSE;
48   ELSE
49     Display.ActiveMovementIndexR.needs_update := TRUE;
50   END_IF

```

### ***Ohjelma 7. Ohjelma Movementspeed\_Curve***

Aina *Movementspeed\_Curve* ohjelman alussa asetetaan aktiivinen piste, jonka määrittää käyttöliittymän avulla koneen ohjaaja. Kun arvoja kalibroidaan, aktiivinen piste on se, jonka ohjausarvoa voidaan muuttaa ja jonka virtausarvoa luetaan. Kuten

ohjelmasta 7 havaitaan, mahdollinen arvon säätäminen suoritetaan ennen tilakoneeseen siirtymistä. Aina, kun käyttöliittymässä ollaan muussa näkymässä kuin arvojen kalibrointiin luodussa näkymässä, tilakone on tilassa *IDLE*. Kun siirrytään kalibrointiin, asettaa käyttöliittymä *Movement* muuttujan arvoksi 1, joka viittaa nostoliikkeeseen. Koska *Movementspeed\_Curve* ohjelmaa kutsutaan jokaisella syklillä, vasta tässä vaiheessa asetetaan visualisoinnin alkuarvot. Käyttämässämme visualisoinnissa on mahdollista piilottaa halutut pisteet ja viivat, mutta alustuksessa tämä ei ole toivottavaa. Pisteet ja viivat asetetaan näkyviksi riveillä 22–25.

Näytölle annetaan piirrettävien pisteiden koordinaatit pikseleinä, joiden välille visualisointi piirtää viivat. Jotta haluttu pikseli voidaan laskea, tarvitaan skaalauskerroin. Kertoimet lasketaan molemmille akseleille riveillä 26 ja 27. Skaalausarvoja käytetään ohjelman metodeissa.

Ohjelmassa 7 esitellään tilakoneesta ainoastaan tila *LIFT*, tämän tilan lisäksi tilakoneessa on vielä tilat *TILT* ja *EXT*. Nämä kolme tilaa ovat kuitenkin toiminnaltaan identtisiä, joten niitä kaikkia ei ole tarve erikseen esitellä. Käyttöliittymään tulee lukea konfiguraatitiedoston arvot liikenopeuksille. Kuitenkin objekteja pyrittiin luomaan mahdollisimman vähän, täten sekä konfiguraatitiedoston arvot, että mitatut arvot välitetään samoilla objekteilla. Tästä seuraa se, että ennen kalibrointia tulee lukea alkuarvot. Tämä suoritetaan ohjelman 7 rivillä 37, kun kutsutaan metodia *ReadInitialParameters*. Kun kyseinen metodi on suoritettu, se saa totuusarvon tosi, joka tallennetaan muuttujaan *InitialReadReady*. Tämän jälkeen voidaan aloittaa kalibrointi metodilla *ReadParameters*. Ohjelman *Movementspeed\_Curve* lopussa on *CASE*-rakenne, jossa kutsutaan metodia *ReadParameters*, kun *InitialReadReady* totuusarvo on tosi.

Metodin *ReadInitialParameters* rajapinnassa luodaan kolme *boolean*-tyyppistä muuttujaa. Rajapinta on esitetty ohjelmassa 8. Yksi muuttuja on ulostulo, jolla kerrotaan *Movementspeed\_Curve* ohjelmalle, että arvojen luku on valmis.

```

1 METHOD ReadInitialParameters
2 VAR
3   Check : BOOL := FALSE;
4 END_VAR
5 VAR_OUTPUT
6   Ready : BOOL;
7 END_VAR
8 VAR_STAT
9   Reading : BOOL := FALSE;
10 END_VAR
```

**Ohjelma 8.** Metodin *ReadInitialParameters* rajapinta

Ohjelmassa 8 luodaan myös muuttujat *Check* ja *Reading*, jotka alustetaan arvoon epätosi. *Reading* muuttuja on staattinen, joten se säilyttää arvonsa kutsujen välillä. Testausvaiheessa havaittiin, että simulaattorissa liikenopeuksien kalibroinnissa on käytetty 9 pistettä 11 sijaan. Tätä varten luotiin *Check* muuttuja, jotta voidaan tunnistaa, onko kalibrointi tehty 9 vai 11 pisteellä.

Ohjelmassa 9 on esitetty *ReadInitialParameters* metodi, jonka avulla luetaan konfiguraatiodoston arvot ja ne lähetetään visualisointiin, joka piirtää ne käyttöliittymän kuvaajaan.

```

1 IF Reading = FALSE THEN
2   Check := FALSE;
3   FOR i := 0 TO MAX_POINTS - 1 DO
4     CurveParameters.HideInitialPoint[i] := TRUE;
5     CurveParameters.HidePoint[i] := TRUE;
6     Display.ControlValueR[i].needs_update := TRUE;
7     Display.FlowValue[i].needs_update := TRUE;
8     Reading := TRUE;
9   END_FOR
10  ELSIF READING = TRUE THEN
11   FOR i := 0 TO MAX_POINTS - 1 DO
12     IF Display.ControlValueR[i].needs_update = FALSE AND
13        Display.FlowValue[i].needs_update = FALSE THEN
14       IF Display.ControlValueR[i].value = 0 AND
15          Display.FlowValue[i].value = 0 THEN
16         IF Check = False THEN
17           Check := TRUE;
18         ELSIF Check = TRUE THEN
19           FOR j := i-1 TO MAX_POINT-1 DO
20             CurveParameters.HideInitialPoint[j] := TRUE;
21             CurveParameters.HidePoint[j] := TRUE;
22             CurveParameters.HideInitialLine[j] := TRUE;
23             CurveParameters.HideLine[j] := TRUE;
24           END_FOR
25         END_IF
26       ELSE
27         Check := FALSE;
28       END_IF
29       CurveParameters.DrawPoints[i][X_INDEX] := ORIGO[X_INDEX];
30       CurveParameters.DrawPoints[i][Y_INDEX] := ORIGO[Y_INDEX];
31       CurveParameters.InitialDrawPoints[i][X_INDEX] :=
32         TO_UINT(Display.ControlValue[i].value*X_scale+ORIGO[X_INDEX]);
33       CurveParameters.InitialDrawPoints[i][Y_INDEX] :=
34         TO_UINT(ORIGO[Y_INDEX]-Display.FlowValue[i].value*Y_scale);
35       IF i = MAX_POINTS-1 THEN
36         IF Check = TRUE THEN // Used only if 9 points in calibration
37           FOR j := 0 TO MAX_POINTS - 3 DO
38             Points[j] := Points[j]-1; // Reorder points
39           END_FOR
40           Points[9] := 4; // Point 4 is zeropoint
41           Points[10] := 4;
42         END_IF
43         Reading := FALSE; //Reset value
44         Ready := TRUE;
45       END_IF
46     ELSE
47       EXIT; // If values are not updated
48     END_IF
49   END_FOR
50 END_IF

```

### ***Ohjelma 9. ReadInitialParameters metodi***

Metodin alussa riveillä 1-9 päivitetään objektisanakirjasta halutut arvot. Tämä tulee tehdä aina ensimmäisenä, koska joka liikkeen arvoja lukiessa käytetään samoja objekteja. Kun kontrollerille on ensin ilmoitettu, että aloitettiin esimerkiksi kääntöliikkeen kalibrointi,

päivittää kontrolleri objektisanakirjaan kyseisen liikkeen pisteiden arvot. Tämän jälkeen arvot päivitetään *Display* GVL muuttujiin. Kun arvo on päivitetty, asettuu kyseinen *needs\_update* bitti epätodeksi. Täten voidaan olla varmoja, että arvo on päivitetty ja tätä tarkastellaan riveillä 12 ja 13.

Aiemmin mainittiin, että simulaatiossa käytetään 11 pisteen sijaan 9 pistettä liikenopeuksien kalibroinnissa. Kuitenkin järjestelmässä on 11 pistettä käytössä joka tapauksessa, tällöin kaksi viimeistä pistettä asetetaan nollapisteksi. *Check* muuttujaa käytetään tunnistamaan, jos järjestelmä lukee peräkkäin kaksi nollapistettä. Jos havaitaan kaksi nollapistettä peräkkäin, kyseiset pisteet ja niihin johtavat viivat piilotetaan riveillä 19–24. Ohjelmassa 6 luodaan muuttuja *Points*, joka on jono. Kyseiseen muuttujaan on tallennettu haluttu järjestys käsitellä pisteet. Jos kalibroidaan 9 pisteellä, muokataan pisteiden järjestys ja asetetaan kaksi viimeistä pistettä nollapisteksi riveillä 36–42. Tämän jälkeen, kun konfiguraatitiedoston arvot on luettu, voidaan siirtyä liikenopeuksien mittaamiseen.

Kun alkuarvot on luettu ja piirretty käyttöliittymään, tulee seuraavaksi mitata liikenopeudet. Mitattujen liikenopeuksien piirtäminen käyttöliittymän kuvaajaan tapahtuu metodilla *ReadParameters*. Ohjelmassa 10 on esitetty kyseinen metodi. Huomioitavaa on se, ettei metodille *ReadParameters* tarvitse määrittää rajapinnassa uusia muuttujia.

```

1 IF Display.ControlValueW[Points[CurrentPoint]].needs_update = FALSE THEN
2   FOR i := 0 TO MAX_POINTS-1 DO
3     CurveParameters.HideInitialPoint[i] := TRUE;
4     CurveParameters.HidePoint[i] := TRUE;
5   END_FOR
6   CurveParameters.HideInitialPoint[Points[CurrentPoint]] := FALSE;
7   CurveParameters.HidePoint[Points[CurrentPoint]] := FALSE;
8   Display.ControlValueR[Points[CurrentPoint]].needs_update := TRUE;
9   Display.FlowValue[Points[CurrentPoint]].needs_update := TRUE;
10  CurveParameters.DrawPoints[Points[CurrentPoint]][X_INDEX] :=
11    TO_UINT(Display.ControlValueR[Points[CurrentPoint]].value*X_scale)
12    + ORIGO[X_INDEX];
13  CurveParameters.DrawPoints[Points[CurrentPoint]][Y_INDEX] :=
14    ORIGO[Y_INDEX] -TO_UINT(Display.FlowValue[Points[CurrentPoint]].value
15    *Y_scale);
16 END_IF

```

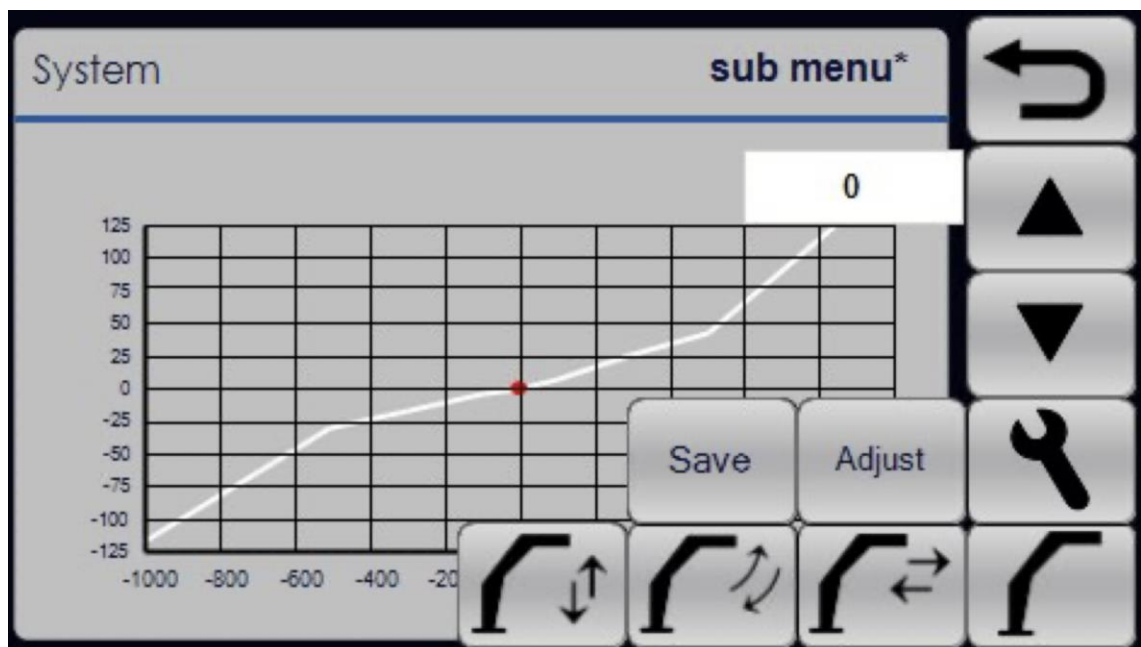
### **Ohjelma 10. Metodi *ReadParameters***

Ohjelmassa 10 ensimmäisellä rivillä tarkastetaan, että haluttu arvo on päivitetty. Tämä johtuu siitä, jos ohjausarvoa on muutettu ei kannata ennen sen päivittämistä lähettää arvoja visualisointiin. Jotta voidaan korostaa käyttöliittymässä kalibroinnissa olevaa pistettä, ensin riveillä 2–5 piilotetaan kaikki pisteet. Tämän jälkeen riveillä 6 ja 7 asetetaan näkyville kalibroinnissa oleva piste. Riveillä 8 ja 9 pyydetään päivittämään

kontrolli- ja virtausarvot, jonka jälkeen riveillä 10–15 lasketaan pisteille pikselisijainnit ja lähetetään ne visualisointiin. Huomioitavaa on se, ettei riveillä 8 ja 9 pyydetyt arvojen päivitykset ehdi toteutua ennen kuin pisteiden sijainnit lasketaan ja lähetetään visualisointiin. Tämä ei kuitenkaan aiheuta ongelmia, koska arvot päivittyvät seuraavalle syklille. Toisin kuin metodia *ReadInitialParameters* metodia *ReadParameters* suoritetaan jatkuvasti, kunnes vaihdetaan kalibroitavaa liikettä tai lopetetaan kalibrointi.

### 6.4.2 Käyttöliittymän ohjelmointi

Käyttöliittymässä päätettiin käyttää jo aiemmin xCrane-järjestelmässä käytettyä kuvaajan visualisointia. Tämän oheen tuli tehdä napit, joilla voidaan vaihtaa kalibroitavaa pistettä, säätää kalibroitavan pisteen ohjausarvoa ja vaihtaa kalibroitavaa liikettä. Käyttöliittymä on esitetty kuvassa 24.



**Kuva 24.** Liikenopeuksien kalibroinnin käyttöliittymä. Kuva simulaattoritesteistä.

Kuvassa 24 on esitetty luotu käyttöliittymänäkymä. Oikeassa reunassa näkyy luodut näppäimet. Alimmalla napilla mahdollistetaan kalibroitavan liikkeen vaihto, jota painamalla tulee näkyviin sen vasemmalla puolella olevat kolme näppäintä. Näistä kolmesta napista painamalla valitaan kalibroitava liike, jonka jälkeen ainoastaan oikeassa reunassa oleva näppäin jää näkyviin. Vastaavaa toimintoa on hyödynnetty pisteen säätämisen mahdollistamisessa. Painamalla näppäintä, jossa on jakoavain, tulee esiin *Adjust* ja *Save* näppäimet. *Adjust*-näppäintä painamalla tulee näkyviin aktiivisen pisteen ohjausarvo tekstikenttään. Kuvassa aktiivisen pisteen ohjausarvo on 0. Ohjausarvoa voidaan säätää nuolinäppäimillä. Jos pisteitä ei tarvitse säätää ja haluttu liike on valittuna, ei näkyvissä ole kuin oikeassa reunassa olevat näppäimet.

Kuvassa 24 on luettu nostoliikkeen alkuarvot, jotka on piirretty kuvaajaan valkoisella viivalla. Vihreällä viivalla kuvaajaan piirtyy mitatut pisteet. Pystyakselilla on virtausarvot ja vaaka-akselilla ohjausarvot. Kuvassa 24 ei ole vielä mitattu pisteitä, joten vihreää viivaa ei ole piirretty. Jos pisteen ohjausarvon säätö ei ole aktiivisena, nuolinäppäimillä voidaan vaihtaa aktiivista pistettä. Pisteet mitataan yksitellen painamalla nuolta ylöspäin ja edelliseen pisteeseen voidaan siirtyä painamalla nuolta alaspäin.

## 7. TULOSTEN ANALYSOINTI

Tässä luvussa analysoidaan saatuja tuloksia. Ensimmäisenä käydään läpi käyttöönottoprosessin simulaattoritestien tuloksia. Tämän jälkeen tehdään johtopäätöksiä tuloksien perusteella, minkä jälkeen pohditaan mahdollisia jatkokehityskohteita.

### 7.1 Testaaminen simulaattorilla

Kun järjestelmä oli todettu PLC-simulaatioilla toimivaksi, voitiin aloittaa järjestelmän testaaminen luvussa 5.2 esitellyn simulaattorin avulla. Simulaattoritestit aloitettiin tarkastelemalla, tunnistaako järjestelmä, kun puomista on ajettu haluttu liike maksimipituuteen. Kuvassa 25 on esitetty lähtötilanne, kun haluttua liikettä ei ole vielä ajettu maksimipituuteen.



**Kuva 25.** Nostoliikkeen poikkeaman määrittämisen lähtötilanne

Kuvasta 25 nähdään, että olemme aloittaneet antureiden poikkeaman määrittämisen käyttöliittymässä. Käyttöliittymästä nähdään myös, ettei järjestelmä anna virheellisesti tietoa, että liike olisi maksimipituudesta. Tämä voidaan todeta siitä, ettei kuvassa 22 esitettyä OK-kuvaketta näy oikeassa alareunassa.

Alkutilanteen jälkeen haluttu liike, eli nostoliike, ajettiin maksimipituuteen. Tämän jälkeen ohjaus pysäytettiin hetkeksi ja taas aloitettiin liikkeen ohjaaminen. Koska maksimipituus

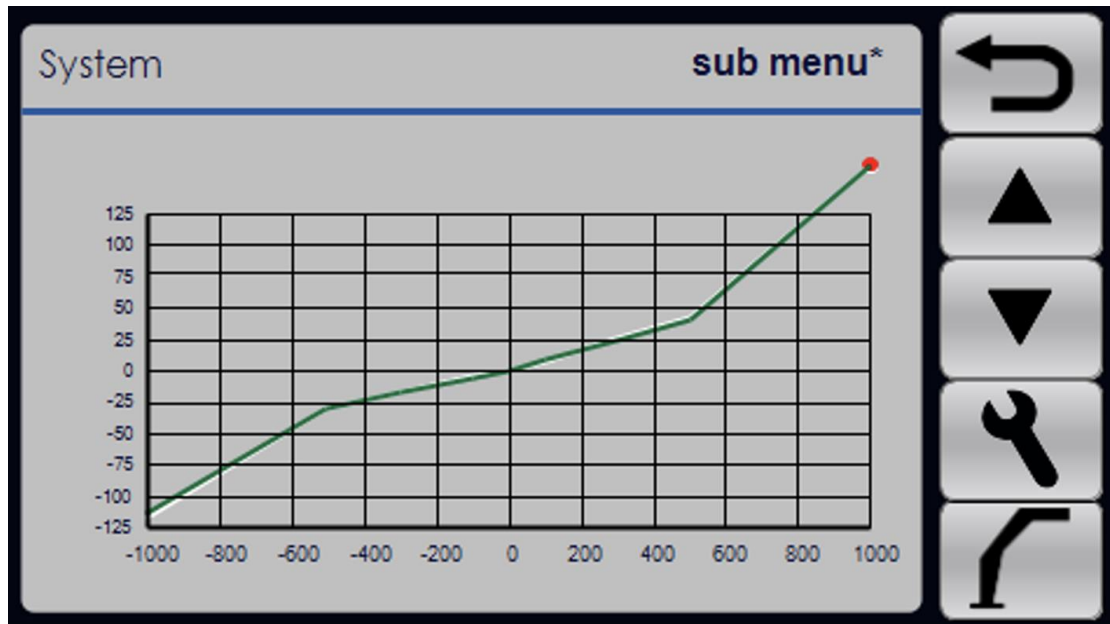
oli saavutettu, ei liikettä tapahtunut ohjauksesta huolimatta. Järjestelmä tunnisti tilanteen ja OK-kuvake tuli näkyviin. Toiminta toistettiin myös taittoliikkeelle sekä jatkeelle onnistuneesti. Kun poikkeaman määrittämiseen luotu toiminta oli todettu toimivaksi, siirryttiin testaamaan liikenopeuksien määrittämistä käyttöliittymän avulla.

Käyttöliittymä on toteutettu niin, että liikenopeuksien kalibroinnissa näytölle piirtyy ensin kuvaaja, josta näkyy konfiguraatitiedoston arvot. Kuvaaja on esillä kuvassa 26.



**Kuva 26.** Käyttöliittymään piirtyvä kuvaaja

Kuvasta 26 havaitaan, ettei kuvaaja mahdu täysin sille varatulle alueelle. Kuvassa näkyy myös mitatuille arvoille näkyvä vihreä kuvaaja. Kuvassa nollepiste on aktiivisena. Kalibrointi suoritettiin painamalla ylöspäin osoittavasta nuolinäppäimestä, jonka avulla edetään mittauspisteissä eteenpäin ja mitataan jokaisessa pisteessä liikenopeus. Kuvassa 27 näkyy käyttöliittymän näkymä simulaattorilla suoritettujen mittausten jälkeen. Kuva on otettu kehitysympäristön näkymästä, koska siten saavutetaan parempi tarkkuus kuvaan.



**Kuva 27.** Käyttöliittymän kuvaaja nostoliikkeen liikenopeuksien kalibroinnin jälkeen. Kuvaajasta voidaan päätellä, että liikenopeuksien kalibrointi toimii simulaattorilla. Tarkat numeeriset arvot ovat taulukossa 1.

Taulukko 1. *Nostoliikkeen liikenopeuskalibroinnin arvot*

Ohjausarvo	Konfiguraatitiedoston liikenopeusarvot (l/min)	Mitatut liikenopeusarvot (l/min)
-1000	-118	-117
-500	-29	-29
-300	-17	-18
-100	-5	-5
0	0	0
100	8	8
300	26	26
500	44	44
1000	164	164

Taulukon 1 arvot todistavat kuvasta 27 luodun havainnon liikenopeuskalibroinnin toimivuudesta simulaattorilla. Ainoastaan ohjausarvolla -300 ja -1000 on pieni ero arvoissa, mutta ero on minimaalinen ja voi johtua osittain pyöristyksestä. Työtä tehdessä

havaittiin myös, ettei arvot aina ole välttämättä täysin identtiset mittausten välillä, joten tulosta voidaan pitää erinomaisena.

Kuvassa 27 valkoinen kuvaaja ei peity täysin. Tämä voi johtua taulukossa 1 olevien erojen lisäksi näytön suhteellisen pienestä resoluutiosta sekä arvojen pyöristämisestä laskutoimituksissa. Kuvaajan antamaa tulosta voidaan kuitenkin pitää tarpeeksi tarkkana.

Simulaattorilla testattiin myös ohjausarvojen muuttamista, vaikka tulokset olivat mitatessa erinomaiset. Myös ohjausarvon muuttaminen ja uudella ohjausarvolla mittaaminen onnistui. Testit suoritettiin myös taittoliikkeelle sekä jatkeelle.

## 7.2 Johtopäätökset

Luvussa 7.1 esitettyjen tulosten mukaan, voidaan luoda johtopäätös, että työn tavoitteeseen päästiin. Tutkimuksessa oli tarkoitus selvittää, voidaanko metsätyökoneen xCrane-kärkiohjausjärjestelmän käyttöönottoprosessi toteuttaa järjestelmän omalla käyttöliittymällä CODESYS-kehitysympäristön sijaan. Tuloksena on, että voidaan toteuttaa ja luotu järjestelmä on valmis testattavaksi oikealla nosturilla.

Järjestelmään onnistuttiin luomaan mekanismi, joka tarkastelee nosturin toimintaa antureiden poikkeamia määrittäessä. Tällä voidaan estää käyttäjää tallentamasta konfiguraatiodostoon virheellisiä arvoja. Liikenopeuksien mittaaminen, mittaustulosten visualisointi sekä vertailu onnistuttiin toteuttamaan järjestelmän näytölle. Saavutettuihin tuloksiin voidaan olla tyytyväisiä.

Työn tavoitteena oli yksinkertaistaa käyttöönottoprosessia niin, ettei jatkossa tarvittaisi CODESYS-kehitysympäristöä tuntevaa ja käyttöä hallitsevaa henkilöä, joko läsnä tai etänä, käyttöönottoprosessin suorittamiseen, eikä tarvitsisi luoda yhteyttä tietokoneen ja TEC152-kontrollerin välille. Tämäkin tavoite saavutettiin.

Kaiken kaikkiaan, voidaan todeta, että työssä päästiin tavoitteisiin. Tulokset on todettu simulaattoriympäristössä. Käyttöönottoprosessista onnistuttiin tekemään selkeä ja helppokäyttöinen, eikä sen suorittamiseen vaadita koulutusta tai tietokonetta.

## 7.3 Tulevaisuuden jatkokehitys

Järjestelmästä löydetään myös jatkokehityskohteita. Ensimmäinen selkeä jatkokehityskohde on todentaa toimivuus oikean nosturin avulla. Tämän jälkeen isoimmat jatkokehityskohteet ovat lisäyksiä käyttöliittymään.

Antureiden poikkeaman määrittämisen tarkoituksena on mahdollistaa puomin asennon riittävän tarkka määrittäminen. Järjestelmää voidaan kehittää niin, että käyttöliittymään piirtyisi kuva, joka kuvaisi puomin asentoa. Tämän avulla saataisiin lisättyä varmuutta poikkeamien määrittämisen onnistumisesta, kun puomin asentoa voitaisiin vertailla näytölle piirtyneeseen asentoon. Järjestelmän tulisi päivittää asentoa näytölle jatkuvasti, jotta puomin liikkeiden muutokset lasketulle asennolle voitaisiin todentaa.

Liikenopeuksien kalibroinnissa yksi jatkokehityskohde olisi näyttää näytöllä aktiivisen pisteen ohjausarvo, sekä virtausarvo. Arvot tulisi näyttää sekä mitatuille tuloksille, että konfiguraatiodiestosta. Jatkossa tulisi myös tutkia mahdollisuuksia liikenopeuksien mittaamisen automatisointiin.

# LÄHTEET

- [1] Technion Oy, "Technion Oy, ". Saatavilla: <https://technion.fi/fi/yritys/>. [Haettu 26 syyskuu 2023].
- [2] Technion Oy, "xCrane PRO, ". Saatavilla: <https://technion.fi/fi/metsajarjestelmat/xcrane-pro-fi/>. [Haettu 27 elokuu 2023].
- [3] "Kronos, ". Saatavilla: <https://kronos.fi/fi/tuote/kargo-121/>. [Haettu 7. heinäkuu 2023].
- [4] J. Uusitalo ja V.-P. Kivinen, Metsäteknologian perusteet, Helsinki: Tapio, 2023.
- [5] H. Kauranne, J. Kajaste ja M. Vilenius, Hydraulitekniikka, Helsinki: SanomaPro, 2013.
- [6] M. Geimer, Mobile working machines, Warrendale, PA : SAE International, 2020.
- [7] S.-M. Hirvonen, *Liikkuvan työkonteen turvallisuus*, Diplomityö. Hydrauliiikan ja automatiikan laitos, 2014.
- [8] O. Lindroos, O. Ringdahl, P. La Hera, P. Hohnloser ja T. Hellström, "Estimating the Position of the Harvester Head- a Key Step towards the Precision Forestry of the Future?," *Croatian Journal of Forest Engineering*, tammikuu 2015.
- [9] P. Multanen, *IHA-1102 Hydrauliiikan ja koneautomaation perusteet*, 2020.
- [10] Technion Oy, "Technion xCrane PRO feature overview" 1 lokakuu 2020.. Saataville: <https://www.youtube.com/watch?v=NRUX9gubMkA&t=390s>. [Haettu 11 heinäkuu 2023].
- [11] L. Vlacic, M. Parent ja F. Harashima, Intelligent vehicle technologies, Oxford : Butterworth-Heinemann, 2001.
- [12] R. Davis, A. Burns, R. Brill ja J. Lukkien, "Controller Area Network (CAN) schedulability analysis : refuted, revisited and revised," *Real-time systems, Vol. 35*, pp. 239-272, 30. tammikuu 2007.
- [13] CAN in Automation, "can-cia.org, ". Saatavilla: <https://www.can-cia.org/can-knowledge/can/can-history/>. [Haettu 13 heinäkuu 2023].
- [14] K. Palovuori, "EE.ELE.520 Verkotetut sulautetut järjestelmät -luentokalvot," 2022.
- [15] "Electrical Academia," . Saatavilla: [www.electricalacademia.com](http://www.electricalacademia.com). [Haettu 26 heinäkuu 2023].
- [16] S. A. Shweta, D. P. Mukesh ja B. N. Jagdish, "Implementation of Controller Area Network (CAN) Bus (Building Automation)," tekijä: *Advances in Computing, Communication and Control*, 2011.
- [17] S. Liu, Engineering Autonomous Vehicles and Robots: The DragonFly Modular-based Approach, Newark: Wiley, 2020.
- [18] C. Technologies, "Copperhill Technologies," 10 kesäkuu 2019. Saatavilla: <https://copperhilltech.com/blog/controller-area-network-can-bus-physical-layer-and-bus-topology/>. [Haettu 12 elokuu 2023].
- [19] M. D. Natale, A. Ghosal, P. Giusto ja H. Zeng, Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice, New York, NY: Springer-Verlag, 2012.
- [20] C. i. Automation, "CANopen - The standardized embedded network, ". Saatavilla: <https://www.can-cia.org/canopen>. [Haettu 12 elokuu 2023].
- [21] National Instruments, 31 maaliskuu 2023. Saatavilla: <https://www.ni.com/fi-fi/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-basics-of-canopen.html>. [Haettu 17 elokuu 2023].
- [22] R. Siegwart, I. R. Nourbakhsh ja D. Scaramuzza, Introduction to Autonomous Mobile Robots, MIT Press, 2011.
- [23] B. Siciliano ja O. Khatib, Springer Handbook of Robotics, Springer International Publishing AG, 2016.

- [24] J. Manner, A. Mörk ja M. Englund, "Comparing forwarder boom-control systems based on an automatically recorded follow-up dataset," *Silva fennica*, 2019.
- [25] Kronos, "Kronos Product Catalogue,". Saatavilla: [https://kronos.fi/wp-content/uploads/2023/08/produktkatalog\\_juni-2023\\_eng\\_komprimerad.pdf](https://kronos.fi/wp-content/uploads/2023/08/produktkatalog_juni-2023_eng_komprimerad.pdf). [Haettu 25 elokuu 2023].
- [26] A. Sokolov, A. Seliverstov ja Y. Sukhanov, "Forest machine automation and ergonomics," *E3S Web of Conferences*, 2023, Vol.389, p.3002, Article 03002, 2023.
- [27] T. Zemánek ja P. Filo, "Influence of Intelligent Boom Control in Forwarders on Performance of Operators," *Croatian Journal of Forest Engineering*, 2022.
- [28] MathWorks, "Inverse Kinematics,". Saatavilla: <https://www.mathworks.com/discovery/inverse-kinematics.html>. [Haettu 26 elokuu 2023].
- [29] B. Löfgren ja J. Wikander, "Kinematic Control of Redundant Knuckle Booms," *International Journal of Forest Engineering*, 2009.
- [30] J. Manner, O. Gelin, A. Mörk ja M. Englund, "Forwarders crane's boom tip control system and beginner-level operators," *Silva Fennica*, 2017.
- [31] Technion Oy, "xMove Sensor," 27 marraskuu 2018. Saatavilla: <https://technion.fi/wp-content/uploads/xmove-001-01-data-sheet.pdf>. [Haettu 28 elokuu 2023].
- [32] Technion Oy, "KÄYTTÖOHJE xCrane PRO,". Saatavilla: <https://technion.fi/wp-content/uploads/xcrane-pro-user-manual-v1504-fi.pdf>. [Haettu 27 elokuu 2023].
- [33] Topcon, "Technical Data Sheet OPUS A3 ECO Basic," 17 tammikuu 2018. Saatavilla: <https://technion.fi/wp-content/uploads/XCRANEDISP250-C0000-data-sheet.pdf>. [Haettu 27 elokuu 2023].
- [34] Technion Oy, "TEC152,". Saatavilla: <https://technion.fi/wp-content/uploads/tec152-brochure-fi.pdf>. [Haettu 27 elokuu 2023].
- [35] CODESYS, "CODESYS - The comprehensive software suite for automation technology,". Saatavilla: <https://www.codesys.com/the-system.html>. [Haettu 28 elokuu 2023].
- [36] CODESYS, "Why CODESYS?,". Saatavilla: <https://www.codesys.com/the-system/why-codesys.html>. [Haettu 28 elokuu 2023].
- [37] CODESYS, "CODESYS VISUALIZATION,". Saatavilla: <https://www.codesys.com/products/codesys-visualization.html>. [Haettu 28 elokuu 2023].
- [38] K. H. John ja M. Tiegelkamp, IEC 61131-3: Programming Industrial Automation Systems Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids, Berlin, Heidelberg : Springer Berlin Heidelberg : Imprint: Springer, 2010.