jamk.fi

Pienmetsäkoneen käyttöliittymä

Markus Pohjola

Opinnäytetyö
Marraskuu 2014

Automaatiotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Pohjola, Markus	Opinnäytetyö	25.11.2014
	Sivumäärä 64	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Pienmetsäkoneen käyttöliittymä

Koulutusohjelma
Automaatiotekniikan koulutusohjelma

Työn ohjaaja(t)
Seppo Rantapuska
Jaakko Fonselius

Toimeksiantaja(t)
Vision Systems Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja valmistaa Tehojätkä-pienmetsäkoneeseen uusi käyttöliittymä. Opinnäytetyö oli osa isompaa projektia, jossa metsäkoneen aikaisempi PC-pohjainen ohjausjärjestelmä päivitettiin luotettavampaan ja käyttäjäystävällisempään järjestelmään. Aikaisempi järjestelmä korvattiin Bosch Rexroth RC 28-14 - mobiilihydrauliikkaohjaimella sekä Hydac Hy-eVision 7 tuuman työkonenäytöllä. Näin saatiin aikaiseksi erittäin luotettava ja metsäolosuhteisiin soveltuva työkone.

Opinnäytetyön keskeisiä aihealueita ovat metsäkoneenohjaimen ja työkonenäytön välisen CAN-väylän sekä sen kautta liikkuvien viestien kuvaaminen. Käyttöliittymän toiminta ja sen sisältämät eri ominaisuuden on myös kuvattu opinnäytetyössä. Lisäksi dokumentissa kerrotaan, kuinka näytön sovellus on ohjelmoitu käyttäen Codesys 3.5– ohjelmointiympäristöä ja mitä eri ominaisuuksia se pitää sisällään.

Opinnäytetyön tuloksena saatiin valmistettua asiakkaan vaatimukset täyttävä ja tuotantoon kelpaava metsäkoneen käyttöliittymä. Käyttöliittymä on tälläkin hetkellä käytössä kaikissa uusissa Usewoodin valmistamissa Tehojätkä-pienmetsäkoneissa.

Avainsanat (asiasanat)
Metsäkoneautomaatio, CAN-Väylä, HYDAC Hy-eVision, CODESYS 3.5, Käyttöliittymä

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Pohjola, Markus	Bachelor's thesis	25.11.2014
		Language of publication: Finnish
	Number of pages 64	Permission for web publication: x

Title of publication
Forest machine user interface

Degree programme
Automation Engineering

Tutor(s)
Rantapuska, Seppo
Fonselius, Jaakko

Assigned by
Vision Systems Oy

Abstract

The aim of the thesis was to design and produce a new user interface for a Tehojätkä-forest machine. This thesis was part of a larger project, in which the previous PC-based control system of the forest machine was updated to be more reliable and user-friendly. The previous system was replaced by Bosch Rexroth RC 28-14 -mobile hydraulics controller and Hydac Hy-eVision 7 inches machine display. The final result was a control system, which is very reliable and has been designed for forestry environment.

The central topics of this thesis were to describe how the CAN-bus works and how the CAN-messages are transferred between the mobile hydraulic controller and the machine display. The operation of the user interface and all of its different properties also present a central part of this thesis. This thesis document also describes how the application of the display is programmed using Codesys 3.5 programming environment and what different features it contains.

The result of the thesis was a forest machine interface, which meets all the requirements set by the assigner company and is suitable for production. This user interface is included in all new Tehojätkä-forest machine manufactured by Usewood oy.

Keywords/tags (subjects)
Forest machine automation, CAN-bus, HYDAC Hy-eVision, CODESYS 3.5, User interface

Miscellaneous

Sisältö

Käsitteet **5**

1 Johdanto **6**

1.1 Opinnäytetyön taustat 6

1.2 Vision Systems Oy 7

1.3 Usewood Oy 7

1.4 Tehojätkä-pienmetsäkone 7

2 Työkonenäyttö **10**

2.1 Näytön kuvaus 10

2.2 Näytön liitynnät 11

2.3 Näytön muistit 12

2.4 Näytön reaaliaikakello 12

2.5 Näytön grafiikan toteutus 13

3 Codesys 3.5 -ohjelmointiympäristö **16**

3.1 Yleiskuvaus 16

3.2 Ohjelmointiympäristön käyttäminen 19

3.2.1 Projektin aloitus 19

3.2.2 I/O-määritykset 20

3.2.3 Ohjelmakirjastot 21

3.2.4 Visualisointi 21

4 CAN-väylä **23**

4.1 Fyysinen rakenne 23

4.2 Viestikehys 24

4.3 CANopen-protokolla 26

4.3.1 Protokollan rakenne 26

4.3.2 CANopen-manager 28

4.3.3 CANopen-objektikirjasto 29

4.4 VS-CAN-protokolla 30

5 Käyttöliittymän suunnittelu **32**

5.1 Vaatimusmäärittely 32

5.2 Vaatimukset 33

6 Käyttöliittymän toteutus 34
6.1 Yleiskuvaus käyttöliittymän toiminnasta 34
6.2 Tiedonsiirto 35
6.3 Parametrit 36
6.4 Vikakoodit 38
6.5 Käyttötapaloki 38
6.6 Tuntilaskurit 39
6.7 Näytön ohjaaminen 40
6.8 Käyttöliittymän rakenne 41
6.8.1 Päänäyttö 41
6.8.2 Asetukset 42
6.8.1 Huoltonäkymä 43
6.8.2 Työlaitenäkymät 43
7 Käyttöliittymän testaaminen 45
7.1 Testausympäristö 45
7.2 Sovelluksen testaaminen 46
7.3 Suoritetut testit 47
7.3.1 Metsäkoneen ajaminen ja työlaitteet 47
7.3.2 Parametrien kirjoittaminen ja lukeminen 47
7.3.3 Käyttötapalokin kirjoitus 48
7.4 Vikatilanteen 48
8 Pohdinta 48
8.1 Asiakkaan palaute 48
8.2 Parannusehdotukset 49
8.3 Oma arvio opinnäytetyöstä 50
Lähteet 52
Liitteet 54
LIITE 1. Työkonenäytön rakenne 54
LIITE 2. Codesys 3.5-ohjelmointiympäristö 55
LIITE 3. Parametritaulukko 56
LIITE 4. Asetussivun logiikan ohjaaminen 58
LIITE 5. Asetussivun grafiikan ohjaaminen 59

LIITE 6. Parametrien kirjoitusfunktio 60
LIITE 7. Käyttötapalokin kirjoitusfunktio 61
LIITE 8. FRAM-muistin käsittelyfunktiot 62
LIITE 9. Can-viestien vastaanottofunktio 63
LIITE 10. Käyttötapaloki 64

KUVIOT

KUVIO 1. Tehojätkä-pienmetsäkone 8
KUVIO 2. UW-40-risuraivain 9
KUVIO 3. Mense-raivauspää 9
KUVIO 4. Hy-eVision työkonenäyttö 10
KUVIO 5. Codesys 3.5 -visualisointityökalu 13
KUVIO 6. Codesys 3.5, toimilohkoesitystapa 16
KUVIO 7. Codesys 3.5, tikapuukaavioesitystapa 17
KUVIO 8. Codesys 3.5, ST-esitystapa 17
KUVIO 9. Codesys 3.5, käskylistaesitystapa 18
KUVIO 10. Codesys 3.5, SFC-esitystapa 18
KUVIO 11. Codesys 3.5, projektinluonti 19
KUVIO 12. Codesys 3.5, IO-kanavat 20
KUVIO 13. Codesys 3.5, ohjelmakirjastot 21
KUVIO 14. Päänäytön rakenne 22
KUVIO 15. Codesys 3.5, Loopcode-funktio 22
KUVIO 16. CAN-väylän signaali-tasot 23
KUVIO 17. CAN-väylän rakenne. 24
KUVIO 18. NRZ-linjakoodaus. 24
KUVIO 19. CAN-viestikehyksen rakenne 25
KUVIO 20. CanOpen-protokollan rakenne 26
KUVIO 21. CanOpen-kommunikointiobjektit 27
KUVIO 22. CanOpen-solmun tilakone 28
KUVIO 23. CANOpen-objektikirjasto 29

KUVIO 24. VS-CAN-protokolla, analogiaviestit 30
KUVIO 25. VS-CAN-protokolla, parametrien kirjoitusviesti 31
KUVIO 26. Codesys 3.5, Task manager 35
KUVIO 27. Codesys 3.5, muuttujien määritys 36
KUVIO 28. Parametrienkirjoitusfunktio 37
KUVIO 29. Metsäkoneen parametrisivu 37
KUVIO 30. Metsäkoneen vikakoodit 38
KUVIO 31. Metsäkoneen tuntilaskurit 40
KUVIO 32. Metsäkoneen päänäyttö 42
KUVIO 33. Metsäkoneen asetukset. 42
KUVIO 34. Metsäkoneen huoltonäkymä 43
KUVIO 35. UW-40 työlaitenäkymä 44
KUVIO 36. Kasvinsuojeluruiskun asetukset 44
KUVIO 37. Työlaitekohtainen ohjesivu 45
KUVIO 38. Metsäkoneen testausympäristö 46

Käsitteet

Bosch Rexroth RC28-14	Mobiilihydrauliikkaohjain
CAN-väylä	Työkoneissa käytetty tiedonsiirtoväylä
CANopen	Ylemmän tason CAN-protokolla
COB-id	CAN-viestin id-numero
Codesys 3.5	Ohjelmointiympäristö
DeviceNet	Ylemmän tason CAN-protokolla
ECU	Electronic Controller Unit, ohjainyksikkö
EDS	Electronic data sheet, laiteajuri
FBD	Function Block Diagram, ohjelmointikieli
FLASH	Haihtumaton muistityyppi
FRAM	Ferroelectric RAM, muistityyppi
GSM	Global System for Mobile
GPS	Global Position System, paikannusjärjestelmä
IL	Instruction list, ohjelmointikieli
J1939	Ylemmän tason CAN-protokolla
LD	Ladder Diagram, ohjelmointikieli
NRZ	Non Returning Zero, linjakoodaus
PDO	Process Data Object, CAN-viestityyppi
POU	Program Organization Unit
Task manager	Ohjelman suoritusajan asetustyökalu
UML	Unified Modeling Language, mallinuskieli
UW-40	Metsäkoneen työlaite risujen raivaamiseen
UW-180	Metsäkoneen työlaite puiden kaatamiseen
SFC	Sequence Function Chart, ohjelmointikieli
SDO	Service Data Object, CAN-viestityyppi
ST	Structured Text, ohjelmointikieli
XML	Extensible Markup Language, merkintäkieli

1 Johdanto

1.1 Opinnäytetyön taustat

Tässä opinnäytetyössä käsiteltiin projektia, jonka toteutus aloitettiin Usewood oy:lle vuonna 2013. Projektin on toteuttanut Vision systems Oy, jossa aloitin työharjoittelijana kesällä 2013. Projektin tarkoituksena oli korvata pienmetsäkoneen edellinen PC-pohjainen ohjausjärjestelmä mobiilihydrauliikkaohjaimella sekä työkonenäytöllä. Mobiilihydrauliikkaohjaimeen siirtymisen tarkoituksena oli parantaa metsäkoneen luotettavuutta sekä pienentää sen vikaantumisen mahdollisuutta. Hydrauliikkaohjaimena projektissa käytettiin Bosch Rexroth RC28-14-ohjainta sekä näyttönä Hydac Hy-evision 7 tuuman työkonenäyttöä. Näytön pääasiallinen tehtävä on välittää informaatiota käyttäjälle koneen toiminnasta sekä mahdollistaa eri työkoneparametrien muuttaminen. Parametreina toimivat erilaiset työlaitteisiin liittyvät painerajat, ja muut työlaitteen toimintaan vaikuttavat asetukset. Lisäksi näyttöä käytetään mahdollisen vian paikallistamiseen sekä koneen toimintaa seuraavien lokitiedostojen tallentamiseen.

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa Tehojätkä-pienmetsäkoneen graafinen käyttöliittymä. Käyttöliittymä oli tarkoitus toteuttaa liikkuviin työkoneisiin suunniteltuun työkonenäyttöön. Näytön ohjelmointiin käytettiin Codesys 3.5-ohjelmointiympäristöä, jonka kuvaaminen onkin yksi opinnäytetyön keskeisistä teemoista. Lisäksi opinnäytetyössä perehdyttiin tiedonsiirtoon CAN-väylässä.

Opinnäytetyössä toteutettiin selkeä ja tuotantoon kelpaava metsäkoneen käyttöliittymä. Käyttöliittymän avulla voidaan asetella metsäkoneen ja sen eri työlaitteiden parametreja. Lisäksi käyttöliittymää voidaan käyttää mahdollisten vikatilanteiden selvittämiseen sekä tiedonkeruuseen metsäkoneesta.

1.2 Vision Systems Oy

Opinnäytetyön toimeksiantaja oli Vision Systems Oy. Se on vuonna 1985 perustettu yritys, joka sijaitseen Jyväskylän Kirrissä. Yritys on erikoistunut optoelektroniikkaan ja konenäköön perustuviin mittaus ratkaisuihin. Sen asiakaskunta koostuu paperi-, sellu-, metalli-, teräs-, lasi-, elintarvike- ja rengasteollisuuden suuryrityksistä sekä eri alojen laite- ja konevalmistajista. Vision Systems on toimittanut maailman laajuisesti yli 2700 konenäköjärjestelmää, ja optoelektronista ratkaisua. (Yritys 2014.)

Toinen merkittävä yrityksen toimiala on sen tarjoamat suunnittelupalvelut. Suunnittelupalvelut tarjoaa asiakkaille apua erilaisten automaatio ja sulautettujenjärjestelmien projektien toteuttamiseen. Hyvänä esimerkkinä suunnittelupalveluiden tarjoamista palveluista, voidaan pitää tässä opinnäytetyössä kuvattua pienmetsäkoneen ohjausjärjestelmän päivittämistä. (Yritys 2014.)

1.3 Usewood Oy

Opinnäytetyön loppuasiakas oli Usewood oy. Se on vuonna 2007 toimintansa aloittanut yritys, joka tuottaa laitteita ja palveluita taimikon ja nuoren metsän hoitoon. Yrityksen myymät Tehojätkä-pienmetsänhoitokoneet ovat heidän itsensä suunnittelemia. Yritys sai alkunsa, kun sen yksi perustajista, lääkärinäkin toimiva Jussi-Pekka Usenius etsi toimivaa metsäkonetta omien metsiensä hoitamiseen. Sopivaa konetta ei kuitenkaan ollut saatavilla, joten hän päätti valmistaa koneen itse. Ensimmäisestä valmistuneesta metsäkoneesta onkin vierähtänyt jo muutamia vuosia ja Tehojätkästä on tähän päivään mennessä valmistunut jo viisi eri sukupolvea. Yritys kehittää jatkuvasti tuotteitaan, jotta metsänhoito olisi mahdollisimman helppoa ja tehokasta. (Tietoa meistä 2014.)

1.4 Tehojätkä-pienmetsäkone

Tehojätkä on erityisesti taimikonhoidon tarkkuustyöhön kehitetty pieni ja kevyt metsäkone. Voimanlähteenä koneessa on dieselmoottori, joka pyörittää

hydrauliikkapumppua. Koneen liikuttaminen ja työlaitteiden käyttäminen toimii täysin hydraulisesti. Konetta on saatavilla mallista riippuen joko 4, 6 tai jopa 8-pyöräisenä. Kuviossa 1 on kuusipyöräinen tehojätkä pro, joka on varustettu erityisesti energiapuunkorjaamiseen käytettävällä UW180 -hakkuupäällä.

KUVIO 1. Tehojätkä-pienmetsäkone

Pienmetsäkoneeseen on saatavilla useita eri työlaitteita, risuraivaimista aina erilaisiin hakkuukouriin. Lisäksi risuraivain voidaan varustaa vesakontorjuntaan käytettävällä torjunta-aineruiskulla. Tätä varten metsäkoneen etuosaan voidaan lisätä 200 litran säiliö, josta aine annostellaan automaattisesti työlaitetta käytettäessä. Kuviossa 2 on esitelty UW-40-risuraivain, jota käytetään esimerkiksi vesakon raivaamineen. (Koneet 2014.)

KUVIO 2. UW-40-risuraivain

Tehojätkään voidaan liittää myös muiden kun Usewoodin valmistamia työlaitteita. Esimerkiksi kuviossa 3 on nähtävillä risukonraivaamiseen käytettävä Mense-raivauspää. Raivauspään toiminta perustuu terän edestakaiseen liikkeeseen ja sen työskentelyleveys on mallista riippuen 1–2,3 m. (Toimintaperiaate 2014.)

KUVIO 3. Mense-raivauspää

2 Työkonenäyttö

2.1 Näytön kuvaus

HY-eVision² 7.0 on Hydac:n valmistama, erityisesti työkoneisiin suunniteltu näyttöpaneeli (ks. kuvio 4). Näyttö toimii täysin itsenäisenä yksikkönä liittyen ulkomaailmaan CAN-väylän kautta. Näyttö on kooltaan 7”, ja sen resoluutio on 800 x 400 pikseliä. Näyttöä ohjataan sen etupaneelissa olevilla kymmenellä painonapilla, joiden toiminta on vapaasti ohjelmoitavissa. Käyttöjärjestelmänä näyttö käyttää räätälöityä Linux-käyttöjärjestelmää, ja sen päällä pyörivän näyttösovelluksen ohjelmointiin käytetään Codesys 3.5-ohjelmointiympäristöä. Näyttöä on saatavilla myös isompana 10.4” versiona sekä varustettuna kosketusnäytöllä. Lisäksi näyttöön voidaan halutessa liittää sisäinen GSM/GPS-moduuli. (Hy-evision 7.0 user manual 2014)

KUVIO 4. Hy-eVision työkonenäyttö

2.2 Näytön liitynnät

Näytön liitynnät ulkomaailmaan ovat monipuoliset. Peruskäytössä näytön liityntänä käytetään CAN-väylää, niitä on yhteensä kaksi kappaletta. CAN-väylät tukevat ylemmän tason CAN-protokollia. Näitä ovat CANopen, DeviceNet sekä J1939. CAN-väylän pääasiallinen tehtävä on siirtää prosessidataa näytön ja varsinaisen ohjainlogiikan välillä. (Hy-evision 7.0 user manual 2014)

Sovelluksen lataamiseen näytölle sekä sovelluksen debukaukseen käytetään ethernet-liityntää. Ethernet-liityntä mahdollistaa myös näytön tiedostojärjestelmän tarkastelun terminaaliyhteyden välityksellä. Mikäli käyttävässä tietokoneessa ei ole kyseistä liityntää, voidaan tiedostojärjestelmän selaamiseen käyttää myös näytössä olevaa RS-232 sarjaliikenneyhteyttä. (Hy-evision 7.0 user manual 2014)

Näytön ohjaamiseen käytettään sen etupaneelissa olevia painonappeja, joita on kymmenen. Painonapit ovat vapaasti ohjelmoitavissa sekä sisältävät taustavalaistuksen halutessa. Näyttöön on myös mahdollista liittää kaksi erillistä videosignaalia, jotka voidaan esittää näytöllä. Liityntä sisältää myös 12 V:n jännitesyötön kameralle, joka saadaan ohjelmallisesti kytkettyä päälle ja pois. Näin saadaan esimerkiksi peruutuskamera kytketyksi näyttöön. (Hy-evision 7.0 user manual 2014)

USB-liityntä mahdollistaa sekä näytön ohjelmistopäivitykset että tiedostojen tallentamisen näytön muistista. Tämä on erittäin käytännöllinen tapa siirtää erilaisia lokitiedostoja näytön ja PC:n välillä. (Hy-evision 7.0 user manual 2014)

Näyttö sisältää myös antureita sen sisäisen toiminnan tarkasteluun. Näitä ovat suorittimen syöttöjännitteen mittaus sekä kotelon sisäisen lämpötilan mittaus. Näitä suureita tarkastelemalla voidaan sovellukseen lisätä halutessa hälytyksiä ja varoituksia, mikäli lämpötila nousee liian korkeaksi. (Hy-evision 7.0 user manual 2014)

Lisävarusteena näyttöön on saatavana laajennusmoduuli, joka tuo näyttöön GSM/GPS-ominaisuudet. Tällä lisävarusteella ja TTcontrolin valmistamalla sovelluksella, voidaan ajoneuvoon lisätä etäseuranta ja datankeruumahdollisuus. Sovelluksella on mahdollisuus muuttaa työkoneen parametreja etäyhteyttä käyttäen. Tässä projektissa ei ollut tarvetta käyttää edellä mainittua laajennusmoduulia. Tarkempi kuvaus työkonenäytön rakenteesta selviää liitteestä 1. (Hy-evision 7.0 user manual 2014)

2.3 Näytön muistit

Muistia näytössä on kolmea eri tyyppiä. Varsinainen sovelluksen käyttämä DDR2-muisti on kooltaan 256 Mbit. Sovelluksen varaaman muistin määrä riippuu sen käyttämistä muuttujista sekä kuvaobjekteista. Toisena muistityyppinä on haihtumaton FLASH-muisti, jota on yhteensä 512 Mbit. Tämä muistialue on jaettu kahdeksi 256 Mbit alueeksi, joista toinen on tarkoitettu sovelluksen sekä sen sisältämien kuvien ja mahdollisten pdf-tiedostojen tallentamiseen. Toinen 256 Mbit alue on vapaasti käyttäjän käytettävissä. Muistialue on tarkoitettu esimerkiksi lokitiedostojen säilyttämiseen näytössä. FLASH-tyyppinen muisti ei kestä jatkuvaa kirjoittamista ja lukemista, joten sitä ei ole suotavaa käyttää sellaisten parametrien tallentamiseen, joita joudutaan lukemaan ja kirjoittamaan useasti. Tähän tarkoitukseen näytössä on käytössä siihen suunniteltu muistialue, josta käytetään nimitystä FRAM. FRAM kestää jopa miljoonia päällekirjoituksia, joten se on ihanteellinen muisti esimerkiksi useasti muuttuvien tietojen tallentamiseen. FRAM-muistia näytössä on 4 kt, joka on riittävä määrä useisiin sovelluksiin. (Hy-evision 7.0 user manual 2014)

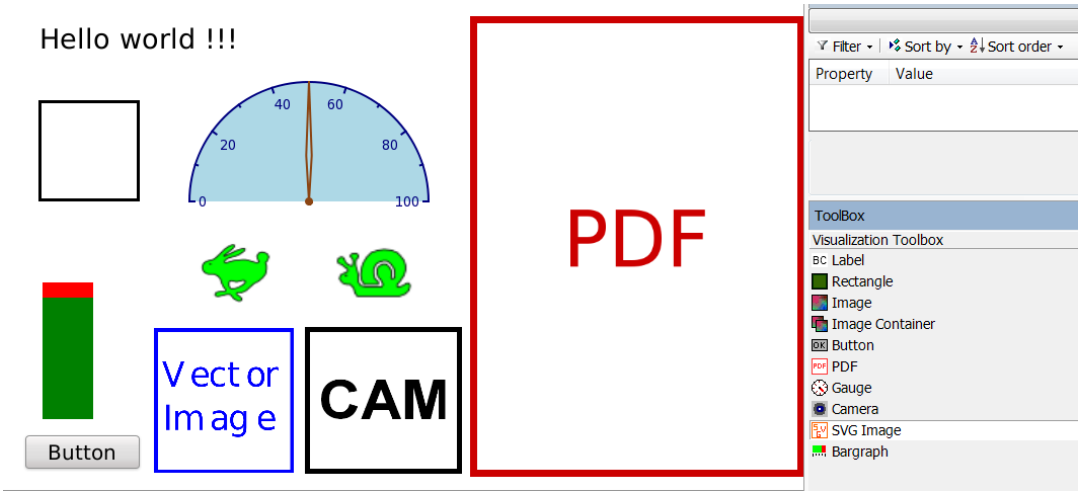
2.4 Näytön reaaliaikakello

Näytön reaaliaikakello on hyödyllinen ominaisuus, jota voidaan käyttää moneen eri tarkoitukseen, kuten kellonajan esittämiseen näytöllä sekä tuntilaskureiden rakentamiseen. Reaaliaikakello on paristovarmennettu ja sille luvataan 14 vuorokauden varmennus täydellä latauksella. Tarkkuudeksi on ilmoitettu +-2 s/vrk, mikä on riittävä tarkkuus useimpiin konesovelluksiin. Esimerkiksi 2 sekunnin lisäys

vuorokaudessa koneen käyttötuntilaskuriin ei vaikuta merkittävästi sen luotettavuuteen. (Hy-evision 7.0 user manual 2014)

2.5 Näytön grafiikan toteutus

HY-eVision² 7.0-näytön grafiikan rakentamiseen käytetään Codesys-ohjelmointiympäristön TTC-Visualization Manager-ominaisuutta, jonka avulla projektiin voidaan luoda uusia sivuja (ks. kuvio 5). Uuden sivun luomisen jälkeen voidaan siihen lisätä tarvittavat elementit Visualization Toolbox-valikosta. Vaihtoehtoina on 10 erilaista elementtiä, joilla kullakin on oma käyttötarkoituksensa. Seuraavaksi käydään läpi Vizualization-valikon toiminnot yleisellä tasolla. (Codesys 3.5 -ohjelmointiympäristö. 2014)



KUVIO 5. Codesys 3.5 -visualisointityökalu

Label-elementti on tekstikenttä, joka voi sisältää joko kiinteän tekstin tai dynaamisesti muuttuvan tekstikentän. Tekstin perusominaisuuksiin kuuluu fontin ja sen koon muuttaminen sekä tekstintasaus. Tekstiin, kuten lähes kaikkiin muihinkin elementteihin voidaan käyttää parametreja, joita ovat värin ja paikan muuttaminen, häivytys sekä tekstin kierto. Kaikkia näitä ominaisuuksia pääsee vapaasti muokkaamaan ohjelmassa viittaamalla tekstinominaisuuksissa määritettyyn tekstin-nimeen sekä valitsemalla muokattava ominaisuus ja antamalla sille arvo.

Rectangle-elementti mahdollistaa neliön tai suorakaiteisen objektin liittämisen sivulle. Rectangle-elementtiä voidaan muokata viittaamalla objektille annettuun nimeen sekä valitsemalla muokattava ominaisuus. Ominaisuudet ovat samoja kuin edellä kuvatun label-elementin ominaisuudet. (Codesys 3.5 Online help 2014)

Image-kentällä voidaan sivulle liittää kuvaobjekti. Tuetut formaatit ovat png sekä bmp. Kuvaelementin muokkaamiseen voidaan käyttää myös edellä kuvattuja ominaisuuksia. Yleisenä efektinä kuvissa on tässä projektissa käytetty SetOpacity-funktiota. Tällä funktiolla voidaan häivyttää objekteja ja luoda siten grafiikkaan erilaisia efektejä. (Codesys 3.5 Online help 2014)

Image container on elementti, johon voidaan määritellä useita kuvia. Esitettävä kuva voidaan valita viittaamalla objektin nimeen sekä valitsemalla metodi SetIndex. Image container-ominaisuutta on hyödynnetty tässä käyttöliittymässä työlaitteiden valintasivulla, jonka avulla asetetaan haluttu työlaite. (Codesys 3.5 Online help 2014)

Button-elementtiä käytetään määrittämään painonappi. Painonapille voidaan määritellä ominaisuutena sen toiminallisuus, joka voi olla palautuva tai palautumaton. Painonapille voidaan myös määritellä vaihtuva kuva sitä painettaessa. Painonapin ominaisuuksiin pääsee viittaamalla sille määritettyyn nimeen sekä valitsemalla muokattava ominaisuus. (Codesys 3.5 Online help 2014)

PDF-elementillä on mahdollista liittää pdf-tiedostoja näytettäväksi näytön sivuilla. Tämä ominaisuus mahdollistaa esimerkiksi työkoneiden käyttöohjeiden lukemisen suoraan näytöltä. Pdf-elementti sisältää metodeja pdf-tiedoston liikuttamiseen näytöllä sekä tiedoston zoomausominaisuuden. Kaikkia elementin ominaisuuksia pääsee muokkaamaan viittaamalla tiedoston nimeen sekä valitsemalla muokattava ominaisuus. (Codesys 3.5 Online help 2014)

Gauge-elementillä voidaan yksinkertaisella tavalla määrittää analoginen mittaristo näytölle. Mittaristolle määritellään sen minimi ja maksimi näyttämä sekä sen fyysinen

koko. Tämän jälkeen sen osoittamaa arvoa voidaan muuttaa viittaamalla mittaristolle annettuun nimeen sekä valitsemalla metodi SetValue. (Codesys 3.5 Online help 2014)

Camera-elementillä voidaan liittää näytön sivulle esimerkiksi peruutuskamerasta tuleva videosignaali. Elementin ominaisuuksista voidaan valita sisään tulevan signaalin kanava, joko channel-1 tai cannel-2. Elementin voi määritellä kattamaan kokonaisen sivun tai vain osan siitä. TTcontrolin ohjelmiston päivittäminen versioon 1.4.1.4 mahdollistaa myös kahden yhtäaikaisen video signaalin esittämisen jaetulla ruudulla. Esitettävää videota voidaan myös kääntää tai peilata näytöllä. Peilikuvana esitettävä video on peruutuskameraa käyttäessä luonnollinen valinta, sillä se helpottaa ympäristön hahmottamista. (Codesys 3.5 Online help 2014)

SVN-image on muutoin samanlainen elementti kuin normaali image-elementti, mutta esitettävän kuvan formaattina käytetään vektorigrafiikalla toteutettuja svn-kuvia. (Codesys 3.5 Online help 2014)

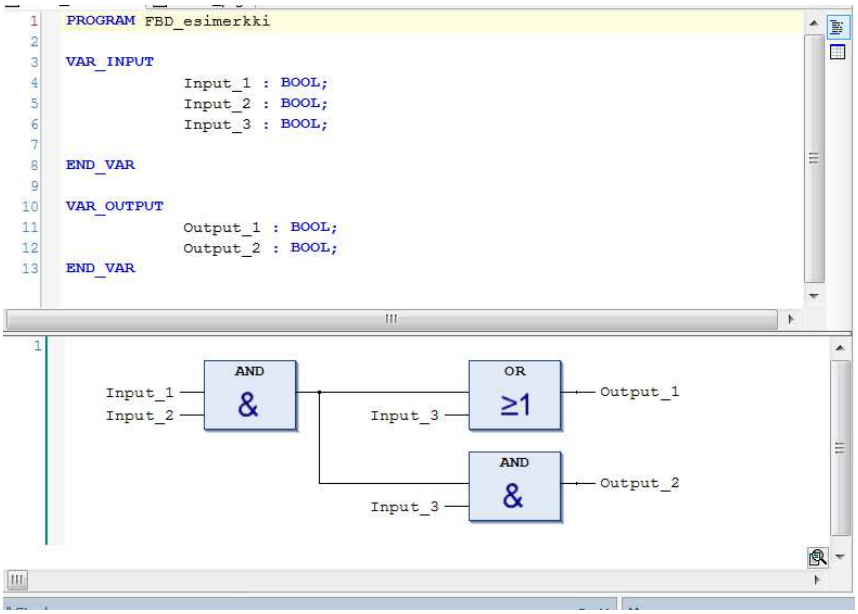
Bargraph-elementillä voidaan liittää pylväspalkki grafiikkasivuille. Palkit ovat havainnollisempi tapa esittää muuttuvaa suuretta, kuin pelkkä numeerinen arvo. Palkille voidaan määrittää sen minimi -ja maksimiarvo, sen koko ja ylähälytysraja. Palkin näyttämää arvoa voidaan muuttaa viittaamalla sille määritettyyn nimeen sekä käyttämällä metodia SetValue. (Codesys 3.5 Online help 2014)

3 Codesys 3.5 -ohjelmointiympäristö

3.1 Yleiskuvaus

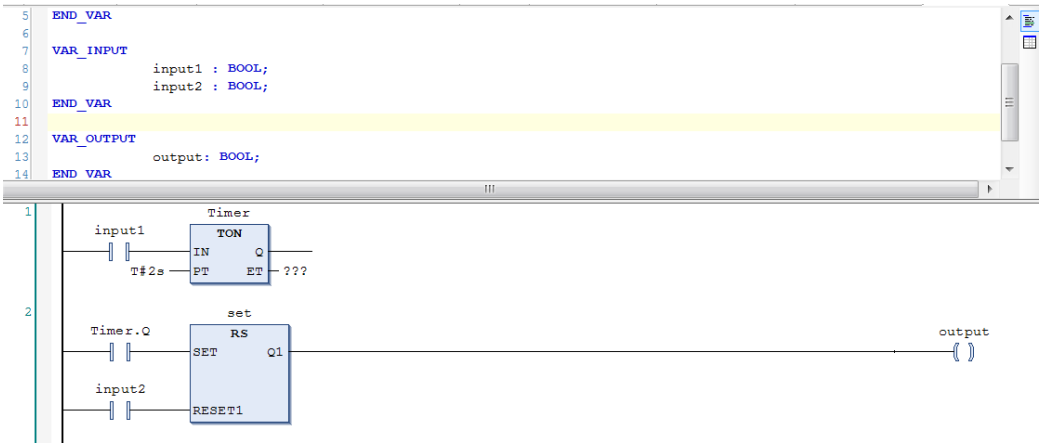
Codesys on saksalaisen 3S-Smart Software Solutions valmistama ohjelmointiympäristö, joka on paljon käytetty etenkin ohjelmoitavien logiikoiden sovelluskehityksessä. Codesys käyttää IEC 61131-3 ohjelmointistandardia, joka määrittää viisi käytettävää ohjelmointikieltä. Ohjelmointikielet ovat seuraavat, FBD (Function Block Diagram), LD (Ladder Diagram), ST (Structure Text,), IL (Instruction List, kieli) ja SFC (Sequential Function Chart). (Codesys ohjelmointiympäristö 2014)

Funtion Block Diagram (FBD) on toimilohkoesitystapa (ks. kuvio 6). Toimilohkoesitystavan etuna on sen selkeä ja helposti luettava rakenne. Se koostuu toimilohkosymboleista, joita voidaan liittää toisiinsa. Ohjelmasuoritus etenee aina ylhäältä alas, ja vasemmalta oikealla. Toimilohkojen sisääntulot ovat niiden vasemmalla ja lähdöt niiden oikealla puolella. (IEC_61131-3 standardi 2014)



KUVIO 6. Codesys 3.5, toimilohkoesitystapa

Ladder Diagram (LD) on tikapuukaavioesitystapa, jonka visuaalinen ilme muistuttaa relekaavioita (ks. kuvio 7). Tikapuuesitystapa saa nimensä juuri tästä relekaaviomaisesta ilmeestään, jossa logiikan tulot ovat vasemmassa ja lähdöt kuvan oikeassa reunassa. (IEC_61131-3 standardi 2014)



KUVIO 7. Codesys 3.5, tikapuukaavioesitystapa

Structure Text (ST) on monipuolinen korkeamman tason esitystapa, jonka rekenne muistuttaa Pascal-kielen rakennetta (ks. kuvio 8). Esitystapa mahdollistaa erilaisten silmukoiden käytön ohjelmassa, kuten IF, IF_ELSE, FOR, WHILE ja CASE rakenteet. (IEC_61131-3 standardi 2014)

```
FUNCTION_BLOCK UiControl
VAR_INPUT

            EnableTX        : BOOL   :=FALSE;
            setHeaterPower  : BYTE := 0;
            setWiperSpeed   : BYTE := 0;
            setGearUp       : BOOL := FALSE;
            setGearDown     : BOOL := FALSE;
```

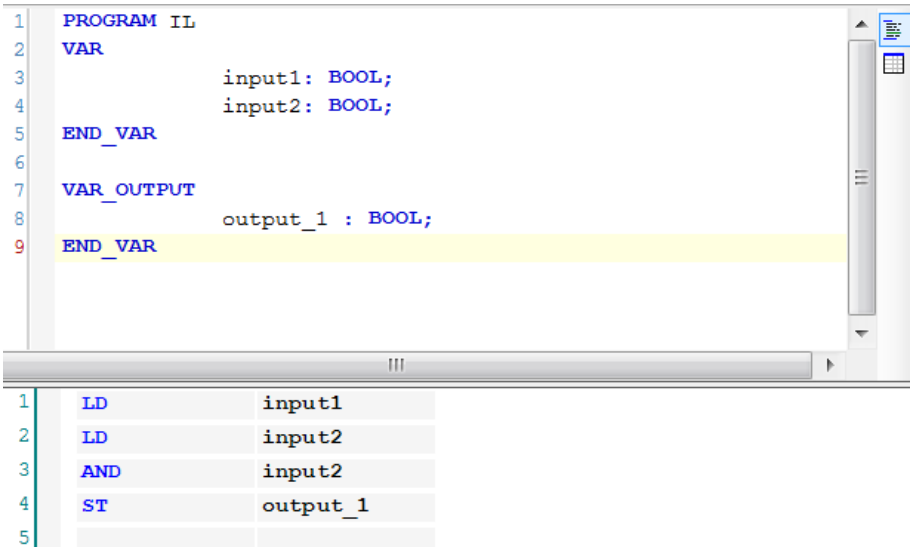
```
    IF setGearUp = TRUE THEN
        GearCommand := 16#40;
        ELSIF
            setGearDown = TRUE THEN
                GearCommand := 16#80;
                ELSE
                 GearCommand := 16#00;
    END_IF

    controlWord := SHL (setHeaterPower, 4) OR setWiperSpeed;   // controlWord

                fbCanTxUiControl(
                        Channel:=0 ,
                        MsgId:= 16#117,
                        CanTxData0:= controlWord ,
                        CanTxData1:= GearCommand ,
                        CanTxData2:= 0,
```

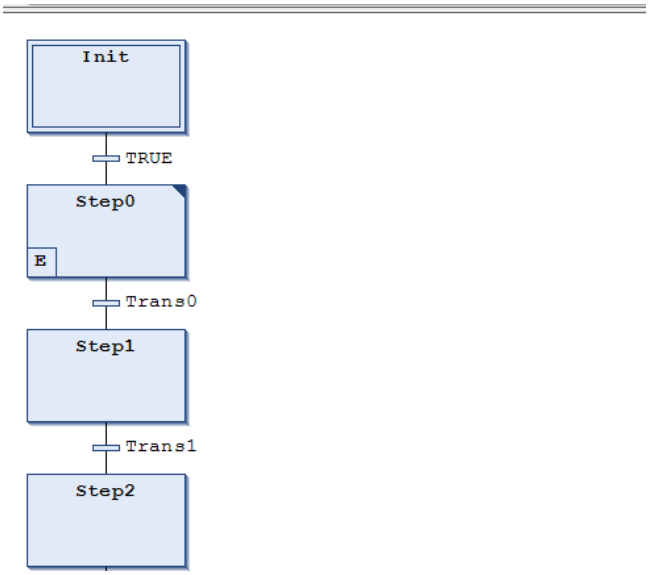
KUVIO 8. Codesys 3.5, ST-esitystapa

IL eli Instruction List on käskylistaohjelmointia (ks. kuvio 9). Ohjelman rakenne koostuu peräkkäisistä käskyistä, joiden mukaan ohjelmansuoritus etenee. (IEC_61131-3 standardi 2014)



KUVIO 9. Codesys 3.5, käskylistaesitystapa

Sequense Flow Chart (SFC), on seqvenssien ohjelmointiin käytetty kieli (ks. kuvio 10). Seqvenssi koostuu toisiinsa liitetyistä itsenäisistä funktioista, joita suoritetaan peräkkäin. (IEC_61131-3 standardi 2014)

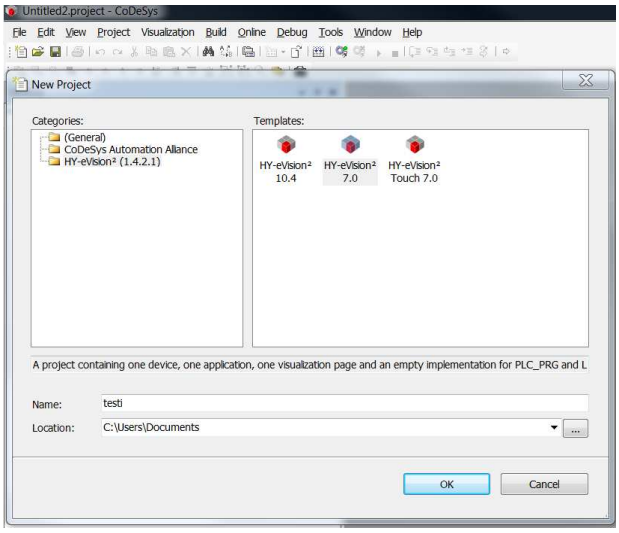


KUVIO 10. Codesys 3.5, SFC-esitystapa

3.2 Ohjelmointiympäristön käyttäminen

3.2.1 Projektin aloitus

Uuden projektin luominen aloitetaan Codesys 3.5-ympäristössä valitsemalla file -> New project. Projekti voidaan luoda tyhjästä, tai käyttäen apuna valmiita pohjia (ks. kuvio 11). Valmiiden pohjien käyttäminen lisää projektiin valitun logiikan sekä sille valitut oletuskirjastot. Lisäsi projektiin lisätään ohjelman pääfunktiona toimiva PLC (prg), joka vastaa yleisesti tunnettua main-funktiota. (Codesys 3.5 Online help 2014)



KUVIO 11. Codesys 3.5, projektinluonti

Ohjelmointiympäristö koostuu projektipuusta (ks. liite 2), jonka alla näkyvät kaikki projektiin liitetyt laitteet, käytetyt kirjastot sekä lisätyt ohjelmafunktiot. Omien funktioiden lisääminen tapahtuu application valikon päällä hiiren oikeaa näppäintä painamalla sekä valitsemalla add object ja POU. Valittavana ovat Program, Function block tai function-tyyppinen ohjelmalohko. (Codesys 3.5 Online help 2014)

”Program” määrittää lohkon normaaliksi ohjelmaksi, joka voi sisältää useita sisään ja ulostulevia signaaleja. Lisäksi Program on ainoa lohko, jolle voidaan määrittää tehtävä-managerilla oma suoritussykli sekä prioriteetti.

”Function block” on tyypiltään funktio, jolle voidaan määrittää useita eri instansseja. Esimerkkinä voidaan pitää tässä projektissa paljon käytettyä päästöhidastukseen käytettävää TON, eli TimeOnDelay-funktiota. Samaa ajastinlohkoa voidaan käyttää ohjelmassa useaan kertaan määrittämällä sille aina eri instanssi.

”Function” on normaali funktio, joka voi sisältää useita tuloja, mutta vain yhden lähdön. Lisäksi perusfunktio eroaa function blockista siinä, että se ei sisällä muistia sisäisten muuttujien tallentamiseen.

3.2.2 I/O-määritykset

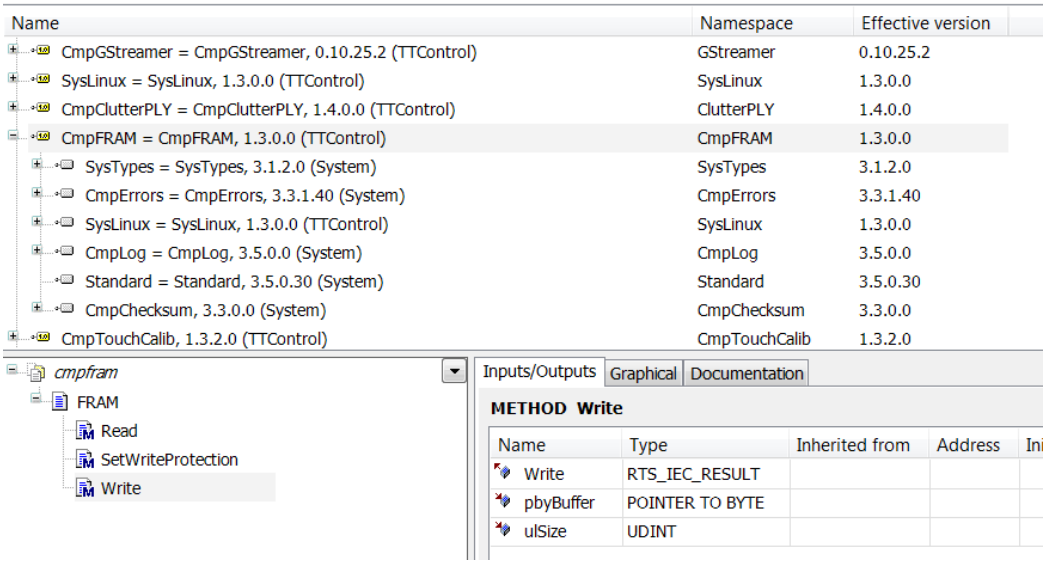
Projektiin valitun logiikan IO-määrityksiä päästään muuttamaan Systems-valikon avulla. Logiikan sisältämät IO-kanavat sekä muut sisäiset muuttuja nähdään System IO-mapping välilehdeltä (ks. kuvio 12). Mikäli logiikan IO-kanavia halutaan lukea ja kirjoittaa ohjelmassa, on ne ensin liitettävä johonkin ohjelman sisältämään muuttujaan. Tämä tapahtuu yksinkertaisesti valitsemalla haluttu IO-kanava sekä määrittämällä muuttuja, joka on aikaisemmin luotu ohjelmaan. Globaaleiden muuttujien käyttäminen IO-kanavien lukemiseen tai kirjoittamiseen mahdollistaa niiden käyttämisen mistä tahansa ohjelman lohkosta. Globaali muuttuja lista voidaan lisätä projektiin hiiren oikealla näppäimellä, sekä valitsemalla AddObject > Global_variable_list. (Codesys 3.5 Online help 2014)

Keyboard						
gKeyboardLight		Keyboard	%QX26.3	BIT		Keyboard backlight
State						
gKey9State		Keyboard	%IX48.2	BIT		Key "F9" state
gKey8State		Keyboard	%IX48.3	BIT		Key "F8" state
gKey7State		Keyboard	%IX48.4	BIT		Key "F7" state
gKey6State		Keyboard	%IX48.5	BIT		Key "F6" state
gKey5State		Keyboard	%IX48.6	BIT		Key "F5" state
gKey4State		Keyboard	%IX48.7	BIT		Key "F4" state
gKey3State		Keyboard	%IX49.0	BIT		Key "F3" state
gKey2State		Keyboard	%IX49.1	BIT		Key "F2" state
gKey1State		Keyboard	%IX49.2	BIT		Key "F1" state
gKey0State		Keyboard	%IX49.3	BIT		Key "F0" state
Count						

KUVIO 12. Codesys 3.5, IO-kanavat

3.2.3 Ohjelmakirjastot

Ohjelman kirjoittamisen kannalta hyvin keskeisessä osassa ovat näytön ohjelmointiympäristön mukana tulleet funktiokirjastot. Nämä sisältävät valmiiksi kirjoitettuja ohjelmalohkoja näytön sekä sen grafiikan ohjaamiseen. Kirjastoja pääsee tarkastelemaan projektipuun library-managerin alta (ks. liite 1). Kuviossa 13 on kuvattu ohjelmakirjastosta löytyvä, näytön FRAM-muistiin kirjoittamiseen käytettävä valmis funktio. Valitettavasti valmiiden funktioiden kuvaukset ja niiden toiminnan selvittäminen on kuvattu erittäin suppeasti. Funktioiden toiminnan selvittäminen ja niiden testaaminen onkin vienyt projektissa todella paljon aikaa. (Codesys 3.5 Online help 2014)

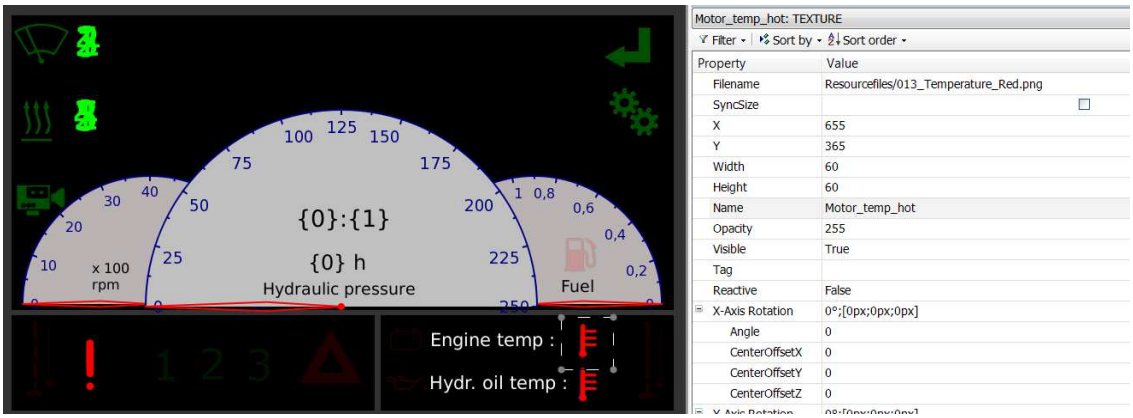


KUVIO 13. Codesys 3.5, ohjelmakirjastot

3.2.4 Visualisointi

Työkonenäytön pääasiallinen tehtävä on välittää informaatiota käyttäjälle metsäkoneen toiminnasta ja sen mahdollisesta vikatilasta. Projektipuun sisältämää TTC vizualization manager-työkalua käyttäen on mahdollista rakentaa erilaisia näyttösivuja, joiden avulla käyttäjä saa tietoa koneen toiminnasta. Grafiikkaeditorilla lisättyjen ikonien ja tekstien ohjaamiseen käytetään omaa erillistä funktiota, jonka nimi on LoopCode (prg). Funktioon lisätään kaikki grafiikkaan ja sen ohjaamiseen

liittyvä logiikka. Seuraavaksi käydään läpi yksinkertainen esimerkki, jonka tarkoitus on havainnollistaa kuinka työkonenäytön grafiikkaa voidaan ohjata. Esimerkkinä käytetään kuviossa 14 näkyvää metsäkoneenpäänäyttöä, ja sen sisältämää moottorin lämpötilaa ilmaisevaa symbolia. Symbolin on tarkoitus ilmaista käyttäjälle mahdollinen moottorin ylikuumeneminen, joka voi johtaa jopa moottorivaurioon. Symboli koostuu punaisesta ja vihreästä lämpötilaikonista, jotka ovat aseteltu päällekkäin. Ikonien ohjaamiseen käytetään kuviossa 15 näkyvää ohjelmaa. Ohjelmaan on aseteltu vakiona moottorin lämpötilaraja, jonka ylittyminen aiheuttaa punaisen lämpötilasymbolin ilmaantumisen näytölle. Symbolien läpinäkyvyyttä säädetään ohjelmakoodissa olevalla SetOpacity-funktiolla. (Codesys 3.5 Online help 2014)



KUVIO 14. Päänäytön rakenne

```
(************************** MOTOR TEMP ICON ***************************************)

IF engineTemp > ENGINE_TEMP_LIMIT THEN                  // engine temp > 104
    Motor_temp_hot.SetOpacity(255);
    Motor_temp_ok.SetOpacity(0);

        ELSE IF engineTemp < ENGINE_TEMP_LIMIT THEN

            Motor_temp_hot.SetOpacity(0);
            Motor_temp_ok.SetOpacity(255);

        END_IF
END_IF
```

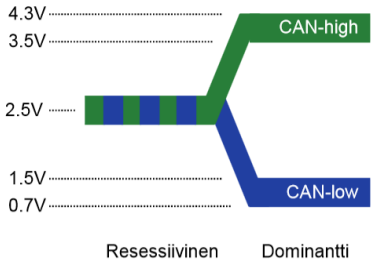
KUVIO 15. Codesys 3.5, Loopcode-funktio

4 CAN-väylä

CAN eli Controller Area Network on sarjamuotoinen tiedonsiirtoväylä. Sen kehittäminen aloitettiin jo vuonna 1983, jolloin ajoneuvoelektroniikkaan erikoistunut yritys nimeltään Bosch julkisti ensimmäiset sitä tukevat sovellukset. Väylä on erityisesti ajoneuvoissa paljon käytetty, koska se vähentää huomattavasti kaapeloinnin tarvetta sekä helpottaa vian paikallistamista. Fyysisesti CAN-väylä koostuu kahdesta parikierretystä johtimesta, joiden välityksellä viestit kulkevat CAN-laitteiden välillä. Väylän toiminta perustuu broadcast-tyyppiseen viestien lähetykseen, jossa ei varsinaisesti ole erillistä isäntälaitetta, joka välittäisi viestejä väylän muille laitteille, vaan jokainen laite voi suoraan kommunikoida toisen laitteen kanssa. (Controller Area Network. 2014)

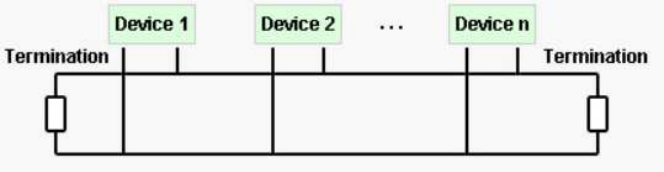
4.1 Fyysinen rakenne

Fyysisesti CAN-väylä koostuu kahdesta johtimesta, joista käytetään termejä CAN_H ja CAN_L (ks. kuvio 16). Looginen 1 ja 0 tulkitaan CAN-väylässä johtimien keskinäisestä jännite-erosta, joka on 1,5 V tai 0 V. Jännite-eroon perustuvaa väylää kutsutaan differenttiaaliseksi väyläksi. Differenttiaalisella väylällä saavutetaan parempi häiriösietoisuus, koska mahdollinen ulkopuolinen häiriö vaikutta sekä CAN_H että CAN_L signaaleihin. Näin ollen ulkopuolinen häiriö saadaan kumottua varvinaisesta signaalista. CAN signaalit voivat olla tilasta riippuen joko dominanttissa tai resistiivisessä tilassa. Termeillä kuvataan johtimessa olevaa jännitetasoa, joka voi olla joko 1.5 tai 3.5 V. Looginen 1 eli dominantti on tila jossa CAN_H johtimeen syötetään 3.5V ja CAN_L johtimeen puolestaan 1.5 V. Resistiivinen eli looginen 0 tarkoittaa tilaa, jossa CAN_H ja CAN_L johtimien keskinäinen jännite-ero on 2.5 V.



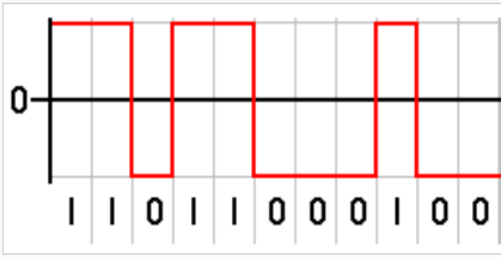
KUVIO 16. CAN-väylän signaali-tasot

CAN-väylän fyysinen rakenne on tyypiltään väylä, johon jokainen solmu kytketään rinnakkain (ks. kuvio 17). Väylän fyysinen maksimi pituus määräytyy käytetyn väylänopeuden mukaan. Väylän suurimmalla nopeudella 1 Mbit/s, väylän pituus rajoittuu 40 m. Muita yleisesti käytettyjä väylänopeuksia ovat 500, 250 ja 125 kbit/s. Väylän molemmissa päissä tulee käyttää 120 Ω päätevastuksia, joiden tehtävän on estää signaalin takaisinheijastumiset väylän päistä. (Saha, H. 2014. Can-väylän toiminta)



KUVIO 17. CAN-väylän rakenne.

CAN-viestien binäärikoodauksessa käytetään ns. NRZ-linjakoodausta (ks. kuvio 18). NRZ-linjakoodauksessa signaalin tilaa ei käytetä alhaalla, ellei signaalin tila todella muutu. NRZ-koodauksen huonona puolena voidaan pitää tahdistuksen puuttumista itse koodauksesta. (Non-return-to-zero. 2014)



KUVIO 18. NRZ-linjakoodaus.

4.2 Viestikehys

CAN-väylässä liikkuvat viestit on pakattu erityisiin viestikehyksiin (ks. kuvio 19). Viestikehysten tarkoituksena on erottaa eri viestit toisistaan, ja välittää ne oikealle vastaanottajalle. Kehyksiä on kahta eri tyyppiä jatkettu, sekä standardikehys. Suurin ero näiden välillä on kehyksen ID-kentässä joka on standardikehyksessä 11-bit pitkä,

sekä jatketussa versiossa 29-bit pituinen. Seuraavassa käydään läpi standardikehyksen eri lohkot ja niiden tarkoitus. (Saha, H. 2014. Can-väylän toiminta)

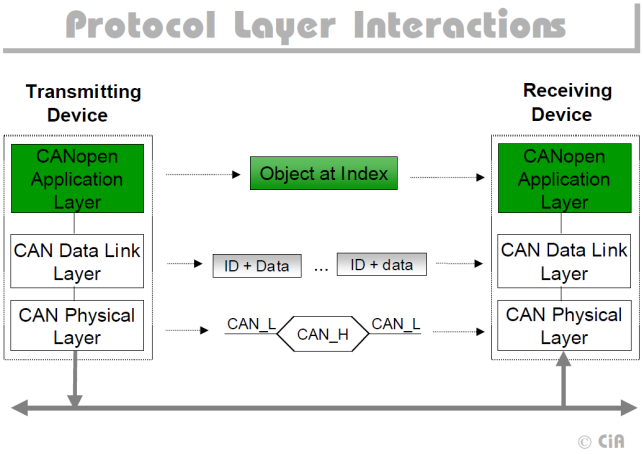
- SOF eli start of frame on aloitusbitti, jonka tarkoituksena on kertoa milloin uusi kehys alkaa, sekä toimia väylän tahdistuksessa.
- Id-numero on joko 11-bittiä tai 29-bittä pitkä. Sen avulla määritetään kenelle väylän solmuista viesti on tarkoitettu. ID-kentän arvo toimii myös prioriteetti arvona, joka määrää mikä on viestien lähetysjärjestys.
- RTR on erikoiskenttä, joka määrittää tietojen kyselyn solmuilta.
- IDE määrittää, käytetäänkö 11-bit vai 29-bit id-kehystä.
- r0 on käytössä, mikäli käytetään jatkettua viestikehystä.
- DLC määrittää kuinka monta tavua varsinaista hyötydataa kehyksessä lähetetään. Valinta voi olla väliltä 0 – 8.
- Datakenttä sisältää kehyksessä solmujen välillä siirrettävän datan, jonka suurin määrä on 8 tavua jokaista lähetettävää kehystä kohden.
- CRC-kentässä lähetetään tarkistus-summa, jonka avulla voidaan todeta vastaanotetun viestin virheettömyys.
- Vastaanottaja kuittaa viestin saapuneeksi, pakottamalla ACK-bitin ylätilaan.
- EOF on viestin lopetuskenttä, joka sisältää kolme resistiivistä bittiä. Kentän avulla voidaan todeta viestin saapuneen kokonaisuudessaan perille.

KUVIO 19. CAN-viestikehyksen rakenne

4.3 CANopen-protokolla

4.3.1 Protokollan rakenne

CANopen on ylemmän tason CAN-protokolla, jota käytetään yleisesti liikkuvissa koneissa tiedonsiirtoprotokollana. Väylään liitettyjä laitteita kutsutaan solmuiksi joita voi olla maksimissa 126-kappaletta, yhdessä ja samassa fyysisessä väylässä. Vaikka CanOpen ei varsinaisesti perustu master-slave tyyppiseen kommunikointiin, sisältää se kuitenkin yleensä verkonhallintasolmun. Verkonhallintasolmu kykenee välittämään verkontoimintaa ohjaavia NMT, eli Network management-viestejä muille väylän solmuille. Verkonhallinta viestien tehtävänä on valvoa ja ohjata väylään liitettyjen muiden solmujen toimintaa. (Canopen protocols. 2014) Kuviossa 20 on esitetty CanOpenin toiminnallinen kuvaus, käyttäen OSI-mallin kerroksia.



KUVIO 20. CanOpen-protokollan rakenne

Fyysisesti CanOpen-viesti välitetään väylään liitettyjen solmujen välillä käyttäen OSI-mallin alinta, eli fyysistä kerrosta. Seuraavassa eli datansiirtokerroksessa määritetään vastaanottaja lähetetylle viestikehykselle. Tähän tarkoitukseen käytetään CAN-viestikehyksen ID-kenttää, joka on CanOpen-standardin mukaan aina 11-bit pitkä. Id-kenttä koostuu solmun yksilöllisestä numerosta, joka voi olla väliltä 1-126 sekä siihen lisätystä kommunikaatio-objektin numerosta. Tästä yksilöllisestä numerosta käytetään nimitystä COB-id, eli Communication Object identification. Tätä numeroa käytetään

määrittämään kenelle verkkoon liitetyistä solmuista lähetetty viesti on tarkoitettu. COB-id numero toimii myös viestin prioriteetin määrittäjänä. Mitä pienempi COB-id numero on, sitä suurempi on sen prioriteetti väylällä. (Canopen protocols. 2014) Kuviossa 21 on listattu CanOpen-standardin viestiobjektit sekä niiden COB-id numerot.

Communication object	COB-ID(s) hex	Slave nodes
NMT node control	000	Receive only
Sync	080	Receive only
Emergency	080 + NodeID	Transmit
TimeStamp	100	Receive only
PDO	180 + NodeID 200 + NodeID 280 + NodeID 300 + NodeID 380 + NodeID 400 + NodeID 480 + NodeID 500 + NodeID	1. Transmit PDO 1. Receive PDO 2. Transmit PDO 2. Receive PDO 3. Transmit PDO 3. Receive PDO 4. Transmit PDO 4. Receive PDO
SDO	580 + NodeID 600 + NodeID	Transmit Receive
NMT node monitoring (node guarding/heartbeat)	700 + NodeID	Transmit

KUVIO 21. CanOpen-kommunikointiobjektit

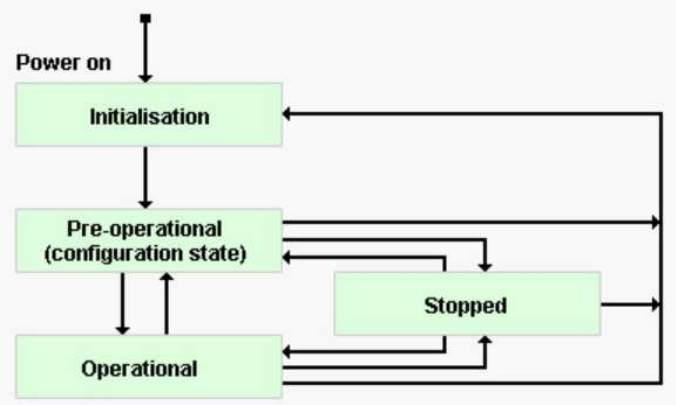
Can-viestin saavuttua oikealle vastaanottajalle käytetään sen tietosisällön tulkitsemiseen OSI-mallin sovelluskerrosta. Tässä kerroksessa saapuva tieto puretaan kehyksestä käyttäen apuna CanOpenin objektikirjastoa (ks. kuvio 23) (Canopen protocols. 2014)

SDO eli Service Data Object on nimensä mukaan objekti, jota käytetään Can-väylässä olevien solmujen asetusten muuttamiseen. SDO-viestien kirjoittaminen perustuu ns. peer-to-peer tyyppiseen kommunikointiin, jossa keskustellaan suoraan valitun solmun kanssa. Piirikytkentäisessä kommunikoinnissa käytetään datan siirtämiseen erillisiä datan pyynti ja datan vastaanottoviestejä. SDO-viestien lähettäminen solmulle edellyttää aina solmun asettamisen ns, PreOperational-tilaan. (Canopen protocols. 2014)

PDO eli prosess data object on objekti, jota käytetään varsinaisen hyötydatan siirtämiseen solmujen välillä. CANopen-protokolla mahdollistaa neljän eri PDO-viestin lähettämisen. Jokainen PDO-viesti voi sisältää maksimissaan 8 tavua prosessidataa. (Canopen protocols. 2014)

4.3.2 CANopen-manager

CANopen manager on normaali solmu CAN-väylässä, joka sisältää myös verkonhallintaan liittyviä ominaisuuksia. Näitä ominaisuuksia ovat solmujen hallintaan käytettävät NMT, eli network management-viestit. CanOpen verkon käynnistyessä kaikki sen sisältämät laitteet siirtyvät ns. pre-operational-tilaan. Tässä tilassa olevat laitteet kykenevät ainoastaan vastaanottamaan verkonhallinta käskyjä sekä SDO, eli service data object -viestejä. SDO-viesteillä välitetään verkon solmuille muun muassa erilaisia konfigurointiin liittyviä parametreja. Solmujen asetusten asettamisen jälkeen laitteet voidaan käynnistää verkonhallintakäskyllä, jonka jälkeen laitteet siirtyvät operational-tilaan ja alkavat lähettämään sekä vastaan ottamaan varsinaista prosessidataa. Prosessidatan välittämiseen solmujen välillä käytetään PDO eli prosess data object-viestejä. (CanOpen heartbeat. 2014) Kuviossa 22 on kuvattu CanOpen-solmun tilakoneen eri tilat ja kuinka siirtyminen tilojen välillä tapahtuu.



KUVIO 22. CanOpen-solmun tilakone

Toinen tärkeä verkonhallintaan liittyvä ominaisuus on solmujen toiminnan jatkuva tarkkailu. Tähän tarkoitukseen on kehitetty kaksi eri protokollaa, jotka ovat nimeltään

hearbeat sekä nodeguarding. Molempien protokollien toiminta perustuun verkossa olevien solmujen tilan toistuvaan kyselemiseen. Mikäli solmu ei tietynajan sisällä vastaa kyselyyn, tulkitaan solmun olevan vikatilassa. Tämä ominaisuus on erittäin olennainen esimerkiksi turvakriittisissä komponenteissa, kuten autojen Airbag-järjestelmissä. Mahdollinen solmun tai verkon vikaantuminen tulee voida havaita välittömästi, ja estää vaaratilanteiden syntyminen. (CanOpen heartbeat. 2014)

4.3.3 CANopen-objektikirjasto

CANopenin ydin on sen objektikirjasto (ks. kuvio 23). Objektikirjastoa käytetään keskitettynä tietovarastona signaaleille ja parametreille. Kirjasto on rakenteeltaan jaettu alueisiin, joita ovat seuraavat: tietotyypit 0x0001- 0x0FFF, solmukohtaiset tunnisteet 0x1000 -0x1FFF, valmistaja ja sovelluskohtaiset 0x2000 -0x5FFF, 0x6000-0x67FF. (Canopen protocols. 2014)

Object Dictionary Layout

Index (hex)	Object
0000	Reserved
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reserved for further use
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardized Device Profile Area
A000-FFFF	Reserved for further use

KUVIO 23. CANOpen-objektikirjasto

4.4 VS-CAN-protokolla

VS-CAN-protokolla on Vision Systemsin kyseiseen projektiin kehittämä, CANopenia kevyempi tiedonsiirtoprotokolla. Päädyin käyttämään omaa protokollaa työkonenäytön sekä mobiilihydrauliikkaohjaimen välillä, säästääkseni aikaa varsinaisen sovelluksen tekemiseen. Kevyempi protokolla ei tarvitse erillisiä EDS-ajureita, vaan kaikki tarvittavat viestit on ennalta määritelty sitä kuvaavaan dokumenttiin. Viestit on ryhmitelty niiden käyttötarkoituksen mukaan ja jokainen ryhmä sisältää 8 tavua varsinaista prosessidataa. Tässä dokumentissa ei ole tarkoitus käydä läpi VS-CAN-protokollaa kokonaisuudessaan, vaan kuvata muutama sen tärkeimmistä viesteistä. Protokolla sisältää tällä hetkellä 25 eri COB-id:llä varustettua viestiä, joilla välitetään prosessidataa näytön ja mobiilihydrauliikkaohjaimen välillä.

Ensimmäisenä esimerkkinä tarkastella CAN-viestiä, jonka avulla voidaan välittää osa analogisista anturiviesteistä näytölle. Kuvio 24 on otettu VS-CAN-protokollan määrittelydokumentista. Kuviosta selviää, mitä prosessidataa viestissä siirretään ja mikä on viestin COB-id. COB-id numeron avulla määritetään, minne näytössä kyseisen viestin sisältämä prosessidata tallennetaan. Kuvasta selviää myös kuinka prosessidata on jaoteltu kahden tavun mittaisiin osiin. Tätä tietoa tarvitaan myös näytön sovelluksen tekemiseen, kun saapuvasta viestistä ”maskataan” halutun mittainen prosessidata. (VS-Can dokumentti 2014)

Sensor value 1

Viestin perustiedot

ID	**0x100**
Lähetys	syklinen, 100 ms
Tavuja	8
Väylä	CAN1

Viestin rakenne

Signaali	**Alku**	**Pituus**	**Selite**	**Tyyppi / yksikkö**
oilTemperature	0	16	Oljynlämpötila	Signed / °C
engineTemp	16	16	Moottorinlämpötila	Signed / °C
fuelLevel	32	16	Polttoaineenmäärä	Unsigned / %
oilLevel	48	16	Oljynmäärä	Unsigned / %

KUVIO 24. VS-CAN-protokolla, analogiaviestit

Toisena esimerkkinä tarkastellaan CAN-viestiä, jonka toiminta on myös keskeisessä osassa metsäkoneen toimintaa. Viestin COB-id on 0x402 ja sen tehtävänä on siirtää parametreja näytön sekä metsäkoneenohjaimen välillä (ks. kuvio 25). Viestin rakenne poikkeaa täysin edellisestä esimerkistä ja sen tulkitsemiseen on olemassa erillinen parametritaulukko (ks. liite 3). Viestin rakenne mukailee metsäkoneenohjaimen sisäistä parametrirakennetta, jossa jokainen parametri on yksilöity seuraavalla tavalla. Parametrit on jaoteltu ohjaimen muistiin eri ryhmiin ja aliryhmiin, joihin viitataan CAN-viestin ensimmäisellä ja toisella tavuilla. Kolmannella tavulla määritetään varsinainen parametrinumero, jonka jälkeen seuraa kahdentavun mittainen data. Kuudennella tavulla kerrotaan ohjaimelle halutaanko parametri lukea, kirjoittaa vai tallentaa ohjaimen muistiin. (VS-Can dokumentti 2014)

Request set device parameters

Viestin perustiedot

ID	**0x402**
Lähetys	syklinen, 100 ms
Tavuja	8
Väylä	CAN1

Viestin rakenne

Signaali	**Alku**	**Pituus**	**Selite**
requestGroup	0	8	Page
requestSubGroup	8	8	Index
requestParameter	16	8	Data LSB
requestDataLSB	24	8	Data LSB
requestDataMSB	32	8	Data MSB
Read / Write / Save	40	8	0 = Read 1 = Write 2 = Save All Values
parameterData6	48	8	Data tavu 6
parameterData7	56	8	Data tavu 7

KUVIO 25. VS-CAN-protokolla, parametrien kirjoitusviesti

5 Käyttöliittymän suunnittelu

Suunnittelun lähtökohtana oli korvata pienmetsäkoneen PC-pohjainen ohjausjärjestelmä uudella mobiilihydrauliikkaohjaimella sekä työkonenäytöllä. Oma roolini projektissa keskittyi näytön ulkoasun sekä toiminallisuuden suunniteluun ja sen toteutukseen.

Suunnittelu aloitettiin käymällä asiakkaan kanssa läpi ominaisuuksia, joita uuteen käyttöliittymään haluttiin. Vaatimukset kirjattiin muistiin, jonka jälkeen saimme käsityksen ohjelman laajuudesta. Tämän jälkeen vaatimuksista kirjattiin käyttötapaukset ja UML-kaaviot. Nämä dokumentit auttavat määrittämään ohjelmistoprojektin laajuuden sekä helpottavat ohjelmiston suunnittelua.

5.1 Vaatimusmäärittely

Graafisessa ulkoasun suunnittelussa tullaan käyttämään pohjana edellisen käyttöliittymän harmaasävyistä värimaisemaa sekä pelkistettyä ulkoasua. Käyttöliittymä tullaan toteuttamaan pääosin TTconrolin symbolikirjastosta löytyvillä valmiilla työkonesymboleilla sekä tarvittaessa itse piirretyillä lisäsymboleilla.

Käyttöliittymässä tulee olla päänäyttö, jota käytetään koneenajamiseen ja joka antaa informaatiota koneen toiminnasta. Päänäytön kautta tulee voida siirtyä työkonekohtaisille sivuille. Käyttöliittymän tulee sisältää selkeitä ja helppokäyttöisiä työkonekohtaisia sivuja, joiden kautta työkoneiden parametreja voidaan muuttaa. Työkonekohtaisia parametreja ovat esimerkiksi UW40-risuraivaimen pyörimisnopeuden esivalinta joko nopeaksi tai hitaaksi. Työkonekohtaisiin sivuihin tulee myös kuulua opastesivu koskien uusia koneenkäyttäjiä. Opastesivun tulee esittää selkeästi ohjaimien vaikutus valitun työkoneen toimintaan.

Näytön tulee voida tallentaa työkoneen käyttötunnit kolmella eri laskurilla. Laskureita ovat virrat päällä -laskuri, käyttötuntilaskuri ja aktiivisten työtuntienlaskuri. Virrat

päällä–laskurin tehtävänä on tallentaa aikaa, jonka koneen hydrauliikkaohjain on ollut käynnissä. Käyttötuntilaskurin tehtävänä on tallentaa aikaa, mikäli metsäkoneen moottori on käynnissä. Kolmannen tuntilaskurin tehtävänä on tallentaa aktiiviset käyttötunnit, jotka tarkoittavat tunteja, jolloin konetta on ajettu tai työlaitetta on käytetty.

Metsäkoneen toiminnasta tulee voida tallentaa lokia. Lokin tarkoituksena on parantaa mahdollisen vikatilanteen syntymisen selvittämistä. Lokiin tulee voida tallentaa metsäkoneen antureiden sekä venttiileiden tilat, joita voidaan myöhemmin tarkastella. Toinen tallennettava asia on mahdolliset vikakoodit. Vikakoodit tulee myös saada tallennettua näytön muistiin sekä saada luettua näytönkautta. Vikakoodit tulee olla selkokielisiä, jolloin huoltomiehen tai käyttäjän on helpompi ymmärtää mikä mahdollisen vian aiheuttaja on.

5.2 Vaatimukset

Seuraavassa on listattu kootusti kaikki kirjatut vaatimukset, jotka koskevat metsäkoneen uutta käyttöliittymää ja sen ominaisuuksia.

1) Ajovaihteen valinta tulee voida tehdä käyttöliittymän kautta.
2) Tuulilasinpyyhintä tulee voida ohjata käyttöliittymästä.
3) Kahvanlämmittimiä tulee voida ohjata käyttöliittymästä.
4) Käyttöliittymän tulee mahdollistaa peruutuskamerankäyttö.
5) Käyttöliittymän tulee sisältää kello ja kalenteri toiminnot.
6) Käyttöliittymässä tulee olla käyttötuntilaskurit (virrat päällä, kone käynnissä, aktiivinen työskentely ja huoltoväli).
7) Käyttöliittymän tulee tallentaa lokia koneen käytöstä.
8) Käyttöliittymästä tulee voida lukea mahdolliset vikakoodit sekä tallentaa ne.
9) Käyttöliittymästä tulee voida muuttaa käytettävää työkonetta.
    I. UW40
    II. UW40D
    III. UW180

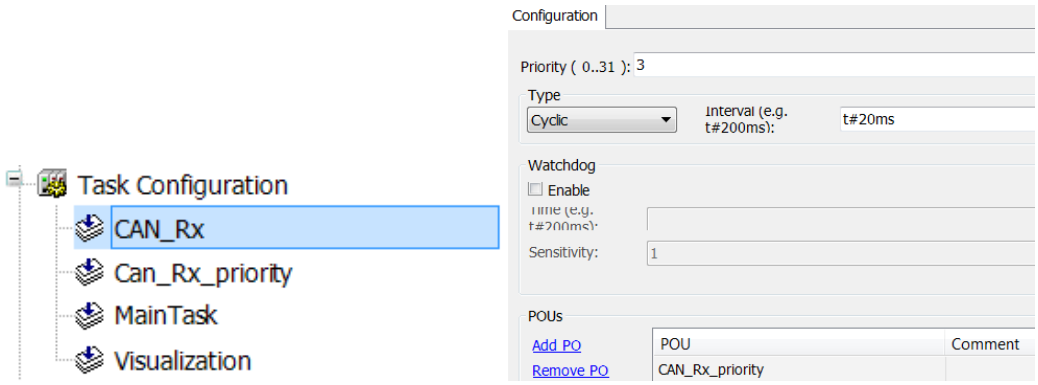
IV. UW160
V. MENSE

10) Käyttöliittymän tulee mukauttaa näkymää valitun työkoneen mukaan.
11) Käyttöliittymän tulee sisältää opastusnäyttö valitun työkoneen mukaan.
12) Käyttöliittymästä tulee voida muuttaa käytössä olevan työkoneen parametreja.
13) Käyttöliittymässä tulee olla toimintoja metsäkoneen I/O-testaamiseen.
14) Käyttöliittymässä tulee olla ominaisuus metsäkoneen parametrien muuttamiseen.
15) Parametrien lukemiseen ja muokkaamiseen on oltava kaksitasoinen salasana suojaus (huoltomies / käyttäjä).
16) Käyttötuntilaskurit on oltava tallessa sekä ECU:ssa että näytössä.
17) Kaikkien ECU:n parametrien on oltava tallennettuja myös näytölle.
18) Käyttöliittymän tulee esittää käytössä olevat versionumerot (ECU ja näyttö).

6 Käyttöliittymän toteutus

6.1 Yleiskuvaus käyttöliittymän toiminnasta

Käyttöliittymän toiminallisuus on jaoteltu Codesys 3.5:n TaskConfiguration-työkalulla neljään erilliseen taskiin, joille jokaiselle on määritelty oma prioriteetti sekä suoritussykli (ks. kuvio 26). Varsinainen näytön toiminta on jaoteltu kahteen itsenäiseen funktioon jotka ovat Main(prg) ja LoopCode(prg). Pääjako toimintojen ohjauksessa on se, että LoopCode ohjaa kaikkia grafiikan esittämiseen vaikuttavia toimintoja, kuten sivujen vaihtoa ja symbolien ohjausta. Main (prg) puolestaan vastaa kaikesta muusta loogisesta toiminnasta, kuten esimerkiksi lokitiedostojen kirjoituksesta ja erilaisten tuntilaskureiden pyörittämisestä.



KUVIO 26. Codesys 3.5, Task manager

CAN-viestien vastaanottoon metsäkoneen ohjaimelta näytölle on varattu kaksi erillistä funktiota. Nämä ovat nimeltään Can_Rx ja CAN_Rx Priotity. Syy tiedonsiirron jakamiseen kahteen eri funktioon oli varmistaa reaaliaikaisen tiedon viiveetön siirto. Reaaliaikaisella tiedolla tarkoitetaan esimerkiksi paineantureiden lähettämän tiedon esittämistä viiveettä näytöllä.

6.2 Tiedonsiirto

Tiedonsiirto työkonenäytön sekä mobiili hydrauliikkaohjaimen välillä on toteutettu käyttäen CAN-väylää. Codesys 3.5 -ohjelmointiympäristössä on mahdollista käyttää valmiiksi määriteltyjä CAN-slave laitteita, mikäli niille tarvittavat EDS-laite ajurit ovat saatavilla. Aiemmin tässä dokumentissa kuvatuista syistä johtuen, olin päättänyt toteuttaa CAN-liikennöinnin käyttäen omaa, hieman kevennettyä CAN-protokollaa. Tästä syystä kaikki CAN-väylään liittyvät määritykset on hoidettu itse, näihin tarkoitusiin rakennetuilla funktioilla. Väylän käyttöönotto ja sen nopeuden määritykset ovat toteutettu erillisessä CAN-väylän alustusfunktiossa (ks. liite 9). Funktio määrittelee väylänopeudeksi 250 kbit/s, viestikehyksen pituudeksi 11-bit, sekä kaikki sallittavat COB-id:t. Varsinaiseen tiedon vastaanottoon käytetään kahta eri funktiota, jotta vältytään viiveiltä tiedonsiirrossa. Ensimmäinen funktio on nimeltään CAN_Rx, jonka sykliseksi ajotaajuudeksi on määritelty TaskConfig-työkalussa 100 ms. Funktion tarkoituksena on vastaanottaa pienemmän prioriteetin prosessidataa, kuten metsäkoneen digitaali I/O-tietoja. Toinen CAN-viestien vastaanottamiseen luotu

funktio on nimeltään CAN_Rx_Priority (ks. liite 9). Sen suoritussykliksi on määritelty 20 ms ja prioriteetiksi korkeampi arvo kuin perus CAN_Rx-funktiolla. Viestien jakamisella kahteen eri funktioon saavutettiin näytön huomattavasti parempi toimivuus, mikä näkyy käyttäjälle analogisten näyttöjen sulavana liikkumisena. Käytännössä nopeasti muuttuvat työlaitteiden painearvot saatiin luettua paremmin näytön grafiikkaan ilman viiveitä.

```
(* ************************** Sensor value 1 ************************** *)

oilTemp         :INT:=0;            // CAN receive data BYTE[1] & BYTE[2]
engineTemp      :INT:=0;            // CAN receive data BYTE[3] & BYTE[4]
fuelLevel       :WORD:=0;           // CAN receive data BYTE[5] & BYTE[6]
oilLevel        :WORD:=0;           // CAN receive data BYTE[7] & BYTE[8]


(* ************************** Sensor value 2 ************************** *)

workPressure    :WORD:=0;           // CAN receive data BYTE[1] & BYTE[2]
hightPressure   :WORD:=0;           // CAN receive data BYTE[3] & BYTE[4]
pumpPressure    :WORD:=0;           // CAN receive data BYTE[5] & BYTE[6]
engineRpm       :WORD:=0;           // CAN receive data BYTE[7] & BYTE[8]
```

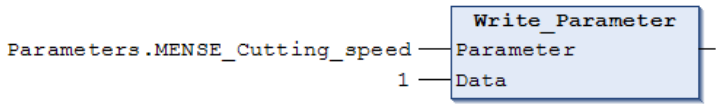
KUVIO 27. Codesys 3.5, muuttujien määritys

CAN-viestien lukemiseen näytöltä ohjaimelle on luotu useita eri funktioita. Näistä keskeisin on parametrien lukemiseen käytettävä Read_Parameters-funktio. Sen pääasiallinen tehtävä on lukea kaikki kone ja työlaiteparametrit laitteen käynnistyksen yhteydessä. Parametrien lukeminen metsäkoneen ohjaimelta tapahtuu lähettämällä erillinen kysely koskien haluttua parametria. Metsäkoneenohjain vastaa lähetettyyn kyselyyn, joka jälkeen parametrinarvo voidaan tallentaa näytön muistissa oleviin muuttujiin (ks. kuvio 27).

6.3 Parametrit

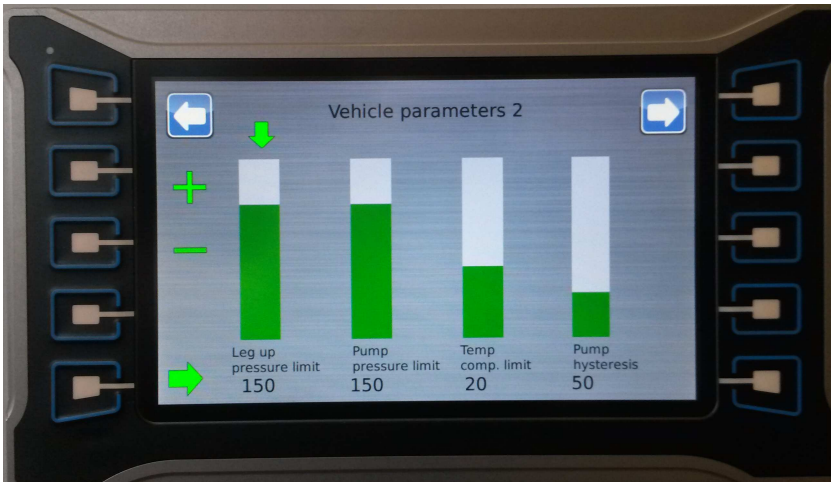
Metsäkoneen toiminnan kannalta ehkä kaikkein keskeisin asian on erilaisten metsäkoneen ja sen työlaitteita koskevat parametrit. Parametrien kirjoittaminen ja lukemine perustuu VS-CAN dokumentissa kuvattuun tapaan siirtää parametreja näytön ja hydrauliikkaohjaimen välillä. Parametrien kirjoittamiseen käytetään siihen tarkoitukseen tehtyä funktiota. Funktiolle annetaan arvoina parametri joka halutaan

muuttaa, sekä parametrille kirjotettava arvo. Parametrien kirjoitusfunktio on esitetty kuviossa 28, käyttäen FBD-esitystapaa. Tarkempikuvaus funktionrakenteesta on kuvattu liitteessä 6, liite sisältää funktion ohjelmakoodin kokonaisuudessaan. Lisäksi liitteessä 3 on havainnollistettu kuinka eri parametrit ovat sijoittuneet näytön FRAM-muistiin. Liitteestä selviää myös miten eri parametreihin viitattaan, käytettäessä kyseistä funktiota.



KUVIO 28. Parametrienkirjoitusfunktio

Parametrien kirjoittamiseen näytöltä metsäkoneenohjaimelle on luotu useita eri näyttösivuja, joista seuraavassa esitellään hyvä esimerkki.



KUVIO 29. Metsäkoneen parametrisivu

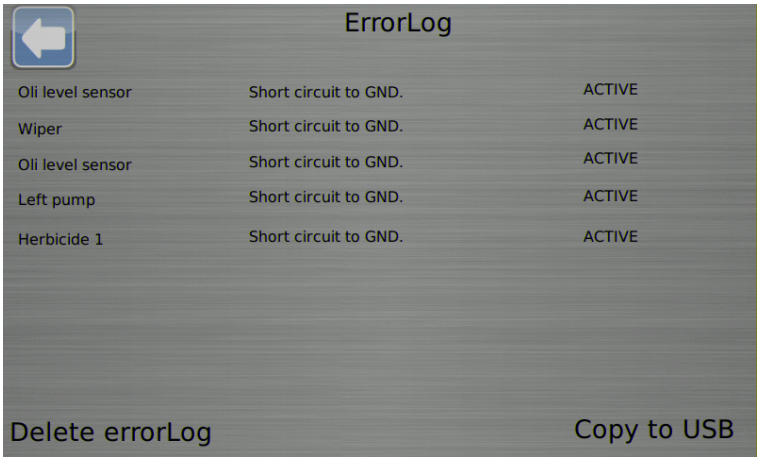
Kuviossa 29 on nähtävissä metsäkoneen parametrien asetussivu. Sivun kautta voidaan muuttaa koneen eri painerajoja sekä määrittää hydrauliikkaöljyn alalämpötilaraja. Lämpötilarajalla estetään metsäkoneentoiminta täydellä teholla, mikäli hydrauliikkaöljy on liian jäykkää.

Valinnat tehdään näytönsivuilla olevilla painonapeilla. Valittu parametri osoitetaan selkeästi nuolisymbolilla valitun parametrin yläpuolella. Valitun parametrin

muuttaminen tapahtuu plus ja miinus ikonien viereisillä painonapeilla. Muuttunut parametrin arvo ilmaistaan sekä pylväspalkilla että numeerisessa muodossa palkin alapuolella. Parametrinvalinta välitetään VS-CAN-protokollan mukaan metsäkoneen ohjaimelle, joka muuttaa toimintaansa valittujen parametrien mukaan.

6.4 Vikakoodit

Metsäkoneen mahdollinen vikaantuminen aiheuttaa vikakoodin tulostumisen näytön vikakoodisivulle, joka on nähtävissä kuviossa 30. Vikakoodi saapuu metsäkoneen ohjaimelta numeerisessa muodossa, joka muutettaan näytönpäässä selkokieliseksi viestiksi. Viesti sisältää tiedon vikaantuneesta kohteesta ja sen aiheuttajasta. Vikaantumisen syy voi olla esimerkiksi oikosulussa oleva ohjausventtiili tai viallinen anturi. Vikakoodit on luettavissa suoraan metsäkoneen käyttöliittymästä sekä tallennettavissa USB-muistitikulle, myöhempää tarkastelua varten. Vikakoodien tallennus formaattina käytetään XML-tiedostoa.



KUVIO 30. Metsäkoneen vikakoodit

6.5 Käyttötapaloki

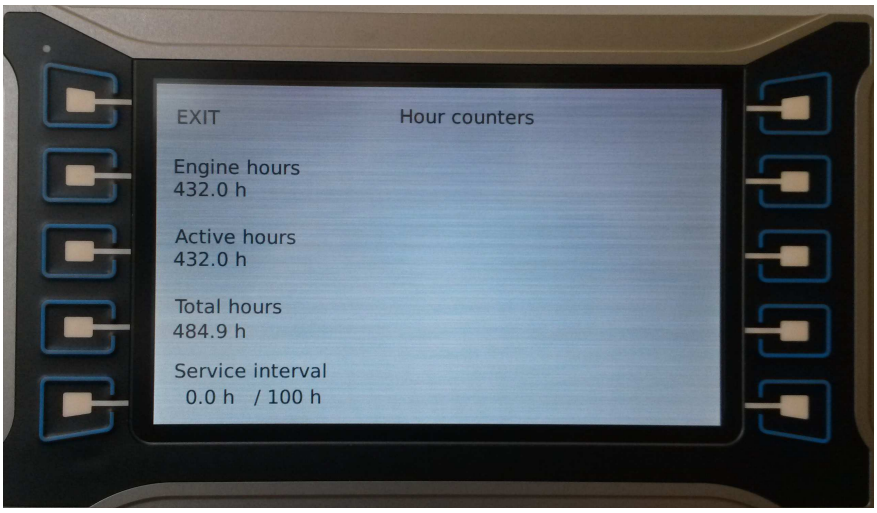
Metsäkoneen toimintaa seurataan erillisellä lokilla, jonka tarkoituksena on tallentaa tietoa koneen toiminnasta ajonaikana. Lokiin tallennetaan tilatietoja metsäkoneen magneettiventtiilien asennoista ja antureiden antamasta datasta. Lokia voidaan

hyödyntää mahdollista vikaa selvitettäessä ja se on myös tallennettavissa USB-muistitikulle. Käyttötapalokin tallennusformaattina on XML-formaatti, joka on avattavissa esimerkiksi Excel-ohjelmistolla. Lokiin tallentuva aikaleima saadaan näytön sisältämästä reaaliaikakellosta, mikä helpottaan mahdollisen vian alkamisajankohdan selvittämistä. Käyttötapalokin kirjoitus XML-tiedostoon on kuvattu osittain liitteessä 7. Lisäksi liitteessä 10 on esimerkki, miltä lokitiedosto näyttää avattuna Excel-ohjelmistolla.

6.6 Tuntilaskurit

Metsäkoneen käyttöliittymään on rakennettu neljä erillistä tuntilaskuria (ks. kuvio 31). Laskureiden tehtävänä on tallentaa muistiin tietoa koneella ajetuista tunneista. Ensimmäisen laskuria käytetään metsäkoneen ohjaimen käynnissä oloajan tallentamiseen. Toisen laskurin tehtävänä on kerätä talteen tunnit metsäkoneen käynnissä oloajasta. Tämän laskurin arvo tulostetaan koneen päänäyttöön josta selviään koneella ajetut tunnit. Kolmatta laskuria käytetään aktiivisten ajotuntien tallentamiseen. Aktiivisilla ajotunneilla tarkoitetaan tunteja, jolloin koneenkuljettaja on ohjannut jotain koneen venttiiliä eli kun koneella on oikeasti tehty töitä. Viimeisen laskurin tehtävänä on toimia huoltolaskurina, joka muistuttaa koneenkäyttäjää määräaikaishuolloista.

Kaikkien laskureiden toiminta perustuu näytön sisältämän reaaliaikakellon antaman ajan hyödyntämiseen. Lasketut tunnit on tallennettu näytön ominaisuuksia kuvaavassa luvussa esiteltyyn FRAM-muistiin, joka on erittäin luotettava ja haihtumaton muistialue. Käyttötuntien laskemiseen, niiden tallentamiseen sekä vertailuun metsäkoneenohjaimen sisältämiin tuntimääriin on ohjelmoitu useita eri funktioita. Funktioiden toimintaa ei kuvata tarkemmin tässä dokumentissa, jotta mahdolliset väärinkäytökset tuntilaskurien käytössä voidaan ehkäistä.



KUVIO 31. Metsäkoneen tuntilaskurit

6.7 Näytön ohjaaminen

Kuten aiemmin on todettu, näytön grafiikanohjaaminen ja varsinaiset loogiset-toiminnot ovat jaettu näytössä omiin ohjelmiinsa. Näppäimistön lukeminen, parametrien kirjoittaminen tai erilaisten ajastimien käyttäminen tapahtuvat MAIN (prg) ohjelmalohkossa. Kaikki näytöngrafiikkaan liittyvät toiminnot, kuten erilaisten ikonien näyttäminen puolestaan tapahtuu LoopCode (prg) ohjelmassa. Seuraavaksi käydään läpi yhden näytönsivun toiminnot kuvattuna sekä MAIN (prg) että LoopCode (prg) ohjelmia tarkastellen. Esimerkkisivuksi on valittu kuviossa 29 näkyvä metsäkoneen asetussivu.

Parametrien muuttamiseen käytetään näytön reunoilla olevia painonappeja. Aluksi muutettava parametri valitaan näytön vasemmanpuoleisella alanapilla. Valinnan jälkeen valitun parametrin arvoa voidaan joko kasvattaa tai pienentää, käyttäen näytön näppäimiä 2 tai 3. Liitteessä 6 on kuvattu osa MAIN (prg) ohjelman kyseiseen näyttösivuun liittyvästä ohjelmakoodista. Liitteestä 6 on nähtävillä kuinka esimerkiksi näytön 2-painikkeen painaminen suurentaa valitun parametrin arvoa 5-yksiköllä. Parametrin arvon muuttamisen jälkeen kutsutaan parametrin kirjoitusfunktiota, joka lähettää metsäkoneen ohjaimelle muuttuneen parametrinarvon, käyttäen CAN-väylää.

Kuviossa 29 oleva parametrien asetussivun grafiikanohjaaminen on puolestaan kuvattu liitteessä 7. Valitun parametrin osoittavaa nuoli-ikonia ohjataan käyttämällä SetPosition-metodia sekä antamalla sille parametrina koordinaatti, johon nuolen halutaan siirtyvän. Pylväspalkkien ohjaamiseen puolestaan käytetään SetValue-funktiota. Funktiolle parametrina asetettu arvo ohjaa palkinkorkeutta. Palkin minimi, maksimi ja oletusarvo asetellaan TTC vizualization manager-valikon avulla.

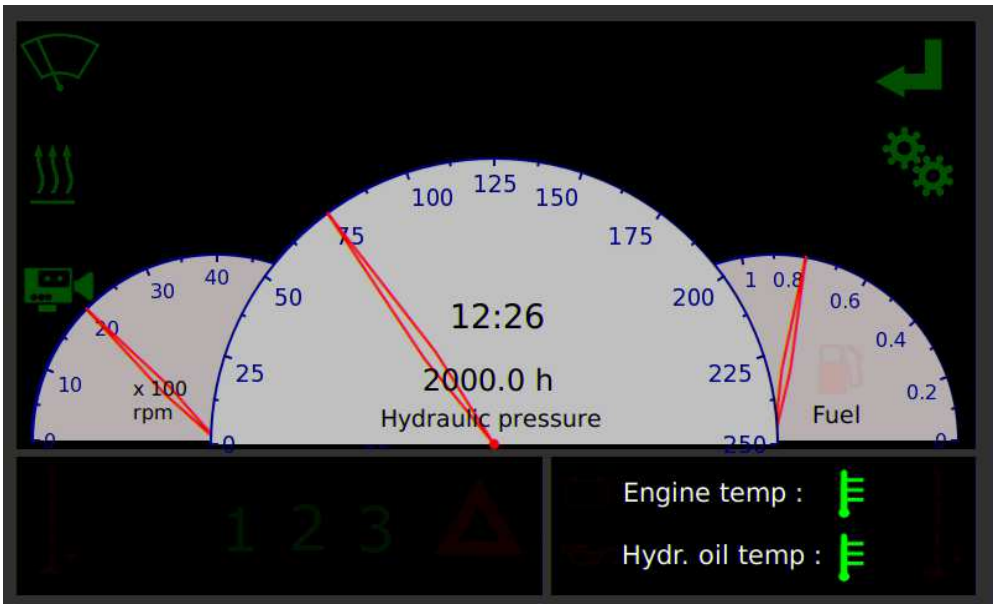
Viimeisenä ominaisuutena esitellään asetusnäytön yläkulmissa olevien sivunvaihtoon käytettävien nuoli-ikonien toiminta. Näytön sivujen selailuun käytettävät nuolet kirkastuvat kun ikonin viereistä painonappia painetaan. Tämä ominaisuus on saatu aikaan käyttämällä liitteessä 7 kuvattu SetOpacity-funktiota. Funktiolla ohjataan valitun ikonin läpinäkyvyyttä asteikolla 0–255.

6.8 Käyttöliittymän rakenne

6.8.1 Päänäyttö

Käyttöliittymän perusta on sen päänäyttö (ks. kuvio 32), jonka tarkoitus on kertoa koneenkäyttäjälle kaikki olennainen koneen toiminnasta. Näyttö sisältää kolme analogista, selkeästi luettavia mittaria, jotka näyttävät moottorin kierrosnopeuden, hydrauliikan työpaineen sekä polttoainemäärän. Näytön alalaidasta löytyvät symbolit valitun vaihteen esittämiseen, tukijalkojen varoitussymbolit ja metsäkoneen moottorin ja hydrauliikkaöljyn lämpötilaa kuvaavat ikonit. Ruudun vasemmassa laidassa olevat ikonit kertovat tuulilasinpyyhkimen ja kahvojenlämmittimen aktivoitumisesta.

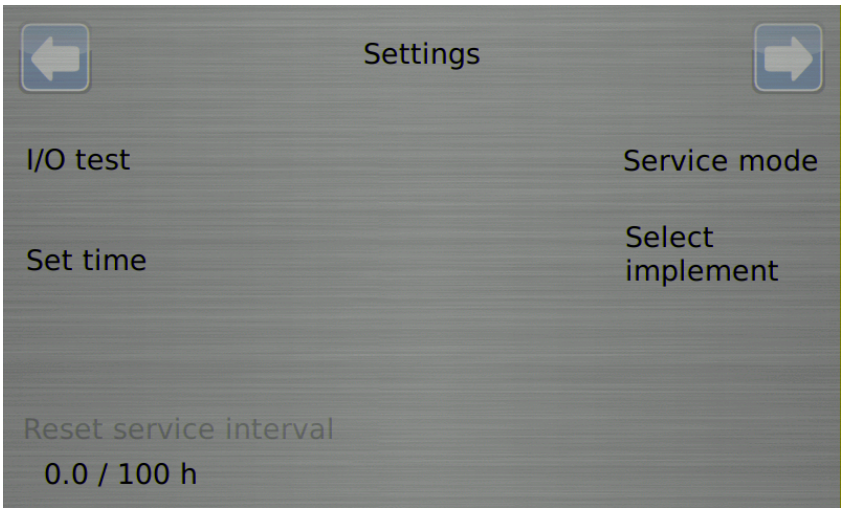
Ikonit ovat sijoiteltu selkeästi vastaamaan näytönlaidalla olevia painonappeja. Vasemmassa laidassa näkyvä kameraikoni ilmestyy näyttöön, mikäli metsäkoneeseen on asennettu peruutuskamera sekä se on aktivoitu kokoonpanoasetuksista. Ikonin viereistä painonappia painettaessa siirrytään peruutuskameranäkymään.



KUVIO 32. Metsäkoneen päänäyttö

6.8.2 Asetukset

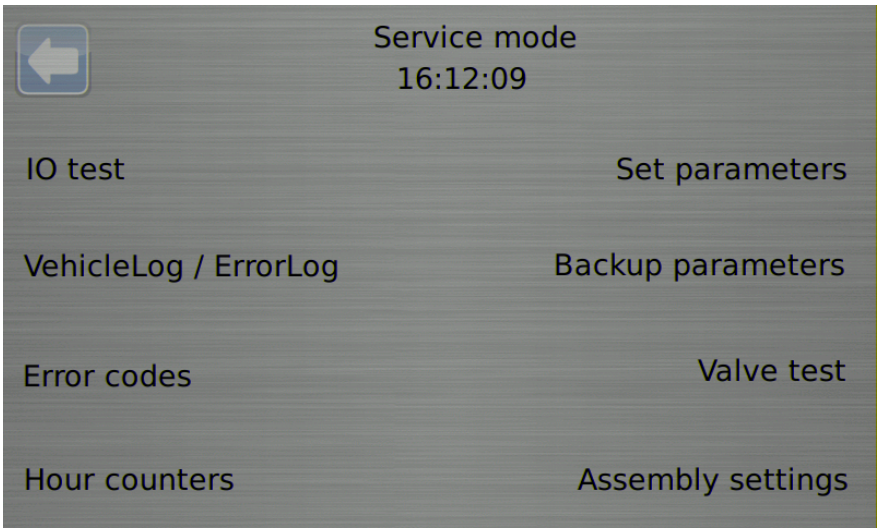
Päänäytön oikeassa ylälaidassa näkyvä ratasikoni avaa käyttäjälle asetussivun (ks. kuvio 33), jonka kautta koneen perusasetuksia voidaan muokata. Sivunkautta voidaan myös vaihtaa käytettävissä olevaa työkonetta. Koneen tehdasasetusten muokkaamiseen ja joidenkin lokitiedostojen lukemiseen käytetään huoltovalikkoa. Tähän valikkoon siirryttäessä käyttäjältä kysytään PIN-koodi, jolla estetään asiaton asetusten muokkaaminen.



KUVIO 33. Metsäkoneen asetukset.

6.8.1 Huoltonäkymä

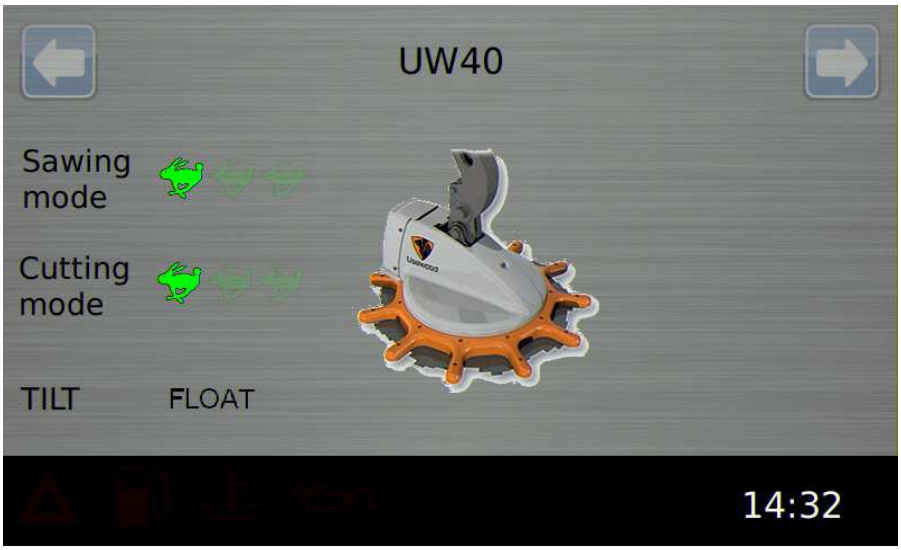
Huoltotilan (ks. kuvio 34) tarkoituksena on helpottaa metsäkoneen mahdollisen vian selvittämistä. Käytössä on työkaluja muun muassa metsäkoneen IO-testaamiseen sekä vika ja käyttötietojen selaamiseen. Huoltotilan kautta voidaan myös muuttaa koneen tehdasasetuksia sekä palauttaa koneen oletusparametrit.



KUVIO 34. Metsäkoneen huoltonäkymä

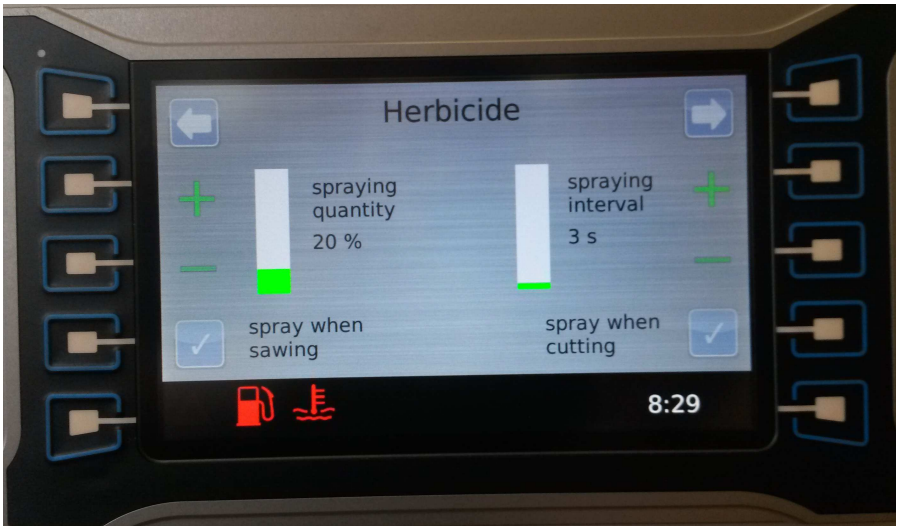
6.8.2 Työlaitenäkymät

Tehojätkä-pienmetsäkoneeseen on saatavilla useita eri työlaitteita, joilla jokaisella on omat toimintonsa ja asetuksensa. Käyttöliittymän perustoiminnallisuuksiin kuuluu oma yksilöllinen työkonenäkymä jokaiselle työlaitteelle, jonka kautta voidaan hallita valitun laitteen toimintaa. Toimintojen valinta tapahtuu näytönreunoilla olevien painonappien avulla. Toimintopyyntö välitetään näytöltä metsäkoneen ohjaimelle, joka suorittaa tarvittavat toimenpiteet. Kuviossa 35 on esitelty UW-40 risuraivaimen työlaitenäkymä. UW40-risuraivaimen raivausterän nopeuden esivalinta voidaan muuttaa näytön kautta, painamalla symbolin vieressä olevaa painonappia. Pyörintänopeus esitetään kolmella jänissymbolilla valitun nopeuden mukaan. Lisäksi Työlaiteasetuksiin kuuluu terän kallistuksen salliminen tai sen estäminen. Tämän asetuksen muuttaminen tapahtuu TILT-valinnan kautta.



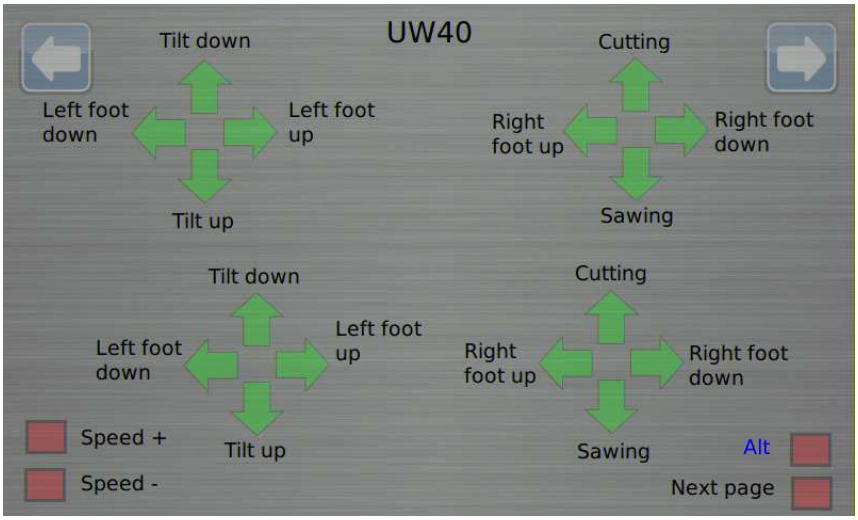
KUVIO 35. UW-40 työlaitenäkymä

Mikäli metsäkoneeseen on asennettu kasvinsuojeluruisku, voidaan sen parametreja muuttaa omalta sivultaan (ks. kuvio 36). Kasvinsuojeluruiskun käyttöikkunan käyttäminen edellyttää ruiskun aktivoimista tehdasasetuksista sekä työlaitesivun oikean ylänuolen painamista. Valittavina parametreina ovat ruiskun työsylinterin liikematkan ja työskentelytaajuuden muuttaminen sekä ruiskun automaattinen aktivointi terän pyöriessä. Parametrien muuttamiseen käytetään plus ja miinus-painikkeita. Kasvinsuojeluruiskun asetukset ovat selkeästi esitettyinä sekä graafisesti että numeerisessa muodossa.



KUVIO 36. Kasvinsuojeluruiskun asetukset

Metsäkoneen uusille käyttäjille on luotu työlaitteen käyttämistä helpottavia ohjesivuja (ks. kuvio 37). Ohjesivulle siirtyminen tapahtuu näytön oikean ylänuolen painamisella, joka avaa kuvan mukaisen näkymän. Näytössä olevat nuolet kuvaavat metsäkoneen ohjaimia ja niiden symbolit kirkastuvat valitun toiminnon mukaan. Ohjesivun sisältö luonnollisesti muuttuu, kun työlaitetta vaihdetaan.



KUVIO 37. Työlaitekohtainen ohjesivu

7 Käyttöliittymän testaaminen

7.1 Testausympäristö

Metsäkoneen ja sen käyttöliittymän testaamisen helpottamiseksi projektissa rakennettiin testausympäristö (ks. kuvio 38). Sen tarkoituksena on mallintaa mahdollisimman hyvin oikeaa metsäkonetta. Testausympäristö sisältää lähes kaikki normaalin metsäkoneen anturit, sen ohjaamiseen käytettävät ohjaimet sekä lähtöihin liitetyt magneettiventtiilit. Projektin edetessä testausympäristö onkin osoittautunut todella hyödylliseksi. Sen avulla metsäkoneen hydrauliikkaohjaimen ja näytön välistä kommunikointia on ollut todella hyvä testata. Huolellisilla testaamisella vältytään usein ohjelmointivirheiltä, jotka paljastuvat vasta normaalissa koneenkäytössä.

KUVIO 38. Metsäkoneen testausympäristö

7.2 Sovelluksen testaaminen

Metsäkoneen hydrauliikkaohjaimen ja näytöntoiminta on testattu ennen jokaista virallista julkaisua, käyttäen siihen tarkoitukseen valmistettua testausympäristöä. Näytön sovellus on rakennettu toiminto kerrallaan ja sen toimintaa on testattu aina jokaisen uuden toiminnon valmistuttua. Testaaminen on suoritettu ”ajamalla” metsäkonetta testiympäristössä ja käymällä läpi kaikki koneen normaalit toiminnot. Toimintoihin kuuluvat muun muassa tarkistaa kaikkien IO-kanavien looginen toiminta sekä näytön ja ohjaimen välisten parametrien kirjoitus. Mikäli kaikki testit saadaan suoritettua onnistuneesti, voidaan sovellus siirtää varsinaiseen metsäkoneeseen. Metsäkoneessa sovelluksen toimintojen testaamista jatkavat Usewoodin ammattikuljettajat, joiden palautteen avulla sovelluksen toimintaa voidaan edelleen kehittää.

7.3 Suoritetut testit

7.3.1 Metsäkoneen ajaminen ja työlaitteet

Käyttöliittymän oikeanlainen toiminta konetta ajettaessa testattiin simuloimalla oikeaa ajotilannetta, käyttäen apuna testausympäristöä. Testauksen tarkoituksena oli ajaa metsäkonetta käyden läpi kaikki sen keskeisimmät toiminnot. Toimintojen oikeanlainen käyttäytyminen voitiin havaita tarkastelemalla työlaitenäytön päänäyttöä sekä testausympäristöön liitettyjä magneettiventtiilejä. Metsäkoneen ohjainten liikuttaminen sai aikaan joidenkin magneettiventtiileiden avautumisen. I/O-listaa apuna käyttäen voitiin varmistua lähtöjen oikeanlaisesta toiminata. Ajonäytön toiminnassa keskityttiin sen grafiikan oikeanlaiseen käyttäytymiseen. Tarkastelun kohteena toimiva sen eri analogiamittarit sekä muut metsäkoneentoiminnasta kertovat ikonit.

Metsäkoneen näytön toiminta testattiin kaikilla siihen liitettävissä olevilla työlaitteilla. Työlaitekohtaisten parametrien oikeanlaisesta toiminnasta voitiin varmistua vai käymällä jokainen työlaite läpi yksittäin. Työlaitekohtaisia parametreja muuttamalla ja venttiilien toimintaa seuraamalla, voitiin varmistua työlaitekohtaistenparametrien oikeanlaisesta vaikutuksesta työlaitteen toimintaan.

7.3.2 Parametrien kirjoittaminen ja lukeminen

Metsäkoneen käyttöliittymä sisältää useita parametrien asetussivuja. Niiden testaamiseksi käytettiin paljon aikaa ja jokaisen parametrin vaikutus tarkastettiin ennen kuin sovelluksen versio voitiin hyväksyä. Muiden kuin suoraan työlaitteisiin liittyvien parametrien oikein kirjoittuminen metsäkoneenohjaimelle, varmistettiin käyttäen apuna Bosch Bodas Service tool-työkalua. Tämän työkalun avulla voidaan muuttaa sekä nähdä kaikki metsäkoneenohjaimen parametrit. Testaus suoritettiin muuttamalla käyttöliittymästä metsäkonekoneen parametreja, jonka jälkeen muutos käytiin tarkistamassa Bodas Service-huoltotyökalulla. (Bosch Bodas Service tool 2014)

7.3.3 Käyttötapalokin kirjoitus

Metsäkoneen käyttötapalokin toiminta testattiin normaalin koneenkäytön yhteydessä. Metsäkoneella ajettiin usean tunnin mittainen käytännön testijakso. Tämän jälkeen tallentunut käyttötapaloki siirrettiin metsäkoneen käyttöliittymästä USB-muistitikulle, ja avattiin XML-tiedostoformaatin ymmärtävällä tekstieditorilla. Lisäksi lokitiedosto avattiin Excel-ohjelmistolla (ks. liite 10), ja todettiin lokitiedoston toimivuus.

7.4 Vikatilanteen

Metsäkoneen käyttöliittymää on kehitetty jo useiden versioiden ajan, ja aina sen toiminta ei ole ollut aivan moitteetonta. Suurimpia kompastuskiviä ja ohjelmiston virheellistä toimintaa ovat aiheuttaneet FRAM-muistiin tallentaminen ja erilaiset parametrien kirjoitus ongelmat. Parametrien kirjoittamiseen ja niiden lukemiseen onkin rakennettu tämän projektin edetessä useita eri funktioita. Tässä dokumentissa kuvattu parametrien kirjoitustapa, ja liitteessä 3. kuvattu parametrirakenne on viimein osoittautunut hyvin toimivaksi ratkaisuksi. Myös näyttöön liitettävän kamerasignaalin näyttäminen näytöllä on aiheuttanut todella paljon virhetilanteita. Ongelmien selvittämistä on vaikeuttanut näytön ohjaamiseen tarkoitettujen funktioiden suppeat toimintakuvaukset.

8 Pohdinta

8.1 Asiakkaan palaute

Parhaan kuvan metsäkoneen ja sen näytön toiminnasta saa kun kuuntelee ammatikseen metsäkonetta ajavien henkilöiden palautetta. Palautteen perusteella onkin tehty paljon parannuksia metsäkoneenohjaimen ja sen työkonenäytön toimintoihin. Seuraava teksti on suora lainaus projektin loppuasiakkaalta, Jussi-Pekka Useniukselta, koskien näytön toimintaa ja sen käytettävyyttä.

”Käyttäjän kommenteiksi sanoisin, että näytön ja koko Tj5-ohjainjärjestelmän toiminta on ollut luotettavaa ja vikaantuminen on ollut minimaalista. Näyttö on selkeä. Toiminnot ovat loogisesti havaittavissa, eikä näytön käyttö edellytä erillistä kouluttautumista. Virheelliset näppäilyt on harvinaisia, eikä niistä aiheudu toiminnallista haittaa. Näyttö on yksinkertainen, eikä sen tutkimiseen tai selailuun kulu ylimääräistä työaikaa. Koneen käyttöhenkilöstölle tarvittu näyttöyksikön käyttäjätuen tarve on ollut minimaalista ja rajoittunut oikeastaan vain service-tason toimintoihin esim. i/o-informaation tulkintaan ja venttiilien käsinohjaukseen. Joitakin toimintahäiriöitä on aiheuttanut näytön ja ecu:n ohjelmaversiohallinnan koordinointivirheet, jolloin näyttö ja ecu eivät ole kommunikoineet oikein ja käyttäjälle on ilmaantunut koneen toimintahäiriö, joka on vaatinut huoltomiehen käynnin maastossa ja ohjelmiston päivityksen.”

8.2 Parannusehdotukset

Työkonenäytön toimintaa voidaan parantaa monilla eri tavoilla. Yksi käyttäjille näkyvä selkeä parannus olisi monikielisyyden lisääminen näyttöön. Työmäärällisesti tämä päivitys on kuitenkin niin iso urakka, että sitä ei ole vielä toteutettu. Monikielisyys lisäisi koneen käytettävyyttä ja avaisi varmasti markkinoita myös uusiin maihin.

Toinen parannusehdotus liittyy metsäkoneen ohjaimen ja näytön väliseen versioidenhallintaan. Tällä hetkellä kentällä on olemassa monia tehojätkä-pienmetsäkoneita, joiden näytön ja metsäkoneen sovellus voitaisiin päivittää uuteen versioon. Valitettavasti tämänhetkinen näytönsovellus ei ota kantaa metsäkoneenohjaimen sovelluksenversioon. Mikäli näytön sovellus päivitetään ilman metsäkoneenohjaimen päivittämistä, seuraa siitä ristiriita parametrien kirjoittamiseen ja lukemiseen. Tähän ongelmaan tulee kiinnittää erityistä huomiota metsäkonetta päivitettäessä, ennen kuin tähän ominaisuuteen on tehty korjaus.

GPS / GSM-laajennuksen lisääminen näyttöön mahdollistaisi monien uusien ominaisuuksien lisäämiseen tämänhetkiseen käyttöliittymään. Paikkatietoa apuna käyttäen olisi esimerkiksi mahdollista tallentaa tietoa metsäkoneella raivatusta pinta-alasta. Pinta-alatietoa vois käyttää suoraan laskutus perusteena. Koneen paikkatieto voitaisiin siirtää myös GSM-verkon avulla vastaanottajalle. Tämä ominaisuus mahdollistaisi metsäkoneen etäpaikannuksen. Paikkatietoa voidaan käyttää myös mahdollisesti anastetun koneen jäljittämiseen. GSM-yhteyttä voidaan käyttää myös metsäkoneen mahdollisen vikatilan selvittämiseen etäyhteydellä. Vikakoodien ja koneen käyttötapa lokitiedostojen lukeminen etäyhteydellä helpottaisi huoltomiesten työtä.

Metsäkoneen käyttöliittymän graafinen ulkoasu on toteutettu tällä hetkellä käyttäen suurimmaksi osaksi kehitysympäristön mukana tulleita valmiita symboleita. Käyttöliittymän ulkoasua voitaisiin parantaa, antamalla graafisen suunnittelijan piirtää siihen yksilöllisempiä symboleita. Lisäksi esimerkiksi päänäytön mittaristo voitaisiin piirtää paremmin vastaamaan juuri kyseistä tuotetta.

8.3 Oma arvio opinnäytetyöstä

Projektin alkaessa alkukesällä 2013, olin juuri tullut harjoittelijaksi Vision Systems oy:n. Minulla ei ollut aikaisempaa kokemusta CAN-väylästä, metsäkoneen näytöksi valitusta näyttöpaneelista, eikä Codesys 3.5-ohjelmointiympäristöstä. Opinnäytetyön tekeminen on ollutkin mitä suurimmaksi osaksi uusien asioiden opiskelua, mikä on ollut erittäin mielenkiintoista. Toisaalta täysin uusien asioiden opiskelu on vienyt yllättävän paljon aikaa.

Roolini projektissa oli selkeä, se keskittyi metsäkoneen käyttöliittymän suunnitteluun ja sen toteuttamiseen. Olin yksi kolmihenkisestä projektiryhmästä, jonka muiden jäsenten vastuulle jäivät koneen uuden sähköjärjestelmän suunnittelu sekä sen sisältämän hydrauliikkaohjaimen sovelluskehitys. Omalta osaltani projekti alkoi vanhaan ohjausjärjestelmään tutustumisella sekä määrittämällä asiakkaan kanssa

käyttöliittymän uudet vaatimukset. Vaatimusten pohjalta aloitin tutustumaan täysin uuteen näyttöpaneeliin, jonka toiminnasta ja sen mahdollisuuksista ei minulla ollut aikaisempaa kokemusta. Varsinainen käyttöliittymän toteutus aloitettiin määrittämällä rajapinta näytön ja metsäkoneenohjaimen välille. Tähän tarkoitukseen määritimme VS-CAN protokollan, jonka dokumentin pohjalta näyttöön toteutettiin tiedonsiirto ohjelmalohkot. Tiedonsiirron toimittua pääsin rakentamaan näyttöön varsinaisia sivuja, joita nykyisessä ohjelmaversiossa on jo yli 40.

Näytönsovellus ja sen eri ominaisuudet on rakennettu vaiheittain, testaten huolellisesti jokainen muutos, ennen seuraavan ominaisuuden lisäämistä. Näytön sovelluskehityksessä hyvänä apuna on toiminut projektin aikana valmistettu testausympäristö. Testausympäristön avulla on voitu varmentaa sovelluksen toiminta, ennen kuin se on luovutettu varsinaisiin kenttätesteihin.

Omasta mielestäni tämä opinnäytetyö onnistui erittäin hyvin. Käyttöliittymästä saatiin tehtyä vaatimukset täyttävä ja sarjatuotantoon kelpaava. Vaikka työ olikin erittäin haasteellinen ja sisälsi paljon uusia opiskeltavia asioita, oli sen tekeminen todella mielekästä. Käyttöliittymän tämänhetkinen rakenne on mielestäni selkeä ja helppokäyttöinen, niin kuin työkoneen käyttöliittymän tuleekin olla. Käyttöliittymään saatiin toteutettua monia ohjelmallisesti saastavia toimintoja, kuten metsäkoneen toimintaa seuraavien lokitiedostojen tallentaminen ja useita erilaisia parametrien tallennus ja kirjoitus toimintoja. Näytön ohjelmiston kehitys alkoi alkukesästä 2013 ja sen tuotekehitys jatkuu edelleen. Näyttöön voidaan lisätä vielä monia mielenkiintoisia ominaisuuksia, joita kuvattiin parannusehdotukset osioissa.

Tämän opinnäytetyön tuloksena saatiin Tehojätkä-pienmetsäkoneeseen valmistettua vaatimukset täyttävä ja toimiva käyttöliittymä. Käyttöliittymä on tälläkin hetkellä käytössä kaikissa uusissa Usewood oy:n valmistamissa metsäkoneissa, ja sen tuotekehitys jatkuu edelleen.

Lähteet

Bosch Bodas Service tool. 2014. Tietoa ohjelmistosta. Viitattu 10.11.2014 http://www.boschrexroth.com/mobile-hydraulics/catalog/ Vornavigation/ Vornavi.cfm?Language=EN&PageID=m4514

CanOpen,NMT. 2014. Tietoa CanOpen-protokollan käyttämästä verkonhallinta ominaisuudesta. Viitattu 30.8.2014.http://www.canopensolutions.com/ english/about_canopen/canopen-management.shtml

CanOpen heartbeat. 2014.Tietoa Canopen-protokollan viantunnistuksesta. Viitattu 30.8.2014.http://www.canopensolutions.com/english/about_canopen/ guarding_heartbeat.shtm

Canopen protocols. 2014. CiA-verkkosivun kuvaus CANpen-protokollan rakenteesta. Viitattu 30.8.2014.http://www.can-cia.org/index.php?id =systemdesign-canopen-protocol

Controller Area Network. 2014. CiA-verkkosivun kuvaus CAN-väylän rakenteesta. Viitattu 30.8.2014. http://www.can-cia.org/index.php?id=systemdesign-can

Codesys 3.5 online help. 2014. Ohjelman sisäinen ohjelmointi ja käyttöopas. Viitattu 30.8.2014.

Codesys 3.5 -ohjelmointiympäristö. 2014. Ohjeita ohjelmointiympäristön käyttämiseen. Viitattu 30.8.2014. http://www.codesys.com/products/ codesys-engineering/development-system.htm

Hy-evision 7.0 user manual. 2014. Työkonennäytön käyttöohje. Sisältyy TTcontrolin asennustiedostoon. Viitattu 30.8.2014. http://www.ttcontrol.com/fileadmin/content/off-highway/files/secure/servicearea/Vision2/HY-eVision__7.0_User_Manual_v1.23.pdf

IEC_61131-3 standardi. 2014. IEC 61131-3-ohjelmointistandardin kuvaus. Viitattu 30.8.2014. http://www.plcopen.org/pages/tc1_standards/iec_61131_3/

Koneet. 2014. Usewood Oy:n tuotteiden esittely. Viitattu 30.8.2014. http://www.usewood.fi/index.php/fi/koneet

Non-return-to-zero. 2014.Wikipedian artikkeli NRZ-linjakoodauksesta.Viitattu 30.8.2014. http://en.wikipedia.org/wiki/Non-return-to-zero

Saha, H. Can-väylän toiminta. N.d. Artikkeli canopen.fi-sivustolla. Viitattu 30.8.2014. http://www.canopen.fi/artikkelit/CAN.pdf

Toimintaperiaate. 2014. Mense-raivauspään esittely. Viitattu 27.10.2014. http://www.mense.fi/tuotteet/raivauspaa/toimintaperiaate

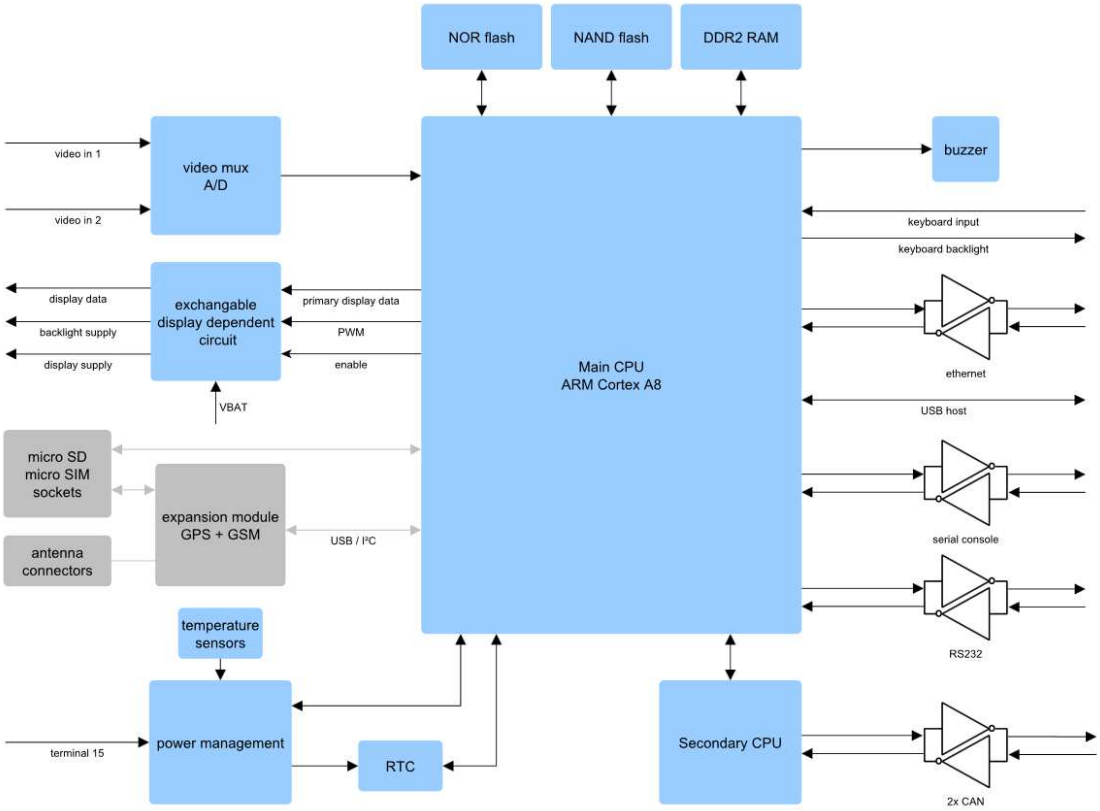
Tietoa meistä. 2014. Tietoa Usewood-yrityksestä. Viitattu 30.8.2014 http://www.usewood.fi/index.php/fi/tietoa

VS-CAN dokumentti 2014. Vision Systemsin sisäinen CAN-dokumentti. Viitattu 10.10.2014

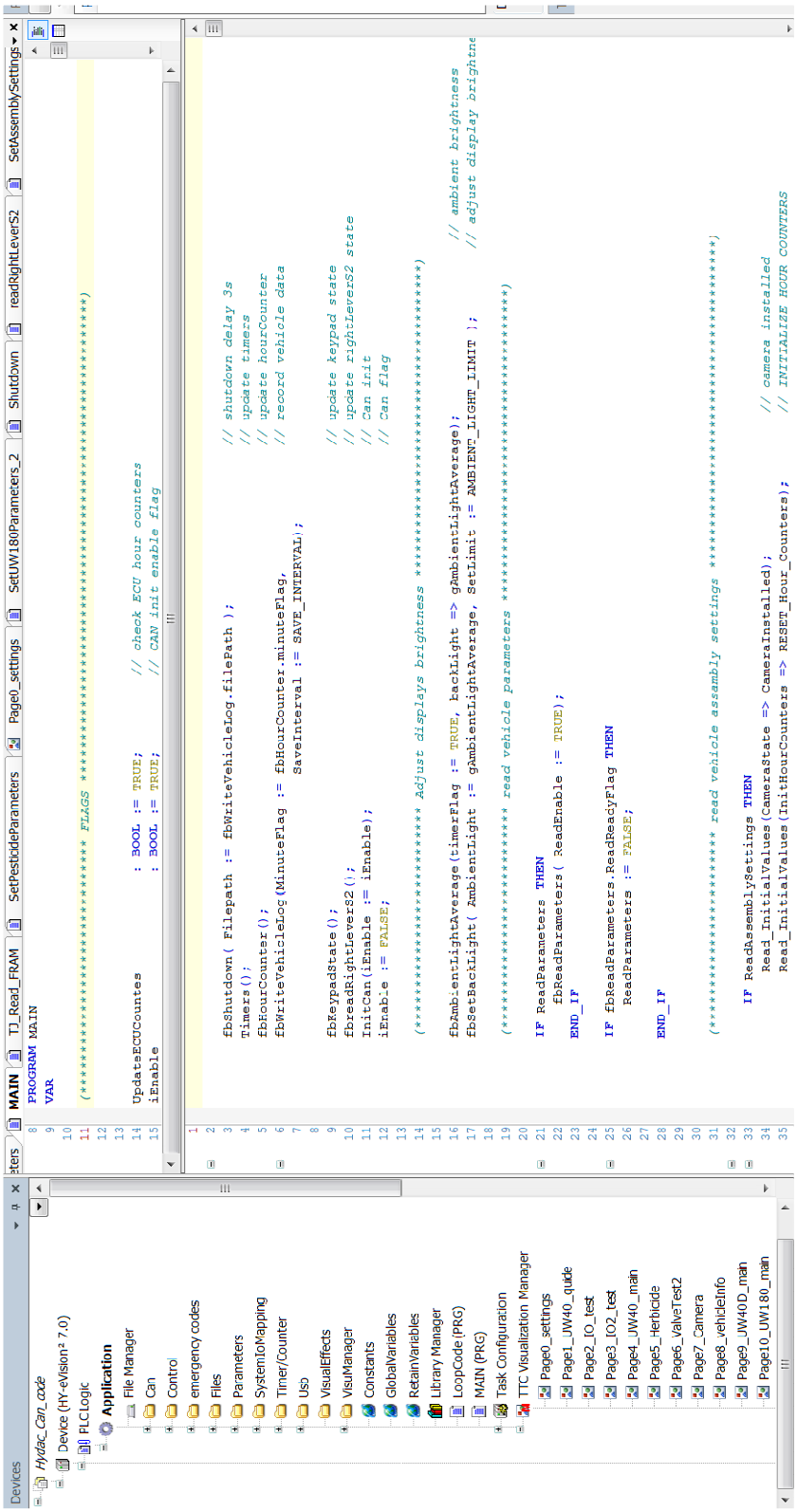
Yritys. 2014. Vision Systems Oy:n esittely. Viitattu 28.10.2014. http://www.visionsystems.fi/index.php?page=yritys

Liitteet

LIITE 1. Työkonenäytön rakenne



LIITE 2. Codesys 3.5-ohjelmointiympäristö



LIITE 3. Parametritaulukko

Metsäkoneen ohjaimen ja näytön välisten parametrien vastaavuustaulukko. Liite sisältää vain osan käytössä olevista parametreista.

	BOSCH RC28-14				Hydac		
Group	Subgroup	Index	Data		FRAM	Parameter	P_number
					FRAM 1	HourCounter > PowerOn	1
					FRAM 2	HourCounter > PowerOn	
					FRAM 3	HourCounter > PowerOn	2
					FRAM 4	HourCounter > PowerOn	
					FRAM 5	HourCounter >Engine On	3
					FRAM 6	HourCounter >Engine On	
					FRAM 7	HourCounter >Engine On	4
					FRAM 8	HourCounter >Engine On	
					FRAM 9	HourCounter > Active working	5
					FRAM 10	HourCounter > Active working	
					FRAM 11	HourCounter > Active working	6
					FRAM 12	HourCounter > Active working	
					FRAM 13	HourCounter > service interval	7
					FRAM 14	HourCounter > service interval	
					FRAM 15		8
					FRAM 16		
					FRAM 17		9
					FRAM 18		
					FRAM 19		10
					FRAM 20		
					FRAM 21		11
					FRAM 22		
					FRAM 23		12
					FRAM 24		
					FRAM 25		13
					FRAM 26		
					FRAM 27		14
					FRAM 28		
					FRAM 29		15
					FRAM 30		
1	1	1	16-bit	MSB LSB	FRAM 31 FRAM 32	Leg PressureLimit	16
1	1	2	16-bit	MSB LSB	FRAM 33 FRAM 34	Gear sequence time	17
1	1	3	16-bit	MSB LSB	FRAM 35 FRAM 36	Parameter 1	18
1	1	4	16-bit	MSB LSB	FRAM 37 FRAM 38	Parameter 2	19
1	1	5	16-bit	MSB LSB	FRAM 39 FRAM 40	Parameter 3	20
1	1	6	16-bit	MSB LSB	FRAM 41 FRAM 42	Parameter 4	13
1	1	7	16-bit	MSB LSB	FRAM 43 FRAM 44	Parameter 5	14
1	1	8	16-bit	MSB LSB	FRAM 45 FRAM 46	Parameter 6	15

1	2	1	16-bit MSB LSB	FRAM 47 FRAM 48	Pump pressure limit	16
1	2	2	16-bit MSB LSB	FRAM 49 FRAM 50	Pump pressure hysteresis	17
1	2	3	16-bit MSB LSB	FRAM 51 FRAM 52	Pressure limit on time	18
1	2	4	16-bit MSB LSB	FRAM 53 FRAM 54	Pressure limitter off time	19
1	2	5	16-bit MSB LSB	FRAM 55 FRAM 56	Limiter activation flow	20
1	2	6	16-bit MSB LSB	FRAM 57 FRAM 58	Limited flow	21
1	2	7	16-bit MSB LSB	FRAM 59 FRAM 60	Automatic limiter off settings	22
1	2	8	16-bit MSB LSB	FRAM 61 FRAM 62	Parameter 7	23
1	3	1	16-bit MSB LSB	FRAM 63 FRAM 64	Temperature compensation limit	24
1	3	2	16-bit MSB LSB	FRAM 65 FRAM 66	Center pump in use	25
1	3	3	16-bit MSB LSB	FRAM 67 FRAM 68	Right pump in use	26
1	3	4	16-bit MSB LSB	FRAM 69 FRAM 70	Left pump in use	27
1	3	5	16-bit MSB LSB	FRAM 71 FRAM 72	Free flow on delay	28
1	3	6	16-bit MSB LSB	FRAM 73 FRAM 74	Free flow off timer	29
1	3	7	16-bit MSB LSB	FRAM 75 FRAM 76	Parameter	30
1	3	8	16-bit MSB LSB	FRAM 77 FRAM 78	Parameter	31
1	4	1	16-bit MSB LSB	FRAM 79 FRAM 80	S1 as pedal	32
1	4	2	16-bit MSB LSB	FRAM 81 FRAM 82	Control max time	33
1	4	3	16-bit MSB LSB	FRAM 83 FRAM 84	Parameter	34
1	4	4	16-bit MSB LSB	FRAM 85 FRAM 86	Parameter	35
1	4	5	16-bit MSB LSB	FRAM 87 FRAM 88	Parameter	36
1	4	6	16-bit MSB LSB	FRAM 89 FRAM 90	Parameter	37
1	4	7	16-bit MSB LSB	FRAM 91 FRAM 92	Parameter	38

LIITE 4. Asetussivun logiikan ohjaaminen

Osa metsäkoneen parametrien asetussivun, logiikkaa ohjaavasta koodista.

```
CASE gParameterSelect OF

1:  // Leg Pressure

    IF fbKeypadState.Key2 AND LegPressureLimit < 200 THEN
            LegPressureLimit := LegPressureLimit +5;
            Write_Parameter(Parameter:=Parameters.Leg_PressureLimit ,
            Data:=WORD_TO_INT(LegPressureLimit));
            Timers.IN_200MS := TRUE;
    END_IF


    IF fbKeypadState.Key3 AND LegPressureLimit >= 5 THEN
            LegPressureLimit := LegPressureLimit -5;
            Write_Parameter(Parameter:=Parameters.Leg_PressureLimit ,
            Data:=WORD_TO_INT(LegPressureLimit));
            Timers.IN_200MS := TRUE;
    END_IF



2:  // Pump Pressure


    IF fbKeypadState.Key2 AND PumpPressureLimit < 200 THEN
            PumpPressureLimit := PumpPressureLimit +5;
            Write_Parameter(Parameter:=Parameters.Pump_pressure_limit ,
            Data:=WORD_TO_INT(PumpPressureLimit));
            Timers.IN_200MS := TRUE;
    END_IF


    IF fbKeypadState.Key3 AND PumpPressureLimit >= 5 THEN
            PumpPressureLimit := PumpPressureLimit -5;
            Write_Parameter(Parameter:=Parameters.Pump_pressure_limit ,
             Data:=WORD_TO_INT(PumpPressureLimit));
            Timers.IN_200MS := TRUE;
    END_IF
```

LIITE 5. Asetussivun grafiikan ohjaaminen

Osa metsäkoneen parametrien asetussivun, grafiikkaa ohjaavasta koodista.

```
(********************** meters ************************************)

LegPressure_meter.SetValue(LegPressureLimit);
PumpPress_meter.SetValue(PumpPressureLimit);
TempCompensation_meter.SetValue(TemperatureCompensationLimit);
PumpHysteresis_meter.SetValue(PumpPressureHysteresis);


(****************************** select parameter ************************)

CASE gParameterSelect OF

1:      indicatorArrow.SetPosition(130,55);
2:      indicatorArrow.SetPosition(294,55);
3:      indicatorArrow.SetPosition(459,55);
4:      indicatorArrow.SetPosition(619,55);

END_CASE

IF gKey0State THEN                      // exit

AnalogArrowLeft.SetOpacity(BUTTON_ON);    // previous page
    ELSE
        AnalogArrowLeft.SetOpacity(BUTTON_OFF);
END_IF


IF gKey5State THEN      // next page

AnalogRightArrow.SetOpacity(BUTTON_ON);
    ELSE
        AnalogRightArrow.SetOpacity(BUTTON_OFF);
END_IF
```

LIITE 6. Parametrien kirjoitusfunktio

Parametrien kirjoitusfunktio. Käytetään parametrien kirjoittamiseen metsäkoneenohjaimelle, CAN-väylän kautta.

```
FUNCTION Write_Parameter     : BOOL
VAR_INPUT
    Parameter                : INT;
    Data                     : INT;
END_VAR
VAR
    fbSendParameter          : Send_TJ_Parameters;
    Group                    : BYTE := 1;
    SubGroup                 : BYTE := 1;
    Index                    : BYTE := 0;
    i                        : INT  := 0;
END_VAR
```

```
FOR i := 1 TO Parameter BY 1 DO
    IF  index < 8  THEN
            Index := Index + 1;
                ELSE
                Index := 1;
                SubGroup := SubGroup +1;
    END_IF

    IF SubGroup > 8 THEN //IF SubGroup > 8 THEN
            SubGroup := 1;
            Group := Group + 1;
    END_IF

END_FOR

fbSendParameter(
    Channel:= 0,
    MsgId:= 16#402,
    Group:= Group ,
    SubGroup:= SubGroup ,
    Index:= Index,
    RequestData:= Data ,
    ReadWrite:= 1,          // write
    CanTxData6:= ,
    CanTxData7:= ,
    hTransmitterId:= ,
    Length:=8 ,
    Enable:= TRUE);          // enable TX
```

LIITE 7. Käyttötapalokin kirjoitusfunktio

Osa metsäkoneen käyttötapalokin, XML-tiedoston kirjoitusfunktiosta.

```
IF minuteCounter > SaveInterval THEN
    minuteCounter := 0;  // reset counter

            (*************** timestamp ************************************************)
                    xmlData[3] := CONCAT (StringHour,':');
                    xmlData[3] := CONCAT (xmlData[3] ,StringMinute);
                    xmlData[3] := CONCAT (xmlData[3] ,':');
                    xmlData[3] := CONCAT (xmlData[3],StringSecond);
            (************ data ****************************************************)
                    xmlData[6]  := INT_TO_STRING ( oiltemp );
                    xmlData[9]  := WORD_TO_STRING ( oilLevel );
                    xmlData[12] := INT_TO_STRING ( engineTemp );
                    xmlData[15] := WORD_TO_STRING ( workPressure );
                    xmlData[18] := WORD_TO_STRING ( pumpPressure );
                    xmlData[21] := WORD_TO_STRING ( fuelLevel );
                    xmlData[24] := BYTE_TO_STRING ( GearStatus );
                    xmlData[27] := WORD_TO_STRING ( pumpL );
                    xmlData[30] := WORD_TO_STRING ( pumpR );
                    xmlData[33] := REAL_TO_STRING ( gActivityPercent);
                    xmlData[36] := WORD_TO_STRING ( engineRpm);
                    xmlData[39] := BYTE_TO_STRING ( gActiveMessages);    // active errorcodes

                    hFile := SysFileOpen(filePath, AM_APPEND, ADR(ResultOpen)); // open file

                    FOR j := 1 TO 41 BY 1 DO

                        WriteText :=xmlData[j];
                        StringLenghtInt := LEN(WriteText);                    //  string lenght
                        StringLength := INT_TO_UDINT (StringLenghtInt);
                        StringLengthSum := StringLengthSum +  StringLength;    // counts characters.
                        SysFileWrite(hFile , ADR(WriteText), StringLength , ADR(ResultWrite));

                    END_FOR

                        SysFileClose(hFile);             // file close
                        i               := i+1;          // counter ++
                        writeDoneFlag   := TRUE;         // flag
                        pathReady       := FALSE;        // flag

END_IF  // end of

                        previousMinute := minuteFlag;       // seve previous state
```

LIITE 8. FRAM-muistin käsittelyfunktiot

FRAM-muistin lukeminen

```
FUNCTION_BLOCK TJ_Read_FRAM
VAR_INPUT
            Parameter      : INT;     // parameter number
END_VAR
VAR_OUTPUT
            Data : WORD;
END_VAR

VAR
    Read_FRAM               : FRAM;                     // FB INIT
    ResultReadFRAM          : RTS_IEC_RESULT;               // result of FRAM read function
    BufferReadFRAM          : ARRAY [1..185] OF WORD;   // read buffer for FRAM

END_VAR
```

```
Read_FRAM.SetWriteProtection(FALSE);             // disable write protection
ResultReadFRAM    := Read_FRAM.Read(ADR(BufferReadFRAM),SIZEOF(BufferReadFRAM));
Data              := BufferReadFRAM[Parameter + 15];
Read_FRAM.SetWriteProtection(TRUE);          // enable write protection
```

FRAM-muistin kirjoitus

```
FUNCTION_BLOCK TJ_Write_FRAM
VAR_INPUT
        Parameter      : INT;     // parameter number
        Data           : WORD;     // data
END_VAR
VAR_OUTPUT
        Write_ready             : BOOL := FALSE;
END_VAR
VAR
        Open_FRAM               : FRAM;          // FB INIT
        ResultReadFRAM          : RTS_IEC_RESULT;    // result of FRAM read function
        ResultWriteFRAM         : RTS_IEC_RESULT;    // result of FRAM write function
        BufferReadFRAM          : ARRAY [1..185] OF WORD;// read buffer for FRAM

END_VAR
```

```
Open_FRAM.SetWriteProtection(FALSE); // disable write protection
ResultReadFRAM    := Open_FRAM.Read(ADR(BufferReadFRAM),SIZEOF(BufferReadFRAM)); // Open

BufferReadFRAM[Parameter + 15] := Data;    // parameter number + 15

ResultWriteFRAM := Open_FRAM.Write(ADR(BufferReadFRAM),SIZEOF(BufferReadFRAM)); // write buffer
Open_FRAM.SetWriteProtection(TRUE); // enable write protection

Write_ready := TRUE;
```

LIITE 9. Can-viestien vastaanottofunktio

Osa vastaanotettavien CAN-viestien alustusfunktiosta

```
// create a buffer which receives Sensor value 1 message 0x100  hReceiver 6
      gCan[0].hReceiver[6] := CL2.CreateMaskReceiver // handle of Driver interface
      (hDriver := gCan[0].hDriver,
      cobIdValue := 16#100,            // identifier: bit value of message
      cobIdMask := 16#1FFFFFFF,        // mask value of message bits to be received
      xRTRValue := FALSE,              // Remote Transmission Request bit value
      xRTRMask := FALSE,               // Remote Transmission Request mask value
      x29BitIdValue := FALSE,          // extended frame format (29 bit) ID value
      x29BitIdMask := FALSE,           // extended frame format (29 bit) ID mask value
      xTransmitValue := FALSE,         // transmit feedback value
      xTransmitMask := FALSE,          // transmit feedback mask value
      xAlwaysNewest := FALSE,          // return always the lastly received message
      eEvent := CB.EVENT.NO_EVENT,     // trigger event when message receive
      xEnableSyncWindow := FALSE,      // use SYNC window
      peError := ADR(gCan[0].eError)   // return error code
       );
```

Osa CAN-viestien vastaanottoon käytettävästä, CAN_RX_Priority-funktiosta.

```
(****************************** hReceiver 6 0x100 ************************************** *)

   fbCanRx(Channel := 0, MsgBuffer := 6, Enable := TRUE, Timeout:=10, NewData =>, Error =>);

   IF fbCanRx.NewData THEN

       // acess to the received data

       CanRxData[0] := gCan[0].CanMsg.Data[0]; // first received data byte
       CanRxData[1] := gCan[0].CanMsg.Data[1]; // second received data byte
       CanRxData[2] := gCan[0].CanMsg.Data[2]; // third received data byte
       CanRxData[3] := gCan[0].CanMsg.Data[3]; // fourth received data byte
       CanRxData[4] := gCan[0].CanMsg.Data[4]; // fifth received data byte
       CanRxData[5] := gCan[0].CanMsg.Data[5]; // sixth received data byte
       CanRxData[6] := gCan[0].CanMsg.Data[6]; // seventh received data byte
       CanRxData[7] := gCan[0].CanMsg.Data[7]; // eighth received data byte

   (* ************   Sensor value 1 message  0x100; ************************** *)

       oilTemp     := SHL (CanRxData[1],8) OR CanRxData[0];     // CAN_RX_DATA BYTE_0 & BYTE_1
       engineTemp  := SHL (CanRxData[3], 8) OR CanRxData[2];  // CAN_RX_DATA BYTE_2 & BYTE_3
       fuelLevel   := SHL (CanRxData[5], 8) OR CanRxData[4];  // CAN_RX_DATA BYTE_4 & BYTE_5
       oilLevel    := SHL (CanRxData[7], 8) OR CanRxData[6];  // CAN_RX_DATA BYTE_6 & BYTE_7

   END_IF
```

LIITE 10. Käyttötapaloki

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	oil_temp	oil_level	motor_temp	work_pressur	pump_pressure	fuel_level	gear	left_pump	right_pump	activity	engine_RPM	error_codes
2	12:9:0	-19	87	34	192	193	99	1	0	0	1,7	2334	11
3	12:19:0	8	81	52	3	31	99	3	0	0	100,0	2445	11
4	12:29:0	13	87	53	5	7	99	2	0	0	79,7	2424	11
5	12:39:0	17	87	54	4	5	100	2	0	0	47,5	2390	11
6	12:49:0	24	81	56	2	49	97	2	0	1	81,4	2403	11
7	12:59:0	28	81	57	2	98	96	2	0	1	66,1	2356	11
8	13:9:0	29	81	57	2	4	89	2	0	0	59,3	2420	11
9	13:19:0	31	81	57	2	62	93	2	0	1	59,3	2321	11
10	13:29:0	32	81	56	4	5	89	2	0	0	49,2	2447	11
11	13:39:0	34	81	56	48	55	89	2	1	1	67,8	2378	11
12	13:49:0	33	81	57	37	41	89	2	1	1	69,5	2337	11
13	13:59:0	34	81	57	2	80	79	2	0	1	61,0	2345	11
14	14:9:0	34	81	57	4	60	72	2	0	1	57,6	2415	11
15	14:19:0	34	81	57	1	62	73	2	0	1	62,7	2377	11
16	14:29:0	33	81	57	22	22	72	2	0	0	67,8	2419	11
17	14:39:0	34	81	57	27	31	72	2	1	1	52,5	2369	11
18	14:49:0	33	81	57	30	33	66	2	1	1	37,3	2421	11
19	14:59:0	35	81	57	46	49	66	2	1	1	72,9	2433	11
20	15:9:0	38	81	57	2	4	66	2	0	0	78,0	2488	11
21	15:19:0	35	81	57	3	5	60	2	0	0	67,8	2522	11
22	15:29:0	35	87	56	39	44	60	2	1	1	49,2	2452	11
23	15:39:0	37	81	56	3	5	56	2	0	0	54,2	2522	11
24	15:49:0	38	81	57	1	1	55	2	0	0	5,1	1330	11
25	15:59:0	35	81	57	30	35	55	2	1	1	55,9	2378	11
26	16:9:0	33	87	56	3	3	51	2	0	0	55,9	2430	11
27	16:19:0	37	81	57	2	68	50	3	0	1	100,0	2315	11