Tommi Kasari

Testaus- ja vianhakutyökalu Epecin ohjausyksiköille

Opinnäytetyö

Kevät 2015

Tekniikan yksikkö

Tietotekniikan koulutusohjelma

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Koulutusohjelma: Tietotekniikan koulutusohjelma

Suuntautumisvaihtoehto: Ohjelmistotekniikka ja sulautetut järjestelmät

Tekijä: Tommi Kasari

Työn nimi: Testaus- ja vianhakutyökalu Epecin ohjausyksiköille

Ohjaaja: Heikki Palomäki

Vuosi: 2015 Sivumäärä: 59 Liitteiden lukumäärä: 4

Tämän opinnäytetyön tarkoituksena on selvittää tekniikka, jolla voidaan toteuttaa Epecin sulautettujen ohjausyksiköiden testaamiseen soveltuva ohjelmistotyökalu. Selvitystyö tehdään Epecin huoltoa varten, jossa työkalua tarvitaan ohjausyksiköiden vianhakuun. Epec Oy on ratkaisutoimittaja, joka on erikoistunut liikkuvien työkoneiden sulautettuihin koneenohjausjärjestelmiin.

Selvitystyössä tutustutaan ensin sulautetun koneenohjausjärjestelmän osa-alueisiin, kuten CAN-väylään, CANopen-protokollaan, ohjausyksiköiden rakenteeseen, sekä erityisesti Epecin 3606-ohjausyksikön I/O-pinnien toimintaan.

Tavoitteena on löytää tekniikka, jonka avulla työkalusta voidaan toteuttaa vaatimusmäärittelyn mukainen. Sopivaa tekniikkaa varten selvitystyössä vertaillaan vaihtoehtoisia toteutustapoja esimerkkisovellusten ja tekniikoihin liittyvien aineistojen avulla. Vertailussa käsitellään muun muassa CODESYS-ohjelmointia, IronPython-ohjelmointikielen käyttöä, sekä ohjausyksiköiden Firmwaren toimintaa.

Selvitystyön lopputuloksena on ratkaisu työkalun toteuttamiseksi, sekä suunnitelma sen ohjelmointia varten.

Avainsanat: Sulautetut järjestelmät, Epec, Ohjausyksikkö, CAN-väylä, testaus, CODESYS, IronPython

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Programming technology and embedded systems

Author: Tommi Kasari

Title of thesis: Testing and debugging tool for Epec control units

Supervisor: Heikki Palomäki

Year: 2015 Number of pages: 59 Number of appendices: 4

The purpose of this thesis was to discover a proper technic that could be used for implementing a software tool for Epec's embedded control units. The study was made for Epec Service where the tool is needed for control unit's fault analysis and testing. Epec is a solution provider that specializes in embedded control systems for mobile machines.

At first this thesis introduces the different parts of an embedded control system including CAN bus, CANopen protocol, structure of control units and especially Epec's 3606-control unit I/O-pins. The goal of this thesis was to find the most suitable technic for the tool implementation that would also pass the requirement specification.

To find the best technic for the tool, alternative methods are compared with the help of example programs and literature on different technics. The comparison will handle, among others, CODESYS programming, IronPython programming language and the functionality of control unit firmware. As a result of this thesis there is a solution for the tool implementation and a plan for the tool program.

Keywords: Embedded system, Epec, Control unit, CAN-bus, testing, CODESYS, IronPython

SISÄLTÖ

Kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

CAN	Tietoliikenneväylä (Controller Area Network).
Node	CAN-väylään liitetty laite, jota kutsutaan solmuksi.
PLC	Ohjelmoitava logiikka (Programmable Logic Controller).
Boot	Prosessorin käynnistysohjelma, jolla Firmware ladataan Flash-muistille.
Firmware	Asiakkaan päivitettävissä oleva laiteohjelmisto, joka toimii laitteiston ja Softwaren välissä.
Software	Laitekohtainen ohjelmisto, joka Epecin ohjausyksiköille ohjelmoidaan CODESYS:illa.
RAM	Haihtuva työmuisti (Random Access Memory).
NVRAM	Varmennettu RAM-muisti (Non-Volatile RAM).
FRAM	Ferrosähköisellä kiteellä varmistettu RAM-muisti (Ferro-electric RAM).
I/O	Input / Output.
DI	Digitaalitulo (Digital Input).
DO	Digitaalilähtö (Digital Output).
AI	Analogitulo (Analog Input).
PI	Pulssitulo (Pulse Input).
FB	Matalaimpedanssinen analogitulo (Feedback).
PWM	Pulssileveysmodulaatio (Pulse-Width Modulation).

1 JOHDANTO

1.1 Työn tausta

Epec Oy valmistaa ohjausyksiköitä liikkuviin työkoneisiin ympäri maailmaa. Vikaantuneet ohjausyksiköt palautuvat useimmiten Epecin huoltoon tutkittavaksi ja korjattavaksi. Ohjausyksiköitä on kymmeniä erialaisia ja niistä osa on takuun aikaisia, ja osa takuun ulkopuolisia. Viimeisen seitsemän vuoden ajan huolto on työllistänyt kolme työntekijää ohjausyksiköiden korjaamiseen ja vian selvittämiseen.

Jokaiselle ohjausyksikön tuoteperheelle on olemassa yksilöity huoltoprosessi, jonka mukaan ohjausyksikön vianhaku ja korjaus suoritetaan. Olennaisena osana huoltoprosessia jokaiselle ohjausyksikön mallille on olemassa vähintään yksi automaattitesteri, jonka avulla tuote pystytään vaivattomasti ja luotettavasti testaamaan.

Nykyisen huoltoprosessin ongelmaksi muodostuu ohjausyksiköiden vianhaun kannalta automaattitestereiden jatkuva käyttötarve uusien yksiköiden valmistuksessa, jolloin huoltoon tulleiden ohjausyksiköiden testaaminen ei aina ole mahdollista. Uusien samanlaisten testereiden hankinta pelkästään huollon tarvitsemaa vianhakua varten ei ole järkevää, sillä niiden hankintakustannukset ovat todella suuret.

Toinen ongelma testereiden käyttämisessä on niiden kankeus ja muokattavuus. Koska testerit on suunniteltu testaamaan mahdollisimman tehokkaasti suuria määriä ohjausyksiköitä, ne on sijoitettu keskeiselle paikalle tuotantoprosessia ja niiden testiohjelmia on vaikea lähteä muokkaamaan vianhakua varten.

1.2 Työn tavoite

Tämän selvitystyön tavoitteena on määritellä millaisella tekniikalla toteutettu laitteisto ja ohjelma riittäisivät Epecin huollolle testaustyökaluksi ohjausyksiköiden vianhakua varten. Selvityksen lisäksi työkalun ohjelmaa varten on tarkoitus tehdä

suunnitelma, jonka perusteella tärkeimmät toiminnot kattava työkalu pystytään toteuttamaan. Työkalu pyritään toteuttamaan mahdollisimman pitkälle Epecin omilla tuotteilla, ja jo käytössä olevilla järjestelmillä. Työkalun tavoitteena ei ole korvata olemassa olevia tuotantotestereitä, vaan tehostaa huollon toimintaa vähentämällä testereiden tarvetta vian selvityksessä. Samalla huollon toiminta tehostuu, mikä näkyy asiakkaille nopeampana palveluna.

1.3 Työn rakenne

Luvussa 2 tarkastellaan ensin yleisellä tasolla työkaluun liittyviä eri osa-alueita ja tekniikoita, joiden tuntemista ja huomioimista tarvitaan työkalun suunnittelussa. Luvussa keskitytään erityisesti CAN-väylän toimintaan, jolla on keskeinen rooli ohjausyksiköiden toiminnassa.

Luvussa 3 tarkastellaan tarkemmin testattavien ohjausyksiköiden I/O-rajapintaa, joita työkalulla on tarkoitus testata.

Luvussa 4 käydään lävitse työkalulle liittyvät vaatimukset, jotka on määritelty opinnäytetyön aloituspalaverissa Epecillä.

Luvussa 5 tarkastellaan työkalun toteuttamiseen soveltuvia vaihtoehtoja. Tarkastelussa käydään läpi lisäksi käytännön esimerkkejä yhden I/O-pinnin ohjaamisesta toteutettavissa olevilla vaihtoehdoilla. Tarkastelussa tehtyjen havaintojen perusteella valitaan lopuksi työkalun toteuttamiseen parhaiten sopiva vaihtoehto.

Luvussa 6 esitetään lopullisen työkalun toteuttamiseksi suunnitelma, jonka pohjalta työkalun kehitys aloitetaan.

Lopuksi luvussa 7 käydään lävitse lyhyt yhteenveto millaisia tavoitteita opinnäytetyönä tehtävälle selvitystyölle asetettiin ja miten niihin vastattiin.

1.4 Epec Oy

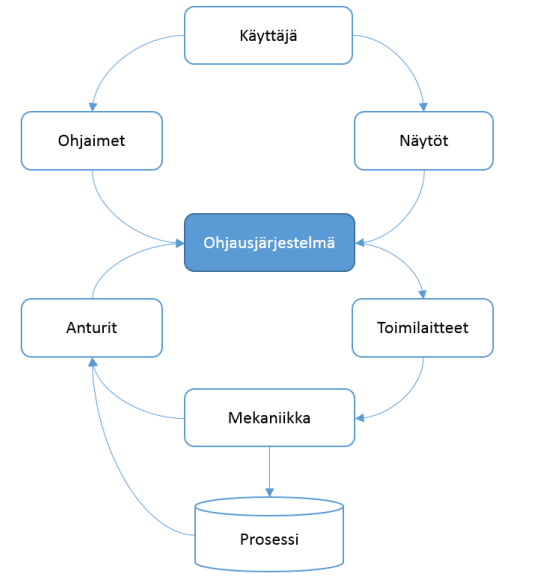
Epec Oy on vuonna 1978 Seinäjoella perustettu liikkuvien työkoneiden koneenohjausjärjestelmien ratkaisutoimittaja. Epec suunnittelee ja valmistaa tuotteet

piirilevystä lähtien valmiiksi tuotteiksi, tekee tarvittaessa niille asiakkaiden toiveiden mukaiset ohjelmat, sekä kouluttaa niiden käytössä. Epecin tuotteet koostuvat pääosin erilaisista koneenohjausjärjestelmien sulautetuista ohjausyksiköistä ja näyttöyksiköistä, jotka liitetään toisiinsa CAN-väylän avulla. Epecin tuotteita käytetään ympäri maailmaa lukuisissa erilaisissa koneissa ja laitteissa, kuten metsä-, kaivos-, maanrakennus-, yhdyskunta- ja maataloussektoreilla. Vuodesta 2004 lähtien Epec on kuulunut Ponsse Oyj:n konserniin. Epecin pääkonttori ja lähes kaikki yrityksen toiminnoista sijaitsevat Seinäjoella. Toimintoihin kuuluvat elektroniikkatuotanto, tuotekehitys, projektipalvelut, huolto sekä asiakastuki. Lisäksi Epecillä on konttori Kiinassa, joka tarjoaa Aasian alueen asiakkaille koulutusta ja asiakastukea. (Epec 2013c.)

2 Sulautettu koneenohjausjärjestelmä

Jokainen työkone tarvitsee toimiakseen jonkinlaisen ohjausjärjestelmän. Ohjausjärjestelmä voi yksinkertaisimmillaan olla täysin mekaaninen. Tietotekniikan avulla toteutetun koneenohjausjärjestelmän avulla koneesta saadaan kuitenkin monipuolisempi, nopeampi ja usein myös luotettavampi kuin pelkällä mekaanisella automaatiolla (Airila 2004, 1). Ohjausjärjestelmän avulla on myös mahdollista parantaa käyttäjän ja koneen turvallisuutta, lyhentää huolto- ja käyttökatkoksia tehokkaammalla vikojen selvittämisellä, sekä pienentää koneen polttoaineenkulutusta ja ympäristönkuormitusta optimoimalla sen käyttötehoja (Epec 2014b, 1).

Tietotekniikalla toteutettu koneenohjausjärjestelmä koostuu tyypillisesti käyttöliittymästä, sekä tulo- ja lähtöliitännöistä, jotka on kytketty ohjausjärjestelmän yhteen tai useampaan tietokoneeseen. Käyttöliittymä koostuu erilaisista näytöistä, napeista ja muista ohjauslaitteista, joita koneen käyttäjä käyttää koneen ohjaamiseen. Lähtöliitäntöihin kytketään koneen toimilaitteet, kuten erilaiset releet ja sähkömoottorit, joilla ohjataan koneen prosessia suorittavaa mekaniikkaa. Tuloliitäntöihin kytketään erilaiset anturit, joiden avulla koneenohjausjärjestelmän tietokone mittaa ja säätää koneen suorittamaa prosessia. (Airila 2004, 2-12.) Koneenohjausjärjestelmän pelkistetty mekatroninen malli on kuvattu kuviossa 1.

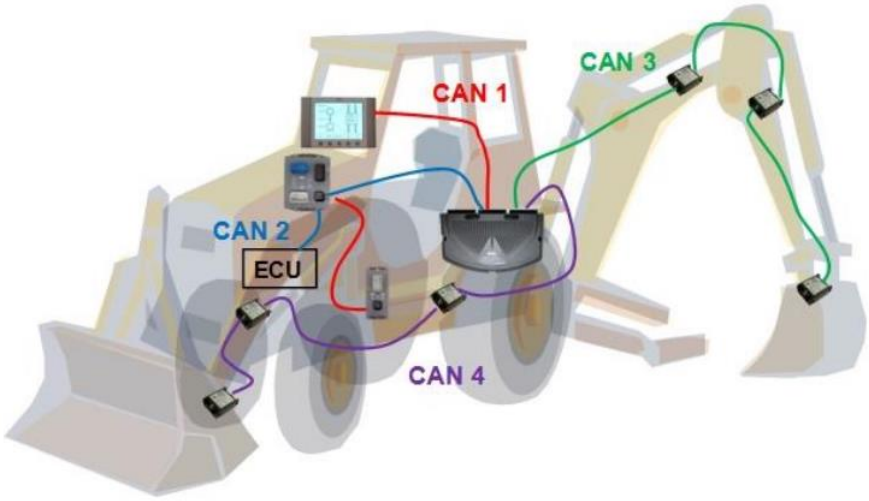


Kuvio 1. Mekatronisen koneen malli (Perustuu Airila 2004, 2)

2.1 Sulautettu järjestelmä

Sulautetulla järjestelmällä toteutetuissa koneenohjausjärjestelmissä perinteisen tietokoneen sijaan käytetään ohjausyksiköitä. Sulautetulla järjestelmällä tarkoitetaan laitetta, joka koostuu elektroniikalla ja mikrotietokoneella toteutetusta järjestelmästä (Koskinen 2004, 7).

Epecin suunnittelemat ja valmistamat ohjausjärjestelmät perustuvat useaan itsenäiseen ja älykkääseen sulautetun järjestelmän ohjausyksikköön ja näyttöön, jotka toisiinsa CAN-väylän avulla liitettyinä muodostavat integroidun kokonaisjärjestelmän (Epec 2014a). Kuviossa 2 on esimerkki ohjausyksiköillä toteutetusta koneenohjausjärjestelmästä.



Kuvio 2. Ohjausyksiköillä toteutettu koneenohjausjärjestelmä (Epec 2013a, 47)

2.2 Ohjausyksikkö

Tyypillinen Epecin valmistama ohjausyksikkö koostuu kotelosta, piirilevystä, piirilevylle juotetuista liittimistä, laiteohjelmistosta eli Firmwaresta, sekä ohjausjärjestelmän ohjelmasta eli Softwaresta.

Kotelo suojaa sisällä olevaa piirilevyä, johon kuuluvat ohjausyksikön liittimet, prosessori, muistit, I/O-rajapintaan liittyvät komponentit, sekä eri tietoliikenne-väylien oheispiirit, kuten esimerkiksi CAN-transceiverit.

Toimiakseen ohjausyksikkö tarvitsee myös ohjelman. Yleisin Epecin ohjausyksiköissä käytettävä ohjelmistoalusta on tällä hetkellä CODESYS 2.3, joka tehdään ohjausyksikölle, joko Epecin projektipalvelussa asiakkaan toiveiden mukaan tai asiakkaan itsensä toimesta. Kuviossa 3 on kuvattu Epecin 3724-ohjausyksikkö, joka on tällä hetkellä uusien ohjausyksiköiden yleisimmin käytössä oleva malli.

Kuvio 3. Epecin 3724-ohjausyksikkö (Epec 2014b, 10)

2.3 Prosessori ja muistit

Mikroprosessorin tehtävänä on lukea mikrotietokoneen muistiin tallennettua ohjelmaa ja suorittaa sen mukaan laitteen toimintoja liitäntäpiirien kautta (Koskinen 2004, 13). Useimmissa Epecin ohjausyksiköissä on käytettävissä kolmenlaista muistia: ohjelman suorittamiseen käytettävää haihtuvaa työmuistia eli RAM-muistia, sekä ohjelman ja sen parametrien tallentamista varten käytettävää haihtumatonta Flash- ja NVRAM-muistia.

Koska Flash-muistin kapasiteetti on näistä suurin, eikä sen sisältö katoa virtakatkonkaan jälkeen ilman erillistä tyhjäystä, sitä käytetään laitteen ohjelmakoodin säilyttämiseen. NVRAM-muistin käyttö on sen sijaan Flash-muistia nopeampaa ja verrattuna Flash-muistiin, sen kirjoituskerrat ovat lähes rajattomat. Tästä syystä NVRAM-muistia käytetäänkin prosessorin käyttömuistina ohjelman parametrien ja muuttujien tallentamiseen. (Koskinen 2004, 36–39.)

Ohjausyksiköiden erilaisten muistien koot ja tyypit on huomioitava testaustyökalua suunniteltaessa, mikäli työkalusta halutaan mahdollisimman helposti erilaisille ohjausyksiköille sopiva.

3606-ohjausyksikön muistien koot jakautuvat seuraavasti:

- Flash 1600 kB, josta Softwarea varten on käytettävissä enintään 768 kB.

- RAM 138 kB, josta ohjelman muuttujia varten käytettävissä 112 kB.

- NVRAM (FRAM) 2 kB, joka on käytettävissä ohjelman parametreja varten.

2.4 CODESYS-sovellus

Kaikkien uusiin koneisiin toimitettavien Epecin ohjausyksiköiden sovellukset tehdään saksalaisen 3S (Smart software solutions) -yrityksen kehittämällä CODESYS:illa. CODESYS on ohjelmankehitysalusta, joka perustuu ohjelmoitavien logiikoiden IEC-61131-3-standardiin (3S 2014, 3).

IEC-61131-3-standardi asettaa vaatimukset ohjelmointialustan ohjelmointikielille, sekä kääntäjälle. Standardiin kuuluu viisi erilaista ohjelmointikieltä: tekstipohjaiset Instruction List (IL) ja Structured Text (ST), sekä graafiset Ladder (LD), Function Block Diagram (FBD), sekä Sequential Function Chart (SFC). (IEC 2013, 9.)

Ohjelmointikielten ja kääntäjien lisäksi CODESYS:ista löytyy myös näyttöyksiköiden käyttöliittymien tekemisen mahdollistava työkalu CODESYS Visualization. CODESYS:illa voidaan myös ladata valmis sovellus suoraan ohjausyksikköön CAN-väylän kautta tietokoneeseen liitetyllä CAN-adapterilla.

2.5 Firmware

Toimiakseen ohjausyksikössä CODESYS-sovellus tarvitsee Bootin ja Firmwaren. Kun ohjausyksikköön kytketään virta, sen Flash-muistilta käynnistetään ensimmäisenä Boot. Bootin tehtävänä on hoitaa Firmwaren lataaminen Flash-muistille ensimmäisessä latauskerrassa sekä Firmwarea päivitettäessä, kun ohjausyksikkö on asetettu ohjelmanlataustilaan. Boot hoitaa myös Firmwaren käynnistämisen, kun ohjausyksikkö käynnistetään normaalisti. Firmwaren avulla

ohjausyksikköön saadaan ladattua CODESYS-sovellus CAN-väylän kautta. Kun Firmware on käynnistynyt, se käynnistää Flash-muistille ladatun CODESYS-sovelluksen. Tämän jälkeen CODESYS-sovellus keskustelee ohjausyksikön prosessorin kanssa Firmwaren välityksellä. (Epec 2012, 10,13.)

2.6 CAN-väylä

Epecin ohjausyksiköissä on jokaisessa mallissa vähintään yksi CAN-liityntä sen liittämiseksi koneenohjausjärjestelmään.

CAN-väylä on Boschin 1980-luvulla kehittämä väylä-pohjainen ja reaaliaikainen viestintäjärjestelmä autoteollisuuden käyttämille hajautetuille laitteille. Väylä on myöhemmin laajentunut käytettäväksi myös muilla sektoreilla, kuten erilaisissa työkoneissa sekä teollisuusautomaatiossa. (Liang & Popovic 2001, 21–23, 60.)

CAN-väylä on toiminnoltaan avoin, sarjamuotoinen ja asynkroninen tiedonsiirto-väylä. CAN-väylän laitteita kutsutaan yleisesti solmuiksi (nodes). Avoimena väylänä solmuja voidaan liittää väylälle sekä poistaa siitä vapaasti, ilman että väylän toiminta häiriintyy, sillä kaikki data lähetetään aina väylän jokaiselle solmulle. Viestit lähetetään samanmittaisina ja peräkkäisinä bittijonoina, jotka alkavat aina viestin ID-tunnisteella. ID:llä määritetään viestin tyyppi ja sen tärkeys. Koska viestiä ei osoiteta millekään tietylle solmulle, kaikki väylän solmut voivat käyttää viestiä. (Liang & Popovic 2001, 21–23.) Tästä syystä CAN-väylään voidaan helposti liittää muun muassa väylän diagnosointiin ja vian selvitykseen tarkoitettuja testilaitteita.

Sarjamuotoisena väylänä CAN-väylän signaalit koostuvat resessiivisestä 1-tilasta, sekä dominantista 0-tilasta. CAN-väylä on aina oletuksena resessiivisessä tilassa, kun mikään solmu ei lähetä väylälle viestiä. Kilpatilanteessa, kun useampi solmu haluaa lähettää väylälle viestin samanaikaisesti, dominantti 0-bitti kumoaa 1-bitin ja menee näin resessiivisen bitin edelle. (Liang & Popovic 2001, 21–23.) Näin ollen laite, jonka viestin ID on pienin, saa jatkaa viestin lähetystä, koska se sisältää eniten 0-tason bittejä.

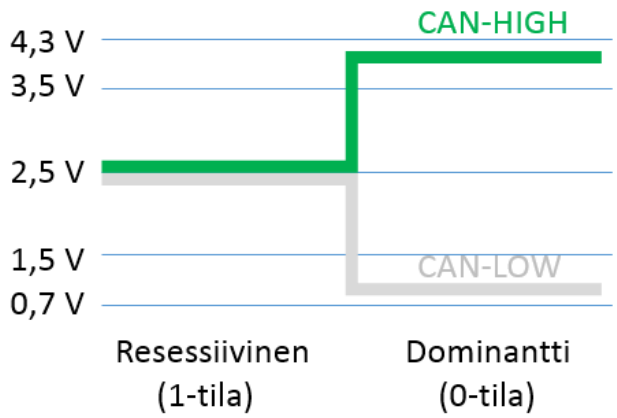
2.6.1 CAN-väylän ISO-OSI-malli

Perinteinen CAN-standardi koostuu ISO:n OSI-mallin kolmesta kerroksesta: fyysisestä kerroksesta (Physical Layer), linkkikerroksesta (Data Link Layer), sekä sovelluskerroksesta (Application Layer).

2.6.2 Fyysinen kerros

CAN-standardin fyysisessä kerroksessa määritellään muun muassa CAN-väylän laitteissa käytettävät CAN-kontrollerit ja -transceiverit, kaapelit ja liittimet, sekä väylän johdinten jännitetasot (Liang & Popovic 2001, 35–40).

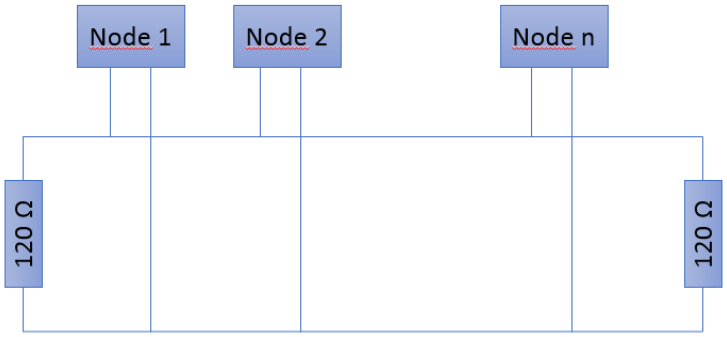
Epecin ohjausyksiköissä käytetään High-speed CAN-standardin fyysistä kerrosta. High-Speed CAN onkin yleisin käytössä oleva fyysinen kerros, joka on määritetty ISO 11898-2 standardissa (Saha 2005, 7–8). Standardi määrittelee CAN-väylälle muun muassa väylän rakenteen, jännitetasot, sallittujen solmujen määrän, sekä väylän pituuden ja tiedonsiirtonopeuden ylärajan (Liang & Popovic 2001, 61). High-speed CAN -väylä koostuu yleensä kierretystä parikaapelista, joiden johtimille käytetään nimitystä CAN-High ja CAN-Low. Johtimien jännitetasoja vastaavat CAN-väylän tilat on kuvattu kuviossa 4.



Kuvio 4. ISO 11898-2 -standardin jännitetasot (perustuu Saha 2005, 8)

Parikaapelilla toteutetulla High-speed CAN -väylällä voidaan saavuttaa enimmillään 1 Mbps tiedonsiirtonopeus, jolloin väylän maksimipituus rajoittuu 40

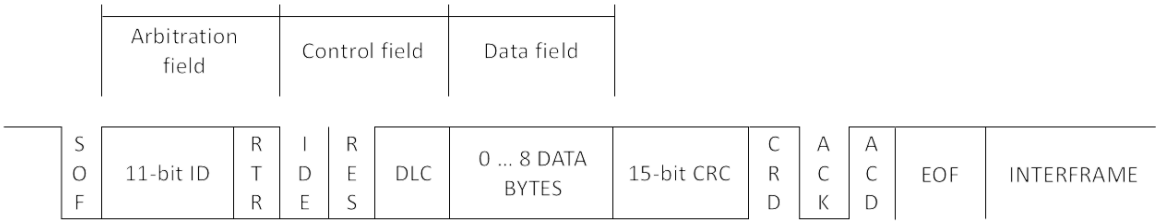
metriin. Signaalien heijastuksien vaimentamiseksi sen molemmat päät tulee terminoida 120 Ω vastuksilla. (Saha 2005, 7-8.) Kuviossa 5 on ISO 11898-2 -standardin mukainen rakenne CAN-väylälle.



Kuvio 5. ISO 11898-2 -standardin mukainen CAN-väylän rakenne

2.6.3 Linkkikerros

CAN-standardin linkkikerroksessa määritellään CAN-väylän viestikehysten rakenne ja sen kautta CAN-väylän luotettavuuteen vaikuttavat toiminnot (Liang & Popovic 2001, 35–36, 41–48). Kuviossa 6 on esitetty normaalipituisen CAN-viestikehyksen rakenne.



Kuvio 6. CAN-viestikehyksen rakenne (Perustuu Liang & Popovic 2001, 28)

Käytettäessä Epecin ohjausyksiköitä niille tarkoitetuilla ohjelmistotyökaluilla CAN-viestikehyksestä tarvitsee käsitellä käytännössä vain sen kolmea kenttää: Arbitration-kenttää, Kontrolli-kenttää ja Data-kenttää.

Arbitration-kenttä pitää sisällään viestin 11-bittisen ID:n, sekä viestien pyytämiseen toiselta solmulta varten tarkoitetun RTR-bitin (Remote Transmission Request).

Kontrolli-kenttä pitää sisällään neljän bitin mittaisen DLC-kentän (Data Length Code), jolla kerrotaan viestissä olevan datakentän pituus 0–8 tavun välillä. Data-kenttä pitää sisällään varsinaisen tiedon, tai toiminnon, joka väylälle lähetetään. (Liang & Popovic 2001, 35–40.) Muiden viestikehyksessä olevien kenttien tarkoitus on käytännössä varmistaa väylän virheetön toiminta eri tilanteissa ja ne jätetään useimmiten laiteajureiden tai laiteohjelmiston hoidettavaksi.

2.6.4 Sovelluskerros

CAN-standardin ylimmän kerroksen eli sovelluskerroksen tarkoituksena on tarjota käyttäjäsovelluksiin CAN-väylän käyttöön liittyvät toiminnot ja protokollat. Näiden avulla määritellään muun muassa millaisia viestiä väylällä voidaan lähettää ja vastaanottaa. (Liang & Popovic 2001, 48–49.) Yksi CAN-väylän sovelluskerroksien protokollista on CANopen (Saha 2006, 6).

2.7 CANopen

Epecin ohjausyksiköiden sisältämät CAN-liitynnät perustuvat kaikki CANopen-protokollaan. CANopen on kansainvälisten yritysten perustaman CiA-järjestön kehittämä ja standardoima CAN-väyläprotokolla, joka on erityisesti suunniteltu sulautetuille järjestelmille (CiA [Viitattu 23.11.2014]). Keskeisin CANopen-protokollan määrittelemistä toiminnoista on sen tarjoama objektikirjasto OD (Object Dictionary) (Saha 2006, 6–7).

2.7.1 Objektikirjasto

Objektikirjasto toimii väylällä olevien solmujen tietovarastona parametreille ja signaaleille. Sen avulla väylällä olevat solmut ja niiden sisältämät sovellukset ovat tunnistettavissa ja ohjattavissa väylältä. Objektikirjasto on jaettu 65535 16-bittiseen indeksiin, joista kukin indeksi voi sisältää enintään 254 8-bittistä ali-indeksiä. (Saha 2006, 6–7.) Taulukossa 1 on kuvattu objektikirjaston indeksiavaruuden jakautuminen eri toimintojen välillä.

Taulukko 1. Objektikirjaston indeksien jako (Epec 2010, 23)

Index range	Use	Description
0000_h	Reserved	Reserved
0001_h – $025F_h$	Data types	
0260_h – FFF_h	Reserved	Reserved
1000_h – $1FFF_h$	Communication object area	Describes the communication behavior of the device
2000_h – $5FFF_h$	Manufacturer specific area	Describes the application behavior in a manufacturer specific way
6000_h – $9FFF_h$	Device profile specific area	Describes the application behaviour in a standardized way by following a CANopen device or application profile
$A000_h$ – $BFFF_h$	Interface profile specific area	Defines the network variables and system variables according to the related CiA interface specification
$C000_h$ – $FFFF_h$	Reserved	Reserved

Objektikirjaston ensimmäinen 0-indeksi on varattu. Indeksit 1–FFF on varattu tietotyyppejä varten. Indeksit 1000–1FFF ovat liikennöintiparametreja ja solmukohtaisia tunnisteita varten. Näihin tallennetaan muun muassa väylä-liikennöinnin ohjaamiseen tarvittavat parametrit, joihin kuuluu muun muassa TPDO- ja RPDO-viestikehykset, sekä solmun valmistetiedot. Indeksit 2000–5FFF ovat laitteen valmistajan sovelluksen tarvitsemia muuttujia ja signaaleja varten, kuten esimerkiksi laitteen yksittäisen lähdön ohjaamista varten tarvittavat muuttujat. Indeksit 6000–9FFF on jaettu laitteistoprofiileille, joita yhdellä solmulla voi olla kaikkiaan kahdeksan erilaista. Lisäksi objektikirjastoon on varattu indeksi-alueet ohjelmoitavien logiikoiden sekä CANopen-CANopen-gatewayn tarvitsemille prosessimuuttujille ja tulevaisuuden laajennuksille. (Saha 2006, 7–8.)

2.7.2 CANopen-protokollat

Objektikirjaston lisäksi CANopen tarjoaa koneenohjausjärjestelmän peruspalvelui-den toteuttamiseen lukuisia protokollia, joilla saadaan varmistettua koneen turvallinen ja hallittu käynnistyminen, sekä käytön aikainen toiminta (Saha 2006, 8–9).

Yhdellä CANopen-väylällä voi olla kaiken kaikkiaan 127 eri solmua, joista jokaisella on oltava eri tunnistenumero, eli node-ID, väliltä 1–127. Yksi solmuista toimii väylän isäntäsolmuna (Master) ja muut solmut toimivat orjina (Slaves) (Epec 2010, 24–25). Isäntäsolmun tehtävänä on valvoa väylän toimintaa, sekä suorittaa muun muassa väylän solmujen hallittu käynnistyminen NMT-protokollan (Network Management) avulla (Saha 2006, 8–9).

NMT-protokollan lisäksi isäntäsolmun tehtävänä on tyypillisesti hoitaa objektikirjaston arvojen lukeminen ja kirjoittaminen SDO-protokollan (Service Data Object) avulla, sekä hoitaa väylän solmujen konfiguraatioiden hallinta. (Saha 2006, 6–9.)

CANopen tarjoaa lisäksi lukuisia muita protokollia väylällä olevien solmujen käytettäväksi, millä saadaan parannettua järjestelmän luotettavuutta. Näitä ovat muun muassa Heartbeat-protokolla sekä EMCY-protokolla. Heartbeat-protokollan avulla väylällä olevat solmut ilmaisevat muille solmuille määrätyin väliajoin olevansa toiminnassa. (Saha 2006, 6–9.) Heartbeat-protokollan tarjoamien viestien avulla voidaankin helposti varmistaa, että CAN-väylä on ehjä ja kaikki koneeseen kuuluvat solmut ovat kytkettynä ja toiminnassa. Jos jonkin väylällä olevan solmun toiminnassa havaitaan ongelmia, siitä voidaan ilmoittaa muille solmuille EMCY-protokollan avulla. EMCY-protokolla (Emergency) on korkean prioriteetin virheilmoituksia varten tehty protokolla (Saha 2006, 6–9).

Sovelluskohtaisten viestien lähettämiseen CANopen-väylällä voidaan käyttää kahta protokollaa, joko SDO-protokollan mukaisia osoituksia suoraan objektikirjastoon solmulta toiselle tai PDO-protokollan (Process Data Object) mukaisia korkeamman prioriteetin omaavia viestejä, jotka lähetetään koko väylälle kulutettavaksi. SDO-protokollan mukaiset osoitukset lähetetään yleensä isäntäsolmun ja orjasolmun välillä, joista toinen solmu pyytää dataa ja toinen solmu vastaa pyyntöön. PDO-protokollia on puolestaan kahta laatua, lähetettäviä viestejä varten oleva TPDO-protokolla (Transmit PDO), sekä vastaanotettavia viestejä varten oleva RPDO-protokolla (Receive PDO). TPDO-viestien lähettämiseen voidaan käyttää useampaa eri tapaa. TPDO-viesti voidaan määrittää lähetettäväksi ennalta määrätyn väliajoin tai jonkin solmun sisäisen muuttujan tilan muuttuessa. Vaihtoehtoisesti TPDO-viesti voidaan määrittää

lähetettäväksi, jonkun toisen solmun pyynnöstä esimerkiksi RTR-bitin avulla tai synkronoidusti SYNC-protokollan (Synchronization) mukaisen viestin avulla. (Saha 2006, 6–9.)

Jokainen väylällä oleva solmu voidaan puolestaan määrittää vastaanottamaan tietty TPDO-viesti RPDO-protokollan avulla, kun sen kommunikointiobjektin tunniste on sama kuin lähetetyllä viestillä. Kommunikointiobjektin tunniste eli COB-ID (Communication Object ID) on CAN-viestien ID:stä käytetty nimitys CANopen-standardissa (Saha 2006, 6–9). COB-ID koostuu viestissä käytettävän kommunikointiobjektin 4-bittisestä tunnisteesta, sekä 7-bittisestä node-ID:stä (IXXAT [Viitattu 27.12.2014]). CANopen-protokollan kommunikointiobjekteille on ennalta määritetyt tunnisteet, jotka on esitetty taulukossa 2.

Taulukko 2. CANopen protokollien viestitunnisteet (Epec 2010, 37)

Communication object	COB-ID (hex)	Slave nodes
NMT	0	Receive only
SYNC	80	Receive only
EMCY	80+NODE-ID	Transmit
TIMESTAMP	100	Receive only
PDO	180+NODE-ID	1. Transmit PDO
	200+NODE-ID	1. Receive PDO
	280+NODE-ID	2. Transmit PDO
	300+NODE-ID	2. Receive PDO
	380+NODE-ID	3. Transmit PDO
	400+NODE-ID	3. Receive PDO
	480+NODE-ID	4. Transmit PDO
	500+NODE-ID	4. Receive PDO
SDO	580+NODE-ID	Transmit
	600+NODE-ID	Receive
NMT node monitoring	700+NODE-ID	Transmit

3 Ohjausyksiköt ja toiminnot

Epecin uusimpien sukupolvien ohjausyksiköihin kuuluu neljä tuoteperhettä, joille huollolla ei ole tällä hetkellä omaa testausympäristöä. Näitä ovat 3000-, 4000-, 5000- ja 6000-sarjan ohjausyksiköt.

Tämän opinnäytetyön selvityksessä on päätetty käyttää 3606-ohjausyksikköä, joka on Epecin 3000-sarjan ohjausyksiköistä pienikokoisin ja yksinkertaisin, mutta sisältää siitä huolimatta lähes kaikki samat toiminnot, jotka löytyvät Epecin muista ohjausyksiköistä. 3606-ohjausyksikkö käyttää myös samaa prosessoria kuin 3724- ja 4602-mallit, mikä helpottaa testiohjelman muuntamista eri malleille. 3606-ohjausyksikkö on esitetty kuviossa 7.

Kuvio 7. 3606-ohjausyksikkö (Epec 2014b, 11)

3.1 Ohjausyksiköiden I/O-tyypit

Ohjelmoitavia ohjausyksiköitä varten tehdyn IEC 61131-2 -standardin mukaan PLC-järjestelmän rajapinnassa tulisi olla vähintään yksi alas vetävä digitaalinen sisääntulo, sekä yksi ylös vetävä lähtö (IEC 2007, 37–42). Koneen erilaisten toimilaitteiden ohjaamista ja antureiden lukemista varten Epecin ohjausyksiköt sisältävät useita erilaisia sisään- ja ulostuloja eli I/O-pinnejä. I/O-pinnit voidaan pinnin tyypistä riippuen ohjelmoida, joko syöttämään jännitettä ulos pinnistä tai lukemaan pinnistä sisälle syötettyä jännitettä tai virtaa. (Epec 2013a, 12–13.)

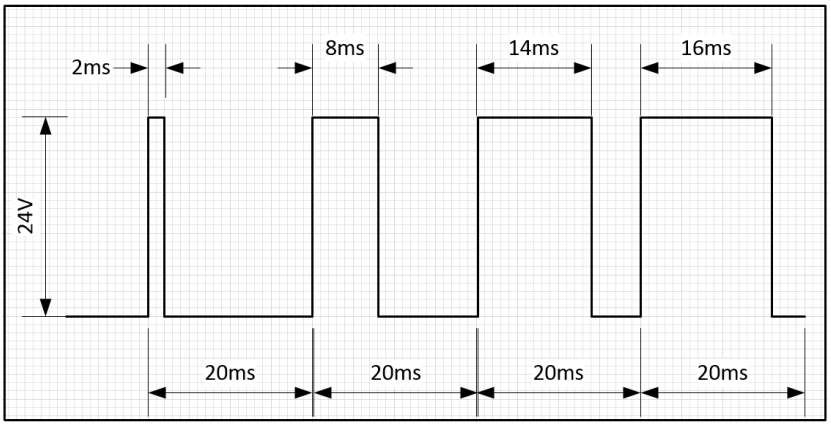
3606-ohjausyksikössä on yksi liitin, jossa on kaikkiaan 35 pinniä. Näistä 21 pinniä on käytettävissä I/O-pinneinä. Eri pinnien toiminnollisuudet on lueteltu Taulukossa 3.

Taulukko 3. 3606-ohjausyksikön I/O-taulukko (Epec 2013a, 12)

Max Amount	Pint Type	Info	DI	AI	FB	PI	DO sourcing	DO sinking	PWM
8	PWM/DO/DI_Type037_1	3 A	X				X		X
5	DI/PI_Type039_1	pull-down to GND	X			X			
4	AI/DI_Type062_1	0-5 V / 0-22 mA	X	X					
4	FB/AI_Type061_1	0-1 A			X				

3.2 PWM/DO/DI-pinnit

Koneen toimilaitteiden, kuten erilaisten venttiilien ja moottoreiden ohjaamiseen käytetään tyypillisesti PWM-lähtöjä. PWM-ohjaus perustuu ulostulevan jännite-pulssin taajuuden ja leveyden muuttamiseen (Koskinen 2004, 107). Kuviossa 8 on esimerkki PWM-signaalista, jossa taajuus on 50 Hz ja pulssin leveys vaihtelee 2 ms - 16 ms välillä.

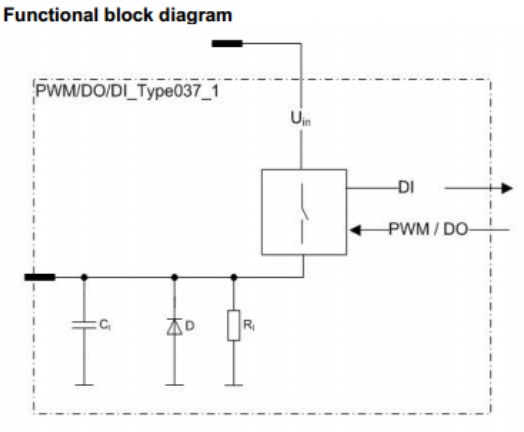


Kuvio 8. Esimerkki PWM-signaalista 50 Hz:n taajuudella.

Toinen tyypillinen tapa ohjata koneen toimilaitteita, kuten releitä ja lamppuja, on käyttää digitaalisia lähtöjä.

Digitaalisia lähtöjä on kahdenlaisia, ylös vetäviä (sourcing), sekä alas vetäviä (sinking). Ylös vetävä digitaalinen lähtö syöttää virtaa pinniin liitetylle laitteelle, kun alas vetävä digitaalinen lähtö kytkee pinniin liitetyn laitteen maahan.

3606-ohjausyksikön PWM/DO/DI-pinnejä voidaan käyttää kolmella eri tapaa: PWM-lähtönä, ylös vetävänä digitaalilähtönä tai ylös vedettävänä digitaalitulona (Epec 2013a, 13–14). Kuviossa 9 on esitetty PWM/DO/DI-pinnien kytkentäperiaate 3606-ohjausyksikössä.



Kuvio 9. 3606-ohjausyksikön PWM/DO/DI-pinnin toiminta (Epec 2013a, 14)

Kuviosta 9 nähdään, että pinnin toiminnot on toteutettu kahdella prosessoriin kytketyllä ohjauksella eli DI-ohjauksella ja PWM/DO-ohjauksella. PWM/DO-ohjaus ohjaa pinnin FET:iä ja DI-ohjaus välittää pinnin jännitetilan prosessorille.

Suoritettaessa PWM/DO/DI-pinnille vianhakua, korjaajan tulee mitata pinnistä ulostuleva jännite oskilloskoopin avulla. Sen avulla nähdään ulostulevan jännitteen suuruus, signaalin muoto, sekä nousevan ja laskevan reunan jyrkkyys, joista voidaan päätellä toimiiko pinni oikein. Jotta mittaus voidaan suorittaa, korjaajan on ensin kytkettävä pinni päälle testaustyökalun avulla.

PWM/DO-pinnin kytkemistä varten on tiedettävä kyseistä pinniä ohjaavan prosessorin kanava ja portti. Toiminnon käyttöönottoa varten kyseiselle portille pitää ensin määrittää sen tila, käytetäänkö sitä sisääntulona vai lähtönä, onko käytössä sisäistä ylös- tai alas-vetovastusta ja onko kanava käännettynä vai ei. Käytettäessä pinniä PWM-lähtönä, PWM/DO-portti asetetaan PWM-lähdöksi.

Ennen portin päälle kytkemistä prosessorille määritetään myös porttia vastaavan laskurin taajuus sekä pulssisuhde. Käytettäessä pinniä digitaalilähtönä PWM/DO-portti asetetaan pelkäksi lähdöksi, eikä muita asetuksia tarvita. (Epec 2013b, 14–17.) Pinnin testaamisen kannalta ei ole väliä suoritetaanko mittaus digitaali- vai PWM-lähtönä, mutta ulostulevaa PWM-signaalia on helpompi tarkastella oskilloskoopin avulla.

Ulostulevan signaalin lisäksi korjaajan tulee myös testata pinnin digitaalitulon toiminta. Pinnin DI-toiminto toimii tavallisen digitaalitulon tavoin ja sen asettamisessa on huomioitava samat asiat kuin digitaalilähdön asettamisessa. Huomioitavaa on se, että DI-toiminnolla on eri portti kuin PWM/DO-toiminnolla. Tämä mahdollistaa pinnin käyttämisen perinteisen digitaalitulon lisäksi sen PWM/DO-lähdön diagnosointiin. Näin ollen kun pinniä testaan lähtönä, korjaajan tulee myös selvittää pinniä vastaavan DI-portin tila prosessorilta, joka onnistuu testaustyökalun avulla. Lähdön lisäksi pinni tulee testata myös ylös vedettävänä digitaalisena sisääntulona. Tällöin korjaajan on huomioitava, että PWM/DO-ohjaus pitää olla kytkettynä pois päältä, ennen kuin pinniin syötetään jännite ulkoapäin.

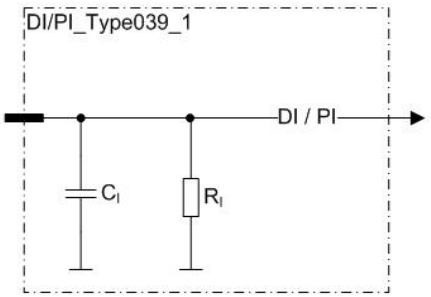
3.3 DI/PI-pinnit

Koneen antureiden ja ohjauslaitteiden, kuten painikkeiden kytkemiseksi ohjausyksiköstä tarvitaan digitaalisia sisääntuloja. Kuten on digitaalilähtöjä, myös digitaalituloja on kahta tyyppiä, alas vedettäviä ja ylös vedettäviä. Prosessoriin kytketyllä digitaalitulolla voi olla myös sisäinen laskuri, jonka avulla digitaalituloa voidaan käyttää myös pulssitulona.

3606-ohjausyksikön DI/PI-pinnejä voidaan käyttää kahdella tapaa, joko tavallisena ylös vedettävänä digitaalitulona tai pulssitulona (Epec 2013a, 15). Digitaalitulon testaamista varten riittää, kun pinnin portti asetetaan ensin sisääntuloksi. Tämän jälkeen pinniin syötetään jännite ulkoapäin ja pinnin tila luetaan prosessorilta sitä vastaavasta rekisteristä testaustyökalun avulla.

Pulssitulon testaamista varten prosessorin portille on asetettava lisäksi laskuri ja mitattavaa suuretta vastaavat asetukset. Tämän jälkeen pinniin syötetään funktio-

generaattorilla, tai toisella ohjausyksiköllä PMW-signaali, joka luetaan testaus-työkalun avulla. 3606-ohjausyksikön pulssituloilla voidaan laskea pulssin nousevaa ja laskevaa reunaa tai molempia, sekä mitata pulssin taajuutta ja leveyttä (Epec 2013a, 15). Pinnin kytkentäperiaate on kuvattu kuviossa 10.



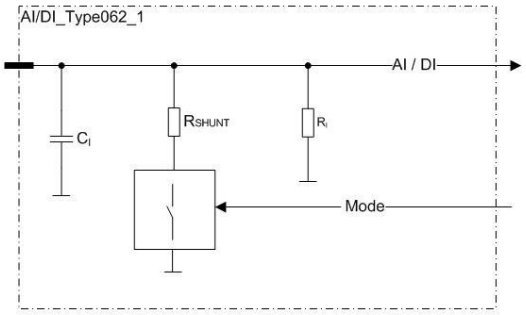
Kuvio 10. 3606-ohjausyksikön DI/PI-pinnin toiminta (Epec 2013a, 15)

3.4 AI/DI-pinnit

Analogitulon eli analogisen jännitevertailijan avulla voidaan mitata jännitteen suuruutta. Analogitulon toiminta perustuu A/D-muuntimeen, jolla ohjausyksikön analogituloon tuotava jännitesignaali muunnetaan prosessorin ymmärtämäksi digitaaliseksi viestiksi. (Koskinen 2004, 106.) Koneenohjausjärjestelmässä analogituloa voidaan käyttää esimerkiksi jännite- tai virtatietoa syöttävälle anturille, tai ohjauslaitteiden joystickia varten.

Epecin 3606-ohjausyksikön AI/DI-pinnejä voidaan käyttää kolmella eri tapaa: jännitettä tai virtaa mittaavana analogitulona sekä digitaalitulona. Pinniin kytketyn jännitteen tai virran suuruus luetaan raaka-arvona suoraan prosessorin kanavasta, johon pinniä vastaavan prosessorin portti on kytketty. Pinnin mittaustapaa vaihdetaan muuttamalla analogitulon kytkentään prosessorilta tulevaa Mode-ohjausta. Kun prosessorin Mode-ohjaus on kytkettynä päälle, pinniin syötetty virta kulkee virranmittaukseen tarkoitetun vastuksen läpi maahan, jolloin prosessorin A/D-muuntimella saadaan mitattua virta 0–22 mA välillä. Ilman ohjausta pinni toimii jännitettä mittaavana analogitulona, tai digitaalitulona, jolloin sillä saadaan mitattua jännite 0–5 V välillä. (Epec 2013a, 16–17.) Pinnin digitaalitulon toiminta määritellään täysin ohjelmallisesti, joten sen toimintaa työkalulla ei tarvitse

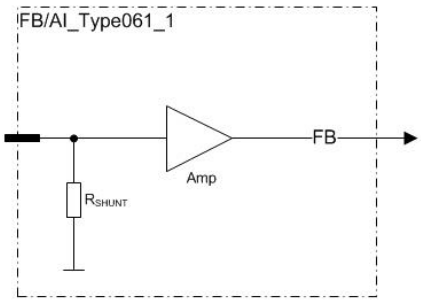
erikseen testata. Sen sijaan analogitulon toiminta jännitetulona ja virtatulona pitää testata erikseen. Molempien toimintojen testaus suoritetaan samalla tavalla. Pinnin tyyppi asetetaan ensin halutuksi testaustyökalun avulla. Tämän jälkeen pinniin syötetään ulkoapäin erisuuruista jännitettä tai virtaa, jonka suuruus luetaan testaustyökalulla. Pinnin kytkentäperiaate on esitetty kuviossa 11.



Kuvio 11. 3606-ohjausyksikön AI/DI-pinnin toiminta (Epec 2013a, 17)

3.5 FB/AI-pinnit

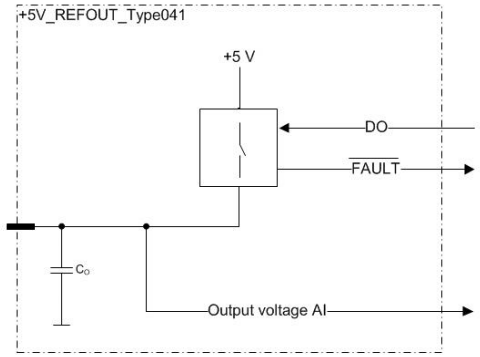
Koneen PWM-ulostulolla ohjattavien laitteiden säätämistä varten tarvitaan monesti laitteen läpi kulkevan virran mittaamista. Tätä virran mittausta varten ohjausyksiköistä löytyy sitä varten tehtyjä matalaimpedanssisia analogituloja, joista käytetään nimitystä Feedback-tulo. Feedback-tulon ero tavalliseen analogituloon onkin ainoastaan sen suurempi virran kesto sekä se, että sillä voidaan mitata pelkästään siihen kytketyn laitteen virtaa. (Epec 2013a, 17.) Feedback-tulon kytkentäperiaate on kuvattu kuviossa 12.



Kuvio 12. 3606-ohjausyksikön FB/AI-pinnin toiminta (Epec 2013a, 17)

3.6 +5V REF-pinnit

Perinteisten I/O-pinnien lisäksi 3606-ohjausyksikössä on kaksi +5 V:n referenssijännitettä tarjoavaa pinniä, joita voidaan valvoa analogisesti ja digitaalisesti, sekä ohjata ohjelmallisesti vikatilanteiden varalta päälle ja pois (Epec 2013a, 18). Pinnin testaamista varten kytkennän DO-ohjaus on asetettava ensin lähdöksi ja kytkettävä päälle testaustyökalun avulla. Tämän jälkeen, jos kytkentä toimii oikein, pinniin tulee +5 V:n jännite, jonka korjaaja mittaa pinnistä oskilloskoopin avulla. Lisäksi korjaajan tulee tarkastaa testaustyökalulla pinnin FAULT-ohjausta vastaavan portin tila, joka tulee olla 0-tilassa, sekä Output voltage AI -portin A/D-arvo, jonka tulee vastata +5 V:n jännitettä. Kytkennän toimintaperiaate on kuvattu kuviossa 13.



Kuvio 13. 3606-ohjausyksikön REF-pinnin toiminta (Epec 2013a, 18)

4 Vaatimusmäärittely

Työkalun selvitystyö alkoi kesäkuussa 2014 Epecillä pidetyn aloituspalaverin pohjalta vaativuusmäärittelyllä sekä vaihtoehtoisten toteutustapojen kartoittamisella.

Laitteisto tulee koostumaan Epecin omista tai jälleenmyytävistä tuotteista, kuten sulautetusta näyttöyksiköstä tai PC-ohjelmasta ja USB-väyläisestä CAN-adapterista. Hyödyntämällä Epeciltä löytyviä tuotteita ja työkaluja erillisiä laitehankintoja ei tarvitse tehdä ja kustannukset saadaan pidettyä matalana. Testilaitteisto pyritään pitämään mahdollisimman yksinkertaisena, jolloin varsinaiset mittaukset ja ohjausyksiköiden lähtöihin kytkettävät oheislaitteet jäävät testaajan huolehdittavaksi tämän opinnäytetyön ulkopuolelle.

Testausohjelma tullaan kehittämään joko tietokoneen Windows-käyttöjärjestelmälle tai Epecin sulautetulle näyttöyksikölle CODESYS-ohjelmointialustalla.

Testausohjelmalla tulee pystyä lukemaan ja ohjaamaan kaikkien Epecin 3000-sarjan ohjausyksiköiden IO-toimintoja, jotka on lueteltu luvussa 3.

Testausohjelman käyttöliittymän tulee olla helposti opittava ja sen tulee ohjata korjaajaa mittausten suorittamisessa, sekä estää mahdollisten virhekytkentöjen syntymistä.

Testausohjelma ei saa poistaa testattavan ohjausyksikön Flash-muistissa olevaa ohjelmaa. Vaihtoehtoisesti, jos ohjelman säilymistä Flash-muistilla ei voida testausohjelmalla varmistaa, se pitää pystyä ottamaan testausohjelmalla talteen ja lataamaan takaisin.

Ohjelman ylläpito ja laajentaminen uusille tuotteille on myös otettava huomioon ohjelman suunnittelussa niin, että olemassa olevia kirjastoja ja funktioita voidaan hyödyntää mahdollisimman monipuolisesti, ja että uusien tuotteiden lisääminen työkaluun olisi mahdollisimman helppoa.

Työkalun varsinainen vaatimusmäärittely löytyy liitteestä 1.

5 Vaihtoehtojen selvitys

Testaus- ja vianhakutyökalun toteuttamisessa on kaksi ratkaistavaa ongelmaa: millä menetelmällä työkalun käyttöliittymä ja ohjelma toteutetaan, ja miten testattavan ohjausyksikön I/O-pinnejä saadaan luettua ja ohjattua CAN-väylän kautta. Vaatimusmäärittelyn perusteella työkalun toteuttamiseksi mietittiin vaihtoehtoisia toteutustapoja, joiden soveltuvuutta tarkastellaan seuraavaksi.

1. Testattavan ohjausyksikön ohjaaminen CAN-väylän kautta sulautetulla näyttöyksiköllä, lataamalla ohjausyksikköön CODESYS 2.3 -sovellus, tai käyttämällä hyödyksi ohjausyksiköiden RAM-muistille ladattavaa testerisovellusta. Näyttöyksikön sovellus ja käyttöliittymä tehdään CODESYS-ohjelmistoalustalla.

2. Testattavan ohjausyksikön ohjaaminen tietokone-ohjelmalla CAN-väylän kautta, käyttämällä hyödyksi ohjausyksiköiden RAM-muistille ladattavaa testerisovellusta.

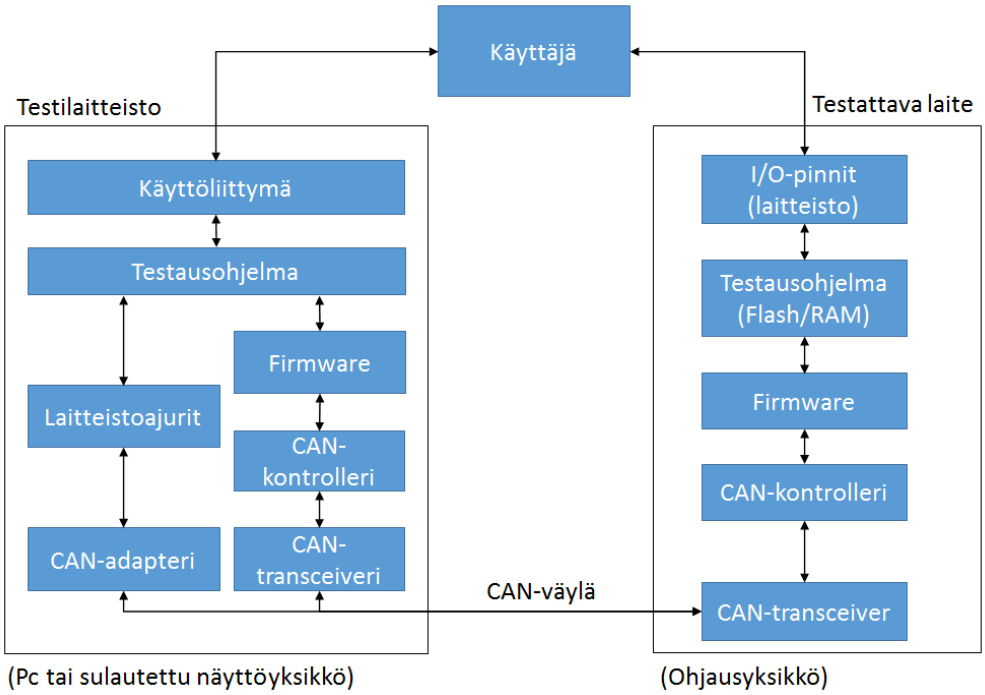
3. Testattavan ohjausyksikön ohjaaminen CAN-väylän kautta sulautetulla näyttöyksiköllä, käyttämällä hyödyksi ohjausyksikön Firmwaren Slave-ominaisuutta. Näyttöyksikön ohjelman toteutus tehdään CODESYS-ohjelmistoalustalla.

4. Testattavan ohjausyksikön ohjaaminen tietokoneen CODESYS-ohjelmalla CAN-väylän kautta, käyttämällä hyödyksi CODESYS-ohjelman debug-ominaisuutta.

5.1 Käyttöliittymä ja testausohjelma

Työkalun käyttöliittymän tarkoituksena on toimia tulkkina testilaitteen ohjelman ja testaajan välissä. Käyttöliittymän avulla korjaajan on helppo ohjata useita ohjausyksikön I/O-pinnejä samanaikaisesti päälle ja pois ilman, että hänen tarvitsee tietää miten se käytännössä tapahtuu.

Työkalun testausohjelman tarkoituksena on toimia puolestaan tulkkina käyttöliittymän ja testattavan ohjausyksikön välissä. Testausohjelma pitää sisällään CAN-väylän käyttöön liittyvät toiminnot sekä määrittelyt, mitä I/O-pinnejä eri ohjausyksiköissä on ja miten niitä ohjataan ja luetaan CAN-väylän avulla. Testaus- ja vianhakutyökalun toimintaperiaate on kuvattu kuviossa 14.

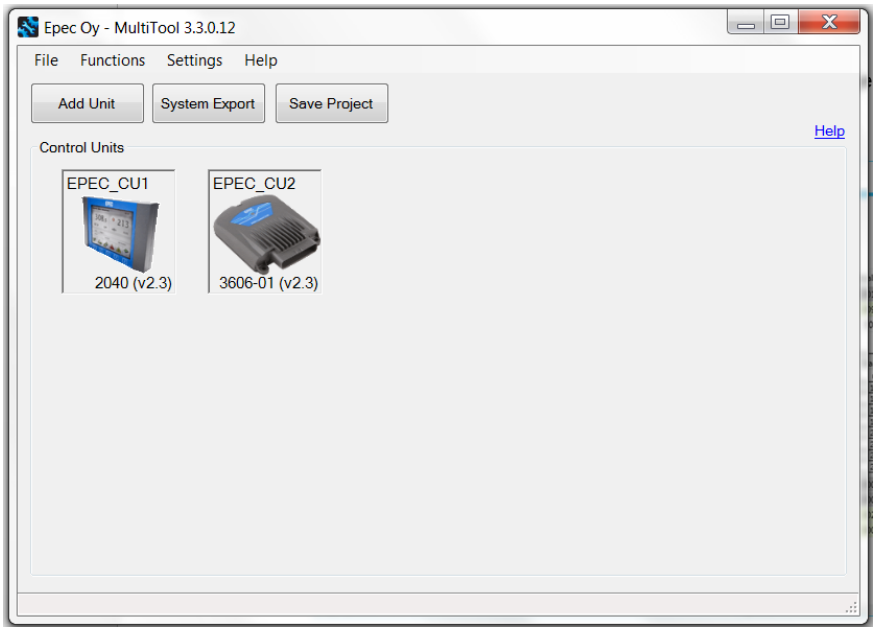


Kuvio 14. Testaus- ja vianhakutyökalun yksinkertaistettu rakenne

5.2 Sulautettu näyttöyksikkö

Työkalun ohjelman ja käyttöliittymän tekemiseksi Epecin sulautetulle näyttöyksiköllе ainoa järkevä vaihtoehto on käyttää hyväksi Epecin MultiToolia ja CODESYS-ohjelmointialustaa.

MultiTool on Epecin valmistama graafinen järjestelmän suunnittelu- ja konfigurointityökalu, jolla määritellään Epecin ohjausyksiköille CODESYS-ohjelmaan tarvittavat alustukset kommunikointi- ja I/O-rajapinnalle (Epec 2014c). Kuviossa 15 on kuvakaappaus Multitoolilla aloitetusta projektista, johon on lisätty samaan CAN-verkkoon yksi näyttöyksikkö ja ohjausyksikkö.



Kuvio 15. Epecin Multitool

5.2.1 2040- tai 6107-näyttöyksikkö

Sulautetulle näytölle löytyy Epecin tuotevalikoimasta kaksi vaihtoehtoista näyttöyksikköä.

2040 on vanhempaa sukupolvea oleva 5,7 tuuman näyttöyksikkö, jonka ohjelmointi voidaan tehdä C-ohjelmointikielellä tai CODESYS 2.3 -versiolla. CODESYS 2.3 -versioon löytyy Epecin SDK-paketista kattava kirjasto valmiita toimintoja ja funktioita, joilla testaustyökalu voidaan toteuttaa. Näytön käytöstä sekä sille tehdyistä ohjelmakirjastoista on myös yleisesti paljon kokemusta talon sisällä, jonka vuoksi 2040-näyttöyksikkö olisi hyvä valinta työkalulle. Näyttöyksikön huono puoli on se, että sitä voidaan käyttää ainoastaan sen etupaneelin napeista tai yksikköön erikseen liitettävistä ohjauslaitteista. Tästä syystä 2040-näyttö-yksiköllä toteutetun työkalun käytettävyys ei olisi kuitenkaan parhain mahdollinen. Kuviossa 16 on kuva 2040-näyttöyksiköstä.

Kuvio 16. Epecin 2040-näyttöyksikkö (Epec 2014b, 14)

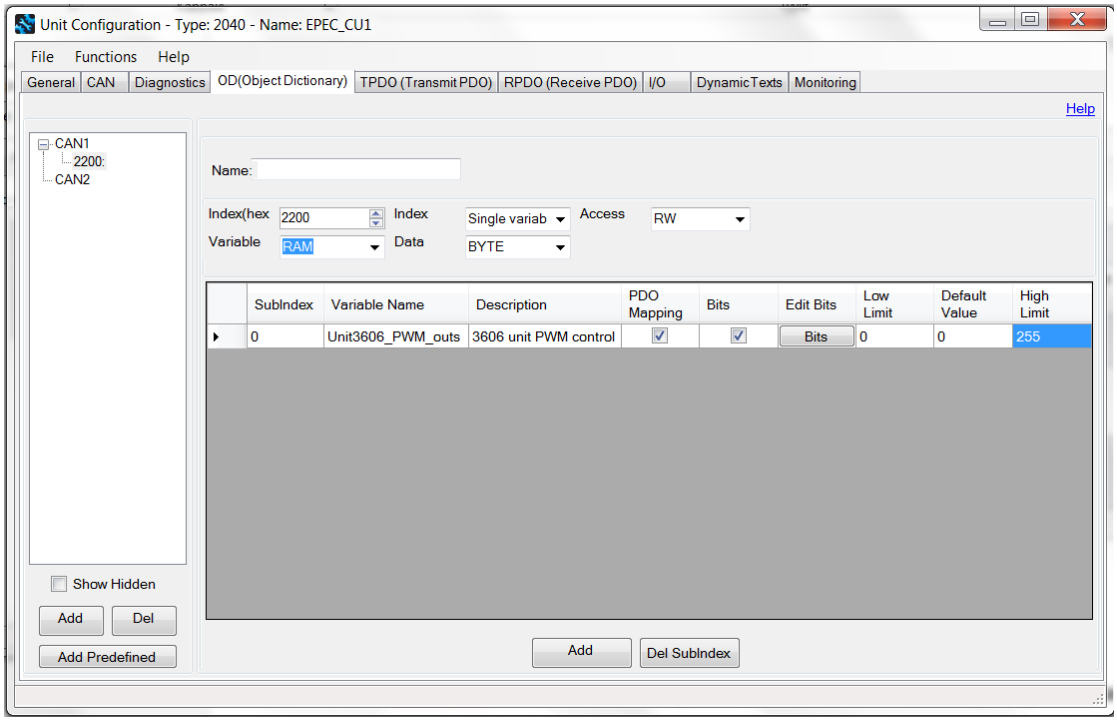
Käytettävyydeltään 2040-näyttöyksikköä parempi olisi Epecin uusi 6107-näyttöyksikkö, jossa on isompi ja käyttöystävällisempi 7 tuuman kosketusnäyttö, sekä huomattavasti enemmän laskentatehoa. 6107:n ohjelmointialustana toimii puolestaan CODESYS:in 3.5 versio, jolle ei ole vielä ihan yhtä kattavaa ohjelmisto-kirjastoa kuin 2040-näyttöyksikölle. Kuviossa 17 on kuva 6107-näyttöyksiköstä.

Kuvio 17. Epecin 6107-näyttöyksikkö (Epec 2014b, 4)

Molemmat näyttöyksiköt täyttävät kuitenkin työkalun tarvitsemat vaatimukset tekniseltä osa-alueelta ja niiden eroavaisuudet ovat lähinnä käytettävyydessä ja CODESYS-ohjelmaversion tuomissa eroavaisuuksissa. Molempien näyttöjen käyttöliittymän suunnitteluun löytyy CODESYS-ohjelmasta oma työkalu CODESYS-visualization, joka mahdollistaa kaikkien huoltotyökalussa tarvittavien käyttöliittymäelementtien ja toimintojen tekemisen. Näin ollen huoltotyökalun käyttöliittymän tekemiseen ei tarvita muita ohjelmia.

5.2.2 CODESYS-testiohjelma

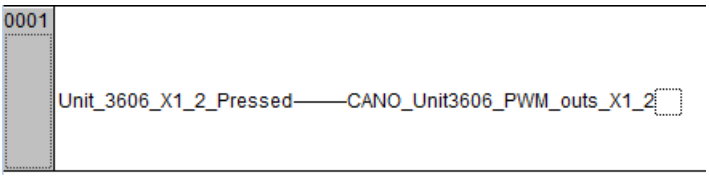
CODESYS-sovelluksilla toteutettuun testaustyökaluun tarvitaan oma CODESYS-sovellus työkaluna käytettävään näyttöyksikköön, sekä jokaiseen ohjausyksikön malliin, jota näyttöyksiköllä on tarkoitus pystyä testaamaan. Sovellusten lisäksi jokaisen ohjausyksikön pinnien ohjaukseen tarvittavat PDO-objektit sekä yksiköiden node-ID:t tulee määrittää ohjausyksiköiden sekä näyttöyksikön objektikirjastoon. Objektikirjaston ja PDO-objektien määrittäminen onnistuu helposti Multitool-ohjelmalla. Objektikirjaston määrittämisen lisäksi Multitoolilla saadaan tehtyä CODESYS-sovellusta varten tarvittavat alustukset ohjausyksikön I/O-pinneille sekä CAN-väylälle. Kuviossa 18 on kuvakaappaus Multitoolista, jossa määritetään lähetettävää TPDO-objektia objektikirjastoon 3606-ohjausyksikön PWM-ulostuloja varten.



Kuvio 18. Objektikirjaston indeksien määrittäminen Multitoolilla

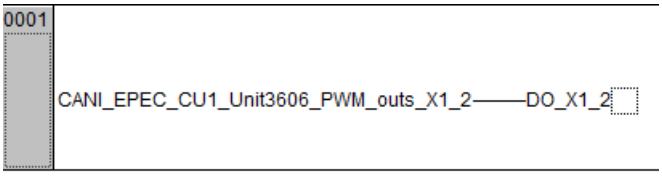
Määrittelyn jälkeen muuttujat viedään Multitoolista sekä ohjausyksikön että näyttöyksikön CODESYS-sovellukseen, jossa ne liitetään joko ohjausyksikön I/O-pinniä ohjaavaan rekisteriin tai haluttuun näyttöyksikön käyttöliittymän elementtiin. Kuviossa 19 on esimerkki näyttöyksikön Main-ohjelmaan FBD-kielellä tehdystä

funktiosta, jolla ohjataan näyttöyksikön kanssa samaan CAN-väylään kytketyn 3606-ohjausyksikön X1-liittimen digitaalilähtöä pinnissä 2. Tieto ohjauksesta välitetään näyttöyksiköltä 3606-ohjausyksikölle Unit3606_PWM_outs-kehyksen X1_2-bitillä, joka on tallennettu objektikirjaston indeksiin 2200.



Kuvio 19. CAN-viestin lähettäminen näyttöyksiköltä 3606-ohjausyksikölle

Testattavan 3606-ohjausyksikön Main-ohjelmaan tehdään vastaavanlainen funktio, joka kytkee X1-liittimen 2-pinnin DO-lähdön päälle, kun näyttöyksikön lähettämän Unit3606_PWM_outs-kehyksen X1_2-bitin tila muuttuu aktiiviseksi. Funktion esimerkki on kuvattu kuviossa 20.



Kuvio 20. CAN-viestin vastaanottaminen 3606-ohjausyksikössä

Analogitulon lukeminen 3606-ohjausyksiköltä tehdään vastaavasti, kuten digitaalilähdön ohjaaminen tehtiin, mutta sen funktiot toteutetaan toisin päin. Kuten digitaalilähtöjä varten, analogituloa varten täytyy myös määritellä muuttuja objektikirjastoon, minkä avulla tieto lähetetään ohjausyksiköltä näyttöyksikölle.

CODESYS-ohjelmalla tehty sovellus on aina ohjausyksikkömallikohtainen, joten jokaiselle testattavan ohjausyksikön mallille täytyy tehdä oma CODESYS-sovellus, joka ladataan näyttöyksiköltä ohjausyksikölle. Kun jokaisen mallin ohjelmat on saatu valmiiksi ja käännetyksi, ne tallennetaan CODESYS-ohjelmasta binääriseksi tiedostoksi ja viedään näyttöyksikölle USB-tikun avulla.

CODESYS-ohjelman latauksessa on huomioitava myös ohjausyksikön node-ID. Jotta näyttöyksikön ja testattavan ohjausyksikön ohjelmat toimisivat keskenään,

niiden on kummankin lähetettävä CAN-viestit oikealle node-ID:lle. CODESYS-ohjelmaa tehtäessä ohjausyksikölle valitaan Multitoolin avulla sen asetuksista haluttu node-ID. Tämän jälkeen, jos ohjelma ladataan ohjausyksikölle, jonka node-ID on eri, se ei ohjelmasta huolimatta osaa vastata näyttöyksikön lähettämiin käskyihin tai kyselyihin, koska se ei tunnista niitä omiksi. Näin ollen ennen ohjelman latausta näyttöyksiköllä on muutettava myös testattavan ohjausyksikön node-ID oikeaksi. Tämä onnistuu ohjelmakirjastoista löytyvällä PARAMETERS_Edit (FB) -funktiolla, jolla pystyy lukemaan ohjausyksikön parametrit sekä muuttamaan niitä (Epec 2014d).

CODESYS:illa tehdyn testiohjelman lataaminen näyttöyksiköltä testattavan ohjausyksikön Flash-muistille onnistuu Epecin ohjelmakirjastoista löytyvällä SWD_APPLICATION_C23 (FB) -funktiolla (Epec 2014d). Vastaavia toimintoja on tehty asiakkaille, jotka käyttävät näyttöyksikköjä kentällä niin sanottuina huolto-näyttöinä, joilla voidaan päivittää koneen ohjausyksiköiden ohjelmat liittämällä näyttöyksikkö koneen CAN-väylään. Testiohjelman latauksen jälkeen ohjausyksikkö käynnistetään uudelleen, jolloin sille ladattu testiohjelma käynnistyy. Tämän jälkeen ohjausyksikön I/O-pinnit ovat käytettävissä näyttöyksiköltä, niin kuin ne on testiohjelmassa määritelty.

5.2.3 Näyttöyksikön soveltuvuus työkaluksi

Työkalun toteuttaminen CODESYS-sovellusten ja näyttöyksikön avulla on selvityksen perusteella melko vaivaton toteuttaa Multitoolin ja valmiiden ohjelmakirjastojen funktioiden avulla. Testisovelluksen lataaminen ohjausyksikön RAM-muistille sen sijaan tuottaa näyttöyksiköllä ongelmia. Muun muassa tutkimustyön testiyksilönä käytettävän 3606-ohjausyksikön RAM-muistin koko on vain 138 kB, josta sovellusta ja sen muuttujia varten on käytettävissä 112 kB. Tarvittavat toiminnot kattavaa CODESYS-sovellusta ei ole mahdollista tehdä näin pienelle muistille. Lisäksi Epecin SDK-paketissa ei ole valmista CODESYS-kirjastoa, jolla sovelluksen saisi ladattua näytöltä ohjausyksikön RAM-muistille Flash-muistin sijaan. Tästä syystä myöskään testerisovelluksen käyttö näyttöyksiköllä toteutetussa työkalussa ei ole mahdollista.

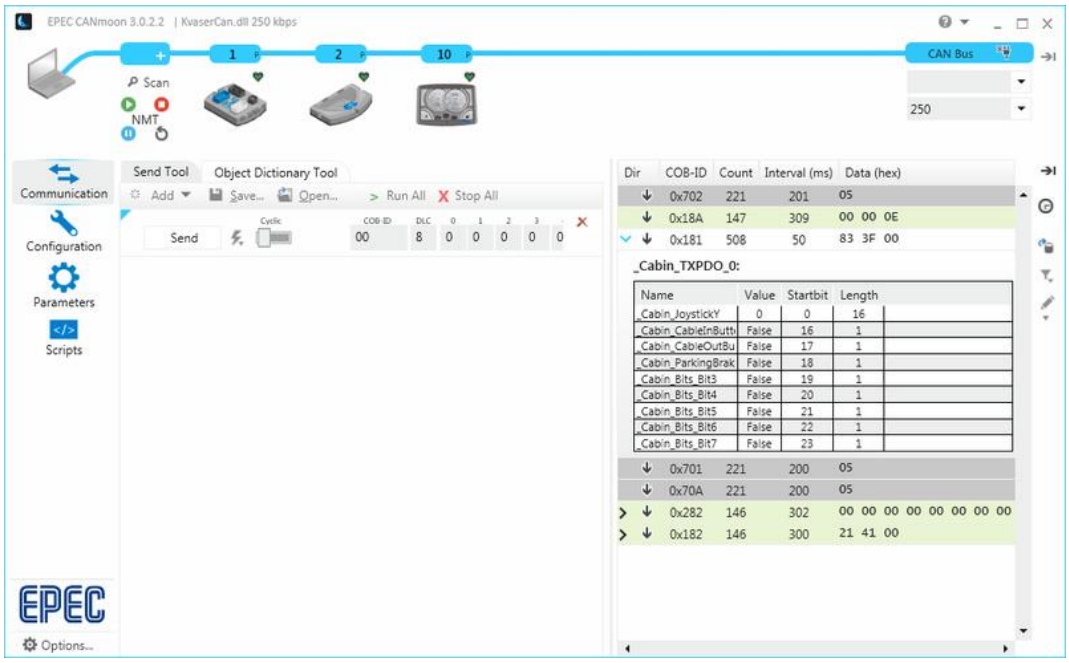
Koska asiakassovelluksen säilyttäminen ohjausyksikön Flash-muistilla on yksi työkalulle asetetuista vaatimuksista, työkalua ei voida toteuttaa vaatimusmäärittelyn mukaisesti näyttöyksiköllä ja CODESYS-sovelluksella. RAM-muistille lataamisen puute ei haittaisi työkalun toteuttamista, mikäli ohjausyksikön sovellus saataisiin ladattua näyttöyksiköllä talteen, mutta Epecin ohjelmakirjastosta ei löydy tällaista funktiota valmiiksi, eikä sellaisen tekeminen opinnäytetyön puitteissa ole mahdollista.

5.3 Windows-pohjainen työkalu

Toinen vaihtoehto testaustyökalun toteuttamiseen on liittää testattava ohjausyksikkö tietokoneeseen USB-liitännäisen CAN-adapterin avulla ja tehdä ohjausyksikön ohjaamiseen tarvittava ohjelma ja käyttöliittymä Windows-käyttöjärjestelmälle. Windowsille vaihtoehtoisia ohjelmointialustoja löytyy lukuisia.

Microsoftin Visual Studio on Windows-käyttöjärjestelmille, sekä web- ja mobiilialustoille suunnattu ohjelmankehitysympäristö (Microsoft 2014). Visual Studio on monipuolinen ohjelmointiympäristö, joka soveltuisi varmasti parhaiten ohjelman tekemiseen, jos sen toteutus aloitettaisiin täysin tyhjältä pöydältä. Kokonaan uuden PC-ohjelman tekemisessä tulee ottaa kuitenkin huomioon I/O-pinnien ohjauksen lisäksi myös CAN-väylän ja CAN-adapterin käsittelyyn liittyvät toiminnot, jotka esimerkiksi Multitool tuo CODESYS:illa tehtyyn ohjelmaan automaattisesti. Näin ollen kokonaan uuden ohjelman tekeminen CODESYS-ohjelmaan verrattuna tulisi kuluttamaan paljon enemmän aikaa Visual Studiolla. Tästä syystä järkevämpää olisikin hyödyntää jo olemassa olevaa ohjelmaa, Epecin tekemää CANmoonia.

CANmoon on CANopen-protokollaan perustuva ohjelmistotyökalu Windows-käyttöjärjestelmälle. Sitä voidaan käyttää CAN-väylän laitteiden valvontaan, vianmääritykseen, sovelluksien lataukseen sekä firmware-ohjelmistojen päivitykseen Epecin valmistamille ohjausyksiköille. (Epec 2014b, 15.) CANmoon-ohjelman näkymä on kuvattu kuviossa 21.



Kuvio 21. CANmoon-ohjelman käyttöliittymä (Epec 2014b, 15)

Koska CANmoon on CAN-väylän diagnosointiin tehty ohjelma, siinä on itsessään valmiiksi kaikki CAN-adapterin ja CAN-väylän käsittelyyn liittyvät toiminnot, joita työkalua varten tullaan tarvitsemaan. CANmooniin on myös mahdollista lisätä omia käyttöliittymiä ja ohjelmia Python-skriptien avulla, mikä mahdollistaa työkalun toteuttamisen CANmoonin aliohjelmana.

Python on oliopohjainen ohjelmointikieli, jota suoritetaan suoraan lennosta Python-suoritusympäristössä. Toisin kuin monet muut ohjelmointikielet, Python-kielen kääntäminen laitteistolle tapahtuu ohjelmakoodin ajon aikana, mikä mahdollistaa muuttujien, metodien ja ohjelmaluokkien vapaamman käytön muihin ohjelmointikieliin verrattuna. (Kokkarinen 2004, 253–254.)

CANmooniin on mahdollista toteuttaa kolmenlaisia ohjelma-skriptejä: Basic, StartUp ja StartUpOverride.

- Basic-tyypin ohjelma-skripti käynnistetään aina erikseen CANmoonin käyttöliittymästä.
- StartUp-tyypin skripti käynnistyy automaattisesti, kun CANmoon käynnistetään.

- StartUpOverride-skripti käynnistyy myös automaattisesti CANmoonin käynnistyessä, mutta se korvaa myös samalla CANmoonin käyttöliittymän skriptiin tehdyllä käyttöliittymällä. (Epec [Viitattu 20.8.2014].)

Ohjelma-skriptit voidaan kirjoittaa millä tahansa tekstieditorilla Python-ohjelmakoodin syntaksin mukaan ja tallentaa tämän jälkeen Pythonin tiedosto-päätteen mukaiseen muotoon. Kun ohjelma-skripti on tallennettu CANmoon-ohjelman Scripts-kansioon, se tulee automaattisesti sisällytetyksi CANmoonin ohjelma-skripteihin. Lisäksi CANmoon suorittaa automaattisesti kaikki kansiossa olevat StartUp-tyyppiset skriptit, kun CANmoon käynnistetään. Kuviossa 22 on esitetty yksinkertainen CANmoonin esimerkki-skripti, joka suoritettaessa avaa näytölle perinteisen Windows-käyttöjärjestelmästä tutun MessageBox-ikkunan.

MBView.py

```
1  import clr
2  clr.AddReference('System.Windows.Forms')
3
4  from System.Windows.Forms import (
5      MessageBox
6  )
7  ScriptName = 'StartUp MessageBox'
8  Description = 'Shows a messageBox on application startup'
9  ScriptType = 'StartUp'
10
11 def RunScript(n):
12     MessageBox.Show('This message came from Script!')
13     return 1;
14
```

Kuvio 22. CANmoonin esimerkki-skripti

Pelkällä Python-ohjelmointikielellä ja Windows Forms -kirjaston elementeillä toteutettu käyttöliittymä ei kuitenkaan sovellu monimutkaisemman käyttöliittymän ja testaustyökalun tekemiseen. Tähän tarkoitukseen sopii paremmin Microsoftin kehittämän .NET Framworkin tarjoamat komponentit, joiden avulla pystytään esittämään tehokkaasti grafiikkaa eri ohjelmointikielillä.

5.3.1 IronPython ja .NET Framework

.NET Framework koostuu ohjelmistokomponenttikirjastosta sekä ohjelmakoodin ajoympäristöstä CLR (Common Library Runtime), joka yhdistää eri ohjelmointikielillä toteutetut ohjelmat toisiinsa ja muuntaa ne tietokoneen laitteistolle sopivaan muotoon (Microsoft 2015a). .NET-yhteensopivasta Python-ohjelmointikielestä käytetään nimitystä IronPython.

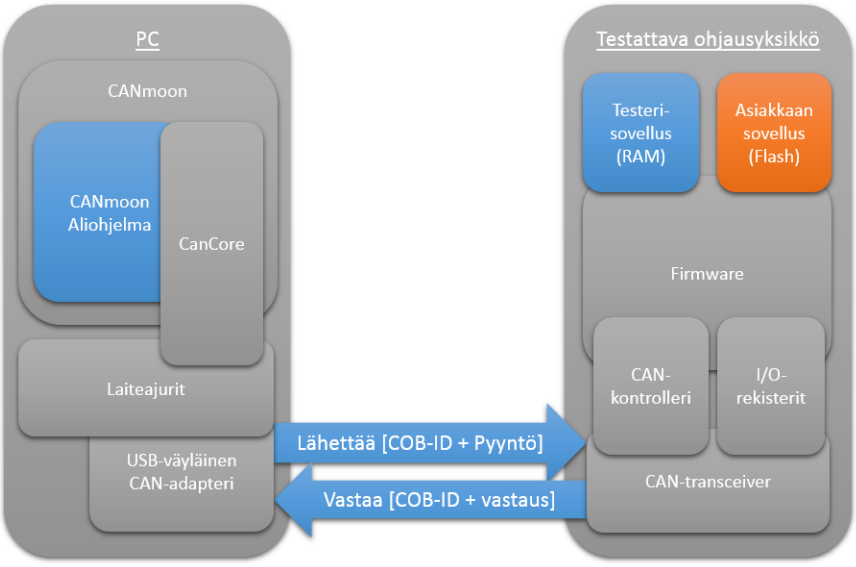
IronPython on avoimen lähdekoodin toteutus Python-ohjelmointikielestä, joka on integroitu Microsoftin .NET Frameworkiin. Tämä mahdollistaa .NET Framework -ohjelmistokomponenttikirjastojen käytön Python-kielessä perinteisten Python-kirjastojen lisäksi. (IronPython [Viitattu 14.1.2015].) Työkalun toteutuksen kannalta IronPythonin tärkein ominaisuus on nimenomaan sen .NET Frameworkin tarjoamat ohjelmistokomponentit käyttöliittymän tekemiseen. Varsinainen käyttöliittymä toteutetaan Python-kielen sijaan XAML-ohjelmointikielellä (Extensible Application Markup Language), joka on Microsoftin kehittämä ohjelmointikieli WPF-pohjaisille käyttöliittymille.

WPF (Windows Presentation Foundation) taas on Microsoftin kehittämä grafiikan esitysjärjestelmä perinteisille sekä selainpohjaisille ohjelmille. WPF:n ydin on resoluutiosta riippumaton vektoripohjainen grafiikkamoottori, jolla toteutetaan ohjelman graafinen ulkoasu. Sen avulla pystytään esittämään ohjelmaa suorittavan laitteen näytöllä muun muassa tekstiä, kuvia, animaatioita sekä 2D- ja 3D-grafiikkaa. WPF kuuluu valmiiksi Microsoftin .NET Frameworkiin ja on näin ollen yhteensopiva sen muidenkin ohjelmistokomponenttikirjastojen kanssa. WPF sisältää grafiikkamoottorin lisäksi myös ohjelman ja käyttöliittymän yhdistämisen mahdollistavat luokat ja sidokset (Bindings), sekä itse käyttöliittymän ulkoasuun kuuluvat komponentit. (Microsoft 2015b.) Visual Studiolla tehdystä WPF-käyttöliittymästä saadaan suoraan tallennettua XAML-tiedosto, joka sisältää kaikki käyttöliittymän elementit ja elementtien ominaisuudet. IronPythonin ja .NET Framework -kirjastojen avulla XAML-tiedostoa voidaan käyttää CANmoonin ohjelma-skripteissä.

5.3.2 Testerisovellus

Jokaiselle Epecin ohjausyksiköllе löytyy elektroniikkatuotannosta sen toimintojen testaamista varten rakennettu testeri eli testausasema. Asema lataa ohjausyksikön prosessorin sisäiseen RAM-muistiin sovelluksen, jonka avulla ohjausyksikön I/O-pinniä, muistia, dataväyliä ja muita toimintoja pystytään ohjaamaan ja lukemaan CAN-väylän kautta (Epec 2013b, 4–5). Testattavat toiminnot on kirjoitettu jokaiselle ohjausyksiköllе tehtyihin testijonoihin, joissa määritellään ohjaukseen tarvittavat CAN-viestit, sekä testerin ohjelman sekvenssit. Näiden avulla testerillä käytettävään ohjelmaan on määritelty milloin mitäkin ohjausyksikön prosessorin porttia tai toimintoa ohjataan päälle ja pois, ja miten toimintoja mitataan pinneistä testerin ohjelmallisesti ohjattavilla mittalaitteilla.

Testerin ohjelman perusperiaate onkin yksinkertaistettuna vain CAN-viestien lähettämistä ja vastaanottamista, kun ohjausyksiköiden automaattiset pinnien mittaukset ja jännitteiden syötöt jätetään pois. CANmoonista löytyy valmis ohjelmistokomponenttikirjasto CanCore, joka tarjoaa riittävät toiminnot vastaavan ohjelman toteuttamiseen. CanCore-kirjasto tarjoaa muun muassa valmiit luokat CAN-viestien lähettämiseen ja vastaanottamiseen, jotka ovat testerisovelluksen osalta kaksi tärkeintä toimintoa. Kuviossa 23 on esitetty testerisovelluksen ja CANmoonin aliohjelmana toteutettavan testaustyökalun toimintaperiaate.



Kuvio 23. CANmoonin aliohjelmana toteutetun työkalun toimintaperiaate

Testerisovelluksen käyttäminen CANmoonin skriptillä toteutetussa ohjelmassa alkaa ensin sovelluksen lataamisesta testattavalle ohjausyksikölle CAN-väylän kautta. Testiohjelman lataamiseen voidaan käyttää samaa CanLoader-ohjelmaa, jota käytetään tuotannon automaattitestereillä. CanLoader hoitaa koko lataus-prosessin CAN-väylän alustamisesta latauksen onnistumisen tarkistamiseen ja lopulta testerisovelluksen käynnistämiseen.

Kun ohjelma on saatu ladattua ohjausyksikköön, se on heti valmis vastaan-ottamaan komentoja CAN-väylän kautta. CANmoonin ohjelma-skriptissä käskyjen lähettäminen toteutetaan CanCore-kirjaston CAN-luokan Transmit-metodilla (Epec [Viitattu 20.8.2014]). Metodin käyttäminen yksittäisen digitaalilähdön päälle kytkemiseksi on kuvattu kuviossa 24.

```
def SetOutputON(self, sender, args):
    cobID = System.UInt32(2015)
    data = System.Array[System.Byte]((0x05,0x13,0x00,0x08,0x00,0x00,0x00,0x00))
    CAN.Transmit(cobID, data)
```

Kuvio 24. CAN-viestin lähettäminen IronPython-skriptillä

Transmit-metodia varten tarvitaan kaksi parametria, vastaanottavan solmun COB-ID, jonka määrittelee testerisovellus, sekä ByteArray-taulukkona lähetettävä data. Lähettävän datan pituus ja sisältö vaihtelee ohjattavasta portista ja toiminnosta riippuen. Transmit-metodilla lähetetty CAN-viesti on aina kahdeksan tavun mittainen, joten siksi metodille ei tarvitse kertoa data-kehyksen pituutta.

Mikäli testerisovellukselle lähetettävällä viestillä halutaan kysyä ohjausyksikön jonkin pinnin, kuten esimerkiksi analogitulon tilaa, saapuvia viestiä varten tehdään oma kuuntelija-luokka, joka rekisteröi testerisovelluksen lähettämät CAN-viestit. Kuviossa 25 on esitetty yksinkertainen CAN-väylää ”kuunteleva” luokka, joka tallentaa lähettäjän COB-ID:n, sekä saapuneen viestin tekstimuodossa omiin muuttujiinsa.

```
class Receiver():
    def __init__(self):
        handler = EventHandler[MessageEventArgs](self.receive)
        CAN.RegisterMessageListener(handler)

    def receive(self, sender, args):
        cobId = args.Message.COB_ID.ToString()
        datamessage = args.Message.DataString.ToString()
```

Kuvio 25. CAN-viestin vastaanottaminen IronPython-skriptillä

5.3.3 CANmoonin skriptin soveltuvuus työkaluksi

Testerisovelluksen käyttäminen ohjausyksikön rajapintana tarjoaa monia valmiita ratkaisuja työkalun toteuttamiseksi. Testerisovelluksesta voidaan hyödyntää suoraan sellaisenaan testerisovelluksen latausohjelma, sekä jokaisen ohjausyksikön testerisovellus, jolloin erillistä ohjelmaa testattaville ohjausyksiköille ei tarvitse tehdä lainkaan. Näiden lisäksi testerisovelluksen käyttämisessä on myös muita etuja, jotka tekevät testerisovelluksen käyttämisestä varteenotettavan vaihtoehdon. Näitä ovat muun muassa se, että:

- Testerisovellus kattaa kaikki ohjausyksikön elektroniikan testaamiseen tarvittavat toiminnot.
- Testerisovellukset kuuluvat tuotekehitysprojektien Gate-malliin, joka takaa uusille ohjausyksiköille tulevaisuudessakin vastaavanlaiset testerisovellukset.
- Sovellus ladataan testattavan ohjausyksikön RAM:lle, jolloin huoltoon tulevan ohjausyksikön Flash-muisti pysyy koskemattomana.

Testereille määritettyjä CAN-viestiä ei voi kuitenkaan käyttää sellaisenaan CANmoonille tehtävässä ohjelmassa, koska ne on tarkoitettu suoritettavaksi ennalta määrätyssä järjestyksessä. Testerisovelluksella toteutetun työkalun haasteet liittyvätkin CANmoonin ohjelma-skriptinä toteutetun ohjelman tekemiseen, näistä merkittävin haaste on toimintojen aktivoimiseen tarvittavien CAN-viestien määrittely ja hallinnointi ohjausyksikön prosessorin porttien tasolla. Siinä missä CODESYS:illa tehdyssä ohjelmassa Multitool määrittelee mitä ohjausyksikön

pinniä milläkin prosessorin portilla ohjataan, testerisovellusta käytettäessä tämä pitää tehdä CANmoonin ohjelma-skriptiin alusta alkaen itse.

5.4 Firmwaren Slave-tuki

Kolmas vaihtoehto toimintojen aktivoimiseen olisi käyttää hyödyksi CANopen Slave pohjaista Firmwarea. CANopen Slave yksikkö ei tarvitse ohjelmaa I/O-pinnien ohjaamiseen, vaan koko ohjausyksikön ohjaus voidaan suorittaa samaan väylään liitetyllä Master yksiköllä (Epec 2014e). Ohjaus toimii käytännössä kuten testerisovellus, mutta CAN-väylän kautta tulevat ohjaukset käännetään prosessorille ohjelman sijaan testattavan ohjausyksikön Firmwaressa.

Työkalun toteuttaminen Firmwaren avulla on vaihtoehdoista kaikkein lähimpänä ohjausyksikön rautaa, eli toisin sanoen sen mikroprosessoria, sillä työkaluun tarvittavat toiminnot ohjelmoitaisiin suoraan prosessorille tulkattavalla C-kielellä. Toteutuksessa olisi mahdollista käyttää jonkin verran ohjausyksiköiden nykyisiä Firmwareja pohjana, mutta käytännössä jokainen ohjausyksikkö vaatisi kokonaan uuden Firmwaren tekemisen. Vastaavanlainen Slave-tuki on tällä hetkellä saatavilla ainoastaan Epecin 2038 Slave -ohjausyksikölle, jonka Firmwarea ei voida hyödyntää uudemman sukupolven 3000-, 4000- ja 5000-sarjan ohjausyksiköissä.

Jokaisen Firmwaren lisäksi työkaluun tarvittaisiin näyttöyksikkö ja näyttöyksikölle tarvittavat toiminnot mahdollistava CODESYS-ohjelma. Ohjausyksikön saapuessa huoltoon näyttöyksiköllä päivitettäisiin sille ensin Slave-yhteensopiva Firmware, jonka jälkeen ohjausyksikön I/O-pinnejä ohjattaisiin CAN-väylän kautta.

Slave-Firmwaren toteuttaminen ohjausyksiköille on työkalun toteutustavoista selvästi vaativin. Toteutustavan etuna olisi kuitenkin se, että sitä varten kehitettävää Firmwarea voitaisiin hyödyntää myös Epecin asiakkaiden käytettäväksi.

5.5 Suora ohjaus CODESYS-ohjelmistolla

Yksinkertaisimmillaan työkalun toteuttaminen saattaisi onnistua pelkästään CODESYS-ohjelmointityökalun Debug-toimintoa käyttämällä. Testattavalle

ohjausyksikölle tehdään ensin konfigurointi Epecin Multitoolilla, jolla määritetään mitä pinnejä ohjausyksiköstä halutaan käyttää ja miten ne asetetaan. Tämän jälkeen CODESYS:illa tehdään tyhjä sovellus, joka ladataan ohjausyksikköön CAN-väylän kautta suoraan CODESYS:ista. Ohjelman latauksen jälkeen CODESYS:illa muodostetaan yhteys ohjausyksikön ja PC:n välille Login-toiminnolla. Lopuksi ohjausyksikköön ladattu ohjelma käynnistetään, jonka jälkeen kaikki ohjausyksikölle Multitoolilla määritetyt I/O-pinnit ovat luettavissa ja ohjattavissa CODESYS:in Resources-ikkunan IO_RAW-välilehdeltä. Lähtöjen kytkeminen päälle tai pois onnistuu I/O-pinniä vastaavan Boolean-tyyppisen muuttujan arvoa muuttamalla ja pakottamalla tämän jälkeen arvot voimaan Force Values -komennolla. Analogi- ja digitaalitulopinnien tilat puolestaan näkyvät suoraan ikkunassa pinniä vastaavan muuttujan arvossa.

5.5.1 CODESYS-ohjelmointityökalun soveltuvuus työkaluksi

Käyttämällä pelkästään CODESYS-ohjelmointityökalua sekä Epecin Multitoolia, pystytään toteuttamaan kaikkien I/O-pinnien ohjaus, mitä työkalulta vaaditaan. Vaihtoehdon suurin etu on myös se, että ohjelman tekeminen Epecin jokaiselle tuotteille onnistuu muutamassa tunnissa. Vaihtoehto on kuitenkin siitä huono, että se vaatii käyttäjältä hieman tuntemusta tarvittavista työkaluista. Lisäksi tämä menetelmä tuhoaa ohjausyksikön Flashilla olevan ohjelman, niin ettei sitä voida tutkia tai käyttää enää CODESYS:illa tehdyn ohjelman latauksen jälkeen. Vaatimusmäärittelyn mukaan testattava ohjausyksikkö pitää pystyä testaamaan niin, että Flash-muistilla oleva ohjelma ei häviä. Vaihtoehtoisesti ohjelma on voitava ottaa talteen ohjausyksikön Flash-muistilta sen palauttamista varten, mutta tämäkään ei onnistu suoraan CODESYS-ohjelmointityökalulla.

5.6 Toteutustavan valinta

Syyskuussa Epecillä pidetyssä opinnäytetyöpalaverissa käytiin läpi selvityksen pohjalta työkalun toteutukselle soveltuvia vaihtoehtoja. Taulukossa 4 on vertailtu eri vaihtoehtojen välisiä eroja.

Taulukko 4. Työkalun toteutustapojen vertailu

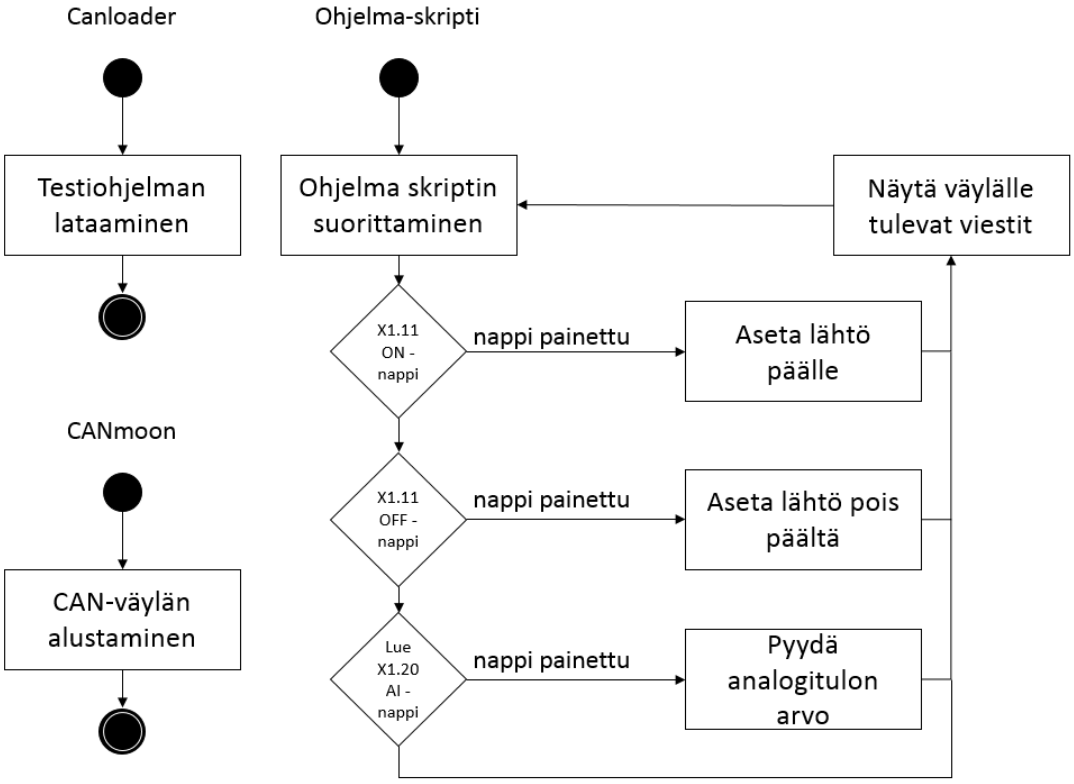
Toteutustapa	**Näyttöyksikkö ja CODESYS-sovellus**	**CANmoon + testerisovellus**	**Firmwaren Slave-tuki**	**CODESYS-ohjelmointityökalu**
Vaadittava työmäärä alussa	Kohtalainen	Suuri	Suurin	Hyvin pieni
Tarvittava materiaali	Näyttöyksikkö, kaapelit	PC, CAN-adapteri, kaapelit	Näyttöyksikkö, kaapelit	PC, CAN-adapteri, kaapelit
Vanhan ohjelman säilyminen muistissa, tai sen talteen ottaminen	Tällä hetkellä ei onnistu kumpikaan vaihtoehdoista	Vanha ohjelma säilyy	Vanha ohjelma säilyy	Tällä hetkellä ei onnistu kumpikaan vaihtoehdoista
Vaadittava työmäärä lisättäessä uusia tuotteita työkaluun	Kohtalainen (Vaatii uuden ohjelman tekemisen molempiin yksiköihin)	Pieni (Testerisovellus valmiina, mutta vaatii prosessorin porttien määrittelyn CANmoonin ohjelmalle)	Ei pystytä tässä vaiheessa vertailemaan. Vaatii ainakin joiltain osin uuden Firmwaren tekemisen.	Hyvin pieni (Vaatii uuden ohjelman)

Vaikka CODESYS-ohjelmointityökalun käyttäminen ohjausyksiköiden testaamiseen olisi työmäärältään ylivoimaisesti pienin, ja siitä huolimatta kattanut jokaisen I/O-pinnien testaamisen tarvittavat toiminnot, työkalua päätettiin lähteä kehittämään testerisovellusta hyödyntäen CANmoonin ohjelma-skriptillä. Toteutustavan edut muihin vaihtoehtoihin verrattuina ovat:

- Olemassa olevien testerisovelluksien hyödyntämisen tuomat edut
- Ohjelmakoodin ja luokkien uudelleen käyttö eri ohjausyksiköiden malleille
- Yhden työkalun tarjoama kokonaisuus, jossa CANmoonin ennestään tarjoamat toiminnot yhdistyvät uuteen työkaluun
- Tällä hetkellä ainut toteutettavissa oleva vaihtoehto, jolla ohjausyksikön Flash-muistilla oleva sovellus jää talteen

6 Suunnitelma työkalun ohjelmalle

Vaihtoehtoja selvitettäessä CANmoonille tehtiin yksinkertainen ohjelma-skripti, jolla saatiin 3606-ohjausyksiköstä ohjattua X1.11-pinnin digitaalilähtö päälle ja pois, sekä luettua X1.20-analogitulopinnin kanavan AD-arvo. Ennen ohjelma-skriptin suorittamista, ohjausyksikköön täytyi ensin ladata erillisellä Canloader-ohjelmalla testisovellus. Tämän jälkeen avattiin CANmoon-ohjelma, josta asetettiin vielä oikeat asetukset CAN-väylän käyttämistä varten. Ohjelman toimintaperiaate on kuvattu kuviossa 26.

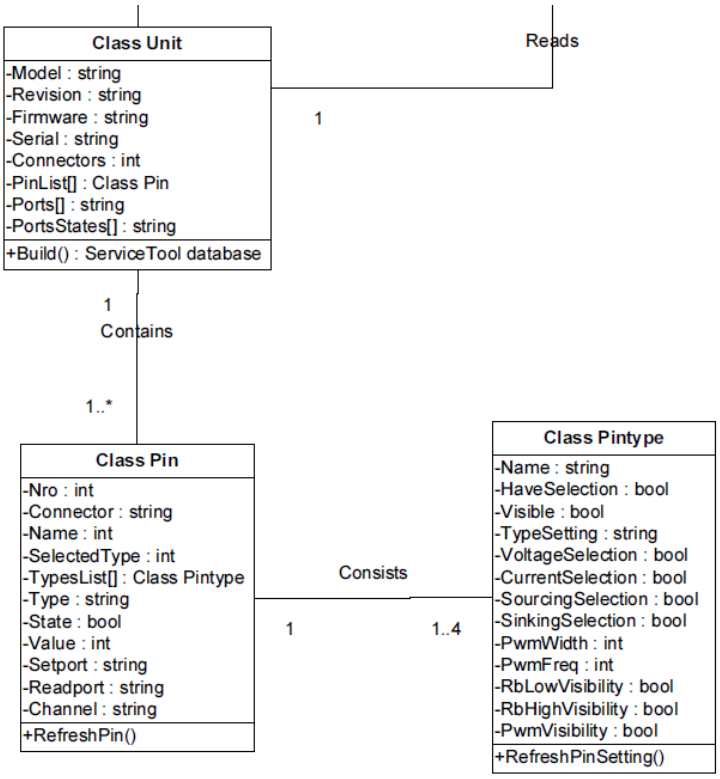


Kuvio 26. Esiselvitystä varten tehdyn testausohjelman toimintaperiaate

6.1 Usean pinnin ohjaus

Selvitystä varten tehdyllä sovelluksella ei pysty kuitenkaan ohjaamaan tai lukemaan kuin yhtä pinniä kerrallaan, ja pinnin on oltava tällöin digitaalilähtönä tai analogitulona. Useamman lähdön ohjaamista varten on huomioitava se, että testerisovelluksen mukaisilla CAN-viesteillä ei ohjata ainoastaan yhtä lähtöä, vaan

viestillä ohjataan prosessorin samassa porttiryhmässä olevaa useampaa porttia. Koska ohjausyksiköllä suoritettavalla testerisovelluksella ei ole mahdollista suorittaa maski-operaatioita vastaanotetun viestin porteille, jokainen kyseisellä CAN-viestillä asetettavista porteista pitää olla etukäteen tiedossa, ettei toisen portin päälle kytkeminen sammuta muita portteja. Tätä varten sovellukseen tehdään olio, jonka avulla pinnejä pystytään hallinnoimaan ja jokaisen pinnin tila voidaan huomioida ennen CAN-viestien lähettämistä. Olion käyttäminen mahdollistaa myös pinnien asetuksien tallentamisen, jolloin pinnejä voidaan ohjata muullakin tavoin, kuin pelkästään digitaalilähtöinä, tai analogitulona. Kuviossa 27 on esitetty ohjelmaa varten tehtävät Unit-, Pin- ja Pintype-luokat.



Kuvio 27. Ohjausyksikön ja pinnien luokkarakenne

Työkalun ylläpitämisen helpottamiseksi Pin- ja Pintype-olioiden parametrit luetaan ulkoisista tiedostoista, niin sanotuista pinnikartoista, joka pitää sisällään tiedot kyseisen CAN-väylälle kytketyn ohjausyksikön mallin pinneistä, pinnien tyypeistä, sekä niihin liittyvistä parametreista. Näin ollen uusille ohjausyksikön malleille riittää, että työkaluun lisätään vain uusi pinnikartta. Pinnin ollessa analogitulona Value-parametriin luetaan CAN-väylältä kyseistä pinniä vastaavan kanavan AD-

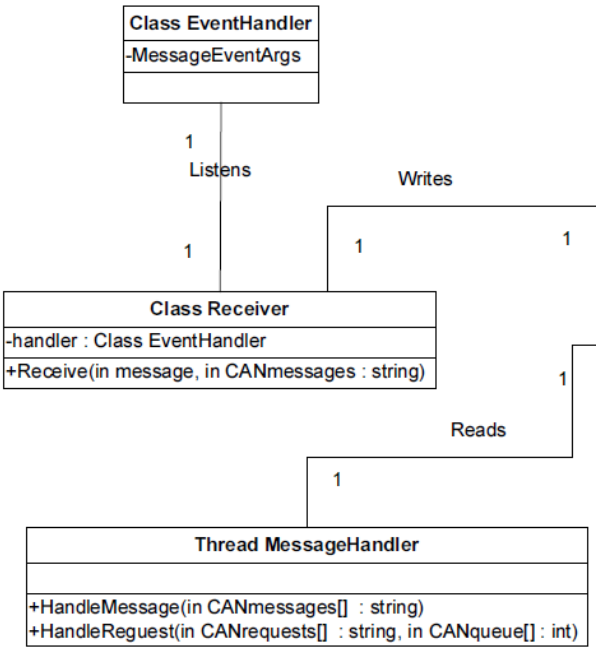
arvo. State-parametri toimii kaikille lähdöksi asetetuille pinneille ohjaavana parametrina, ja se on käytettävissä käyttöliittymän pinnilistan CheckBox-elementistä. Tulopinneillä State-parametri vastaa pinnin portilta luettua tilaa, joka näytetään vastaavasti pinnilistan CheckBox-elementissä. Tällöin CheckBox-elementti on asetettu ReadOnly-tilaan. Unit-olion ja sen Pin-olioita sisältävä pinnilista liitetään DataBindien avulla käyttöliittymän DataGrid-elementtiin, jolloin parametrien ja käyttöliittymän välille saadaan luotua molempiin suuntiin toimiva yhteys. Kuviossa 28 on esitetty pinnilistana toimivan DataGrid-elementin määrittely käyttöliittymää varten tehtävästä XAML-tiedostosta.

```
<DataGrid Name="PinList" ItemsSource="{Binding}" AutoGenerateColumns="False" >
    <DataGrid.Columns>
        <DataGridTextColumn Header="Connector"  Binding="{Binding Connector}" IsReadOnly="True" Width="100"/>
        <DataGridTextColumn Header="Pin"  Binding="{Binding Name}" IsReadOnly="True" Width="100"/>
        <DataGridTextColumn Header="Type" Binding="{Binding Type}" IsReadOnly="True" Width="140"/>
        <DataGridTextColumn Header="Value" Binding="{Binding Value}" IsReadOnly="True" Width="100"/>
        <DataGridCheckBoxColumn Header="State" IsReadOnly="False"
        Binding="{Binding State, NotifyOnSourceUpdated=True, UpdateSourceTrigger=PropertyChanged}" Width="60"/>
    </DataGrid.Columns>
</DataGrid>
```

Kuvio 28. Pinnilistaa vastaavan DataGrid-elementin XAML-koodi

6.2 CAN-viestien hallinnointi

Selvitystyötä varten tehdyssä ohjelmassa on myös ongelma analogitulojen lukemiseen liittyen. Luettaessa analogituloa ohjausyksiköltä, ohjausyksikkö vastaa kyselyyn CAN-viestillä, joka pitää sisällänsä ainoastaan viestityypin COB-ID:n, sekä kyseisen analogitulon kanavan raaka-arvon. Tämän vuoksi useampia analogituloja luettaessa lukeminen on suoritettava ennalta sovitussa järjestyksessä, jotta saapuvat arvot voidaan yhdistää oikeille pinneille. Väylältä luetut analogitulojen CAN-viestit on ensin eroteltava muista väylän viesteistä sen COB-ID:n perusteella. Tämän jälkeen viestin arvo lisätään analogitulojen arvoille tarkoitettuun jonoon, josta ne luetaan omassa taustasäikeessä FIFO-periaatteella kutakin pinniä vastaavan Pin-olion Value-parametriin. Kuviossa 29 on esitetty CAN-viestien vastaanottamista varten tehtävä Receiver-luokka, sekä viestien käsittelyä varten omassa taustasäikeessä ajettava MessageHandler-luokka.



Kuvio 29. CAN-viestien lähetys, vastaanotto ja viestien käsittely

Kun CAN-väylälle tulee viesti, se käynnistää EventHandlerin, joka kutsuu Receiver-olion Receive-metodia. Receive-metodi lisää vastaanotetun viestin globaaliin CANmessages-listaan, josta se on luettavissa MessageHandler-säikeellä. MessageHandler-säikeellä on kaksi tehtävää, käsitellä CANmessages-listaan saapuneita viestejä niiden COB-ID:n mukaan HandleMessage-metodissa, sekä lähettää CAN-väylälle CANrequests-listaan lisättyjä viestejä HandleRequest-metodissa. Lähetettävät viestit, joihin ohjausyksiköltä odotetaan vastausta, lisätään globaaliin CANqueue-listaan. CANqueue-listan sisältöä verrataan väylältä saapuviin viesteihin HandleMessage-metodissa, jossa viestit kohdistetaan toisiinsa. Koska ohjelmassa on kaksi eri säiettä, joista toinen tuottaa sisältöä ja toinen kuluttaa sitä, mahdollisten ristiriitatilanteiden vuoksi viestejä sisältävät listat on aina lukittava sitä käyttävälle säikeelle sen käsittelyn ajaksi.

6.3 Ohjelman käyttöönotto

Ohjelman käyttämisen kannalta puolestaan ei ole mielekästä, että testerisovellus ladataan yhdellä ohjelmalla, CAN-väylän alustaminen toisella ja pinnien ohjaukset kolmannella. Koska ohjelman käytön pitää olla mahdollisimman yksinkertaista,

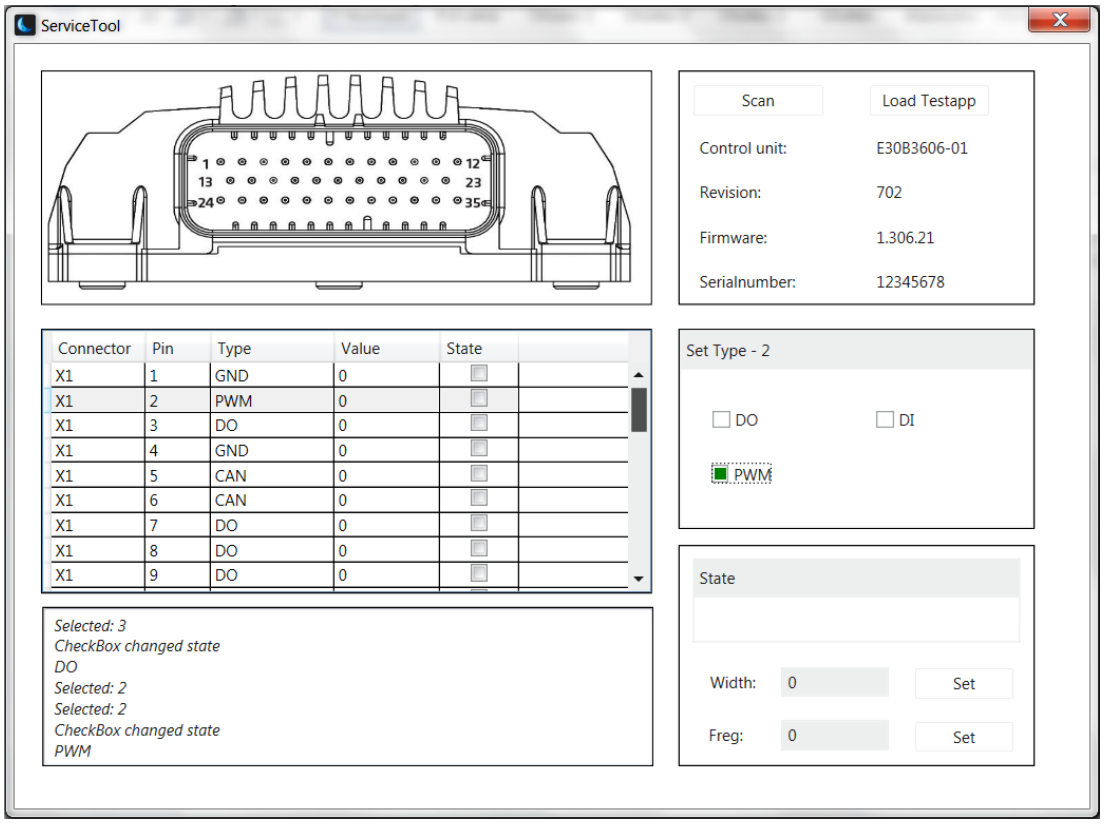
ohjelmanlataus ja CAN-väylän alustaminen tulee sisällyttää ohjelma-skriptiin. Ohjelmanlatausta varten on tehtävä oma säie, jolla pystytään suorittamaan toisia ohjelmia suoraan ohjelma-skriptistä. Tässä tapauksessa ohjelma-skriptistä pitää pystyä suorittamaan Canloader-ohjelmaa. Ulkoisen prosessin käynnistäminen onnistuu .NET Frameworkin System.Diagnostics-kirjaston Process-luokan sekä ProcessStartInfo-luokan avulla (Microsoft 2015d). CAN-väylän alustaminen onnistuu puolestaan CanCore-kirjaston CAN-luokan avulla (Epec [Viitattu 20.8.2014]). CAN-väylän alustaminen ja ohjelmaan lataukseen tarvittava ohjelmakoodi on kuvattu kuviossa 30.

```
444     #CAN-bus -> offline
445     if CAN.IsOnline:
446         CAN.StopCANCore()
447 
448     p = Process()
449     p.StartInfo.UseShellExecute = False
450     p.StartInfo.RedirectStandardOutput = False
451     p.StartInfo.FileName = 'Resources\Scripts\ServiceTool\Loader\canLoader2.exe'
452     p.StartInfo.Arguments = arguments
453     try:
454         p.Start()
455         p.WaitForExit()
456     except CANCore.CANCardReservedException:
457         WriteToConsole(self,"CAN IOError")
458         break
459     else:
460         WriteToConsole(self, ("Could not load application to the unit. Error: " + str(p.ExitCode)))
461         break
462 
463     #CAN-bus -> online & initialize
464     CAN.StartCANCore()
465     if CAN.IsOnline:
466         CAN.Initialize(1000)
467 
468     #load defaults to the control unit
469     LoadDefaultGPIOs(self)
470     #show status
471     WriteToConsole(self, "Control unit ready")
```

Kuvio 30. CAN-väylän hallinta ja ohjelmanlataus IronPythonilla

6.4 Käyttöliittymä

Tehokkaamman XAML-kielellä tehdyn käyttöliittymän latausta varten ohjelmaan tarvitaan System.Windows.Markup-kirjaston XamlReader-luokkaa. Käyttöliittymän XAML-tiedosto saadaan ladattua skriptistä suoritettavan ohjelman ikkunalle XamlReader-luokan avulla (Microsoft 2015c). Ohjelman käyttöliittymä toteutettiin opinnäytetyön rinnalla Ohjelmistoprojektin kurssilla. Käyttöliittymän ulkoasu on esitetty kuviossa 31.



Kuvio 31. Valmiin ServiceTool-testaustyökalun käyttöliittymä

Koko ohjelman toimintaa kuvaava luokkakaavio, sekvenssikaaviot sekä tilakaavio löytyvät tämän työn liitteistä 2, 3 ja 4.

7 POHDINTAA JA YHTEENVETO

Tämän selvitystyön tarkoituksena oli selvittää tekniikka, jolla voitaisiin toteuttaa Epecin sulautettujen ohjausyksiköiden testaamiseen soveltuva ohjelmistotyökalu. Selvityksen lisäksi työkalun ohjelmaa varten oli tarkoitus tehdä suunnitelma, jonka perusteella ohjelma voitaisiin toteuttaa.

Selvitystyössä vertailluista vaihtoehdoista työkalun toteuttamiseksi valittiin tietokonepohjainen sovellus, joka tultaisiin toteuttamaan Epecin CANmoon-ohjelmaan sisällytettävällä IronPython-ohjelma-skriptillä. Ohjelma-skriptissä hyödynnetään Epecin elektroniikkatuotannon testereissä käytettäviä testeri-sovelluksia, joilla työkalua varten tarvittavaa työmäärää saadaan pienennettyä. Vertailua varten tehdyllä yksinkertaistetulla ohjelmalla onnistuttiin ohjaamaan yhtä ohjausyksikön digitaalilähtöä, sekä lukemaan yhtä analogituloa. Nämä havainnot riittävät määrittämään työkalun toteuttamiseen tarvittavat tekniikat ja ratkaisut. Ohjelmaa varten käytetty tekniikka olikin vertailtavista vaihtoehdoista ainoa toteutettavissa oleva ratkaisu, joka täytti työkalun vaatimusmäärittelyn osalta kohdan, jossa ohjausyksikön alkuperäinen ohjelma pitää pystyä säilyttämään Flash-muistilla. Vaatimus olisi ollut todennäköisesti toteutettavissa myös Slave-toimintoa tukevalla Firmwarella, mutta sen toteuttamista varten tehtävä selvitystyö olisi ollut niin laaja, että se olisi yksistään riittänyt opinnäytetyöksi. Slave-toimintoa tukevan Firmwaren tekeminen ei olisi myöskään onnistunut ilman Epecin ohjaus-yksiköiden Firmwaresta vastaavien asiantuntijoiden apua.

Kun käytettävä tekniikka oli selvillä, lopullista työkalua alettiin suunnittelemaan luokkakaavion ja sekvenssikaavioiden avulla. Suunnitteluun liittyi myös paljon varsinaisen ohjelmakoodin tekemistä ja testaamista, sillä tekniikan perustana toimiva CANmoon-ohjelman versio oli melko uusi Epecillä, eikä työkalua vastaavia aliohjelmia ollut juurikaan aikaisemmin tehty. Lopputuloksena työkalua varten saatiin tehtyä keskeisimmät toiminnot kattavan ohjelman luokkakaavio, sekä sekvenssikaaviot.

Varsinaista työkaluakin ehdittiin toteuttaa jonkin verran suunnittelun loppuvaiheilla, sillä samanaikaisesti koululla järjestettiin projektikurssi, jossa työkalulle

suunniteltiin ja toteutettiin sen käyttöliittymä, joka pystyttiin liittämään aiemmin teknisten ratkaisujen kokeiluja varten tehtyihin ohjelmakoodeihin.

Tämän selvitystyön jälkeen työkalun ohjelman kehitys jatkuu Epecillä tässä työssä tehtyjen suunnitelmien avulla. Suunnitelmien perusteella valmiilla työkalulla pitäisi pystytä testaamaan kaikista 3000-sarjan ohjausyksiköistä jokainen digitaalitulo, digitaalilähtö, PWM-lähtö sekä analogitulo. Nämä riittävät vaatimusmäärittelyn mukaisesti 3000-sarjan ohjausyksiköiden testaamiseen vian etsimisen kannalta. Koska työkalua ei ole vielä täysin toteutettu, suunnitelmissa voi tulla vastaan virheitä. Työkalun valmistumista estäviä kriittisiä virheitä ei voi kuitenkaan enää tulla, sillä sen keskeisimmät toiminnot on testattu ja todettu toimiviksi.

Muiden kuin 3000-sarjan ohjausyksiköiden testausta tässä selvitystyössä ei tutkittu, sillä niiden sisällyttäminen työkaluun vaatii huomattavasti lisää selvitys-työtä ja aikaa. Epecin huollon lopullisena tavoitteena kuitenkin on, että yhdellä työkalulla pystyttäisiin testaamaan kaikki Epecin ohjausyksiköt, jolloin huolto ei olisi enää lainkaan riippuvainen elektroniikkatuotannon testereistä vian etsimisessä. Työkaluksi valitun tekniikan merkittävä etu onkin se, että se perustuu pohjimmiltaan elektroniikkatuotannossa käytettäviin testereihin, jolloin sen laajentaminen muillekin kuin 3000-sarjan ohjausyksiköille pitäisi olla mahdollista.

LÄHTEET

3S. 2014. CODESYS Imagebrochure. [Verkkojulkaisu]. Saksa: 3S-Smart Software Solution GmbH. [Viitattu 19.8.2014]. Saatavana: https://www.codesys.com/download/category/broschueren.html

CiA. Ei päiväystä. CANopen. [Verkkosivu]. CAN in Automation (CiA). [Viitattu 23.11.2014]. Saatavilla: http://www.can-cia.org/index.php?id=171

Epec. 2010. CAN CANopen. [Verkkojulkaisu]. Epec Oy. [Viitattu 30.1.2015]. Saatavana Epecin extranetistä: http://epec.planeetta.com/Public/Downloads/Training%20material/Common%20knowledge/CAN_CANopen.pdf. Vaatii käyttöoikeuden.

Epec. 2012. Architecture Specification 3000 control unit. Epec Oy, 10, 13. Vaatii käyttöoikeuden.

Epec. 2013a. Epec 3606 Control Unit Technical document. [Verkkojulkaisu]. Epec Oy. [Viitattu 19.8.2014]. Saatavana Epecin extranetistä: http://epec.planeetta.com/Public/Technical_Documents/3606/3606_MAN000552.pdf. Vaatii käyttöoikeuden.

Epec. 2013b. Toiminnallinen määrittely 3000 Control units Tuotantotestaussovellus. 4. Versio. Epec Oy, 4–5, 14–17. Vaatii käyttöoikeuden.

Epec. 2013c. Epec Oy - Control system solutions for extreme conditions (company video, English version). [Video]. Epec Oy. [Viitattu 22.2.2015]. Saatavana: https://www.youtube.com/watch?v=yZD3b3tZ_D8

Epec. 2014a. Ohjausjärjestelmät. [Verkkosivu]. Epec Oy. [Viitattu 24.10.2014]. Saatavana: http://www.epec.fi/fi/ohjausjarjestelmat/

Epec. 2014b. Epec Product Gatalogue. [Verkkojulkaisu]. Epec Oy. [Viitattu 24.10.2014]. Saatavana: http://www.epec.fi/@Bin/138199/Epec+Product+Catalogue+2015_WEB.pdf?Action=Save

Epec. 2014c. MultiTool. [Verkkosivu]. Epec Oy. [Viitattu 24.10.2014]. Saatavissa: http://www.epec.fi/fi/ohjausjarjestelmat/ohjelmointi-ja-huolto-ohjelmat/multitool/

Epec. 2014d. EPEC Programming and Libraries Manual. [Verkkosivu]. Epec Oy. [Viitattu 25.10.2014]. Saatavana Epecin extranetistä: http://epec.planeetta.com/Public/Manuals/EPEC_Programming_And_Libraries/EpecProgrammingAndLibrariesManual_MAN000538.htm. Vaatii käyttöoikeuden

Epec. 2014e. Epec 38 CANopen Slave M1 ohjausyksikkö. [Verkkosivu]. Epec Oy. [Viitattu 16.11.2014]. Saatavana: http://www.epec.fi/fi/ohjausjarjestelmat/epec-38-canopen-slave-m1/

Epec. Ei päiväystä. CANmoon Scripting Manual. [Verkkosivu]. Epec Oy. [Viitattu 20.8.2014]. Saatavana Epecin extranetistä: http://epec.planeetta.com/Public/Manuals/CANmoon3/API_html/index.html. Vaatii käyttöoikeuden

IEC. 2007. IEC 61131-2 Programmable controllers – Part 2: Equipment requirements and tests. International Electrotechnical Commission, 37–42.

IEC. 2013. IEC 61131-3 Standard Programmable controllers – Part 3: Programming languages. International Electrotechnical Commission, 9.

IronPython. Ei päiväystä. IronPython. [Verkkosivu]. IronPython Community. [Viitattu 14.1.2015]. Saatavilla: http://ironpython.net/

IXXAT. Ei päiväystä. CANopen Basics - Predefined Connection Set. [Verkkosivu]. IXXAT Automation GmbH. [Viitattu 27.12.2014]. Saatavilla: http://www.canopensolutions.com/english/about_canopen/predefined.shtml

Kokkarinen, I. 2004. Java, Prolog ja Python. Tehokas näkökulma ohjelmointiin. Helsinki: Edita.

Koskinen, J. 2004. Mikrotietotekniikka Sulautetut järjestelmät. 1. uud. p. Helsinki: Otava.

Liang, N. & Popovic, D. 2001. The CAN bus. Teoksessa: Vlacic, L., Parent, M. & Harashima, F. Intelligent Vehicle Technologies. Oxford: Butterworth-Heinemann, 21–23, 27–31, 35–49, 60–61.

Microsoft. 2014. Visual Studio with MSDN. [Verkkosivu]. Microsoft Corporation. [Viitattu 7.11.2014]. Saatavana: http://www.visualstudio.com/products/visual-studio-with-msdn-overview-vs

Microsoft. 2015a. Overview of the .NET Framework. [Verkkosivu]. Microsoft Corporation. [Viitattu 14.1.2015]. Saatavilla: http://msdn.microsoft.com/en-us/library/zw4w595w%28v=vs.110%29.aspx

Microsoft. 2015b. Introduction to WPF. [Verkkosivu]. Microsoft Corporation. [Viitattu 14.1.2015]. Saatavilla: http://msdn.microsoft.com/en-us/library/aa970268%28v=vs.110%29.aspx

Microsoft. 2015c. XamlReader Class. [Verkkosivu]. Microsoft Corporation. [Viitattu 26.1.2015]. Saatavilla: https://msdn.microsoft.com/en-us/library/system.windows.markup.xamlreader(v=vs.110).aspx

Microsoft. 2015d. Process Class. [Verkkosivu]. Microsoft Corporation. [Viitattu 27.1.2015]. Saatavilla: https://msdn.microsoft.com/en-us/library/system.diagnostics.process(v=vs.110).aspx

Saha, H. 2005. CAN-väylä. [Verkkolehtiartikkeli]. FLUID Finland 4 - 2005, 7–8. [Viitattu 7.1.2015]. Saatavilla: http://www.canopen.fi/artikkelit/CAN.pdf

Saha, H. 2006. CANopen perusteet. [Verkkolehtiartikkeli]. FLUID Finland 1 - 2006, 6–9. [Viitattu 7.1.2015]. Saatavilla: http://www.canopen.fi/artikkelit/CANopen.pdf

LIITTEET

Liite 1: Vaatimusmäärittely

Liite 2: Luokkakaavio

Liite 3: Sekvenssikaaviot (1-4)

Liite 4: Tilakaavio

Liite 1: Vaatimusmäärittely

Huoltotyökalun vaatimusmäärittely

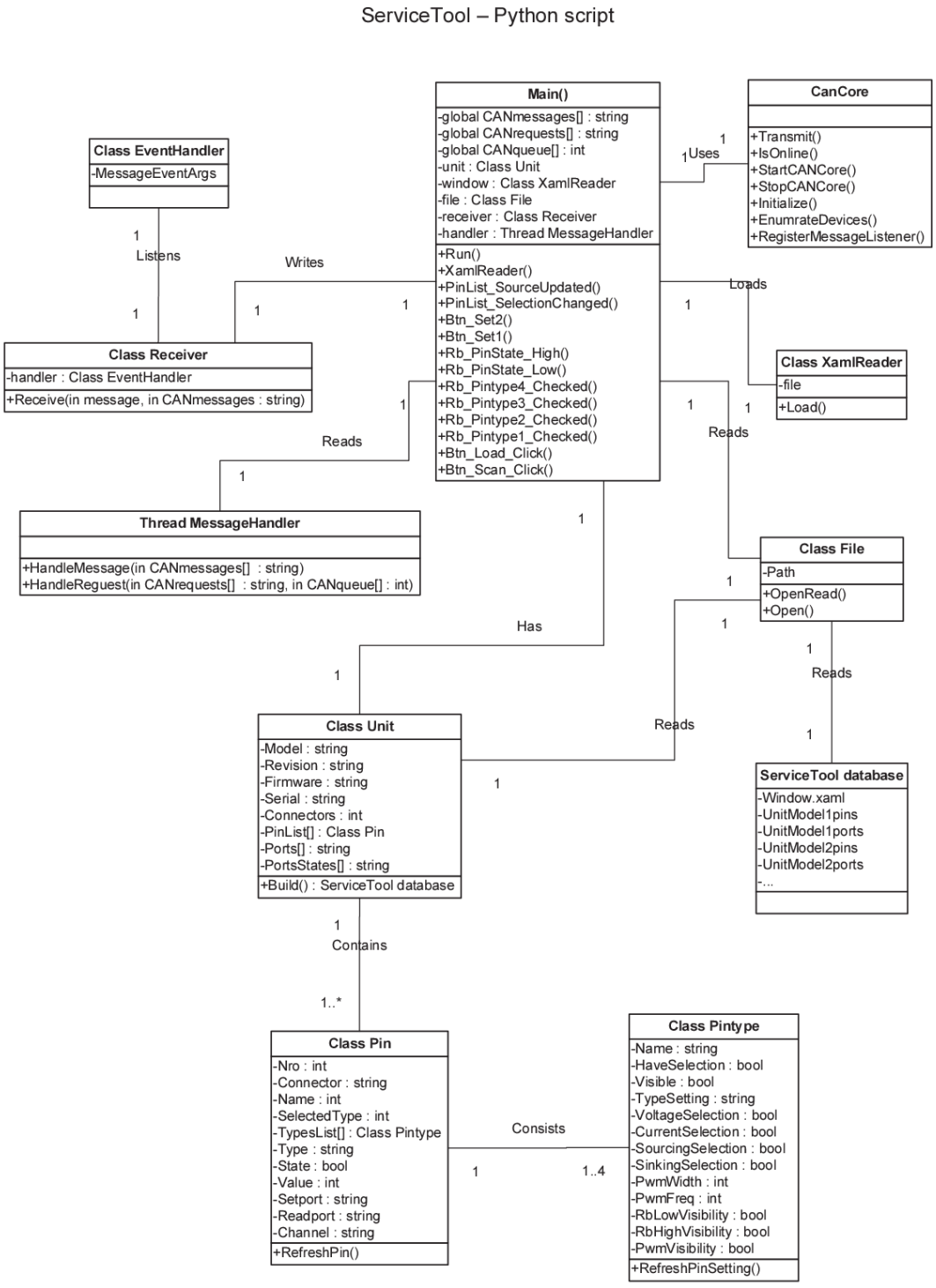
- Huoltotyökalun tarkoitus on auttaa ohjausyksiköiden testaamisessa ja vian selvityksessä.
- Työkalu koostuu ohjelmasta ja käyttöliittymästä, ohjelmaa pyörittävästä laitteesta, sekä työkalun ja testattavan tuotteen liityntärajapinnasta.
- Testattava tuote liitetään työkaluun Can-väylän kautta.
- Työkalulla on voitava testata:
    - Tavoite 1. – Epecin 3606 tuotetta
    - Tavoite 2. – Epecin 3000 sarjan tuotteita
    - Tavoite 3. – Epecin 3000-, 4000- ja 5000-sarjan tuotteita
    - Tavoite 4. – Epecin 2000-, 3000-, 4000- ja 5000-sarjan tuotteita
- Tuotteista tulee pystyä testaamaan sen jokaista I/O-liitäntää, kaikilla niillä tavoilla, joilla niitä voidaan käytännön sovelluksissa käyttää.
- Testattavan tuotteen toiminnoista on pystyttävä kytkemään päälle, sekä muuttamaan niiden ominaisuuksia (1. tavoitteen mukaan):
    - PWM-ulostulo, sen pulssin leveys ja taajuus
    - DO-ulostulo (sinking, sourcing)
- Testattavasta tuotteesta on pystyttävä lukemaan (1. tavoitteen mukaan):
    - DI-tulo (sinking, sourcing)
    - AI-tulo, sekä jännitetulona, että virtatulona
    - FB-tulo
    - PI-tulo

- Testit pitää pystyä suorittamaan niin, että:
    - Testattavan tuotteen muistissa oleva ohjelma ei muutu eikä häviä, tai ennen testiä ohjelma pitää pystyä lataamaan luotettavasti talteen sen uudelleen lataamista ohjausyksikölle.
    - Testattavan tuotteen koteloa ei tarvitse avata.
- Työkalulta vaadittavat ominaisuudet:
    - Työkalun hankintakustannukset tulee pyrkiä pitämään mahdollisimman alhaisina.
    - Työkalun ohjelma pitää pystyä kopioimaan kaikille yrityksessä sitä tarvitseville ilman erillisiä ohjelmistolisenssejä.
    - Työkalun ylläpito ja kehitys tulee tapahtumaan muun työn ohessa, joten sen tulee olla mahdollisimman vaivatonta ja helppoa.
- Työkalun ohjelman kehityksessä tulisi mahdollisuuksien mukaan:
    - käyttää olemassa olevia ohjelmia ja ohjelmakirjastoja
    - tehdä uusia uudelleen käytettäviä ohjelmakirjastoja
    - tehdä luokkia ja hyödyntää olio-ohjelmoinnin tarjoamia etuja
    - hyödyntää modulaarisuutta
    - hyödyntää ohjelman kehittämiseen tarkoitettuja tehokkaita apuohjelmia
- Työkalun käyttöönotto tulee olla mahdollisimman helppoa, näin ollen:
    - Ohjelman käyttöliittymän tulee olla mahdollisimman helppokäyttöinen, niin ettei sen käytön aloittamiseen tarvita erillistä koulutusta tai tietämystä ohjelmoinnista.
    - Ohjelman käyttöohjeen tulee löytyä ohjelmasta itsestään.

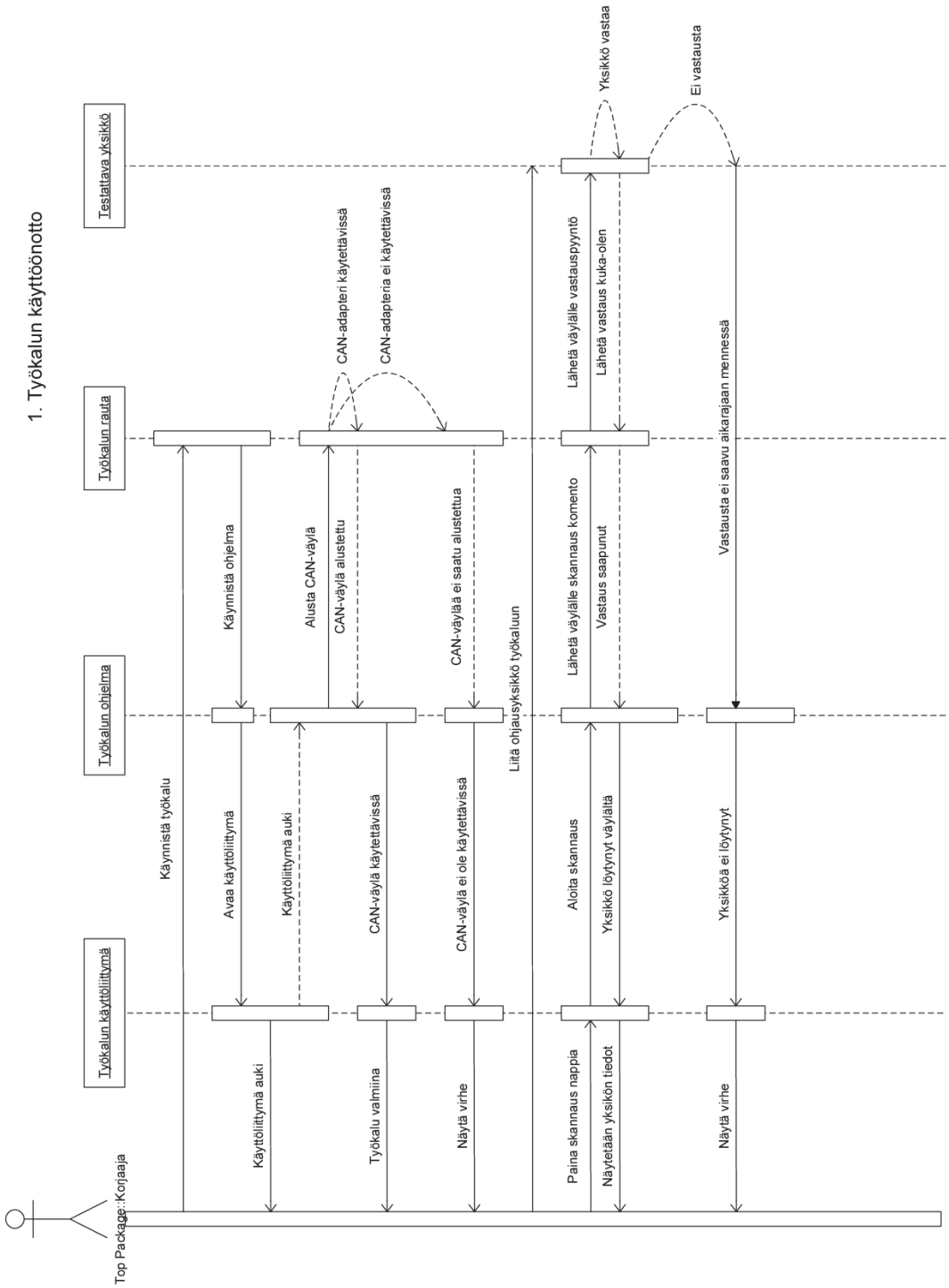
- Ohjelman tulee tarjota käyttäjälle testaamisen kannalta oleelliset ja tarvittavat tiedot, kuten ohjausyksikön pinnien numerot ja tyypit, sekä estää mahdollisien virhetilanteiden syntymisen.

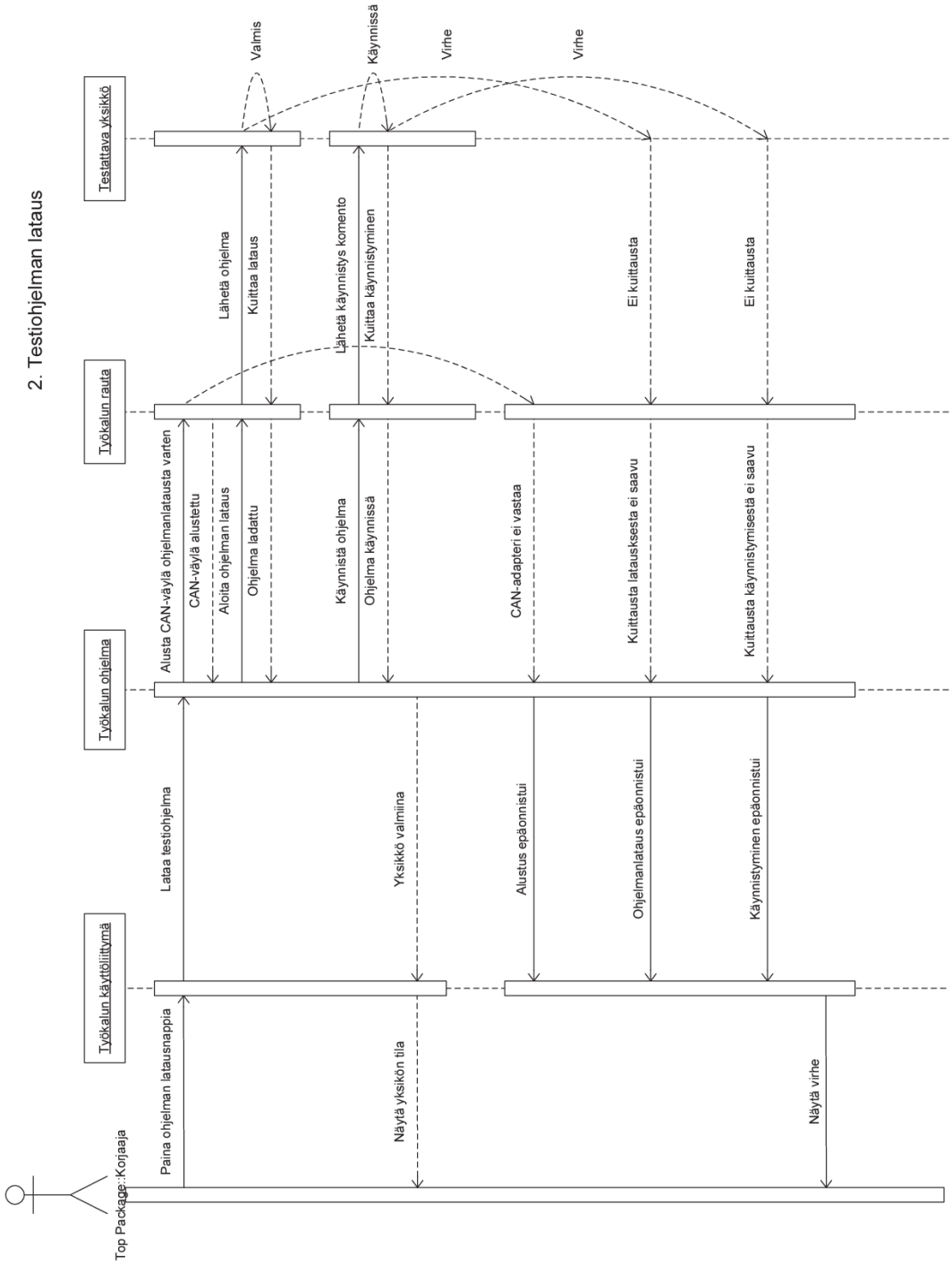
- Tarvittavien laitteiden asennus tulee onnistua alaa vastaavan ammatillisen koulutuksen, tai kokemuksen omaavalta henkilöltä ohjeiden avulla.

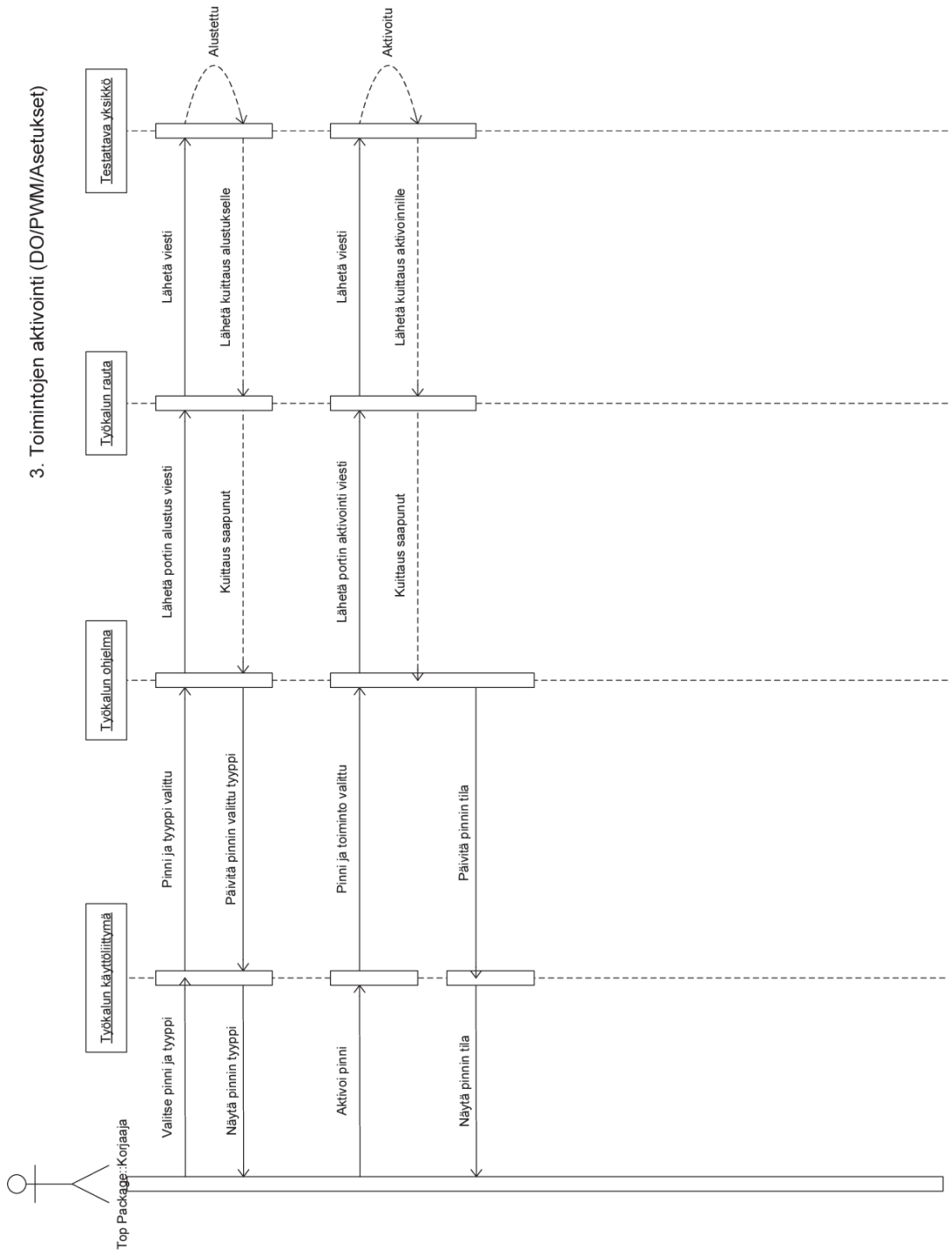
Liite 2: Luokkakaavio

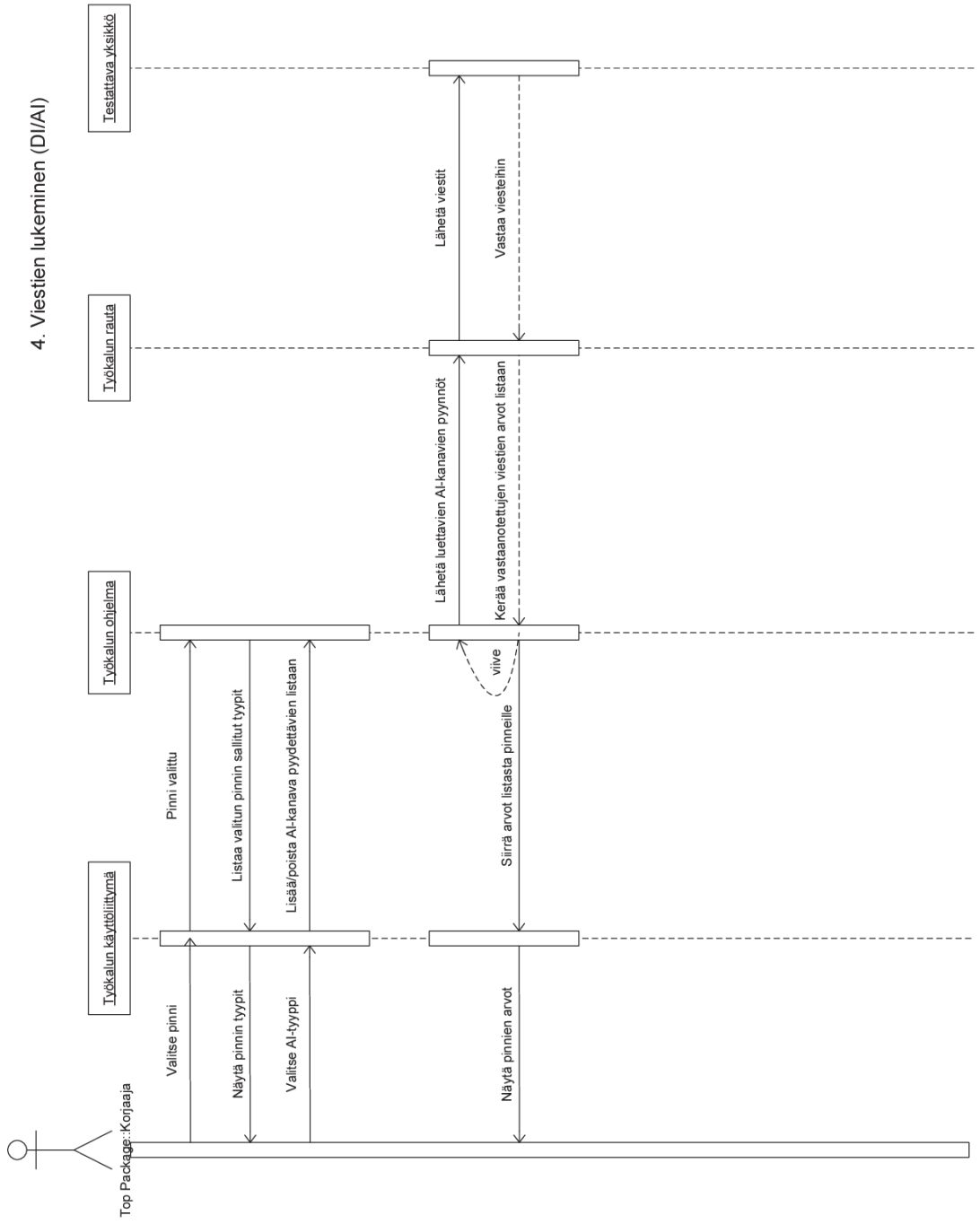


Liite 3: Sekvenssikaaviot (1-4)




2. Testiohjelman lataus
Testattava yksikkö
Työkalun rauta
Työkalun ohjelma
Työkalun käyttöliittymä
Top Package::Korjaaja
Valmis
Käynnissä
Virhe
Virhe
Lähetä ohjelma
Kuittaa lataus
Lähetä käynnistys komento
Kuittaa käynnistyminen
Ei kuittausta
Ei kuittausta
Alusta CAN-väylä ohjelmanlatausta varten
CAN-väylä alustettu
Aloita ohjelman lataus
Ohjelma ladattu
Käynnistä ohjelma
Ohjelma käynnissä
CAN-adapteri ei vastaa
Kuittausta latauksesta ei saavu
Kuittausta käynnistymisestä ei saavu
Lataa testiohjelma
Yksikkö valmiina
Alustus epäonnistui
Ohjelmanlataus epäonnistui
Käynnistyminen epäonnistui
Paina ohjelman latausnappia
Näytä yksikön tila
Näytä virhe


3. Toimintojen aktivointi (DO/PWM/Asetukset)
Top Package::Korjaaja
Työkalun käyttöliittymä
Työkalun ohjelma
Työkalun rauta
Testattava yksikkö
Valitse pinni ja tyyppi
Näytä pinnin tyyppi
Pinni ja tyyppi valittu
Päivitä pinnin valittu tyyppi
Lähetä portin alustus viesti
Kuittaus saapunut
Lähetä viesti
Lähetä kuittaus alustukselle
Alustettu
Aktivoi pinni
Näytä pinnin tila
Pinni ja toiminto valittu
Päivitä pinnin tila
Lähetä portin aktivointi viesti
Kuittaus saapunut
Lähetä viesti
Lähetä kuittaus aktivoinnille
Aktivoitu


4. Viestien lukeminen (DI/AI)
Testattava yksikkö
Työkalun rauta
Työkalun ohjelma
Työkalun käyttöliittymä
Top Package::Korjaaja
Lähetä viestit
Vastaa viesteihin
Lähetä luettavien AI-kanavien pyynnöt
Kerää vastaanotettujen viestien arvot listaan
viive
Pinni valittu
Listaa valitun pinnin sallitut tyypit
Lisää/poista AI-kanava pyydettävien listaan
Siirrä arvot listasta pinneille
Valitse pinni
Näytä pinnin tyypit
Valitse AI-tyyppi
Näytä pinnien arvot

Liite 4: Tilakaavio