Kuisma Rainio

Demolaitteen käyttöönotto ja ohjelmointi:

IoT:n soveltaminen liikkeenohjauksessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Kone- ja tuotantotekniikka

Insinöörityö

18.5.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Kuisma Rainio Demolaitteen käyttöönotto ja ohjelmointi: IoT:n soveltaminen liikkeenohjauksessa 37 sivua + 2 liitettä 18.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Kone- ja tuotantotekniikka
Suuntautumisvaihtoehto	Koneautomaatio
Ohjaajat	Lehtori Heikki Paavilainen Myyntipäällikkö Olli Sihvo

Insinöörityön tavoitteena oli ottaa käyttöön Bosch Rexroth Oy:lle suunniteltu demolaite, toteuttaa laitteelle liikkeenohjausohjelma, luoda Android-pohjaiseen näyttöön käyttöliittymä laitteen operointiin käyttäen Industry 4.0 -teknologiaa sekä tutustua Open Core Engineering -teknologiaan.

Industry 4.0:lla tarkoitetaan niin sanottua neljättä teollista vallankumousta, jolla viitataan informaatioteknologian ja teollisuuden sulautumiseen. Open Core Engineering -teknologia on Bosch Rexrothin luoma tapa soveltaa Industry 4.0 käytännössä. Open Core Engineering -teknologian avulla voidaan hyödyntää perinteistä tietokoneohjelmointia ja älylaitteita automaatiosovelluksien ohjauksessa ja tiedonkeruussa. Tässä työssä ohjattiin servomekanismeja Android-applikaatiolla rinnan PLC-ohjelman kanssa.

Toteutettu demolaite koostuu kolmesta servoakselista. Laite on tikanheittopeli, jota käyttäjä ohjaa Android-tablettitietokonetta kallistelemalla, siihen luodun käyttöliittymän avulla. Lisäksi laitteeseen ohjelmoitiin automaattinen työkierto sellaisia tilanteita varten, joissa sillä ei ole käyttäjää. Tablettitietokone yhdistetään servokäyttöihin WLAN-verkon yli, ja ohjaus toimii reaaliajassa. Android-applikaatio ja PLC-ohjelma vaihtavat tietoja keskenään ohjelman suorituksen aikana.

Työlle asetetut tavoitteet saatiin täytettyä lukuun ottamatta osana liikkeenohjausta olevaa värähtelynvaimennusteknologian käyttöönottoa, jota aikataulullisista syistä ei ehditty toteuttamaan. Laitteen kehitystyötä tullaan kuitenkin jatkamaan tämän insinöörityön ulkopuolella.

Avainsanat	Open Core Engineering, IoT, liikkeenohjaus

Author Title Number of Pages Date	Kuisma Rainio Commissioning and Programming of a Demo Device: Applying IoT to Motion Control 37 pages + 2 appendices 18 May 2017
Degree	Bachelor of Engineering
Degree Programme	Mechanical engineering
Specialisation option	Machine automation
Instructors	Heikki Paavilainen, Senior Lecturer Olli Sihvo, Sales Manager

The objective of this Bachelor's thesis was to commission a demo device developed for Bosch Rexroth Ltd. In addition, the aim was to implement a motion control program for the device, to create a user interface in an Android based smart device for the operation of the demo device using Industry 4.0 technology, and also to familiarize with Open Core Engineering.

Industry 4.0 stands for the so called fourth industrial revolution, and it refers to the merger of information technology and the industry. Open Core Engineering is a method created by Bosch Rexroth, for applying Industry 4.0 in practice. With Open Core Engineering one can utilize conventional computer programming and smart devices for controlling and harvesting information from automation applications. In this particular task, servo mechanisms were controlled by an Android application simultaneously with a PLC program.

The device in question consists of 3 servo axes. It is a dart throwing game, which a user controls by tilting an Android tablet, by means of a user interface created for it. In addition, an automatic cycle for the device was created for situations when it has no user. The tablet is connected to the servo drives over a WLAN network, and the control works in real time. The Android application and the PLC program exchange information during the execution of the program.

In conclusion, the targets set for the thesis were achieved, except for the commissioning of the vibration damping technology, which was left outside the thesis due to lack of time. In future, however, the development of the device will be continued.

Keywords	Open Core Engineering, IoT, Motion control

Sisällys

Liitteet

Liite 1. Toimilaitteen paikkaohje ja todellinen sijainti ajan funktioina

Liite 2. Moduulien moottorien valinnat (ote keväällä 2016 tehdystä projektista)

1 Johdanto

Keväällä 2016 olin toteuttamassa Bosch Rexroth Oy:lle projektia, jossa suunniteltiin laite demonstroimaan yrityksen kehittämää värähtelynvaimennusteknologiaa messuilla ja vastaavissa tapahtumissa. Bosch Rexroth Oy päätti toteuttaa laitteen käytännössä. Tämän insinöörityön tavoitteena oli ottaa kyseinen laite käyttöön, toteuttaa laitteelle liikkeenohjausohjelma, luoda Android-pohjaiseen näyttöön käyttöliittymä laitteen operointiin käyttäen Industry 4.0 -teknologiaa sekä tutustua Open Core Engineering -teknologiaan. Industry 4.0 - ja Open Core Engineering -teknologioissa on kyse viime aikoina paljon esillä olleesta esineiden internet[1] -käsitteestä.

Tässä projektissa toteutettavan laitteen pääkomponentit ovat kolme sähköservomekanismein ohjattavaa lineaaritoimilaitetta. Ohjauksen toteuttaminen poikkeaa perinteisestä servomekanismien ohjauksesta, koska ohjaus toteutetaan osittain Android-laitteella reaaliajassa WLAN[2]-verkon yli. Vastaavaa ei ole koskaan aikaisemmin toteutettu Suomessa, ja se on myös maailmalla toistaiseksi harvinaista teknologiaa. Syksyllä 2017 Helsingissä tullaan järjestämään teknologiamessut, jossa tässä projektissa aikaansaatua laitetta on tarkoitus esitellä.

Bosch Rexroth AG on voimansiirron, ohjauksen ja liikkeenhallinnan ratkaisujen maailmanlaajuinen toimija. Yritys valmistaa ja kehittää ratkaisuja erilaisiin automaattisiin prosesseihin. Sen toiminta myynti- ja huoltoverkostoineen sekä tuotantolaitoksineen kattaa yli 80 maata. Yrityksen liikevaihto vuonna 2016 oli 5 miljardia euroa ja henkilöstömäärä 29 500 [1]. Bosch Rexroth Oy on toiminut Suomessa vuodesta 1978. Yrityksen tuoteohjelmaan kuuluvat teollisuushydrauliikan, sähkökäyttöjen ja ohjausjärjestelmien, lineaari- ja kokoonpanotekniikan ja mobilehydrauliikan komponentit, järjestelmät sekä huoltopalvelut. Yrityksen toimipaikat sijaitsevat Vantaalla ja Tampereella, ja vuonna 2016 se työllisti 100 henkeä. Samana vuonna yrityksen kokonaismyynti oli 82,5 miljoonaa euroa [1].

[1] Internet of things, IoT

[2] Wireless Local Area Network, langaton lähiverkkoyhteys

2 Industry 4.0

Industry 4.0 -käsitteellä viitataan niin kutsuttuun neljänteen teolliseen vallankumoukseen. Tässä yhteydessä teolliset vallankumoukset voidaan kategorisoida seuraavasti:

1. Vuosi 1780 – Mekanisaatio, veden valjastaminen tehonsiirtoon ja höyryvoima
2. Vuosi 1870 – Massatuotanto, kokoonpanolinja ja sähkö
3. Vuosi 1960 – Tietokoneet ja automaatio
4. Vuosi 2012 – Esineiden internet, älykäs automaatio, yhdistetty teollisuus ja kyberfyysiset järjestelmät[3].

Industry 4.0:n keskiössä on data. Data on neljännen teollisen vallankumouksen raaka-aine, ja sen selkäranka on esineiden internet. Esineistä tulee älykkäämpiä, jotta ne voivat suorittaa toimintoja riippumattomasti ja kommunikoida ympäristönsä ja muiden esineiden kanssa internetin välityksellä. Industry 4.0 kuvaa informaatioteknologian ja teollisuuden sulautumista. Teollisuudessa on tarkoituksenmukaista:

- välttää ylimääräisiä kustannuksia, seisokkiaikoja ja pullonkauloja
- minimoida hävikki, virheet ja varastossa olevat tuotteet
- optimoida ja kehittää tuottavuutta
- parantaa tuotteiden laatua
- nopeuttaa ja helpottaa palvelua
- saada aikaan tuotteiden ja osien täydellinen jäljitettävyys niiden koko elinkaarelle.

Informaatioteknologian sulautuminen teollisuuteen mahdollistaa kaiken tämän automaation ja tietokoneohjelmistojen lisääntyessä jatkuvasti tuotannossa.

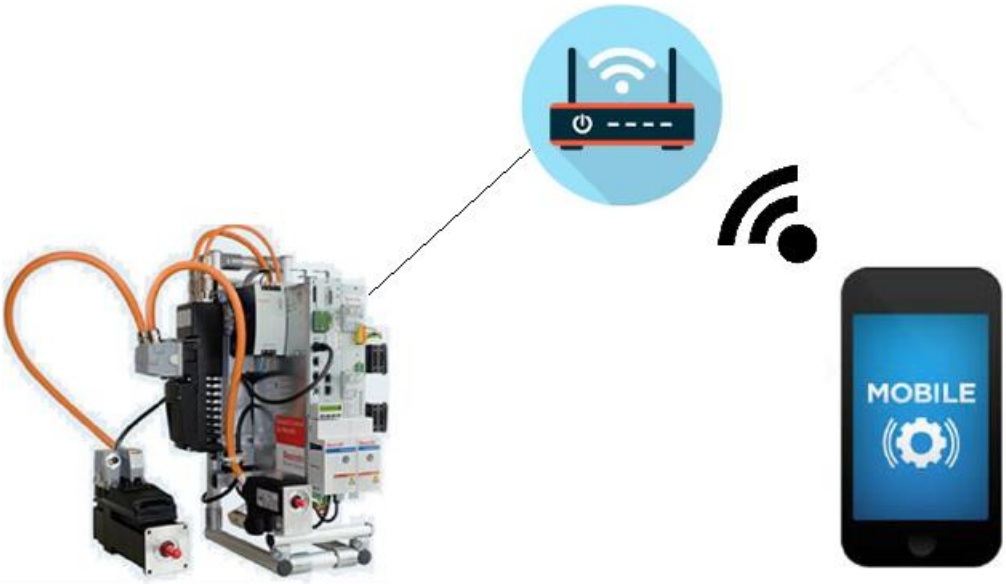
[2.]

[3] Cyber-Physical System (CPS). Kyberfyysisillä järjestelmillä tarkoitetaan mekaanisfyysisiä järjestelmiä, jotka sisältävät antureita ja toimilaitteita, tietokoneohjelmistoa ja yhteyksiä ympäristöön. Älypuhelin on IT-maailman esimerkki kyberfyysisestä laitteesta.

3 Open Core Engineering

Open Core Engineering (OCE) on Bosch Rexrothin luoma käsite. OCE yhdistää aikaisemmin erillään olleet PLC:n[4] ja IT:n[5] kokonaisuudeksi, joka käsittää avoimia standardeja, ohjelmointityökaluja, ohjelmapaketteja ja Open Core -rajapinnan sallimaan uudenlaisen vapauden automaatiosovelluksiin. OCE mahdollistaa perinteisten tietokoneohjelmointikielten käytön IEC 61131-3[6] -kielten rinnalla automaatiosovelluksissa.

Käytännön ilmentymä edellä mainituista asioista on, että ohjelmointiympäristöjä ja ohjelmointikieliä, joita ei aikaisemmin ole voinut käyttää automaation toimilaitteiden ohjauksessa, voi nyt käyttää ohjaukseen yhdessä PLC:n kanssa tai ilman. Kyseessä olevassa työssä tullaan käyttämään Android-ohjelmointia ja PLC-ohjelmointia rinnan saman järjestelmän ohjaamisessa. Kuvassa 1 on esitelty tässä työssä sovellettava Open Core -verkon topologia.



Kuva 1. Open Core -verkon topologia [3]. Kuvassa vasemmalla on servokäyttö ja siihen yhdistetty servomoottori. Servokäyttö on yhdistetty Android-laitteeseen WLAN-reitittimen välityksellä.

[4] Programmable logic controller, ohjelmoitava logiikka

[5] Informaatioteknologia

[6] Kansainvälinen standardi, joka käsittelee PLC-ohjelmointikieliä.

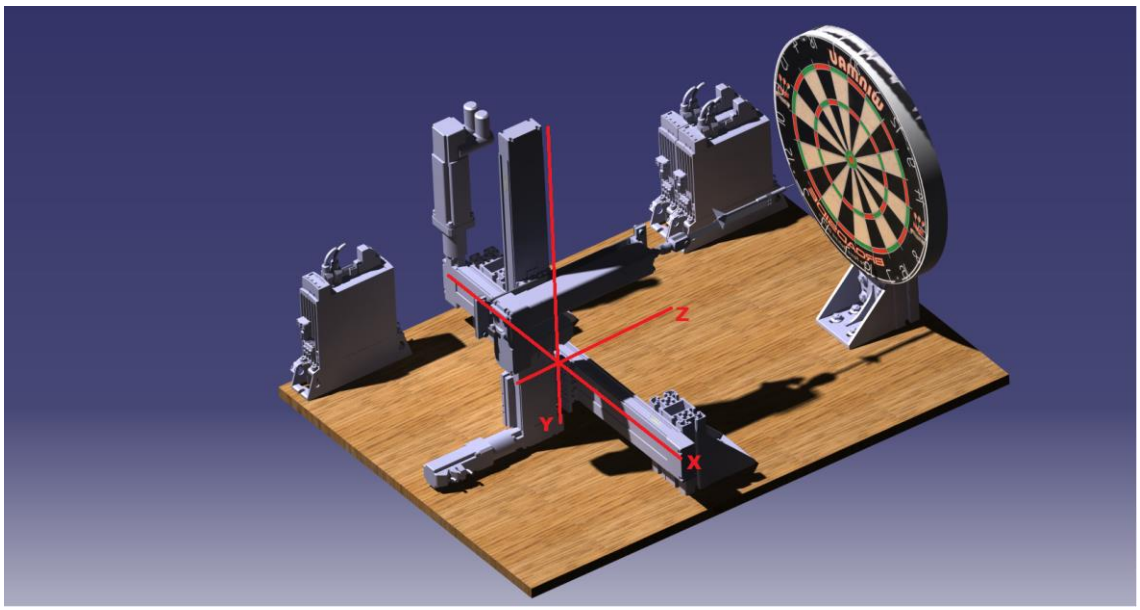
OCE mahdollistaa Industry 4.0:n toteuttamista käytännössä. Yksi mahdollinen sovellutus on, että verkkoon kytketyt koneet lähettävät reaaliaikaista dataa omasta tilastaan esimerkiksi niistä vastaavan henkilön kannettavaan älylaitteeseen[7]. Näin huoltotoimenpiteitä voidaan suorittaa ennen vikatilanteita tai vikatilanteessa välittömästi vian ilmetessä. OCE mahdollistaa myös esimerkiksi poikkitieteellisen laitteiden suunnittelun. Reaaliaikaisten simulaatio-ohjelmistojen avulla esimerkiksi sähköinsinööri, mekaniikkasuunnittelija ja ohjelmoija voivat samaan aikaan suorittaa tehtäviä, jotka aikaisemmin on pitänyt suorittaa porrastetusti. Yksi tämän projektin kannalta olennainen OCE:n mahdollistama sovellutus on älylaitteiden käyttö toimilaitteiden liikkeenohjaukseen. Kyseiset laitteet sisältävät valmiiksi suuren määrän antureita ja oheislaitteita, joita voidaan käyttää hyväksi osana toimilaitteiden ohjausta. Käytännössä esimerkiksi servokäyttöjä voidaan ohjata älypuhelimen kiihtyvyysantureiden dataa hyväksi käyttämällä kirjoittamatta riviäkään PLC-koodia.

[3; 4; 5.]

4 Toimintasuunnitelma

Kuvassa 2 nähdään demolaitteen alkuperäinen 3D-malli, jonka pohjalta laitetta lähdettiin toteuttamaan. Laitteen kokoonpanon suorittaa erillinen työryhmä. Laitteeseen on tullut joitakin alkuperäisestä suunnitelmasta poikkeavia rakenteellisia muutoksia sekä komponenttilisäyksiä, mutta ne eivät vaikuta ohjauksen toteuttamiseen. Laitteen kokoaminen ei kuulu tähän insinöörityöhön, mutta se on siihen vahvasti kytköksissä, koska kokoonpanossa esiintyvät tekijät vaikuttavat toimilaitteiden liikkeisiin ja täten ohjaukseen.

[7] Älylaitteiksi käsitetään esimerkiksi älypuhelimet ja tablettitietokoneet.



Kuva 2. Demolaitteen 3D-malli. Laitteessa on kolme toisiinsa kiinnitettyä lineaaritoimilaitetta, joista yksi liikkuu x- toinen y- ja kolmas z-akselin suunnassa. Z-akselin suunnassa liikkuva laite on sähkömekaaninen sylinteri, jonka männänvarren päässä on kiinnitettynä värähtelevä terästanko, johon on kiinnitetty tikka.

Toteutettava demolaite, on interaktiivinen ”tikanheittopeli”. Laitetta ohjataan käyttäjän toimesta tablettitietokoneella, jossa on Android-käyttöjärjestelmä. Tarkoitus on saada toimilaitteet liikkumaan Android-laitetta kallistamalla. Toiminta olisi mahdollista toteuttaa ilman PLC:tä ohjaamalla servokäyttöjä Open Core -rajapinnan kautta suoraan servokäyttöjen parametreihin kirjoittamalla, mutta koska laitteelle halutaan myös automaattinen käyttäjästä riippumaton työkierto, toteutetaan ohjaus tekijän osaaminen huomioon ottaen mahdollisimman suurelta osin PLC-ohjelmalla.

Suunnitelma laitteen toiminnasta on, että käyttäjä ohjaa kahta lineaaritoimilaitetta tikkataulun etupintaa vastaan samansuuntaisessa tasossa (xy-tasossa, ks. kuva 2) Android-laitetta kallistamalla kokoonpanon määrittämällä etäisyydellä tikkataulusta. X-suuntaisen toimilaitteen kiihtyvyys saa sähkömekaanisen sylinterin päässä olevan värähtelijän värähtelemään, jolloin värähtelynvaimennusteknologiaa voidaan esitellä. Tikka ”heitetään”, eli sähkömekaaninen sylinteri suorittaa iskun z-suunnassa, kun sille annetaan komento Android-laitteelta, jolloin laitetta kallistamalla ei voi enää vaikuttaa xy-tason liikkeisiin. Suorituksen jälkeen laite ohjautuu automaattisesti asemaan, jossa tikan kärki on taulun etupinnan rajaaman tason ulkopuolella ja käyttäjä voi suorittaa seuraavan ”heiton”. Ollessa vailla käyttäjää, laite suorittaa automaattista työkiertoa. Automaattinen kierto voi olla esimerkiksi, että tikka ”heitetään” taulun kaikkiin lukuihin vuorotellen. Tarkkaa työkiertoa ei ole välttämätöntä lyödä lukkoon suunnitteluvaiheessa, vaan sen voi päättää

testivaiheessa. Ohjelmallisesti sen toteuttaminen on verrattain helppoa. Peliä voi pelata joko värähtelynvaimennuksella tai ilman, jolloin huomataan värähtelynvaimennuksen tuoma etu.

Tarkoitus on valjastaa tablettitietokoneen antureita ohjauksen käyttöön. Kiihtyvyysantureita käytetään havaitsemaan muutoksia laitteen kallistuskulmassa. Kiihtyvyysanturi on sähkömekaaninen laite, joka mittaa staattisia tai dynaamisia kiihtyvyyksiä. Kiihtyvyysantureita on erityyppisiä, mutta kaikkien yhteinen toimintaperiaate perustuu siihen, että anturissa oleva komponentti liikkuu suhteessa anturin rungossa olevaan toiseen, kiinteään komponenttiin. Komponenttien välisestä suureen muutoksesta, esimerkiksi jännite tai kapasitanssi, päätellään kiihtyvyyden suuruus. [6, 21 - 7.]

Suunnitelma on toteuttaa Android-sovellus, jossa Android-laitteen kiihtyvyysanturien tiedot tallennetaan muuttujiin niiden arvojen muuttuessa. Kirjoitetaan muuttujien arvot servokäytön globaaleihin rekisteriparametreihin (tyhjiin muistipaikkoihin) Open Core -rajapinnan välityksellä. Käytetään muuttujia PLC-koodissa liikkeenohjaustoimintolohkoissa[8]. Vastaavasti PLC:ltä kirjoitetaan servokäytön tyhjiin muistipaikkoihin, jotka luetaan Android-laitteella. Toteutetaan Android-laitteeseen yksinkertainen käyttöliittymä, jolla toimilaitteita ohjataan.

On selvitettävä, mikä on järkevin tapa havaita Android-laitteen käytön lopettaminen käyttäjän toimesta, jotta saadaan automaattinen työkierto alkamaan. Yksi vaihtoehto on käyttää tähän jotain toista Android-laitteen anturia, esimerkiksi etäisyysanturia, mikäli Android-laite on varustettu sellaisella. Kun anturi ei havaitse kohdetta riittävän lyhyellä etäisyydellä tietyn ajan kuluessa, päätellään että laite on vailla käyttäjää. Kappaleessa 6.3 käsitellään Android-laitteen antureita tarkemmin.

[8] Motion function block

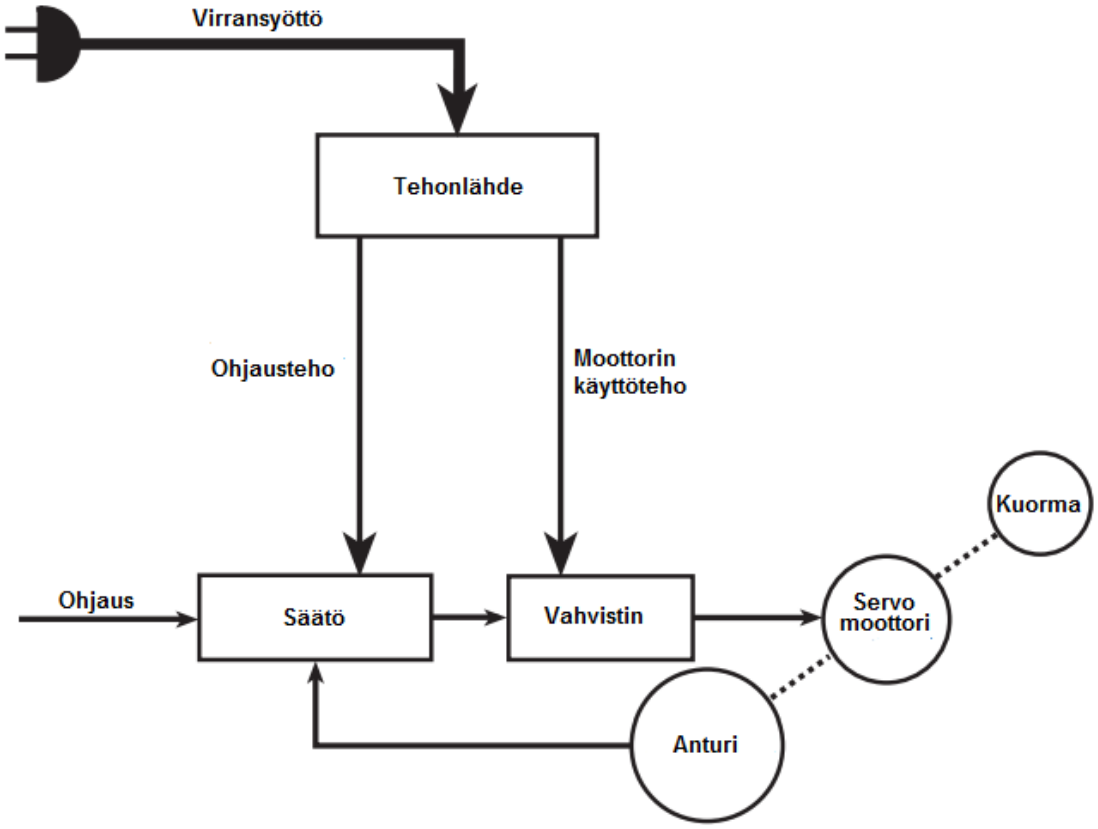
5 Ohjelmisto ja laitteisto

5.1 Servojärjestelmä

Ohjausjärjestelmien yhteydessä sana servo tarkoittaa takaisinkytkettyä säätöjärjestelmää, joka koostuu ohjattavasta laitoksesta, ohjaimesta ja takaisinkytkennästä. Lähtökohtaisesti servojärjestelmän ei tarvitse sisältää moottoria.

Toisinaan kuulee puhuttavan servomoottoreista aivan kuin termillä sinänsä tarkoitettaisiin jotain tiettyä moottorityyppiä. Servojärjestelmässä itse moottori voi kuitenkin olla erityyppinen vaihtovirta- tai tasavirtasähkömoottori. Mikä tekee tietystä moottorista servomoottorin, on moottoriin integroitu anturi, joka lähettää tietoa takaisin ohjaukselle esimerkiksi moottorin nopeudesta tai asemasta. Näin muodostuu takaisinkytketty säätöjärjestelmä ja servomekanismi (kuva 3). Moottorinohjauksessa parhaimman mahdollisen tarkkuuden saavuttamiseksi käytetään enkooderilla, eli moottorin pituusakselin ympäri tapahtuvaa pyörimistä mittaavalla anturilla, takaisinkytkettyjä servomekanismeja [7, s.199].

Sähkökäyttöinen servomekanismi tarvitsee tehonlähteen, eli servokäytön, ja ohjauksen. Servokäyttö on pohjimmiltaan vahvistin, joka vastaanottaa sähköisen komentosignaalin ohjausjärjestelmältä (esim. PLC:ltä), vahvistaa sen ja johtaa sähkövirtaa servomoottorille tuottaakseen käskysignaaliin verrannollisen liikkeen. Servomoottorin enkooderi lähettää moottorin todellisen sijainnin/pyörimisnopeuden takaisin servokäytölle. Säätöpiiri vertailee todellista sijaintia/pyörimisnopeutta annettuun ohjeeseen ja säätelee moottorille menevää jännitettä korjatakseen niiden eroa. [8, s.11.]



Kuva 3. Lohkokaavio servomekanismin periaatteesta, nuolilla kuvataan signaalin etenemissuuntaa [8, s. 11].

5.2 IndraDrive Cs HCS01.1E-W0006

IndraDrive Cs HCS01.1E-W0006 on projektissa käytettävä servokäyttö. Servokäytön joitakin käyttökohteiden kannalta olennaisia tietoja on lueteltu seuraavassa [9]:

Tehonlähde:	110…230 V, yksi- tai kolmivaihe vaihtovirta, 2.8 A tai 1.2 A, 50…60 Hz
Ulostulo:	0…230 V, kolmivaihe vaihtovirta, 2.0 A, 0…1600 Hz
Ohjaustehonlähde:	24 V tasavirta, 3.4 A maksimi

Kyseinen servokäyttö soveltuu hyvin messu- ja esittelykäyttöön, koska sitä voidaan syöttää yksivaiheisella verkkovirralla.

5.3 Moottorit ja toimilaitteet

Projektissa toimilaitteiden liikkeen aikaansaamiseksi käytetään kestomagnetoituja kolmivaihe-tahtimoottoreita. Tahtimoottoreissa moottorin kestomagnetoitu roottori pyörii tahdissa staattorivirran luoman pyörivän magneettikentän kanssa. Näin ollen roottorin pyörimisnopeus on sidottu jännitelähteen taajuuteen. Moottorin pyörimisnopeutta saadaan siis muutettua moottorille menevän jännitteen taajuutta muuttamalla. [8, s. 3, 5.]

Demolaitteessa x- ja y-suunnan liikkeissä käytetään hammashihnavetoisia lineaaritoimilaitteita, joita ajetaan vaihteen välityksellä, välityssuhteella 5. Z-suunnan liike toteutetaan sähkömekaanisella sylinterillä. Sähkömoottori käyttää sylinterin mäntää hammashihnan välityksellä välityssuhteella 1.

5.4 Kiihtyvyysanturi QG30-KAX-2,0E-AI-K

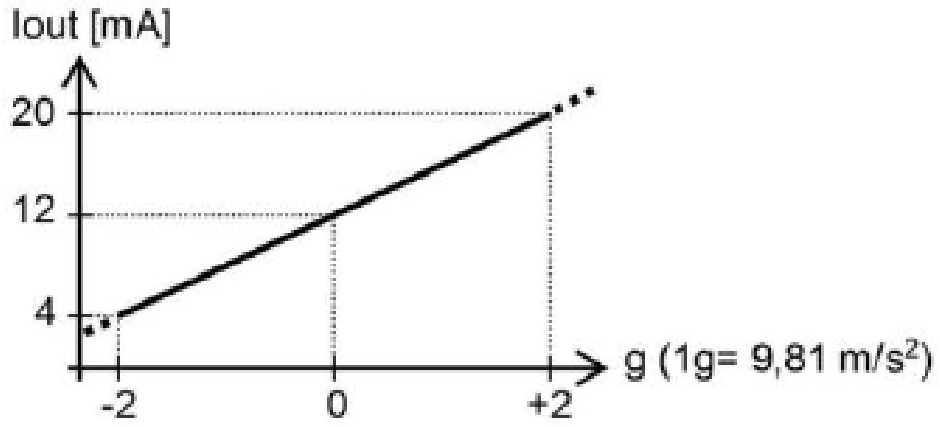
Värähtelynvaimennus toteutetaan värähtelijätangon päähän kiinnitetyn kiihtyvyysanturin lähettämän datan avulla. Taulukossa 1 ja kuvassa 4 on ilmoitettuna kiihtyvyysanturin olennaisia tietoja [10].

Taulukko 1. Kiihtyvyysanturi QG30-KAX-2,0E-AI-K. Tietoja ei ole suomennettu käännösvirheiden välttämiseksi.

QG30-KAX-2,0E-AI-K General specifications v20140313	
Supply voltage	10 - 30V dc
Current consumption	≤ 10 mA (output signal excluded)
Measuring range	± 2 g
Accuracy (2σ)	overall 0,08g typ. (offset excluded)
Offset error	± 0,07 g
Non linearity	< ± 0,05 g
Sensitivity error	< ± 3,5%
Resolution	0,004 g
Output signal	4 - 20 mA
Response time	< 0,3 ms

Kiihtyvyysanturin tarkkuus (accuracy) on tyypillisesti 0,08 poisluettuna *offset error*. Jos siis kiihtyvyysanturilta luetaan esimerkiksi arvo 2 g, voi arvo todellisuudessa olla välillä 2 g ± (0,08 g + 0,07 g), eli välillä 1,85 g–2,15 g. *2σ* tarkoittaa että kyseisen valmistajan valmistamien kiihtyvyysanturien tarkkuus on normaalijakautunut ja taulukon tarkkuus pätee kahden keskihajonnan päässä keskiarvosta oleviin antureihin, eli 95,45 %:iin kaikista

valmistetuista antureista [11]. Kuvassa 4 nähdään kiihtyvyysanturin lähtösignaali kiihtyvyyden funktiona.



Kuva 4. Kiihtyvyysanturin lähtösignaali kiihtyvyyden funktiona

5.5 IndraWorks Engineering, IndraLogic 2G

IndraWorks Engineering on Bosch Rexrothin ohjelmistokehys muun muassa PLC-ohjelmointiin ja servokäyttöjen parametrien säätöön. Tässä työssä servokäytöt ohjelmoidaan käyttäen hyväksi kyseistä ohjelmointikehystä ja siihen liitettyä IndraLogic 2G PLC-ohjelmointityökalua.

5.6 IndraMotion MLD (Motion Logic Drive-based)

IndraMotion MLD on IndraDrive Cs-servokäyttöön sulautettu PLC. Käytännössä tämä tarkoittaa servokäyttöön asennettua valmisohjelmistoa. Servokäyttöä voidaan siis käyttää itsenäisenä liikkeenohjausyksikkönä, ilman ylemmän tason ohjausyksikköä (erillistä PLC:tä). Toisaalta usean akselin järjestelmässä, kuten tämä projekti, yksi servokäyttö toimii muiden järjestelmään liitettyjen servokäyttöjen erillisenä ohjausyksikkönä. [12.]

5.7 EAL SDK – Easy Automation Library Software Development Kit

EAL SDK on Open Core -rajapinta korkean asteen ohjelmointikielen sekä IndraDrive Cs-servokäytön välillä. Kyseessä olevan työn yhteydessä tämä tarkoittaa käytännössä, että

EAL SDK sisältää valmiita kirjastoja Android Studioon (ks. seuraava kappale), jotka mahdollistavat yhteyden muodostamisen Android-laitteen ja servokäytön välille sekä tarjoavat pääsyn servokäytön eri komponentteihin (parametreihin).

5.8 Android Studio

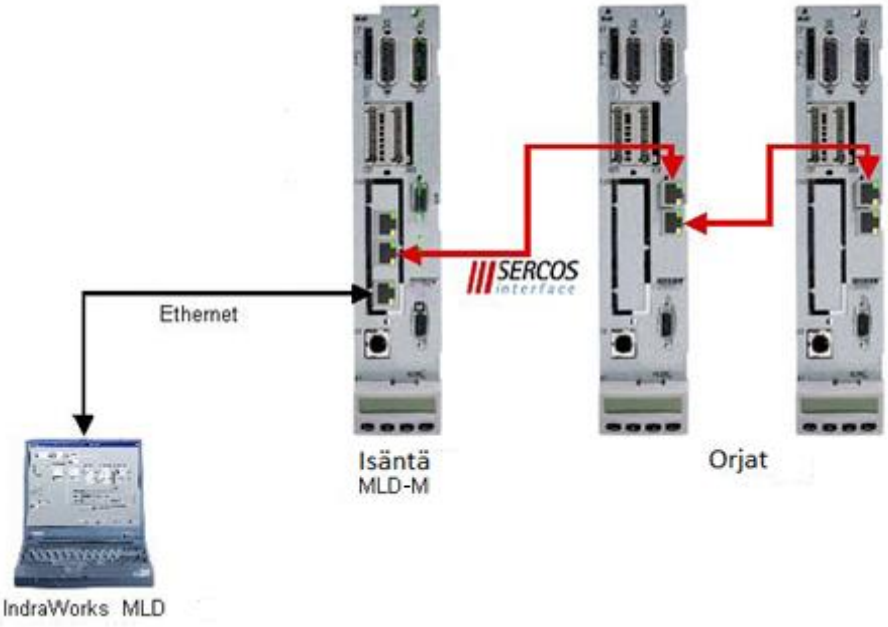
Android Studio on Java-pohjainen virallinen sovelluskehitysympäristö laitteille, joissa on Googlen Android-käyttöjärjestelmä [13]. Tämän insinöörityön tarkoitus ei ollut tehdä tekijästä Android-ohjelmoijaa, eikä tekijällä ollut aikaisempaa kokemusta Java- tai Android-ohjelmoinnista, joten työn Android-osuus suoritettiin mahdollisimman pitkälti valmiita malleja apuna käyttäen.

6 Työn suoritus

6.1 Servojen käyttöönotto

Tässä työssä ohjataan kolmea servomoottoria kolmella erillisellä servokäytöllä. Kyseisissä servokäytöissä on sulautettu PLC-yksikkö (MLD), joka ohjelmoidaan tietokoneella, tarkoitukseen sopivassa ohjelmointiympäristössä (IndraWorks Engineering).

Kuvassa 5 on esitettynä IndraMotion MLD-M -systeemin topologia. MLD-M viittaa MLD:n variaatioon, jossa isäntä-servokäyttöön on yhdistetty orja-servokäyttöjä Sercos 3 -väylän yli. Ohjelmointitietokone ja isäntänä toimiva servokäyttö kommunikoivat Ethernet-väylän yli. MLD-M -topologia sallii yhteensä kahdeksan servokäytön liittämisen toisiinsa. [14, 4.4.]



Kuva 5. IndraMotion MLD-M-systeemin topologia [14, 4.4].

IndraDrivs Cs -servokäytön käyttöönotto tapahtuu seuraavasti:

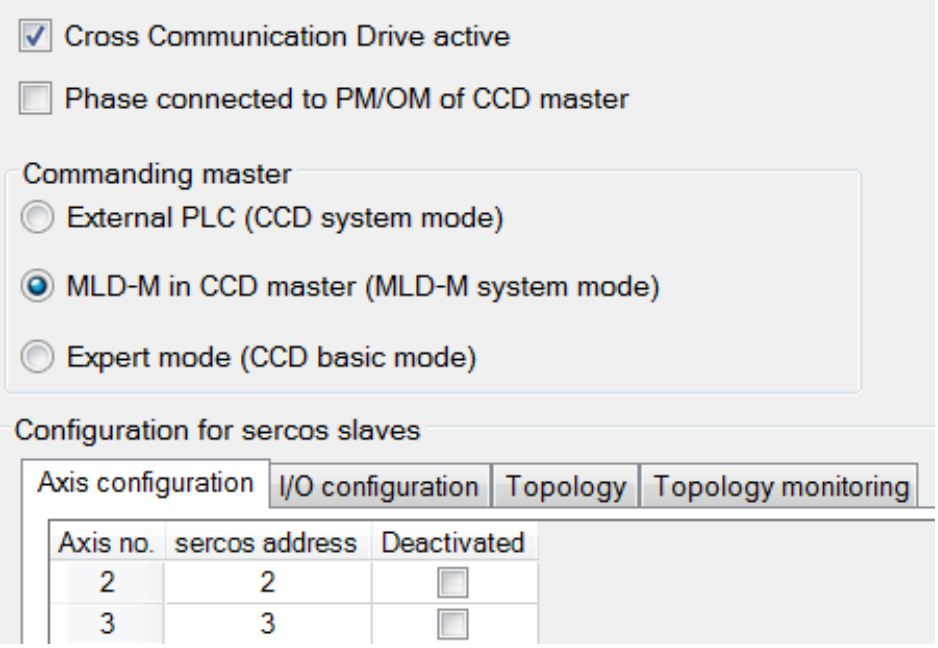
1. Yhdistetään isäntä-servokäyttö Ethernet-kaapelilla tietokoneeseen ja liitetään tietokone ja servokäyttö samaan IP-alueeseen. Servokäytössä on näytöllinen paneeli, josta servokäytölle saa määriteltyä IP-osoitteen. Yhdistetään orja-servokäytöt järjestelmään Sercos 3 -kaapelilla (ethernet-kaapeli) kuvan 5 osoittamalla tavalla.

2. Luodaan IndraWorks-engineering -ohjelmistokehyksessä uusi projekti ja yhdistetään servokäyttö kyseiseen projektiin.

3. Ladataan laiteohjelmistolta servokäytölle perusparametrit, millä varmistetaan puhdas aloitus projektille.

4. Määritellään kahdelle muulle orja-servokäytölle Sercos-osoitteet. Tämä onnistuu myös servokäyttöjen ohjauspaneelista. Osoitteet voivat olla muotoa kokonaisluku, tässä tapauksessa 2 ja 3. Ohjelmisto havaitsee orja-servokäytöt automaattisesti.

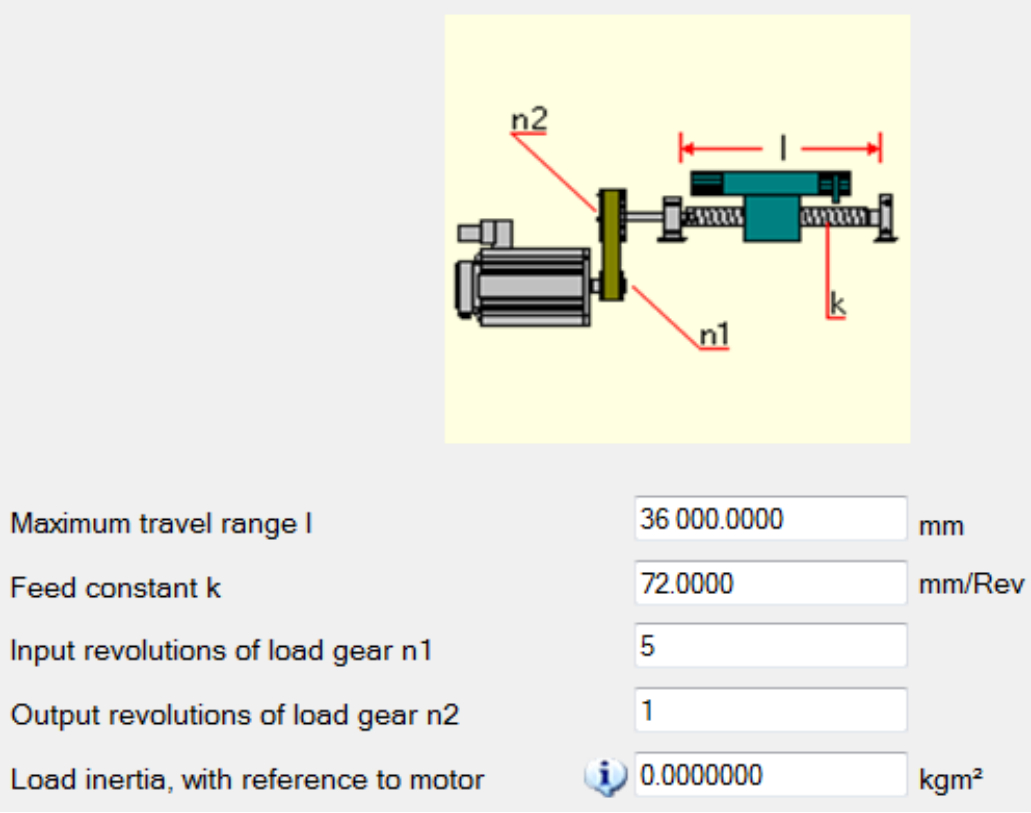
5. Määritellään *Sercos basic settings* -dialogista *Cross Communication Drive* aktiiviseksi ja valitaan *Commanding master* -otsikon alta *MLD-M in CCD master* aktiiviseksi (kuva 6). Aktivoidaan orja-servokäytöt. Nyt kaikki servokäytöt ovat yhdistetty projektiin ja ne voidaan parametrisoida.



Kuva 6. Orjien yhdistäminen isäntään Sercos 3 -väylän yli.

6. Jotta toimilaitteita voidaan ajaa tarkoilla nopeus- ja paikkaohjeilla, tulee mekaniikan skaalaus konfiguroida. Toimilaitteet ovat lineaaritoimilaitteita, joten on tarkoituksenmukaista määritellä niille lineaarinen skaalaus rotaatioskaalauksen sijaan. Lisäksi välityssuhde moottorin ja vaihteen välillä sekä syöttö (mm/kierros) tulee määritellä (kuva 7). Kyseiset arvot on ilmoitettu laitteiden tyyppikilvissä. X- ja y-suunnassa välityssuhde on 5 ja syöttö 72 mm/kierros. Z-suunnassa välityssuhde

on 1 ja syöttö 10 mm/kierros. Toimilaitteiden liikkeille esitetään laskelmat kappaleessa 6.5.1.



Kuva 7. Mekaaninen välitys moottorin, vaihteen ja toimilaitteen välillä x- ja y- suunnan toimilaitteissa.

7. IndraWorks-ohjelmisto tunnistaa automaattisesti servokäyttöihin yhdistetyt Bosch Rexrothin servomoottorit ja määrittelee akseleille säätöparametrit. Periaatteessa akseleita voi ajaa näillä parametreilla ja niin tässä työssä toimittiin. Toimilaitteiden vetotapa vaikuttaa säätöparametrien sopivuuteen kyseiseen applikaatioon. Eräs tapa määritellä säätöparametrien tyydyttävyys on kuunnella toimilaitteen ääntä sitä ajaessa [15]. PI-säätimen arvoja muuttamalla yritetään saada toimilaite mahdollisimman hiljaiseksi. Ohjelmisto tarjoaa myös mahdollisuuden automaattiseen optimointiin. Automaattisen optimoinnin ei ole kuitenkaan havaittu toimivan täysin tyydyttävällä tavalla ja käytännössä säätöparametrien arvot valitaan kokeilemalla [15]. IndraWorks-ohjelmisto tarjoaa oskilloskooppi-mittatyökalun servokäyttöjen ja ohjausjärjestelmien käyttöönottoon, huoltoon ja testaukseen [16, s. 261]. Käsitellään testausta oskilloskoopin avulla kappaleessa 6.2.

8. Akseleille tulee asettaa referenssipisteet, joiden perusteella paikoitus suoritetaan. Tämä on helpointa tehdä käyttämällä hyväksi servokäytön *easy startup* -toimintoa. Tämä kirjoittaa dataa suoraan servokäytön ohjausparametreihin ohi PLC:n [7, s. 379]. *Easy startup* -toiminnolle löytyy käyttöliittymä *IndraWorks engineering* -ohjelmistokehyksestä. Toiminnon avulla ajetaan toimilaitteet haluttuihin paikkoihin ja referoidaan akselit. Tässä tapauksessa määritellään paikat systeemin fyysisen kokoonpanon perusteella.

9. Aktivoidaan MLD, eli servokäyttöön sulautettu PLC, jolloin servokäyttöjen ohjelmointi voidaan toteuttaa PLC-ohjelmalla. PLC-ohjelmaa käsitellään kappaleessa 6.5.2.

6.2 Oskilloskooppi-mittaustyökalu

Oskilloskoopin avulla voidaan tulkita toimilaitteiden reagointia ohjauskäskyihin ja säätöparametrien sopivuutta kyseiseen applikaatioon. Luotiin ohjelma, jossa yksi akseli laitettiin liikkumaan kahden pisteen välillä, esimerkkinä x-akseli. Valittiin pisteiksi 200 - ja 100 mm:n etäisyydet referenssipisteestä ja nopeudeksi 5000 mm/min. Valittiin oskilloskoopin kuvaajaan piirrettäväksi paikkaohje (Position command value) ja takaisinkytkennältä (enkooderilta) saatu tieto paikasta (Active position feedback value). Suoritettiin mittaus yhden ms:n välein 8192 ms:n ajan. Saatiin liitteen 1 kuvan 1 mukainen kuvaaja. Kuvaajasta nähdään että arvot seuraavat hyvin tarkasti toisiaan. Suurennetaan kuvaa viimeisen huipun kohdalla (liite 1, kuva 2). Nähdään että tullessa 200 mm:n kohtaan todellinen arvo pyrkii hieman karkaamaan ohjearvosta, kuitenkin palatakseen lähelle ohjearvoa hyvin nopeasti. Kyseisessä applikaatiossa todellisen arvon ja ohjearvon erot ovat maksimissaan joitakin kymmeniä mikrometrejä. Tämän applikaation kohdalla ei ollut tarkoituksenmukaista tehdä sen tarkempaa analyysia eikä säätöarvoja tarvitse muuttaa.

6.3 Android-laitteen käyttöönotto

Android-laitteena projektissa käytettiin Samsung GT-P5210-tablettitietokonetta. Laitetta ei erikseen hankittu tätä projektia varten ja se on jo useamman vuoden vanha. Täten oli tarkoituksenmukaista selvittää, mitä antureita kyseisestä laitteesta ylipäätään löytyy. Antureiden havaitsemista varten luotiin applikaatio, joka laitteen käytön aikana tunnistaa kaikki laitteesta löytyvät anturit.

```
private SensorManager SM;
SM = (SensorManager)getSystemService(SENSOR_SERVICE);
System.out.println(SM.getSensorList(Sensor.TYPE_ALL));
```

Viimeisen rivin komento tulostaa antureiden tiedot Android-monitor työkalulla. Seuraavassa on lueteltu laitteesta tunnistetut anturit:

- K2DH Acceleration Sensor
- MS-3R (YAS532) Magnetic Sensor
- MS-x Orientation Sensor
- BH1733 Light Sensor
- ASP01 Grip Sensor
- Screen Orientation Sensor

Kyseisestä tablettitietokoneesta löytyy projektin kannalta kriittinen kiihtyvyysanturi (K2DH Acceleration Sensor). Sen sijaan, laitteesta ei löydy etäisyysanturia, jonka avulla pystyttäisiin melko suoraviivaisesti toteamaan, onko älylaitteella käyttäjää, niin kuin toimintasuunnitelmassa luvussa 4 suunniteltiin. Syy miksi kiihtyvyysanturin käyttäminen käyttäjän poistumisen havaitsemiseen on hankalaa, on anturin herkkyys. Android-laitteen ollessa paikoillaan tasaisellakin pinnalla muuttuvat kiihtyvyysanturin mittaamat arvot. *ASP01 Grip Sensor* viittaa nimensä puolesta siihen, että kyseisellä anturilla voitaisiin tunnistaa, pitääkö joku älylaitetta käsissään. Valitettavasti Android Studio ei tue kyseistä anturia, eli sille ei löydy valmista kirjastoa, joten sitä ei voida hyödyntää [17]. Toisaalta yksi vaihtoehto selvittää anturin avulla, milloin Android-laite on vailla käyttäjää, voisi olla käyttää listalla toisena olevaa *Magnetic sensor* -anturia. Voidaan päätellä, että kyseessä on magneettikenttiä havaitseva anturi. Alustalle, jonne laite lasketaan käytön jälkeen, voisi sijoittaa magneetin, jolloin havaittaisiin että laite ei ole enää käyttäjän käsissä. Häiriötä voi tosin aiheuttaa anturin vaikutusalueelle mahdollisesti pääsevät muut magneettiset kappaleet.

Kyseisen anturin toimintaa kokeiltaessa havaittiin kuitenkin toinen perustavanlaatuinen ongelma. Lähdettäessä toteuttamaan sovellusta Android-laitteelle ilman suoraa koulutusta tai kokemusta kyseisten laitteiden ohjelmoinnista, voi tulla vastaan tilanteita, joissa käyttäjän tekemä ohjelma ja kyseisessä älylaitteessa vaikuttavat taustaohjelmat ovat ristiriidassa keskenään. Tässä tapauksessa esimerkiksi älylaitteen magneettianturille on jo

annettu tehtävä. Älylaite on suunniteltu käytettäväksi suojakotelon kanssa. Kun suojakotelo suljetaan, havaitsee anturi suojakotelossa olevan magneetin, jolloin se aktivoi älylaitteen näytönsäästäjän. Tämän ominaisuuden deaktivoiminen ei onnistu laitteen perusasetuksista. Vertailun vuoksi selvitettiin myös, mitä antureita löytyy hieman uudemmasta LG-merkkisestä älypuhelimesta. Kyseisestä laitteesta löytyy etäisyysanturi.

Antureiden tunnistusohjelma antaa antureista myös erinäistä tietoa. Taulukossa 2 on listattu kiihtyvyysanturin tiedot.

Taulukko 2. Kiihtyvyysanturin tiedot.

Sensor name	K2DH Acceleration Sensor
vendor	STMicroelectronics
version	45800
type	1
maxRange	19.6133
resolution	0.009576807
power	0.011
minDelay	10000

Olennaisinta kiihtyvyysanturin tiedoissa ovat viimeiset neljä kohtaa:

1. ”maxRange” – Kiihtyvyysanturin mittaaman kiihtyvyyden maksimiarvo, joka on ± 19,6133 m/s^2. Kiihtyvyyden muuttaminen nopeusohjeeksi ohjelmalle voidaan toteuttaa kertoimien avulla, joten todetaan kyseinen arvo riittäväksi.

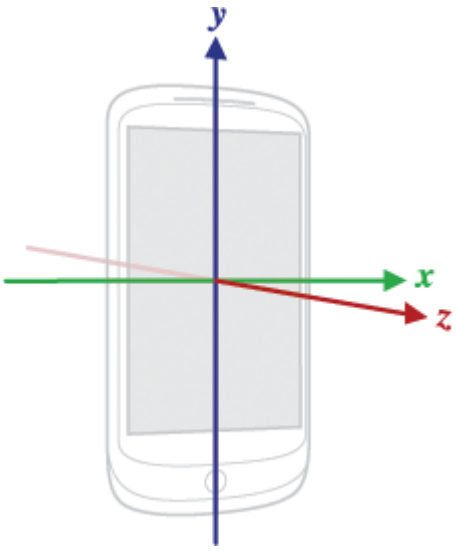
2. ”resolution” – Kiihtyvyysanturin tiedon erotuskyky.

3. ”power” – Kiihtyvyysanturin virrankulutus, jonka yksikön puuttumisen takia arvioidaan olevan 0,011 mA.

4. ”minDelay” – Kyseisestä kiihtyvyysanturista ei löytynyt tarkempaa tietoa, joten tämänkin kohdan merkitys täytyi päätellä. Todennäköisesti sillä tarkoitetaan vähimmäisaikaa, joka yksittäisten mittausten välillä on oltava. Empiiristen kokeiden ja realistisuuden perusteella päätellään ajan yksiköksi mikrosekunti (µs).

   10000 µs = 10 ms = 0,01 s

Käytettäessä Android-laitteen kiihtyvyysantureita ohjelmassa tulee huomioida koordinaatisto, jota Android Studion anturi-sovellusliittymä käyttää (kuva 8) [17].



Kuva 8. Koordinaatisto, suhteessa Android-laitteeseen, jota anturi-sovellusliittymä käyttää [17]. Ohjelmaa suunniteltaessa on tärkeää huomioida käytetyn älylaitteen luonnollinen orientaatio. Se voi olla pystyformaatti (ks. kuva) tai vaakaformaatti. Laitetta käänneltäessä koordinaatisto pysyy samana suhteessa luonnolliseen orientaatioon.

6.4 Android-ohjelmointi

Open Core -rajapinnan käyttöönotto Android Studiossa noudattaa seuraavaa järjestystä:

1. Tietokoneelle tulee olla asennettuna Java 64 Bit RTE[9] sekä Android Studio. Java RTE mahdollistaa Java-ohjelmien ajamisen tietokoneella [18, s. 23].

2. Android-laitteesta tulee aktivoida sovelluskehittäjän asetukset, jotka on lähtökohtaisesti piilotettu, ja sen kautta sallia USB-määrityksistä medianlähetysprotokolla[10].

3. Käytettävälle Android-laitteelle tulee olla asennettuna service.apk, EAL:ään kuuluva applikaatio, joka mahdollistaa Open Core -rajapinnan käytön Android-laitteella [19].

[9] Runtime environment

[10] Media Transfer Protocol

4. Ladataan Bosch Rexrothin esimerkkiohjelma Android Studioon.

5. Lisätään Android-projektiin kaksi Bosch Rexrothin EAL-kirjastoa.

6. Ladataan ohjelma Android-laitteelle.

7. Liitetään WLAN-reititin servokäyttöön ja yhdistetään Android-laite reitittimen välityksellä servokäyttöön.

8. Suoritetaan ohjelma ja kokeillaan, saadaanko yhteys Android-laitteen ja servokäytön välille muodostettua.

Esimerkki-Android-applikaatiossa on liikkeen toteuttavia metodeja[11], jotka kirjoittavat dataa suoraan servokäytön ohjausparametreihin, eikä servokäyttöön sulautettu PLC ole käytössä. Muutetaan ohjelmaa niin, että ei käytetä valmiita liikemetodeja, vaan kirjoitetaan Android-laitteelta PLC:n globaaleihin rekisteriparametreihin arvoja. Näihin päästään käsiksi PLC-koodissa funktioiden avulla. Näin toimimalla systeemin ohjaus saadaan toteutettua pääasiassa PLC-ohjelmalla ja työn Android-osuus jää minimiin. Kuva 9 havainnollistaa edellä mainittua prosessia.

Kuva 9. Ohjelmoinnin topologia [3; 13]. Älylaitteelle ohjelmoidaan applikaatio Android Studiolla, PLC:hen ohjelmoidaan ohjelma IndraLogic-ohjelmointityökalulla. Tämän jälkeen servokäytön ja siihen sulautetun PLC:n ohjaamiseen tarvitaan ainoastaan kyseistä älylaitetta.

Käytettiin esimerkkiohjelmaa projektin Android-applikaation raakamallina. Ohjelmasta saatiin valmiina ohjelmaosuus, jossa Open Core -rajapinnan avulla luodaan WLAN-yhteys Android-laitteen ja servokäytön välille. Kyseinen ohjelma sisältää myös graafisen käyttöliittymän, jonka pohjalta muokattiin tämän projektin käyttöliittymä. Tehtäväksi jäi

[11] Olio-ohjelmointikielissä funktioita kutsutaan metodeiksi.

selvittää, miten Android-laitteen kiihtyvyysantureihin pääsee ohjelman avulla käsiksi, miten servokäytön parametreihin kirjoitetaan tietoa Android-applikaatiolla, miten Android-applikaatiolla luetaan tietoa servokäytön parametreista sekä miten graafista käyttöliittymää muokataan. Ottaen huomioon tekijän rajallisen tietämyksen Java- ja Android-ohjelmoinnista sekä koulutusohjelman, käydään Android-applikaatiosta läpi sen olennaisimmat periaatteet syventymättä sen tarkemmin tietotekniikkaan sen takana. Kuvassa 10 nähdään toteutettu Android-laitteen käyttöliittymä.

Kuva 10. Android-laitteen käyttöliittymä.

Connect-painikkeesta avautuu dialogi (kuva 11), johon kirjoitetaan Android-laitteeseen yhdistettävän laitteen IP-osoite.

Kuva 11. Sisäänkirjautumisdialogi.

Ip Address ilmoittaa yhdistetyn laitteen IP-osoitteen. *Disconnect*-painikkeesta katkaistaan yhteys. *Enable*-painikkeesta aktivoidaan servokäytöt. *Disable*-painikkeesta deaktivoidaan servokäytöt. *Start*-painike käynnistää pelin. *Throw*-painikkeesta tikka ”heitetään”.

Seuraavassa selitetään ohjelman toimintaperiaate kuvan 11 avulla.

1. Käyttäjän ei ole tarkoitus pelin aikana koskea muihin kuin *Start*- ja *Throw*-painikkeisiin.

2. Yhdistämisen jälkeen *Start*-painike on aktiivinen ja *Throw*-painike on inaktiivinen.

3. *Start*-painiketta painaessa, deaktivoituu kyseinen painike ja *Throw*-painike aktivoituu. Toimilaitteet alkavat reagoida Android-laitteen kallistuksiin ja peliaika käynnistyy. Mikäli ”heittoa” ei suoriteta (*Throw*-painiketta paineta) ennalta määrätyssä ajassa, suoritetaan heitto automaattisesti, jolloin *Throw*-painike deaktivoituu. Mikäli käyttäjä painaa *Throw*-painiketta aikaikkunan sisällä, suorittaa kone vastaavat toimenpiteet kuin automaattisesti heittäessä. X- ja y-suunnan toimilaitteet pysähtyvät ja z-suunnan toimilaite suorittaa iskun, eli ”heiton”. ”Heiton” jäl-

keen toimilaitteet palaavat kotiasemaan, jossa tikan kärki on tikkataulun etupinnan rajaaman tason ulkopuolella. *Start*-painike aktivoituu tultaessa kotiasemaan, ja tilanne on sama kuin kohdassa 2.

4. Mikäli *Start*-painiketta ei paineta riittävän pitkään ennalta määrättyyn aikaan, alkaa kone suorittamaan automaattista kiertoa. Automaattisen kierron aikana kone ”heittää” tikan kaikkiin tikkataulun lukuihin pienimmästä suurimpaan ja lopuksi häränsilmään. Tämän jälkeen kierto alkaa alusta. Automaattisen kierron aikana *Start*-painike on aktiivinen. Mikäli painiketta painetaan, päättyy automaattinen kierto välittömästi, *Start*-painike deaktivoituu ja toimilaitteet liikkuvat kotiasemaan. Kotiasemaan tultaessa *Throw*-painike aktivoituu, toimilaitteet alkavat reagoida Android-laitteen kallistuksiin ja peliaika alkaa.

Käyttöliittymästä toteutettiin mahdollisimman käyttäjäystävällinen. Toisin sanoen pidetään vain ne painikkeet aktiivisena, joiden painamisen halutaan aiheuttavan tapahtuman ohjelmassa. Tähän tarkoitukseen käytetään Android-studion *onClick*-metodia, joka tunnistaa, milloin painiketta on painettu kerran. Esimerkiksi kun *Start*-painiketta painetaan, käynnistetään kiihtyvyysanturin lukeminen. Deaktivoidaan *Start*-painike ja aktivoidaan *Throw*-painike. ”Ilmoitetaan” PLC:lle, että käyttäjä ohjaa laitetta, kirjoittamalla tieto PLC:n globaaliin rekisteriparametriin *P-0-1274*.

```
public void onClick(View v) {

 if (v.equals(btnStart)) {

     if(iThrowDart == 0) {

         SM.registerListener(this, accSensor, READINGRATE);

         try {

           btnStart.setEnabled(false);
           btnThrow.setEnabled(true);

           iUserControl = 1;

           String stringUserControl = Integer.toString(iUserControl);
           Parameter parameter = connection.getAxes(0).parameter();
           parameter.writeDataAsString("P-0-1274", stringUserControl);

          } catch (EalException e) {
              e.printStackTrace();
              showErrorDialog(e.getMessage());
          } catch (RemoteException e) {
              e.printStackTrace();
              showErrorDialog(e.getMessage());
          } catch (Exception e) {
```

```
                e.printStackTrace();
                showErrorDialog(e.getMessage());
            }
        }
    }
```

ConnectToDrive on metodi yhteyden muodostamiseen Android-laitteen ja servokäytön välillä.

```
public void ConnectToDrive() {

    try {
        connectedToService = true;
        connection = new EalConnection(getBaseContext(), iEalMethods);
        connection.connect(ipAddress);
        connectedToDrive = true;
        btnConnect.setEnabled(false);
        btnDisconnect.setEnabled(true);
        btnEnable.setEnabled(true);
        lblStatus.setText("Connected To " + ipAddress);
    } catch (EalException e) {
        e.printStackTrace();
        showErrorDialog(e.getMessage());
        ConnectToDrive();
    } catch (RemoteException e) {
        e.printStackTrace();
        showErrorDialog(e.getMessage());
    }
}
```

onSensorChanged on metodi, jota kutsutaan kun havaitaan uusi anturi tapahtuma.

```
public void onSensorChanged(SensorEvent event) {

    yAcc = event.values[0];
    xAcc = event.values[1];

    sendAccelerometerValuesToDrive();
}
```

Metodia kutsutaan myös, vaikkei anturin mittaama arvo poikkeaisi edellisestä, kunhan sillä on tuoreempi aikaleima [17]. Alustaessa anturia ohjelmassa, tulee sille määritellä mittaustaajuus.

```
SM.registerListener(this, accSensor, READINGRATE);
```

Riippumatta siitä minkä arvon mittaustaajuudelle antaa, saattavat Android-järjestelmä ja muut samaan aikaan laitteessa käytettävät applikaatiot vaikuttaa arvoon. Tästä syystä ja laitteen muistin ylikuormittamisen välttämiseksi sekä virran kulutuksen minimoimiseksi, mittaustaajuus kannattaa määritellä mahdollisimman pieneksi mutta kuitenkin

applikaation kannalta riittäväksi. Sopiva arvo mittaustaajuudelle saatiin kokeilemalla ohjelmaa eri mittaustaajuuden arvoilla. Android-laitetta kallistaessa päätettiin, millä mittaustaajuuden arvolla toimilaite vastaa kallistuksiin riittävän nopeasti. Valittiin pienin mahdollinen arvo, joka silti tuottaa tyydyttävän vasteen toimilaitteiden liikkeisiin. Tässä tapauksessa se oli 500 000 µs (0,5 s). Koska *onSensorChanged*-metodia kutsutaan suurella taajuudella, on suositeltavaa pitää metodin sisällä suoritettavat muut toiminnot mahdollisimman vähäisinä, metodin tukkimisen välttämiseksi. Metodin sisällä x- ja y-suunnan kiihtyvyysantureiden tiedot tallennetaan muuttujiin ja kutsutaan metodia *sendAccelerometerValuesToDrive*, joka tallentaa tiedot servokäytön parametreihin.

```
public void sendAccelerometerValuesToDrive() {
    try {

        if (!connection.isConnected()) {
            showErrorDialog("Yhteys pudonnut!!!!!");
            ConnectToDrive();
        }
        else {
            String xAccAsString = Float.toString(xAcc);
            String yAccAsString = Float.toString(yAcc);
            Parameter parameter = connection.getAxes(0).parameter();
            parameter.writeDataAsString("P-0-1270", xAccAsString);
            parameter.writeDataAsString("P-0-1271", yAccAsString);
        }

    } catch (EalException e) {
        e.printStackTrace();
        showErrorDialog(e.getMessage());
        ConnectToDrive();
    } catch (RemoteException e) {
        e.printStackTrace();
        showErrorDialog(e.getMessage());
    } catch (Exception e) {
        e.printStackTrace();
        showErrorDialog(e.getMessage());
    }
}
```

Aluksi metodissa tarkastetaan, onko yhteys servokäytölle olemassa. Jos ei, kutsutaan *ConnectToDrive*-metodia, jossa yritetään muodostaa yhteys uudestaan. Kun yhteys on olemassa, muutetaan liukulukuina olevat kiihtyvyysanturin tiedot merkkijonoiksi, koska *writeDataAsString*-metodi vaatii niiden olevan merkkijonoja, ja kirjoitetaan tiedot servokäytön globaaleihin rekisteriparametreihin.

Kyseisessä projektissa PLC:n ja Android-laitteen tulee vaihtaa tietoja keskenään. On suoraviivaista toteuttaa PLC:llä ohjelma, joka jatkuvasti suorittaa tiettyä toimintoa, tai suorittaa toiminnon saatuaan herätteen esimerkiksi liiketoimintolohkon jostakin haarasta tai servoakselin todellisen sijainnin perusteella. Perinteisillä tietokoneohjelmointikielillä

vastaava ei ole yhtä suoraviivaista. Android-laitteen täytyy saada PLC:ltä tieto milloin jokin tietty toimilaitteiden fyysinen kierto on suoritettu, että voidaan aktivoida/deaktivoida painikkeita Android-laitteen käyttöliittymästä. PLC:ltä ”ilmoitetaan” Android-laitteelle kyseisestä tapahtumasta kirjoittamalla tieto servokäytön globaaliin rekisteriparametriin. Android-applikaation on kuitenkin saatava heräte tiedon lukemiseen. Tämä päätettiin toteuttaa ajastimella, joka lukee parametrien arvoja tasaisin väliajoin.

```
timer.schedule(paramReadTask, 0l, 250);
```

Määritellään ajastimelle kutsuntasykliksi 250 ms.

```
TimerTask paramReadTask = new TimerTask() {
    public void run()
    {
        MainActivity.this.runOnUiThread(new Runnable()
        {
            public void run()
            {
                if(iTimerStart == 1){
                    ParameterReadWrite();
                }
            }
        });
    }
};
```

Ajastinta kutsutaan 250 ms:n välein. Toisin sanoen *ParameterReadWrite*-metodi suoritetaan tällä syklillä. Kyseisessä metodissa luetaan PLC:ltä globaalien rekisteriparametrien arvoja ja sen perusteella aktivoidaan painikkeita Android-laitteen käyttöliittymästä.

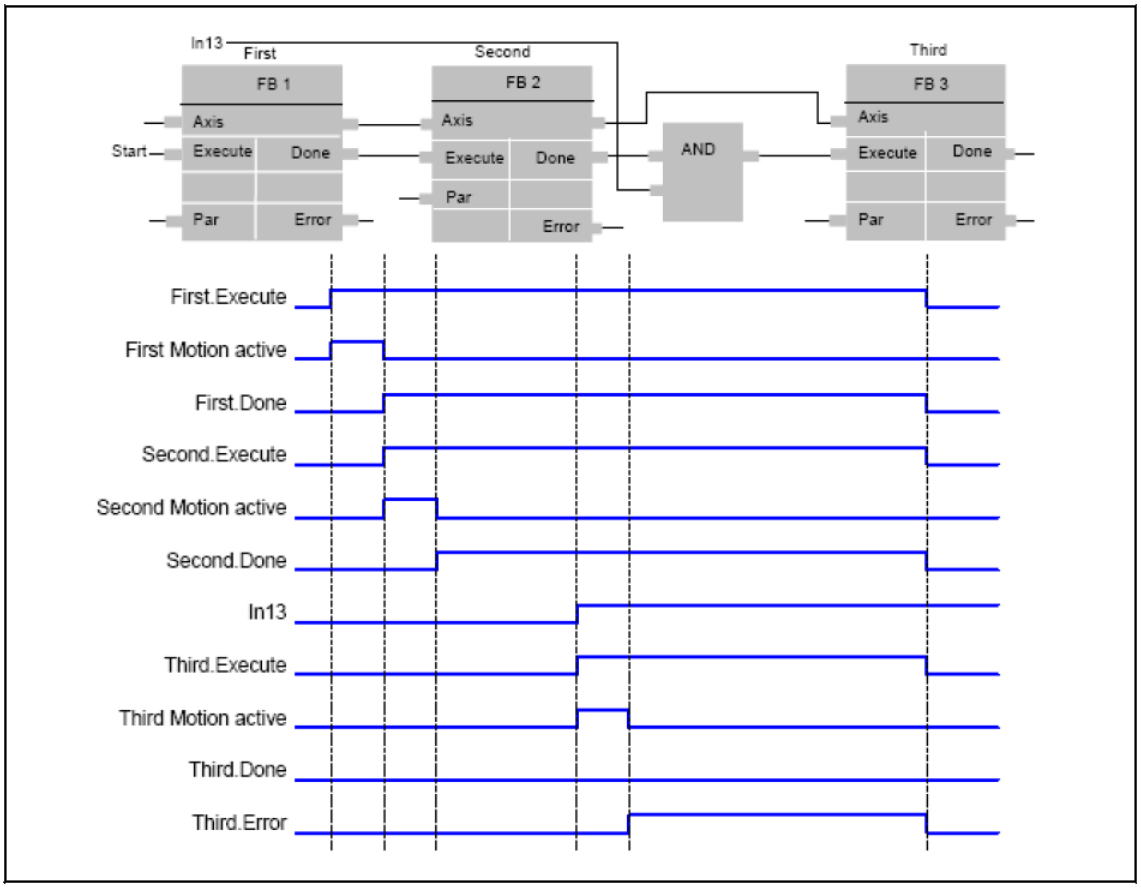
```
public void ParameterReadWrite() {

     try {

      Parameter parameter = connection.getAxes(0).parameter();
      String stringParam1276 = parameter.readDataAsString("P-0-1276");
      String stringParam1277 = parameter.readDataAsString("P-0-1277");

      iParam1276 = Integer.parseInt(stringParam1276);

      iParam1277 = Integer.parseInt(stringParam1277);

      if(iParam1277 == 1){

         btnThrow.setEnabled(false);

        }
      else if(iParam1277 == 0){

         btnThrow.setEnabled(true);
```

```
            // for toggling the throw button
            String stringTwo = Integer.toString(2);
            parameter.writeDataAsString("P-0-1277", stringTwo);
               }

         if(iParam1276 == 2) {

            iThrowDart = 0;
            btnRotate.setEnabled(true);

            String stringThrowDart = Integer.toString(iThrowDart);
            parameter.writeDataAsString("P-0-1275", stringThrowDart);
               }

         else if (iParam1276 == 1){

            iThrowDart = 1;
            btnThrow.setEnabled(false);
            btnRotate.setEnabled(false);

            iUserControl = 0;

            String stringUserControl = Integer.toString(iUserControl);
            parameter.writeDataAsString("P-0-1274", stringUserControl);

            String stringThrowDart = Integer.toString(iThrowDart);
            parameter.writeDataAsString("P-0-1275", stringThrowDart);
               }

           } catch (EalException e) {
               e.printStackTrace();
               showErrorDialog(e.getMessage());
           } catch (RemoteException e) {
               e.printStackTrace();
               showErrorDialog(e.getMessage());
           } catch (Exception e) {
               e.printStackTrace();
               showErrorDialog(e.getMessage());
           }
      }
}
```

6.5 PLC-ohjelmointi

Sallimalla PLC:n ohjata servokäyttöjä, voidaan toimilaitteiden liikkeenohjaus toteuttaa käyttäen hyväksi valmiita liikkeenohjauskirjastossa olevia toimintolohkoja. Toimintolohkojen avulla aktivoidaan servokäytöt, annetaan moottoreille nopeus-, kiihtyvyys ja paikkaohjeet sekä pysäytetään ne.

Kuvassa 12 esitetään PLCopen[12] yhdenmukaisten toimintolohkojen perustoimintaperiaatetta. *Done*-komento on seuraavan lohkon *Execute*-komento. Signaali-aika-diagrammi (kuva 12) havainnollistaa toimintolohkojen sisään- ja ulostulojen toimintaa.



Kuva 12. Toimintolohkojen signaali-aika-diagrammi [12, s. 52].

[12] Organisaatio joka tarjoaa standardoituja liikkeenohjauskirjastoja [20].

6.5.1 Toimintolohkojen liikeohjeet

Messudemon suunnitteluvaiheessa laskettiin toimilaitteille kiihtyvyydet (liite 2, kuva 1, kuva 2). Kiihtyvyys on otettava huomioon toimilaitteille aiheutuvien rasitusten takia sekä värähtelijän saattamiseksi värähtelemään.

Toimilaitteiden nopeudet valittiin kokeilemalla niiden ajamista eri nopeuksilla ja päätettiin pelin toimivuuden kannalta optimaalinen nopeus, ottaen huomioon toimilaitteiden suurimmat sallitut nopeudet. Määritellään suurin sallittu pyörimisnopeus n_{mek} x- ja y-suunnan CKR-070-lineaaritoimilaitteille kaavan 1 mukaan [21].

$$n_{mek} = \frac{v_{max} * 1000 * 60}{\pi * d_3} = \frac{3\,^{m}/_{s} * 1000 * 60}{\pi * 22{,}92\ mm} = 2499{,}816 \ldots\ min^{-1} \quad (1)$$

jossa v_{max} on lineaaritoimilaitteen suurin sallittu lineaarinen nopeus ja d_3 on vetävän hihnapyörän halkaisija. Välityssuhteen ollessa i = 5 saadaan moottorin maksimi pyörimisnopeudeksi n_{mek}:n suhteen kaavan 2 mukaan:

$$i = \frac{n_1}{n_2} => n_1 = 2499{,}816\ min^{-1} * 5 = 12499{,}08 \ldots\ min^{-1} \quad (2)$$

jossa n_1 on moottorin maksimi pyörimisnopeus ja n_2 on lineaaritoimilaitteen suurin sallittu pyörimisnopeus. Toisaalta tiedetään, että moottorin maksimi pyörimisnopeus on 9 000 min^{-1}, joten lineaaritoimilaitteen puolesta moottorin pyörimisnopeutta ei tarvitse rajoittaa. Vaihteen suurin sallittu pyörimisnopeus on n_{vaihde} = 8 000 min^{-1} [22] joten käytettäessä moottoria täysillä kierroksilla, kaavaa 2 soveltaen, vaihde pyörii nopeudella:

$$i = \frac{n_1}{n_2} => n_2 = \frac{n_1}{5} = \frac{9000\ min^{-1}}{5} = 1800\ min^{-1}$$

Tästä saadaan lineaaritoimilaitteelle suurin mahdollinen lineaarinopeus soveltaen kaavaa 1:

$$n_{mek} = \frac{v_{max} * 1000 * 60}{\pi * d_3} => v_{max} = \frac{1800\ min^{-1} * \pi * 22{,}92\ mm}{1000 * 60} = 2{,}16\ {}^{m}/_{s}$$

Z-suunnan toimilaite on EMC-032-sähkömekaaninen sylinteri. Laitteen suurin sallittu lineaarinopeus $v_{max} = 1{,}13\ {}^{m}/_{s}$ sekä syöttö $u = 10\,{}^{mm}/_{kierros}$ selviää laitteen tyyppikilvestä. Sylinteriä pyörittävän moottorin maksimi pyörimisnopeus on $n_{max} = 5\,000\ \mathrm{min}^{-1}$ [23]. Lasketaan kyseisellä pyörimisnopeudella aikaansaatava lineaarinopeus (kaava 3):

$$u * n_{max} = v_{max} => v_{max} = 10\,{}^{mm}/_{kierros} * 5000\ min^{-1} \quad (3)$$

$$= 50000\ {}^{mm}/_{min} = 0{,}83\,{}^{m}/_{s}$$

6.5.2 PLC-ohjelma

Luodaan pääohjelma ja sen rinnalle erillinen aliohjelma. Aliohjelmaan sijoitetaan toimintolohkot. Pääohjelmasta kutsutaan aliohjelmaa ja muutetaan toimintolohkoille tulevien muuttujien arvoja. Määritellään PLC-ohjelman suoritukselle 1 ms sykli, mikä on nopein mahdollinen [12, s. 25]. Ohjelmaa suoritetaan jatkuvasti PLC:n ollessa käynnistetyssä tilassa (*RUN*-tila).

Luodaan toimintolohkot aktivoimaan servokäytöt.

```
fbPower1(
          Enable :=  bAndroidEnableCommand,
          Status => ,
          Error => ,
          ErrorID => ,
          ErrorIdent => ,
          Axis := Axis1);
```

Android-laitteelta on kirjoitettu arvo servokäytön globaaliin rekisteriparametriin, joka sijoitetaan muuttujaan *bAndroidEnableCommand* PLC-ohjelmassa. Käytetään kyseistä muuttujaa servokäyttöjen aktivointi-toimintolohkojen herätteenä. Tehdään jokaiselle akselille vastaava toimintolohko. *Axis*-haarassa määritellään, mikä akseli on kyseessä. On hyvän ohjelmointitavan mukaista nimetä muuttujat selkeästi niiden toimintaa kuvaavalla tavalla sekä ilmoittaa muuttujatyyppi pienillä kirjaimilla ennen ”varsinaista” nimeä.

Toteutetaan x- ja y-suunnan akseleiden liikkeet Android-laitetta kallistaessa *MC_MoveVelocity*-toimintolohkolla.

```
fbMoveVelocity1(
          Execute := bMoveVelo,
          Velocity := rVelo1,
          Acceleration := rAcc1,
          Deceleration: = rAcc1,
          InVelocity => ,
          Active => ,
          CommandAborted => ,
          Erro r=> ,
          ErrorID => ,
          ErrorIdent => ,
          Axis := Axis1);
```

Toimintolohkolle annetaan nopeus-, kiihtyvyys- ja hidastuvuusarvot. Lohkon toiminta käynnistyy *Execute*-haaraan tulevan muuttujan nousevalla reunalla. Näin ollen mikäli samaa lohkoa halutaan käyttää toistamiseen, tulee kyseisen muuttujan arvo käyttää nollassa, jotta saadaan aikaan uusi nouseva reuna. *InVelocity*- ja *Active*-haarat kertovat akselin olevan liikkeessä. Loput haarat ovat käytännössä apuvälineitä häiriöiden ja virheiden kartoittamiseen. Tallennetaan *double integer* -muodossa oleva tieto servokäytön globaalista rekisteriparametrista (P_0_127x), jonne Android-laitteen kiihtyvyysanturin tieto on tallennettu, *real*-muodossa olevaan *rVelo1* -muuttujaan ja kerrotaan se sopivalla kertoimella, ottaen huomioon kappaleessa 6.5.1 lasketut suurimmat mahdolliset nopeudet. Näin saadaan *MC_MoveVelocity*-toimintolohkolle nopeusohje.

```
rVelo1 := DINT_TO_REAL(DV_Axis[1].P_0_1271*2000);
```

Koska nopeusohje toimintolohkolle tulee Android-laitteen kiihtyvyysanturilta, muuttuu sen arvo käytännössä jatkuvasti (ks. kappale 6.3). Näin ollen tulee myös *MC_MoveVelocity*-toimintolohkon *Execute*-haaralle saada jatkuvasti uusi nouseva reuna. Tämä toteutetaan pääohjelmassa seuraavasti.

```
IF bToggleBmoveVelo = FALSE THEN
          bMoveVelo := TRUE;
```

```
ELSE
          bMoveVelo := FALSE;
END_IF

bToggleBmoveVelo := NOT bToggleBmoveVelo;
```

Ohjelmaosuus saa aikaan *bMoveVelo*-muuttujan tilan vaihtumisen jokaisella ohjelmakierroksella.

Kun käyttäjä painaa painiketta Android-laitteelta suorittaakseen ”heiton”, Android-applikaatiossa deaktivoidaan kiihtyvyysanturit, jolloin toimilaitteet eivät enää vastaa Android-laitteen kallistuksiin, sekä kirjoitetaan arvo servokäytön globaaliin rekisteriparametriin, joka luetaan PLC-ohjelmalla, ja voidaan suorittaa heittoliike ja sitä seuraava paluuliike. Määritellään ehtoja ohjelman suoritukselle muuttujien avulla ja Android-laitteelta saadun tiedon perusteella. Toimilaitteiden käyttäjästä riippumattomat liikkeet suoritetaan *CASE*-tilakoneen avulla, esimerkkinä seuraava tikan ”heitto”. Kohdassa 30 on esimerkki ajastimen käytöstä PLC:llä. Tikka viipyy taulussa 3 sekuntia.

```
IF bAndroidThrowDart = TRUE OR bPeliAikaOhi = TRUE THEN
     CASE nState OF

          0:
                    bMoveAbsolute1      := FALSE;
                    bMoveAbsolute2      := FALSE;
                    bMoveAbsolute3      := FALSE;
                    bMoveVelo           := FALSE;
                    rVelo1              := 0;
                    rVelo2              := 0;
                    nState              := 10;
          10:
                    bMoveVelo           := TRUE;
                    nState              := 20;
          20:
                    bMoveVelo           := FALSE;
                    rPos3               := 119;
                    bMoveAbsolute3      := TRUE;
```

```
                DV_Axis[1].P_0_1276 := 1;
                nState              := 30;

        30:
        tonTauluViipyma(IN:=bMove3Done, PT:= T#3S, Q=> bTaulu-
        ViipymaOhi);

                IF bTauluViipymaOhi = TRUE THEN
                        bMoveAbsolute3      := FALSE;
                        rPos3               := 1;
                        bTauluViipymaOhi    := FALSE;
                        nState              := 40;
                END_IF


        40:
                bMoveAbsolute3      := TRUE;
                nState              := 30;
        50:
                IF bMove3Done = TRUE THEN
                        bMoveAbsolute3      := FALSE;
                        rPos1               := 500;
                        rPos2               := 30;
                        rVelo1              := 2000;
                        rVelo2              := 2000;
                        bMoveAbsolute1      := TRUE;
                        bMoveAbsolute2      := TRUE;
                        nState              := 50;
                END_IF
        60:
                IF bMove1Done = TRUE AND bMove2Done = TRUE
                THEN
                        bMoveAbsolute1      := FALSE;
                        bMoveAbsolute2      := FALSE;
                        DV_Axis[1].P_0_1276 := 2;
                END_IF
END_CASE
```

```
ELSIF bAndroidThrowDart = FALSE OR bPeliAikaOhi = FALSE THEN
          nState              := 0;
          DV_Axis[1].P_0_1276 := 0;
END_IF
```

Kierto suoritetaan *MC_MoveAbsolute*-toimintolohkolla, jonka toiminta eroaa *MC_MoveVelocity-toiminto*lohkon toiminnasta siltä osin, että *MC_MoveAbsolute*-toimintolohkolle annetaan paikkaohje nopeus- ja kiihtyvyysohjeen lisäksi.

Android-laitteelle tulee ilmoittaa, milloin automaattiset kierrot ovat valmiita, kirjoittamalla PLC:ltä arvo globaaliin rekisteriparametriin ja lukemalla se Android-laitteella, jotta käyttäjä voi suorittaa seuraavan suorituksen.

```
DV_Axis[1].P_0_1276 := 1;
```

Laitteen automaattinen työkierto, laitteen ollessa vailla käyttäjää toteutetaan kartoittamalla tikkataulu käyttäen kappaleessa 6.1 käsiteltyä *easy startup* -toimintoa. Näin saadaan selville jokaisen kohteen sijainti.

7 Päätelmät

Tämän insinöörityön tavoite oli suorittaa käyttöönotto messudemolaitteelle, toteuttaa laitteelle liikkeenohjausohjelma, luoda Android-pohjaiseen näyttöön käyttöliittymä laitteen operointiin käyttäen Industry 4.0 -teknologiaa sekä tutustua Open Core Engineering -teknologiaan.

Ottaen huomioon lähtökohdat työ oli erittäin haastava. Lisäksi työtä hankaloitti työn aikataulutuksen riippuvuus laitteen fyysisen kokoonpanemisen aikataulusta ja siihen liittyvistä muuttujista. Tästä johtuen ei värähtelynvaimennusta ehditty toteuttaa tämän insinöörityön puitteissa. Lisäksi kokoonpano vaatii muutoksia ennen suurelle yleisölle esittelyä siinä olevien puutteiden takia. Tikkataulu on sijoitettu väärin suhteessa toimilaitteisiin. Toimilaitteiden liikkeet eivät kata koko tikkataulun aluetta vaan ne jäävät vajaaksi kuljettaessa sivuttaissuunnassa toiseen laitaan sekä pystysuunnassa toiseen ääripäähän. Toimilaitteiden liikematkat antavat kuitenkin mahdollisuuden koko alueen kattamiseen.

Työn alussa asetetut tavoitteet saatiin kuitenkin täytettyä, laite toimii luvussa 6.4 kuvatulla tavalla. Projektia tullaan myös jatkamaan insinöörityön ulkopuolella. Värähtelynvaimennus tullaan toteuttamaan, Android-käyttöliittymää kehittämään eteenpäin sekä laitteeseen lisäämään turvalogiikkaa.

Työssä opittiin, että vaativia asioita voi myös lähteä opettelemaan niin sanotusti väärästä päästä. Tällä tarkoitetaan työn Android-osuutta. Työ oli kaiken kaikkiaan hyvin tuottoisa tekijälleen ja siitä jäi käteen edistynyttä kokemusta PLC- ja Android-ohjelmoinnista, servokäytöistä sekä liikkeenohjauksesta.

Lähteet

1 Rexroth Suomessa. 2017. Verkkodokumentti. https://www.boschrexroth.com/fi/fi/yritys/tietoa-bosch-rexrothista/rexroth-suomessa/rexroth-in-finnland-1. Viitattu 6.5.2017

2 Bosch Rexroth. Industry 4.0 Overview. PowerPoint esitys. Sisäinen tietokanta.

3 Bosch Rexroth. Open Core Engineering. PowerPoint esitys. Sisäinen tietokanta

4 Bosch Rexroth. 2015. What can Open Core Engineering do for you? Verkkodokumentti. <https://www.youtube.com/watch?v=zC83xkOrCp0>. 22.12.2015. Viitattu 20.12.2016.

5 Bosch Rexroth. 2016. Understanding the Open Core Interface within automation environment. Verkkodokumentti. <https://www.youtube.com/watch?v=H9jMobqLSYo>. 25.2.2016. Viitattu 20.12.2016

6 Wilamowski, Bogdan M. & Irwin, J. David. 2011. The Industrial Electronics Handbook. Control and Mechatronics. Second Edition. Boca Raton, Florida. CRC Press.

7 Bosch Rexroth. 2016. MPx-20 Version Notes. Edition 03. 15.06.2016. Verkkodokumentti. <https://md.boschrexroth.com/modules/BRMV2PDFDownload-internet.dll/R911345606_03.pdf?db=brmv2&lvid=1195609&mvid=12679&clid=20&sid=BEE4B1350B68874153BB467CBCCB4223.borextc&sch=M&id=12679,20,1195609>. Viitattu 10.02.2017.

8 Servo control facts. A handbook explaining the basics of motion. Baldor Motors and drives. Baldor electric company. Verkkodokumentti. <http://www.baldor.com/Shared/manuals/1205-394.pdf>. Viitattu 10.02.2017.

9 Tyyppikilpi. IndraDrive Cs HCS01.1E-W0006-servokäyttö.

10 DIS Sensors. 2015. Acceleration sensor QG30-KAX-2,0E-AI-K. Datasheet. 13.10.2015. Verkkodokumentti. <https://dis-sensors.com/en/download/pdf/QG30-KAX-2,0E-AI-K.pdf>. Viitattu 03.03.2017

11 Van Andel, Marius. 2017. Product Manager, DIS Sensors bv. Sähköpostiviesti 11.04.2017.

12 Bosch Rexroth. 2011. Rexroth IndraDrive, Rexroth IndraMotion MLD Application Manual. Edition 05. 31.03.2011. Verkkodokumentti. < https://md.boschrexroth.com/modules/BRMV2PDFDownload-internet.dll/R911306084_05.pdf?db=brmv2&lvid=1155655&mvid=12716&clid=20&sid=775042BE728C300A78E39C74FD6DED64.borex-tc&sch=M&id=12716,20,1155655>. Viitattu 03.03.2017

13 Android Studio 2.3. 2017. Verkkodokumentti. <https://developer.android.com/index.html>. Viitattu 10.01.2017.

14 Bosch Rexroth. 2008. Rexroth IndraMotion MLD Getting Started. Edition 01. 16.04.2008. Verkkodokumentti. <https://md.boschrexroth.com/modules/BRMV2PDFDownload-internet.dll/r911319306_01.pdf?db=brmv2&lvid=1133109&mvid=12714&clid=20&sid=1655CF191864BE76DD3D57599938EA1C.borex-tc&sch=M&id=12714,20,1133109>. Viitattu 13.4.2017.

15 Vidqvist, Ville. 2017. Bosch Rexroth Oy, Industrial Applications, Technical Support. Haastattelu 10.4.2017 Vantaalla Bosch Rexrothin tiloissa.

16 Bosch Rexroth. 2016. IndraWorks 14VRS Engineering. Edition 05. 28.09.2016. Verkkodokumentti. <https://md.boschrexroth.com/modules/BRMV2PDFDownload-internet.dll/R911343566_05.pdf?db=brmv2&lvid=1198409&mvid=12716&clid=20&sid=775042BE728C300A78E39C74FD6DED64.borex-tc&sch=M&id=12716,20,1198409>. Viitattu 7.5.2017

17 Sensors overview. Verkkodokumentti. <https://developer.android.com/guide/topics/sensors/sensors_overview.html>. Viitattu 10.01.2017.

18 Vesterholm, Mika & Kyppö, Jorma. 2015. Java-ohjelmointi. 9. uudistettu painos. Espoo. Talentum.

19 Bosch Rexroth. Getting started with EAL4Android in Android Studio. Sisäinen tietokanta. Viitattu 10.01.2017.

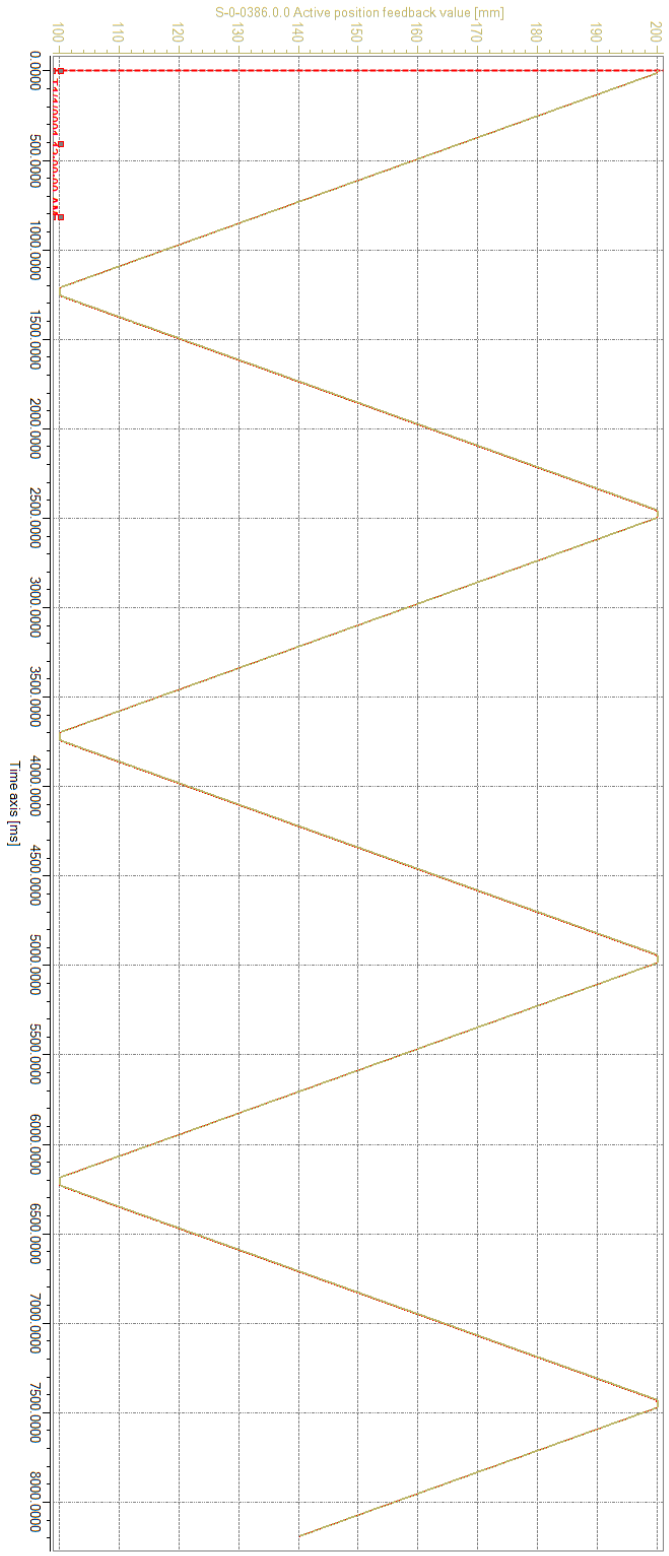
20 TC2 – Motion Control. 2008. Verkkodokumentti. <http://www.plcopen.org/pages/tc2_motion_control/>. Versio 0.41. Viitattu 12.02.2017.

21 Bosch Rexroth. 2017. Compact Modules CKK/CKR. 01.01.2017. Verkkodokumentti. <https://md.boschrexroth.com/modules/BRMV2PDFDownload-internet.dll/CKX_EN_R999000499_2017-01_13_01_2017.pdf?db=brmv2&lvid=117935&mvid=12679&clid=20&sid=7D43F37B894334B943BC9F66107CF7A7.borex-tc&sch=M&id=12679,20,117935>. Viitattu 04.04.2017.

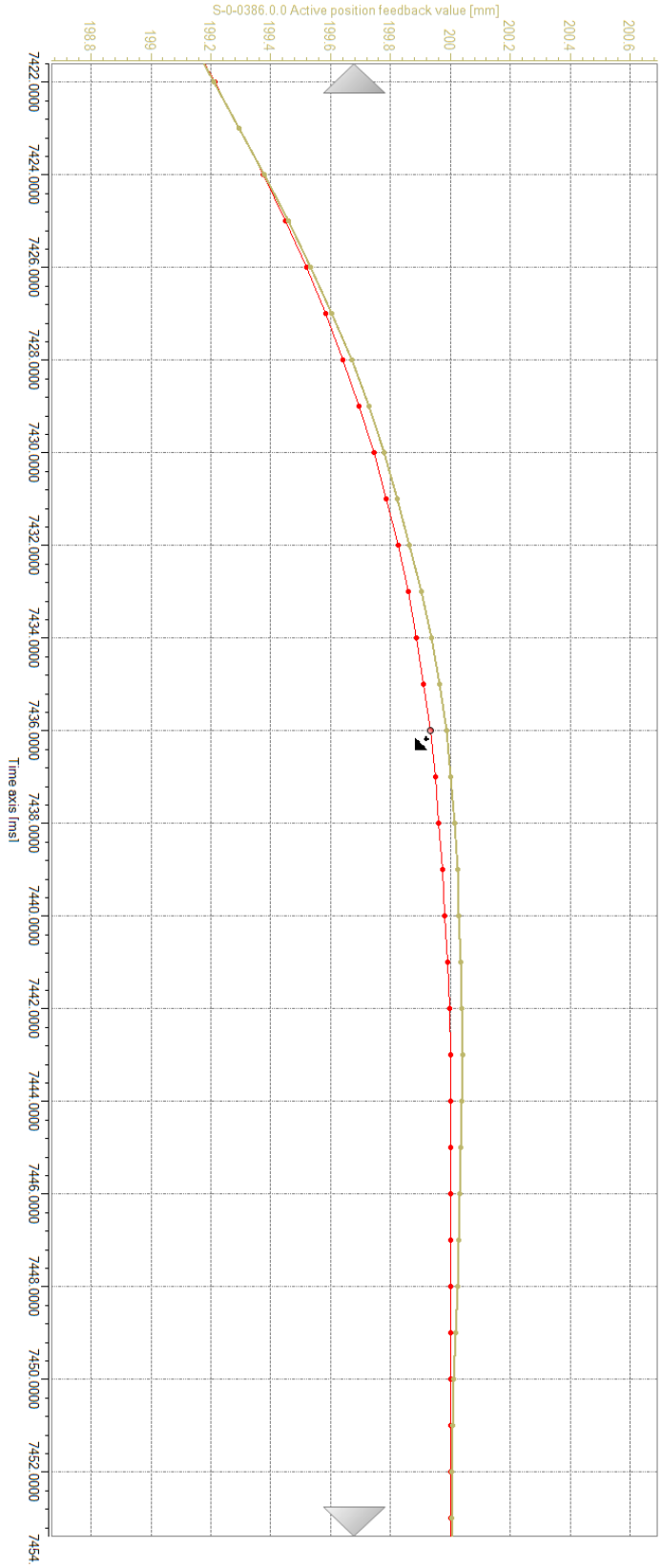
22 Wittenstein. 2017. Product catalog: Chapter CP. Verkkodokumentti. <http://alpha.wittenstein-us.com/products/servogetriebe/spielarme-planetengetriebe/cp-planetengetriebe/>. Viitattu 16.04.2017.

23 Bosch Rexroth. 2015. IndraDyn S MSM Synchronous Motors. Data Sheet. Edition 03.1. 12.11.2015. Verkkodokumentti. <https://md.boschrexroth.com/modules/BRMV2PDFDownload-internet.dll/R911329338_03.1.pdf?db=brmv2&lvid=1191699&mvid=12679&clid=20&sid=FFCDD1743D5CF6C9029BA1663BA6B16C.borextc&sch=M&id=12679,20,1191699>. Viitattu 16.04.2017.

Toimilaitteen paikkaohje ja todellinen sijainti ajan funktioina



Kuva 1. Toimilaitteen paikkaohje ja todellinen sijainti ajan funktioina.

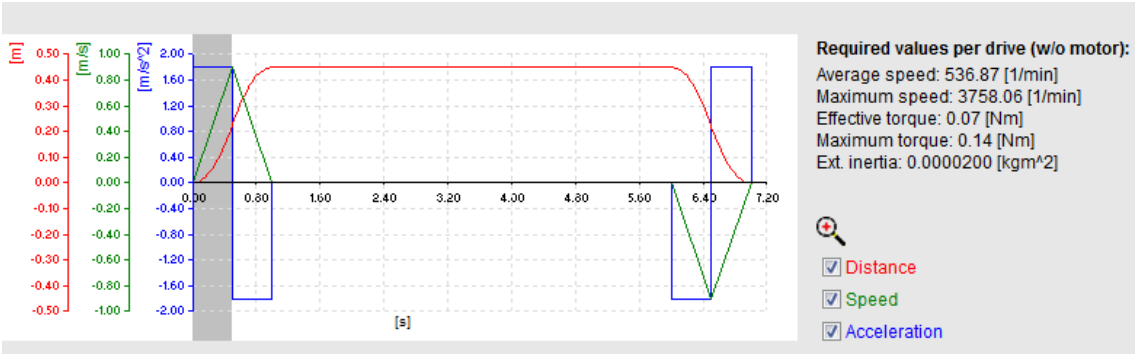


Kuva 2. Kuvaajassa turkoosilla nähdään todellinen paikka ja punaisella paikkaohje

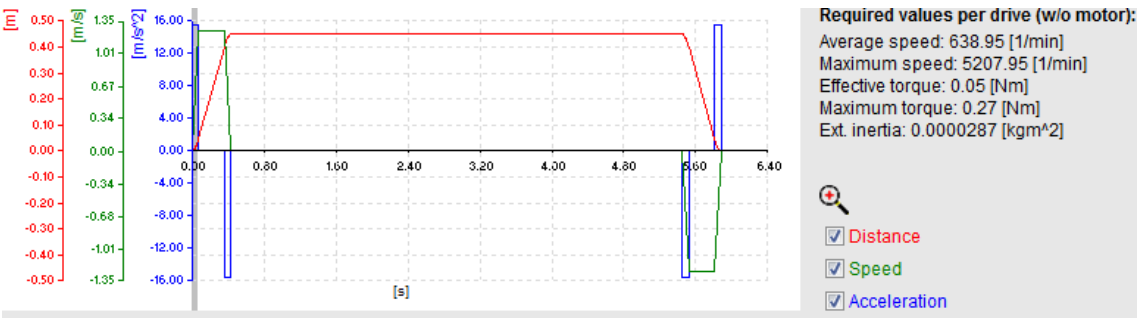
Moduulien moottorien valinnat (ote keväällä 2016 tehdystä projektista)

Bosch Rexrothin IndraSize-ohjelmistolla selvitettiin lineaarimoduuleille tarvittavat moottorit ja moottorinohjausyksiköt. Kokeilemalla työkiertoa eri kiihtyvyyksien arvoilla saimme selville työkierrolle sopivan moottorin. Sivuttais- ja pystysuunnassa liikematka on suurimmillaan 451 mm laidasta laitaan, mikä vastaa tikkataulun halkaisijaa. Kyseinen liike suoritetaan vaakasuunnassa 0,8 sekunnissa ja pystysuunnassa 1 sekunnissa. Vaakaliike on valittu nopeammaksi, jotta värähtelijä saataisiin värähtelemään mahdollisimman suurella amplitudilla.

Ohjelmistolle syötetään haluttu liikeprofiili liikeaikojen ja -matkojen (kuvat 1 ja 2) avulla. Ohjelmisto laskee tarvittavat moottorin pyörimisnopeudet ja vääntömomentit, ja näiden pohjalta saadaan moottorisuositukset, jos sellaiset kyseisillä arvoilla löytyy.



Kuva 1. Pystyliikkeen liikeprofiili



Kuva 2. Vaakaliikkeen liikeprofiili