Tampereen ammattikorkeakoulu

Sähköajoneuvon CAN-väylän datan visualisointi

Elias Kallio

OPINNÄYTETYÖ
Marraskuu 2022

Tieto- ja viestintätekniikka
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Sulautetut järjestelmät ja elektroniikka

KALLIO, ELIAS:
Sähköajoneuvon CAN-väylän datan visualisointi

Opinnäytetyö 55 sivua, joista liitteitä 12 sivua
Marraskuu 2022

Sähköajoneuvojen yleistyessä vaikuttaisi yksi tämän trendin mahdollisista päätöspisteistä olevan henkilöautojen ja muiden ajoneuvojen kokonaan sähköistäminen. Tässä opinnäytetyössä tutkitaan CAN-väylän ja siihen perustuvien standardien soveltamista sähkötoimisen raskaan kaluston kehityksessä.

Opinnäytetyö tehtiin osana Tampereen ammattikorkeakoulun osuutta Helsingin kaupungin rakentamispalveluliikelaitos Staran johtamasta Stara eRetrofit -hankkeesta. Yhteistyössä olivat TAMK:n lisäksi myös Forum Virium Helsinki ja Teknologian tutkimuskeskus VTT. Hankkeen tavoitteena oli yhden Staran kuormaauton muuttaminen sähkökäyttöiseksi. Opinnäytetyön aiheena oli ajoneuvon ohjaamoon tulevaa näyttöä varten tehtävä ohjelmointityö.

Opinnäytetyön tuloksiksi muodostuivat alkuperäiset vaatimukset täyttävä ohjelmisto näyttöä varten sekä toissijaisena tavoitteena ollut CAN- ja J1939-protokollien toimintaa kattavasti kuvaava teoriakokonaisuus.

CAN-väylä, J1939, CODESYS

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Embedded Systems and Electronics

KALLIO, ELIAS:
Visualizing the CAN Bus Data of an Electric Vehicle

Bachelor's thesis 55 pages, appendices 12 pages
November 2022

Electrification is one of the most significant technological challenges of the modern era and it is highly prominent in transportation. Partially or fully electrically powered vehicles are already very common, be it electric buses or hybrid cars. One of the possible end points for this trend is the total electrification of cars and other vehicles. This thesis concerns the function of the CAN bus and the standards derived from it in the development of heavy-duty electric vehicles.

This thesis is a part of the Tampere University of Applied Sciences' contribution to the eRetrofit project which is headed by the Helsinki City Construction Service Stara. Also contributing to the project are Forum Virium Helsinki and VTT Technical Research Center of Finland. The project aims to electrify one of Stara's trucks. The subject of this thesis is the programming work involved in the display which is to be situated in the cabin of the vehicle.

The results of this thesis were a program for the display unit in accordance with the original specifications and a theoretical package describing the working principles of the CAN and J1939 protocols.

CAN bus, J1939, CODESYS

SISÄLLYS

1 JOHDANTO 7
2 CAN-väylä 8
2.1 Fyysinen kerros 9
2.1.1 NRZ ja Bit Stuffing 10
2.2 Siirtokerros 12
2.2.1 Kehysrakenne 12
2.2.2 Kehystyypit 13
2.2.3 Kilpavaraus 14
2.2.4 Virheentarkistus 14
3 SAE J1939 17
3.1 Siirtokerros ja sovelluskerros 18
3.1.1 Viestikehys 18
3.1.2 PGN ja SPN 19
3.1.3 Address Claiming 21
3.1.4 Siirtoprotokolla 22
4 Käytetyt ohjelmistot 25
4.1 CODESYS 25
4.2 Epec Multitool 26
4.3 Vector CANdb++ 28
5 Toteutus 29
5.1 CANdb++ -kirjasto 30
5.2 Multitool-projekti 31
5.3 CODESYS-projekti 33
6 Projektin kulku ja versiot 36
6.1 Vaatimukset 36
6.2 Versio 1 37
6.3 Versio 2 38
7 Pohdinta 41
LÄHTEET 42
LIITTEET 44
Liite 1. Vaatimusluettelo 44
Liite 2. Version 1 visualisaationäkymät ja ”Main”-välilehden koodi. 47
Liite 3. Version 2 visualisaationäkymät ja ”Main”-välilehden koodi. 50
Liite 4. Visualisaationäkymät ruudulla 55

LYHENTEET JA TERMIT

ACK	**Acknowledgement**
BAM	**Broadcast Announcement Message**
CAN FD	**Controller Area Network Flexible Data-Rate**
CAN	**Controller Area Network**
CAN-H	**CAN-High**
CAN-L	**CAN-Low**
CiA	**CAN in Automation**
CRC	**Cyclic Redundancy Check**
CTS	**Clear to Send**
DLC	**Data Length Code**
ECU	**Electronic Control Unit**
EOF	**End of Frame**
FBD	**Function Block Diagram**
IDE	**Identifier Extension**
IFS	**Interframe Space**
IL	**Instruction List**
ISO	**International Organization for Standardization**
J1939	**CAN-pohjainen korkeamman tason protokolla raskaalle kalustolle**
LD	**Ladder Diagram**
NRZ	**Non-Return-to-Zero**
OSI-malli	**Open Systems Interconnection Reference Model, tiedonsiirtoprotokollia kerroksittain kuvaava malli.**
PDU	**Protocol Data Unit**
PGN	**Parameter Group Number**
REC	**Receive Error Counter**
RTR	**Remote Transmission Request**
RTS	**Request to Send**
RZ	**Return-to-Zero**
SAE	**Society of Automotive Engineers**
SFC	**Sequential Function Chart**
SOF	**Start of Frame**

SPN	**Suspect Parameter Number**
ST	**Structured Text**
TEC	**Transmit Error Counter**
TP.CM	**Transport Protocol, Connection Mode**
TP.DT	**Transport Protocol, Data Transfer**

1 JOHDANTO

1980-luvun alussa saksalainen yritys Robert Bosch GmbH arvioi olemassa olevia sarjaväyläjärjestelmiä käytettäväksi henkilöautoissa, mutta yrityksen vaatimukset täyttävää järjestelmää ei löydetty. Täten uuden CAN-väylän (Controller Area Network) suunnittelu alkoi vuonna 1983 ja se julkaistiin vuonna 1986. (CAN in Automation n.d.).

Suunniteltavan väylän tarkoitus oli pääasiallisesti lisätä toimintoja, joita suunnittelijat eivät löytäneet olemassa olevista väyläteknologioista. Sivuvaikutus tehdyistä ratkaisuista oli CAN-väylän johdotuksen merkittävä yksinkertaistuminen aikalaisiinsa verrattuna. Tämä näkyy esimerkiksi säästöinä materiaaleissa, asennusajassa ja järjestelmän painossa. (CAN in Automation n.d.).

Tämä opinnäytetyö tehtiin osana Helsingin kaupungin rakentamispalveluliikelaitos Staran johtamaa eRetrofit -projektia. Projektissa muutetaan yksi Staran käytössä olleista kuorma-autoista sähkökäyttöiseksi ja arvioidaan ajoneuvokaluston sähköistämisen kustannuksia ajoneuvon elinkaaren aikana sekä mahdollisuuksia nopeuttaa sähkökäyttöiseen ajoneuvokalustoon siirtymistä. Muunnoksen jälkeen ajoneuvo otetaan normaaliin käyttöön ja sen suorituskykyä seurataan testijakson ajan. Tämän opinnäytetyön tavoitteena oli luoda ohjelmisto, jonka avulla voidaan välittää ajoneuvon CAN-väylältä tietoa ohjaamossa sijaitsevalle näytölle, jotta kuljettaja voi seurata ajoneuvon tilaa.

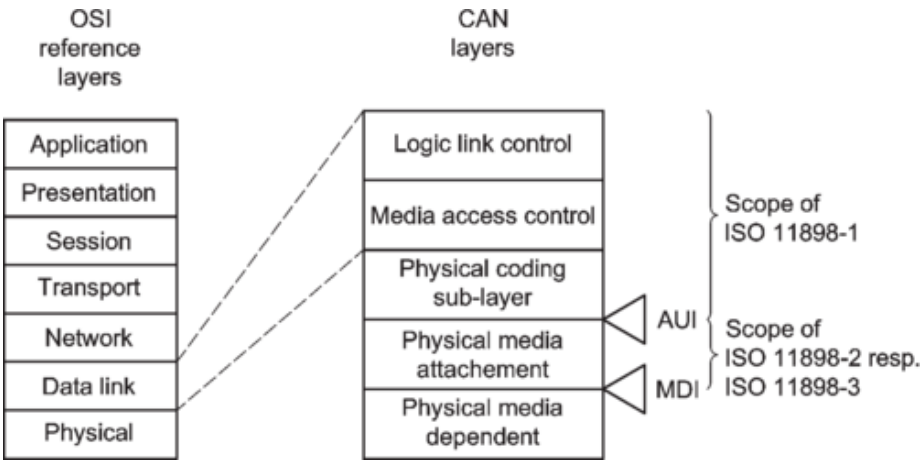
Tämä opinnäytetyö koostuu karkeasti kahdesta osasta: ensimmäisessä perehdytään CAN-väylään ja siitä kehitettyyn raskaan kaluston käyttämään J1939-standardiin. Standardien dokumenttirakenteet kuvaillaan lyhyesti, jonka jälkeen niiden toiminnallisuutta käsitellään syvemmin. Työn toisessa osassa kerrotaan EPEC 6107 -näytön ohjelmiston tuottamisesta ja siihen käytetyistä ohjelmistoista.

2 CAN-väylä

CAN-väylän standardien kehitys on edelleen aktiivista; CAN 2.0 julkaistiin vuonna 1991 ja vuonna 2012 julkaistiin CAN FD (Controller Area Network Flexible Data-Rate) 1.0. Muita tärkeitä virstanpylväitä CAN-väylän kannalta olivat CiA:n (CAN in Automation) perustaminen vuonna 1992 ja väylän standardointi ISO 11898:n mukaisesti vuonna 1993. CiA on kansainvälinen, voittoa tavoittelematon järjestö, joka koostuu CAN-väylän käyttäjistä ja valmistajista. Sen tavoitteena on CAN-väylän jatkokehittäminen ja sen käytön lisääminen eri markkinoilla. Jäsenyrityksiä on nykyään yli 700 ja mukana ovat esimerkiksi kotimaiset Sandvik Mining and Construction Oy ja ABB Oy. (CAN in Automation, n.d.)

ISO 11898-standardi jaettiin vuonna 2003 standardeihin ISO 11898-1, joka määrittelee siirtokerroksen ja ISO 11898-2, joka käsittelee suurinopeuksisen High-speed CAN-väylän fyysistä kerrosta. High-speed CAN on tällä hetkellä fyysisen kerroksen ratkaisuista laajimmassa käytössä. Matalanopeuksista, vikasietoista CAN-väylää koskeva ISO 11898-3 -standardi julkaistiin vuonna 2006. (CAN in Automation, n.d.)

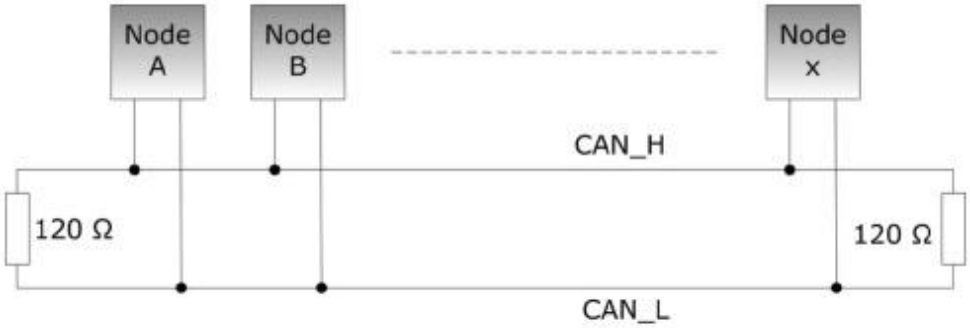
Näiden standardien suhde OSI-malliin (Open Systems Interconnection) on esitelty kuvassa 1. OSI-malli on standardointijärjestö ISO:n (International Organization for Standardization) kehittämä seitsemänkerroksinen, tietoliikennejärjestelmää kuvaava malli, jonka tarkoitus on standardisoida liikennettä monien erilaisten järjestelmien välillä.



KUVA 1. CAN-väylän standardit OSI-mallissa (ISO 11898-2:2016, 2016).

2.1 Fyysinen kerros

ISO 11898-2 -standardin mukainen CAN-väylä koostuu yleisimmin parikaapelista, joka kulkee väylän jokaisen solmun (päätelaitteen) kautta. Korkeilla nopeuksilla kaapelin tulisi olla kierretty ja suojattu, mutta matalilla nopeuksilla tämä ei ole välttämätöntä. Väylä on päätetty molemmista päistä 120 Ω päätevastuksilla, joilla pyritään minimoimaan etenkin suurinopeuksisten signaalien aiheuttamia heijastushäiriöitä. Väylätopologia on esitelty kuvassa 2. Väylä on moni-isäntäinen eli jokainen solmu voi aloittaa lähetyksen omatoimisesti. Loogista rajaa väylän solmujen määrälle ei ole, mutta siihen vaikuttavat solmujen lähetin-vastaanotin-kytkennät. (Alanen 2000, 5)

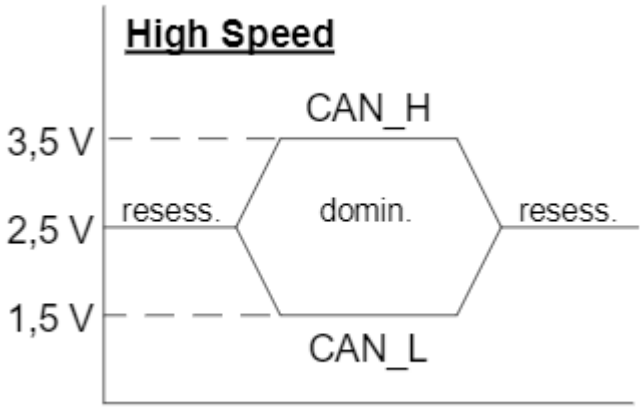


KUVA 2. CAN-väylän fyysinen rakenne (Copperhill Technologies n.d.)

Broadcast-tyyppiset sanomat ja väylän kilpavarausperiaate edellyttävät, että jokaisen solmun on otettava näyte samanaikaisesti, jotta ongelmia ei synny. Vaatimus samanaikaisuudesta ja signaalin siirtotiellä kokema viive asettavat enimmäisrajan väylän fyysiselle pituudelle annetulla siirtonopeudella. High-speed CAN-väylä kykenee parhaimmillaan 1 Mbit/s siirtonopeuteen 40 m kaapelipituudella, mutta nopeutta vähentämällä voidaan mahdollistaa jopa useiden kilometrien väyläpituus. (Alanen 2000, 5)

Tiedonsiirto CAN-väylässä perustuu CAN-L- ja CAN-H-johtimien väliseen jännite-eroon. Väylän dominantti tila on looginen ”0”, joka kumoaa väylällä olevan resessiivisen ”1”-tilan. (Bosch 1991, 8). Väylän dominantissa tilassa CAN-H-johdin ajetaan kohti 3.5 V:n jännitettä ja CAN-L-johdin kohti 1.5 V:n jännitettä. Mikäli

yksikään väylän laitteista ei lähetä dominanttia, palaa väylä resessiiviseen tilaan, jossa molempien johdinten jännite on 2.5 V. (Texas Instruments 2016, 7). Tämä periaate on esitelty kuvassa 3. Johtimet ovat keskenään samanlaisia, joten sähkömagneettisen säteilyn aiheuttamat yhteismuotoiset jännitteen vaihtelut eivät välttämättä aiheuta ongelmia tiedonsiirrossa, koska johtimien välinen jännite-ero pysyy samana.

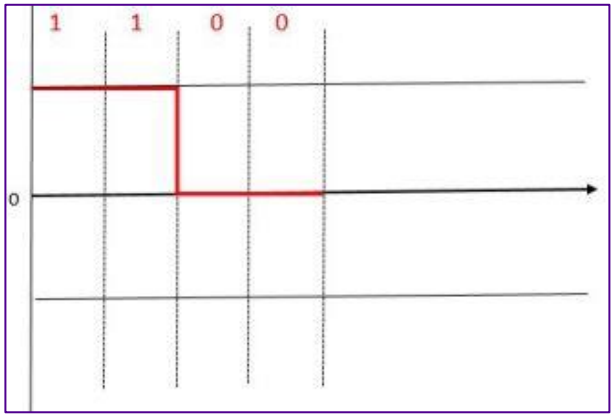


KUVA 3. CAN-väylän jännitetasot (Alanen 2000, 10).

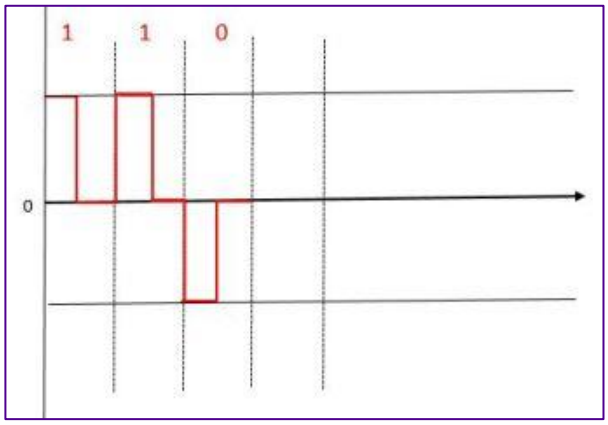
2.1.1 NRZ ja Bit Stuffing

CAN-protokolla käyttää tiedon koodaamiseen NRZ-koodausta (Non-return to Zero). Tässä koodausmenetelmässä ei ole erillistä neutraalia ”lepotilaa”, vaan loogista tilaa ”1” vastaa tietty ehto ja tilaa ”0” toinen ehto. NRZ-koodauksessa tila ei muutu, ennen kuin signaalissa lähetetään erimerkkinen bitti. Useiden samanmerkkisten bittien lähetys voi aiheuttaa synkronointiongelmia lähettäjän ja vastaanottajan välillä, koska laitteet synkronoidaan signaalissa esiintyvien reunojen perusteella.

RZ-koodauksessa taas signaali palaa jokaisen lähetetyn bitin jälkeen neutraaliin lepotilaan, kunnes seuraava bitti lähetetään. Tästä johtuen sillä ei ole samoja NRZ-koodauksen synkronointiongelmia, mutta sen lähetysnopeus on vain puolet NRZ-koodauksen nopeudesta. Ero NRZ- ja RZ-koodauksen välillä on esitelty kuvissa 4 ja 5.



KUVA 4. NRZ-koodauksen periaate (embedclogic n.d.)



KUVA 5. RZ-koodauksen periaate (embedclogic n.d.)

NRZ-koodauksen synkronointiongelman takia CAN-väylällä on käytettävä bit stuffing -toimintoa. Bit stuffingissa koodiin lisätään yksi erimerkkinen bitti, mikäli koodissa esiintyy peräkkäin viisi samanmerkkistä bittiä. Tällä helpotetaan väylän synkronointia ja vähennetään virheiden todennäköisyyttä. Nämä lisätyt bitit poistetaan vastaanottovaiheessa, ennen virheentarkistusta. Bit Stuffing menee pois päältä viestin CRC-kentän (Cyclic Redundancy Check) lopuksi. (Texas Instruments 2016, 6). CRC on yksinkertainen tarkistussummaan perustuva virheentarkistusmenetelmä, jonka avulla väylällä liikkuvien viestien eheyttä pyritään valvomaan.

2.2 Siirtokerros

ISO 11898-1 -standardi käsittelee CAN-protokollan siirtokerrosta, joka on yhteinen kaikille CAN-väylille. Sanomat lähetetään broadcast-tyyppisenä lähetyksenä kaikille väylän laitteille, jotka päättävät viestin tunnisteen perusteella, tarvitseeko niiden vastaanottaa kyseistä viestiä (Kvaser, n.d.).

2.2.1 Kehysrakenne

Viestit liikkuvat väylällä viestikehyksissä, joita on neljä erilaista: datakehys, RTR-kehys (Remote Transmission Request), virhekehys ja ylikuormituskehys. Tämä voi muuttua, mikäli käytetään CAN-pohjaisia korkeamman tason protokollia. Kehysten rakenne voi myös olla näissä protokollissa erilainen. Kuvassa 6 esitellään CAN 2.0A:n mukainen datakehys jaettuna sen sisältämiin kenttiin. CAN 2.0A -standardi on yhteensopiva aiempien CAN-protokollien kanssa. CAN 2.0B -standardi eroaa A:sta vain tunnistekentän pituudessa, joka on 29 bittiä. (Kvaser, n.d.)

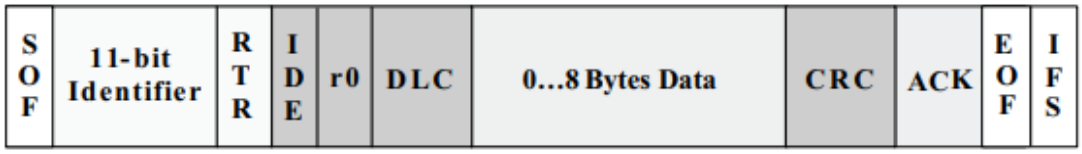
Datakehys nimensä mukaisesti kuljettaa laitteilta saatua dataa CAN-väylällä. Kehys alkaa yhdellä, dominantilla SOF-bitillä (Start of Frame), jolla merkitään viestin alku ja suoritetaan synkronointi. Tätä seuraa 11-bittinen tunnistekenttä, jolla voidaan tunnistaa lähettävä laite ja sen prioriteetti väylän käyttöön. (Texas Instruments 2016, 3)

RTR-bitti on datakehyksessä dominantti. Resessiivinen RTR-bitti tarkoittaa kehyksen olevan RTR-kehys. IDE-bitillä (Identifier Extension) ilmaistaan, onko kehyksessä normaali vai pidennetty tunnistekenttä. Dominantti bitti tässä kohtaa tarkoittaa normaalin pituista tunnistekenttää. Seuraava bitti on varattu tulevaisuutta ajatellen mahdollisia muutoksia varten eikä tarkoita mitään. Neljä DLC-bittiä (Data Length Code) ilmaisevat datakentän pituuden tavuina. CAN 2.0 -kehys voi sisältää maksimissaan 8 tavua dataa. (Texas Instruments 2016, 3)

CRC-kenttä sisältää datan 15-bittisen tarkistussumman sekä yhden resessiivisen välibitin ja se toimii viestikehyksen virheentarkistuksena. Vastaanotetun viestin läpäistessä virhetarkistuksen, kirjoittavat vastaanottajat väylälle dominantin tilan

lähettäjän resessiivisen ACK-kentän (Acknowledgement) päälle. ACK-bittiä seuraa myös yksi resessiivinen välibitti. (Texas Instruments 2016, 3)

Viimeisenä tulee viestin päättymistä ilmaiseva EOF-kenttä (End of Frame), joka sisältää seitsemän resessiivistä bittiä. IFS-kenttä (Interframe Space) toimii kehysten välisenä taukona, jolla annetaan ohjaimelle aikaa viestipuskurin päivittämiseen. (Texas Instruments 2016, 3)



KUVA 6. Dataviestin kehysrakenne (Texas Instruments 2016, 3).

2.2.2 Kehystyypit

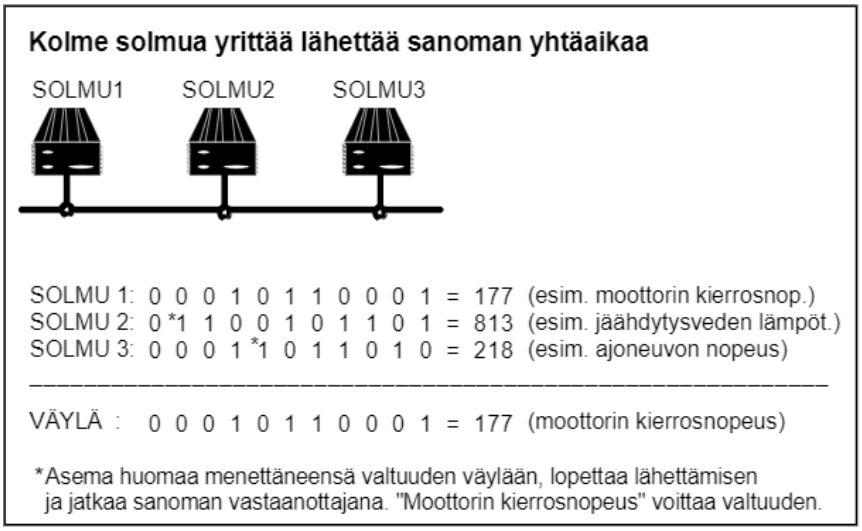
Lähettämällä RTR-kehyksen voi jokin väylän laitteista voi pyytää toista laitetta lähettämään dataa. Tällaisessa kehyksessä RTR-bitti on resessiivinen ja se ei sisällä lainkaan dataa. (Texas Instruments 2016, 3)

Virhekehys eroaa rakenteeltaan huomattavasti datakehyksestä. Sen ensimmäinen kenttä, Error Flag, koostuu 6 virhebitistä, mutta se voi pidentyä 0–6 bitillä. Tätä kenttää seuraa 8-bittinen, resessiivisistä biteistä koostuva välikenttä. Error Flag -kenttä voi koostua resessiivisistä tai dominanteista biteistä, riippuen onko kyseessä passiivinen vai aktiivinen Error Flag. Virhekehys rikkoo bit stuffing -sääntöä, joten sen lähettäminen väylällä johtaa siihen, että kaikki muutkin laitteet alkavat lähettämään virhekehystä. Virhetilanteen ratketessa alkuperäinen lähettäjä lähettää viestinsä uudelleen. (Bosch 1991, 16)

Ylikuormituskehys on muodoltaan virhekehystä vastaava, koostuen 6–12 dominantista bitistä, joita seuraa kahdeksan bittiä pitkä välikenttä. Ylikuormituskehys lähetetään, mikäli laite tarvitsee lisäaikaa sanoman käsittelyyn. (Bosch 1991, 17–18).

2.2.3 Kilpavaraus

Mikä tahansa väylän laitteista voi yrittää tiedonsiirtoa havaitessaan väylän olevan vapaana. Mikäli useampi laite alkaa lähettää dominanttia tilaa, ratkaistaan vuoro kilpavarausperiaatteella. Molemmat laitteet alkavat lähettää omaa viestiänsä, jonka ensimmäinen osio on lähettäjän tunnistenumero. Matalampi tunnistenumero omaa väylässä korkeamman prioriteetin. Laitteet tarkkailevat väylää samalla ja mikäli yksi niistä huomaa lähettävänsä resessiivistä tilaa samalla, kun toinen lähettää dominanttia tilaa, luopuu resessiivisen tilan lähettäjä lähetysvuorostaan. Tämä periaate on esitelty kuvassa 7. Jäljelle jäänyt laite jatkaa viestin lähettämistä. (Alanen 2000,7)



KUVA 7.CAN-väylän kilpavarausperiaate (Alanen 2000, 7).

2.2.4 Virheentarkistus

Jokainen väylän laitteista valvoo väylän liikennettä ja mikäli virhe havaitaan, lähetetään virhekehys. Virhekehyksen muotoilu rikkoo bit stuffing -sääntöä ja aiheuttaa väylällä olevan viestin tuhoutumisen. Tämän jälkeen väylän muut laitteet hylkäävät virheelliseksi havaitun viestin. Virheiden havaitsemiseen on CAN-protokollassa viisi eri tapaa: bittivirhe, bit stuffing -virhe, kehysvirhe, kuittausvirhe ja CRC-virhe. (Texas Instruments 2016, 6)

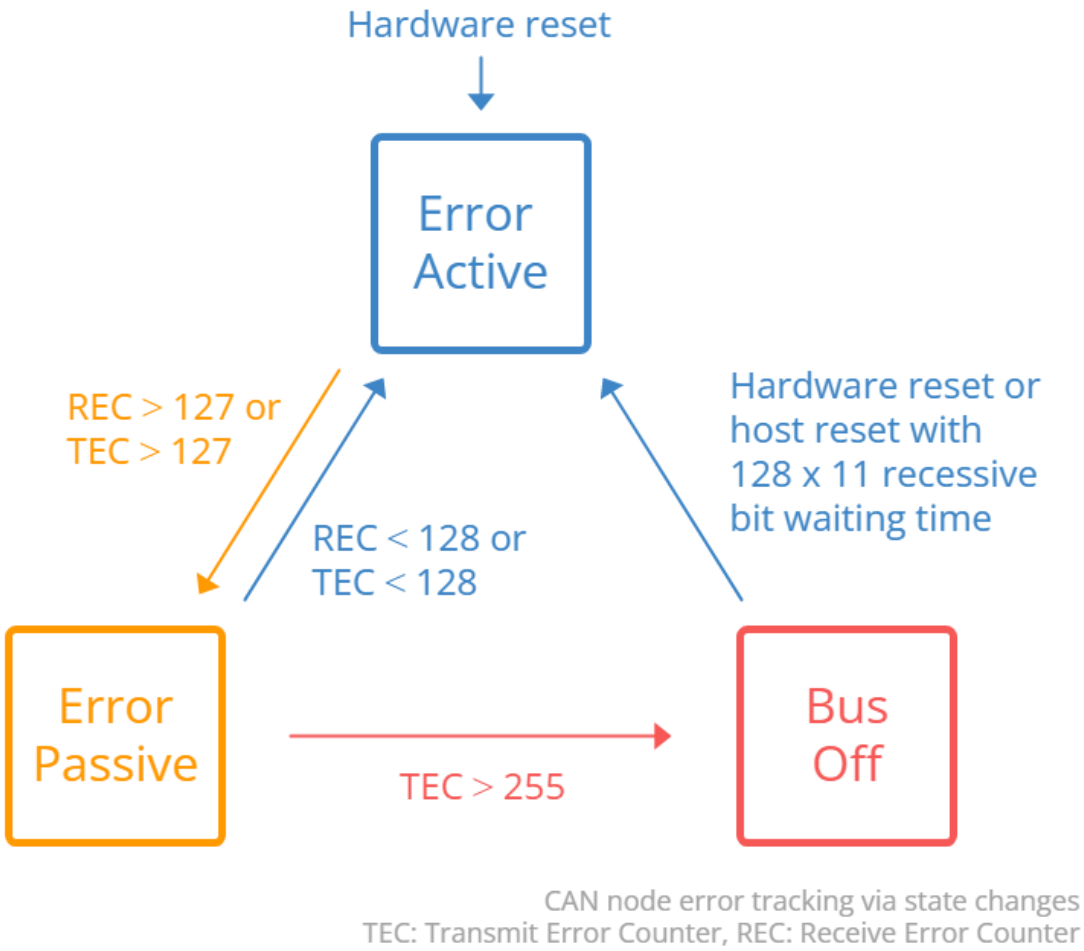
Bittivirhe ja bit stuffing -virhe toimivat järjestelmän bittitasolla. Bittivirhe tarkoittaa sitä, että väylällä havaitaan looginen tila, jota ei lähetetty. Bittivirhettä ei tapahdu kilpavarausprosessin aikana. Bit stuffing -virhe syntyy, jos väylällä esiintyy enemmän kuin viisi samanmerkkistä bittiä peräkkäin. (Bosch 1991, 23)

Loput virheenhavaitsemismekanismit toimivat viestitasolla. Kehysvirheessä jokin viestikehyksen biteistä eivät vastaa ennalta määriteltyä arvoaan. Tällaisia ovat esimerkiksi välibitit CRC- ja ACK-kenttien yhteydessä, joiden tulisi olla resessiivisiä. Kuittausvirhe syntyy, jos viestin lähettäjä ei havaitse ACK-kentän aikana muiden laitteiden lähettämää dominanttia kuittausta. ACK-signaalin lähettävät myös laitteet, jotka eivät vastaanota viestin sisältöä. CRC-virheessä lähetetty tarkistussumma ei vastaa vastaanottajan laskemaa summaa. (Bosch 1991, 23)

Näitä virheitä käsitellään jokaisessa CAN-väylän laitteessa olevien virhelaskureiden avulla. Virhelaskurit ja niiden väliset siirtymät on esitelty kuvassa 8. Virhelaskureita on kaksi, TEC (Transmit Error Counter) eli lähetysvirhelaskuri ja REC (Receive Error Counter) eli vastaanottovirhelaskuri. Laskurit ovat 8-bittisiä, eli niiden maksimiarvo on 255. TEC kasvaa kahdeksalla bitillä jokaista havaittua lähetysvirhettä kohtaan ja REC yhdellä bitillä jokaista vastaanottovirhettä vastaan. Tämä on siksi, että lähetysvirheen todennäköisyys on suurempi, kuin vastaanottovirheen. (Vector, 2021)

Jommankumman rekisterin ylittäessä arvon 127, menee laite Error Passive -tilaan. Tällöin se lähettää virheen havaitessaan passiivisia virhekehyksiä, joiden sisältämät resessiiviset bitit eivät tuhoa väylän liikennettä. Täten se kasvattaa oman virhelaskurinsa arvoa häiritsemättä väylän toimintaa. Mikäli jompikumpi rekisteri ylittää arvon 255, menee kyseinen laite Bus Off -tilaan. Tällöin laite ei lähetä mitään väylälle. Tällä pyritään välttämään väylän ruuhkautuminen viallisen laitteen takia. (Vector, 2021)

Jokainen oikein lähetetty viesti vähentää virhelaskureiden lukemaa. Oikein lähetetyn viestin kriteeri on 11 peräkkäistä resessiivistä bittiä, jotka ovat ACK-kentän välibitti, seitsemän EOF-bittiä sekä kolme IFS-bittiä. Laite, joka on joutunut Bus Off -tilaan siirtyy takaisin normaaliin tilaan (Error Active), kun 128 tällaista sarjaa on havaittu väylällä. Edellä mainitut säännöt eivät kata kaikkia virhelaskureihin liittyviä sääntöjä, mutta ovat niistä keskeisimmät. (Bosch 1991, 24–25)

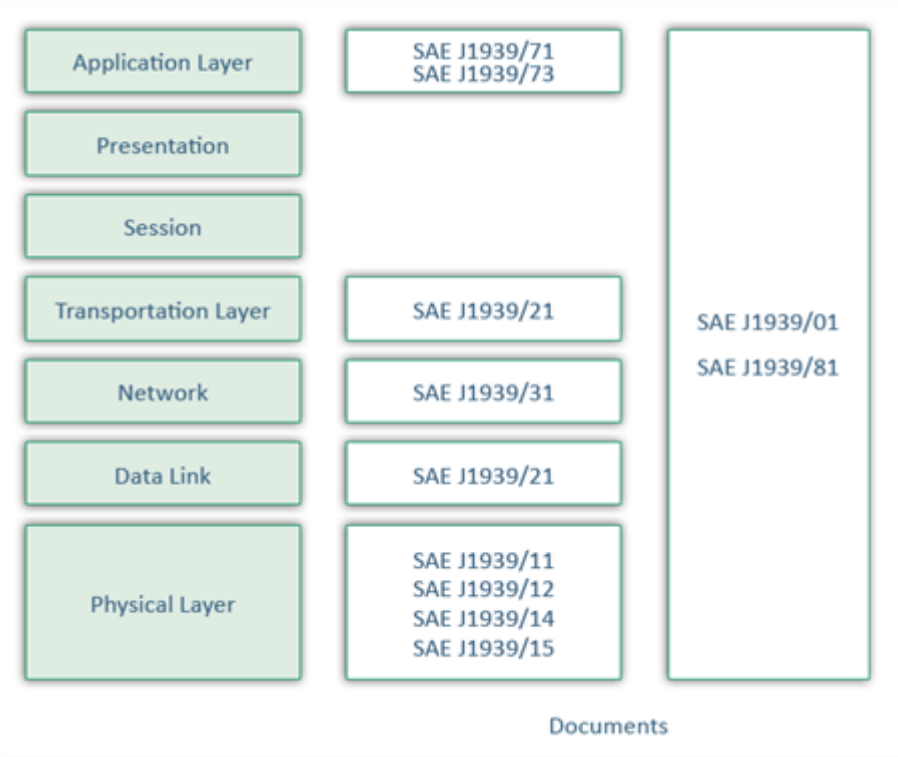


KUVA 8. CAN-väylän virhelaskureiden logiikka (Falch, 2022)

3 SAE J1939

SAE J1939 on Society of Automotive Engineersin julkaisema standardikokoelma, joka määrittelee sähköisten hallintayksiköiden (ECU, Electronic Control Unit) kommunikointia raskaan kaluston CAN-väylällä. J1939-standardit tarjoavat korkeamman tason protokollia, jotka käyttävät CAN-väylää fyysisenä kerroksenaan. Standardikokoelman on tarkoitus mahdollistaa kommunikaatio eri valmistajien hallintayksiköiden välillä. Kuvassa 9 on nähtävissä eri J1939-standardit OSI-mallin kerroksissa. (Falch, 2022).

Keskeisimmät standardit tämän työn kannalta ovat fyysisen kerroksen määrittelevä J1939/11, siirtokerroksen määrittelevä J1939/21 sekä sovelluskerroksen määrittelevä J1939/71. J1939-standardi säilyttää suuren osan CAN-väylän hyödyistä, kuten äärimmäisen varmatoimisuuden ja erinomaisen virheenhavaitsemiskyvyn. Tämä on johtanut J1939-pohjaisten laajennusten käyttöön laajasti eri tarkoituksissa, kuten merenkulun käytössä yleistynyt NMEA2000-standardi ja sotilaskäytössä oleva MilCAN. (Falch, 2022).



KUVA 9. J1939-standardit OSI-mallissa (Ixxat, n.d.)

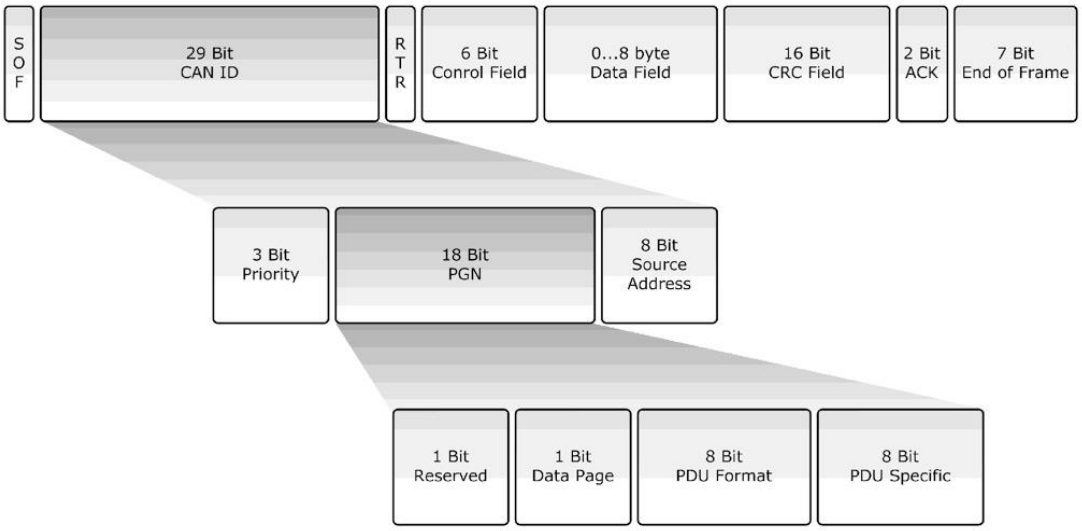
J1939/11:n fyysinen kerros perustuu voimakkaasti ISO 11898 -standardiin, joka määrittelee CAN-väylän fyysistä kerrosta. Se on määritelty kierretyksi, suojatuksi parikaapeliksi, jonka tiedonsiirtonopeus on 250kbit/s. Myöhemmissä fyysisen kerroksen standardeissa määritellään esimerkiksi 500kbit/s nopeudella toimiva väylä ja suojaamattoman parikaapelin käyttö. Väylän maksimipituus on 40 m ja se voi sisältää 30 hallintayksikköä. (Vector, n.d.a) Verkkoa voidaan tarvittaessa jatkaa erillisillä kytkimillä.

3.1 Siirtokerros ja sovelluskerros

J1939:n merkittävimmät erot CAN-protokollaan verrattuna ovat siirto- ja sovelluskerroksissa. J1939:ssä viestit jaetaan niiden sisällön perusteella parametriryhmiin, jotka varustetaan erillisellä PGN-tunnisteella (Parameter Group Number). Lisäksi J1939 -standardikokoelmassa määritellään tapa lähettää yli kahdeksantavuisia viestejä. Seuraavissa alaluvuissa käsitellään näitä toimintoja, sekä muita siirto- ja sovelluskerrosten keskeisiä ominaisuuksia.

3.1.1 Viestikehys

J1939/21-standardissa määritelty viestikehys käyttää CAN 2.0B -standardin mukaista 29-bittistä tunnistekenttää. Viestikenttä pidennetyllä tunnisteella on esitelty kuvassa 10, jossa nähdään myös tunnistekentän eri osat. Tunnistekentässä on havaittavissa merkittäviä eroja verrattuna normaaliin CAN-kehykseen. Ensimmäiset kolme bittiä ilmaisevat viestin prioriteetin, kolmen dominantin 0-bitin ollessa tärkein mahdollinen viesti. PGN-kenttä ilmaisee, minkälaista dataa viesti sisältää. Tunnistekentän kahdeksan viimeistä bittiä ilmaisevat lähettäjän uniikin osoitteen. (Kvaser, n.d.)



KUVA 10. Viestikenttä pidennetyllä 29 bitin tunnisteella (Voss, 2020, muokattu).

3.1.2 PGN ja SPN

PGN:t ovat yksi J1939-standardin keskeisimmistä käsitteistä ja ne määritellään sovelluskerrosta koskevassa J1939/71-standardissa (Voss, 2020). Nämä parametriryhmät sisältävät nimensä mukaisesti joukon yksittäisiä parametrejä, jotka liittyvät toisiinsa. Esimerkki tällaisesta on kuvassa 11, jossa nähdään sähköisen moottorinhallintayksikön parametriryhmä. Kyseinen PGN sisältää tiedon muun muassa moottorin nopeudesta ja väännöstä.

Kuvan 10 mukainen PGN-kenttä on jaettu neljään osaan: Reserved-bitti, Data Page -bitti, PDU Format -tavu (Protocol Data Unit) ja PDU Specific -tavu. Reserved-bitti on varattu protokollan laajentamiselle ja on lähetyksissä täten 0. Data Page laajentaa käytettävien parametriryhmien lukumäärää. PDU Format määrittelee, onko kyseessä broadcast-tyyppinen lähetys vai tietylle laitteelle tarkoitettu viesti. Mikäli PDU Format on 0–239, on kyseessä osoitteellinen viesti ja kentän arvoilla 240–255 viesti on broadcast-tyyppinen. PDU Specific määrittää vastaanottajan osoitteen, jos viesti on osoitteellinen. Broadcast-tyyppisessä viestissä tällä määritetään lisää parametriryhmänumeroita. (Kvaser, n.d.)

PGN61444 - Electronic Engine Controller 1 - EEC1

Transmission Repetition Rate: Engine speed dependent
Data Length: 8 bytes
Data Page: 0
PDU Format: 240
PDU Specific: 4
Default Priority: 3
Parameter Group Number: 61444 (0x00F004)

Bit Start/Byte	Length	SPN ID	SPN Description
1.1	4 bits	899	Engine Torque Mode
2	1 byte	512	Driver's Demand Engine - % Torque
3	1 byte	513	Actual Engine - Percent Torque
4-5	2 bytes	190	Engine Speed
6	1 byte	1483	SA of Controlling Device for Engine Control
7.1	4 bits	1675	Engine Starter Mode
8	1 byte	2432	Engine Demand - Percent Torque

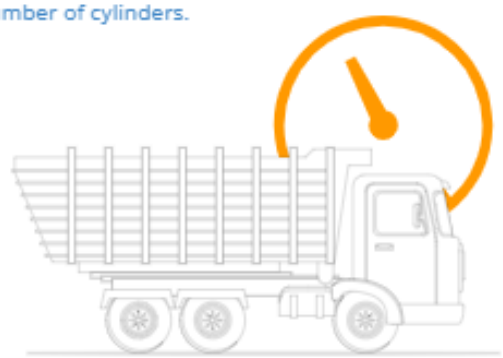
KUVA 11. Esimerkki PGN:n sisältämistä parametreistä (Falch, 2022).

SPN (Suspect Parameter Number) on yksittäisen parametrin tunnistenumero. Kuvassa 12 esitellään yksi PGN 61444:n sisältämistä parametreistä, SPN 190, joka sisältää tiedon moottorin nopeudesta. Kyseisellä SPN:lle on määritetty alku-bitti ja pituus viestin datakentän sisällä sekä useita arvoja, joiden avulla lukema bitteinä voidaan muuntaa kierrosnopeudeksi. Valmiiksi luodut PGN:t ja SPN:t on määritelty standardissa J1939/71.

J1939 SPN **190** | Engine Speed

Actual engine speed which is calculated over a minimum crankshaft angle of 720 degrees divided by the number of cylinders.

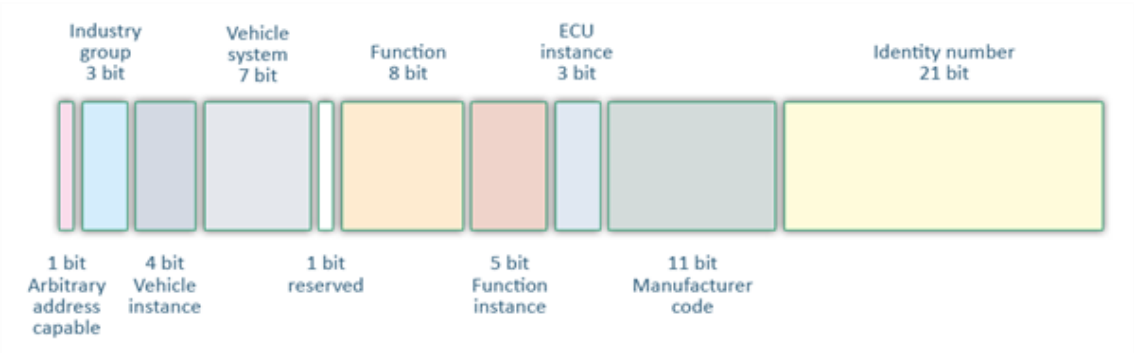
Bit start: **24**
Length: **2 bytes**
Scale: **0.125**
Offset: **0**
Unit: **rpm**
min-max: **0-8031.875**
PGN reference: **61444**



KUVA 12. Parametrin SPN190 sisältö (Falch, 2022).

3.1.3 Address Claiming

Laitteiden osoitteet ovat useimmiten ennalta määriteltyjä ja ne asetetaan järjestelmän käynnistyksen yhteydessä. Myöhemmin verkkoon liittyvät laitteet voivat käyttää Address Claim -toimintoa, jonka avulla niille voidaan antaa osoite. Tämä toimii erillisten PGN:ien avulla ja siihen on kaksi tapaa. Molemmissa tavoissa käytetään väylän laitteiden nimiä. Nimi on kahdeksan tavua pitkä ja sisältää kuvauksen laitteesta. (Ixxat n.d.). Nimikentän formaatti on esitelty kuvassa 13.



KUVA 13. J1939 mukaisen nimikentän formaatti (Ixxat, n.d.).

Ensimmäisessä metodissa laite lähettää väylälle kuvan 14 mukaisen PGN:n, jolla se pyytää väylän muita laitteita lähettämään parametriryhmän 60928 mukaisen viestin. PGN 60928 on Address Claimed -viesti, jossa laite lähettää oman nimensä. Pyytämällä väylän laitteita lähettämään omat nimensä laite saa tiedon väylällä olevista laitteista sekä niiden osoitteista ja voi valita itselleen osoitteen. (Voss, 2019)

Parameter Group Name	**Request**
Parameter Group Number	59904 ($00EA00_{hex}$)
Definition	Requests a Parameter Group from a single device or all devices in the network.
Transmission Rate	User defined (no more 2 to 3 times a second is recommended)
Data Length	3 bytes (CAN DLC = 3)
Extended Data Page (R)	0
Data Page	0
PDU Format	234
PDU Specific	Destination Address (Global or Peer-to-Peer)
Default Priority	6
Data Description	Requested Parameter Group Number = PGN 60928

KUVA 14. Request for Address Claimed -viesti (Voss, 2019).

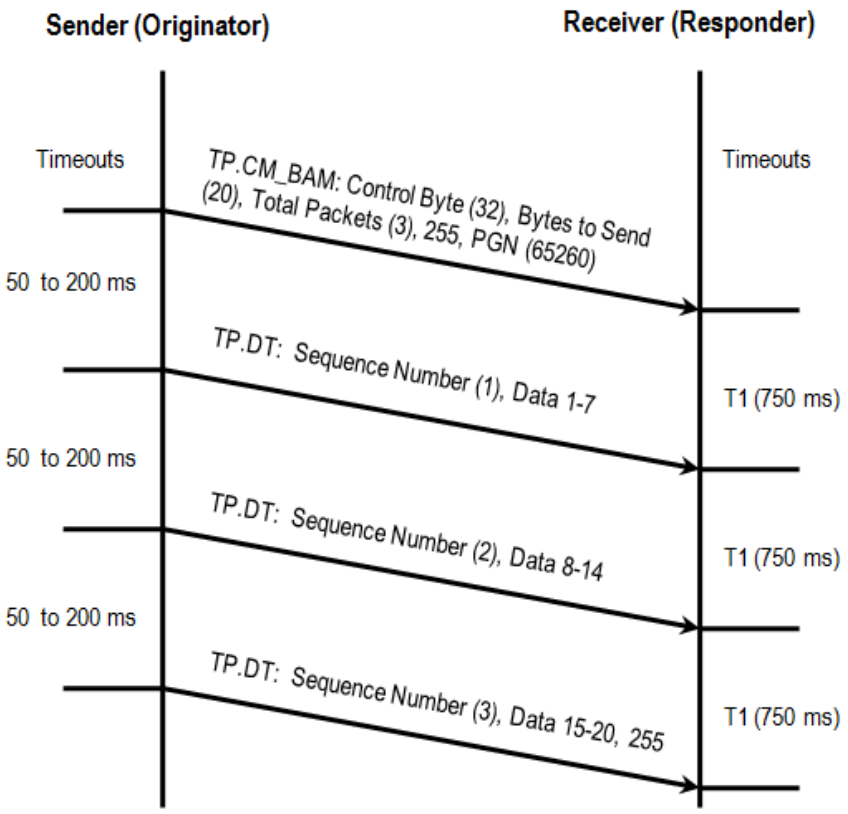
Toinen tapa saada osoite on lähettää Address Claimed -viesti väylälle, jolloin muut laitteet vertaavat pyydettyä osoitetta omiin taulukkoihinsa. Mikäli osoite ei ole käytössä, saa laite pyytämänsä osoitteen ja väylän muut laitteet päivittävät sen taulukkoihinsa. Konfliktitilanteessa laite, jolla on pienempi nimikentän arvo, saa osoitteen. (Kvaser, n.d.)

3.1.4 Siirtoprotokolla

J1939/21 määrittelee tavan lähettää yli kahdeksan tavua pitkiä viestejä, vaikka CAN-väylän datakentän pituus onkin kahdeksan tavua. Tätä ominaisuutta kutsutaan siirtoprotokollaksi (transport protocol). Tämä tapahtuu pilkkomalla viesti seitsemän tavun pituisiksi osiksi, jotka voidaan lähettää CAN-kehyksen sisällä. Kehyksen ensimmäistä datatavua käytetään viestien numerointiin välillä 1–255. Täten lähettävän viestin maksimipituudeksi saadaan 1785 tavua. (Falch, 2022)

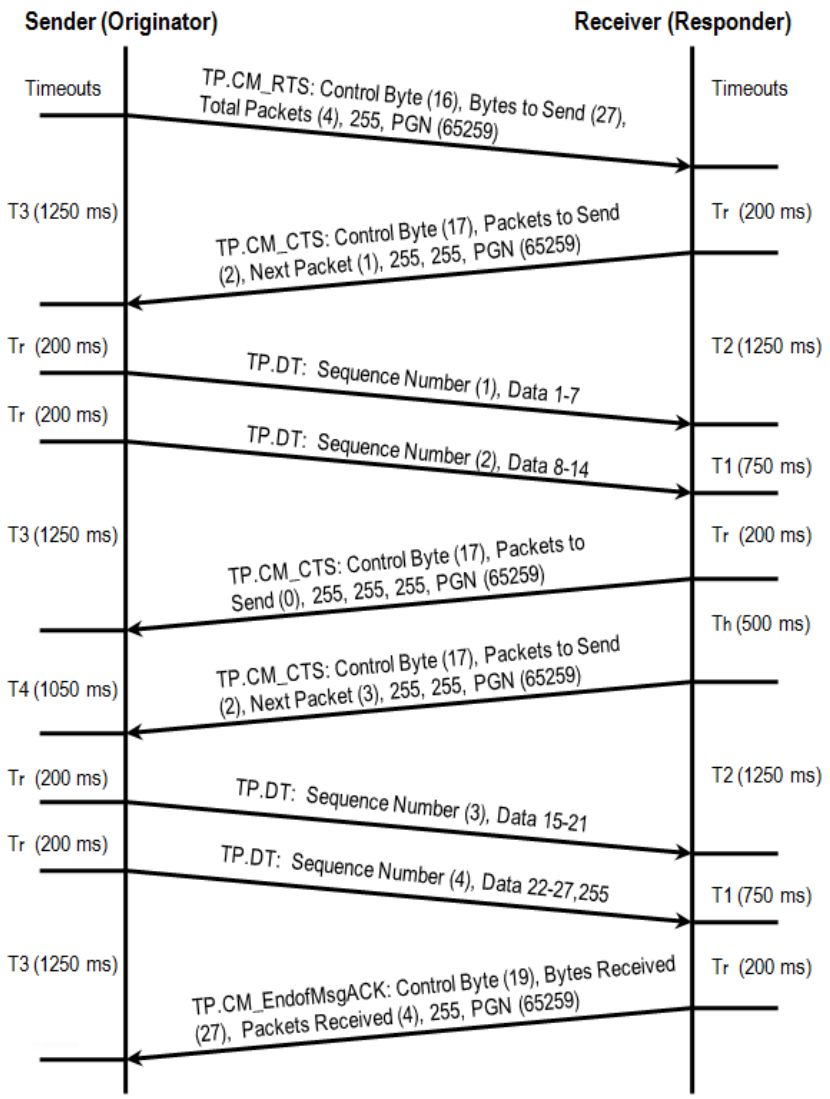
Siirtoprotokollia määritellään dokumentissa kaksi; BAM (Broadcast Announce Message) ja Connection Mode. Näiden monipakettiviestien lähettämiseen käytetään kahta siirtoprotokollassa määriteltyä PGN:ää. Ensimmäinen näistä on TP.CM (Transport Protocol Connection Management), jota käytetään yhteyden hallinnointiin. Datan siirtämiseen käytetään protokollaryhmää TP.DT (Transport Protocol Data Transfer). (Voss, 2018)

BAM-tiedonsiirto on broadcast-tyyppinen ja alkaa BAM-viestillä, jossa ilmoitetaan lähetettävän datan PGN, pituus tavuina ja viestien määrä. Tällainen lähetys ei odota kuittausta vastaanottajilta, vaan viestin lähetys tapahtuu aikarajojen puitteissa. Mikäli aikarajat ylittyvät, yhteys katkaistaan. Muissa tapauksissa yhteys katkaistaan viimeisen paketin saavuttua. Tällainen tiedonsiirtometodi on esitelty aikarajoineen kuvassa 15. (Voss, 2018)



KUVA 15. BAM-tiedonsiirtoprotokollan toiminta (cpeacock 2019).

Connection Mode -tyyppinen tiedonsiirto tapahtuu lähettäjän ja vastaanottajan välisellä yhteydellä. Onnistunut tiedonsiirto taataan käyttämällä kättely- ja kuittausviestejä. Lähettäjä avaa yhteyden lähettämällä RTS-viestin (Request to Send), jolla se tiedustelee, onko vastaanottaja kykenevä vastaanottamaan viestin. RTS-viesti sisältää samat tiedot, kuin BAM-viesti. Mikäli vastaanottaja on kykenevä vastaanottamaan dataa, lähettää se CTS-viestin (Clear To Send), jossa se ilmoittaa seuraavan lähetettävän paketin numeron sekä kuinka monta pakettia se kykenee vastaanottamaan seuraavassa lähetyksessä. Lähettäjä lähettää sovitun määrän viestejä ja jää odottamaan seuraava CTS-viestiä. Vastaanottajan saadessa viimeisen viestin, se lähettää kuittausviestin, jonka jälkeen yhteys suljetaan. Tällainen tiedonsiirto on esitelty kuvassa 16. (cpeacock 2019).



KUVA 16. Connection Mode -tiedonsiirtoprotokolla (cpeacock 2019).

4 Käytetyt ohjelmistot

Seuraavissa alaluvuissa käsitellään opinnäytetyön aikana käytettyjä ohjelmistoja sekä niiden rooleja toteutuksessa. Ohjelmistot on esitelty tärkeysjärjestyksessä. Projektin alussa oletettiin CODESYS- ja EPEC Multitool -ohjelmistojen riittävän toteutukseen, mutta toteutuksen edetessä todettiin olevan tarve myös CAN-kirjastojen muokkaamiseen, joten Vector CANdb++ otettiin käyttöön. Suurin osa toteutuksessa tehdystä työstä tehtiin CODESYS-ympäristössä, käyttäen versiota 3.5. Käytetty Multitoolin versio oli 7.2. ja CANdb++:n versio 3.1.18.

4.1 CODESYS

Codesys on 3S-Smart Software Solutionsin valmistama ohjelmointiympäristö, jota käytetään etenkin ohjelmoitavien logiikoiden sovelluskehityksessä. Ympäristö noudattaa IEC 61131-3 -standardia, joka määrittelee kolme graafista ja kaksi tekstipohjaista ohjelmointitapaa. Graafiset tavat ovat relekaaviota muistuttava tikapuukaavio (LD, Ladder Diagram), toimilohkokaavio (FBD, Function Block Diagram) ja vuokaavio-ohjelmointi (SFC, Sequential Function Chart). (Codesys, n.d.). Graafiset ohjelmointitavat tarjoavat helposti luettavan tavan tuottaa koodia ohjelmoitavan logiikan tarpeisiin ja niiden oppiminen on helpompaa, kuin tekstipohjaisten ohjelmointikielien.

Tekstipohjaiset ohjelmointitavat IEC 61131-3 -standardissa ovat käskylista (IL, Instruction List), joka muistuttaa syntaksiltaan assembly-kieltä ja Pascal-kieltä muistuttava rakenteinen teksti (ST, Structured Text). (Codesys, n.d.). Tekstipohjaiset ohjelmointitavat soveltuvat paremmin suurien ohjelmointikokonaisuuksien toteutukseen, koska ne eivät käytä ennalta määriteltyjä lohkoja ja voivat täten tuottaa saman tuloksen vähemmällä ohjelmoinnilla. Tämän opinnäytetyön ohjelmointi suoritettiin käyttämällä rakenteista tekstiä.

Codesys sisältää myös oman grafiikkaeditorinsa, jonka avulla on mahdollista rakentaa erilaisia näkymiä. Editorissa tehtyjen visualisaatioiden avulla voidaan välittää tietoa käyttäjälle helposti luettavassa muodossa. Editori sisältää valmiita visuaalisia elementtejä, kuten mittareita ja valoja, joita voidaan sijoittaa visualisaa-

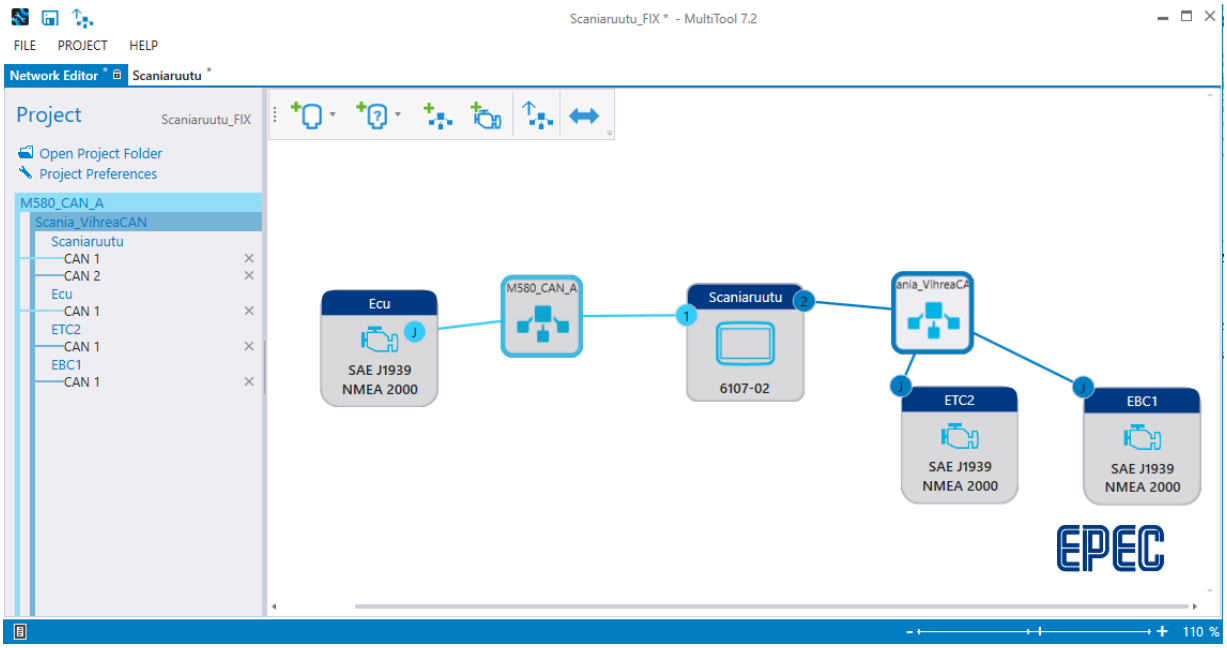
tiosivulle. Näitä elementtejä on mahdollista muokata, esimerkiksi vaihtamalla mittareiden ulkoasua tai asteikon mittaa. Esimerkki visualisaatiosta on esitelty kuvassa 17.

KUVA 17. Codesys-visualisaatio.

Viimeinen tämän työn kannalta keskeisistä ominaisuuksista Codesys-ympäristössä on simulaatio. Nimensä mukaisesti sillä on mahdollista testata järjestelmän toimintaa. Tällä toiminnolla ei tietenkään saa mallinnettua koko järjestelmää, mutta sillä voidaan tarkistaa esimerkiksi ehtolausekkeiden toimintaa tai arvon perusteella ulkoasuaan vaihtavien mittareiden toimintaa.

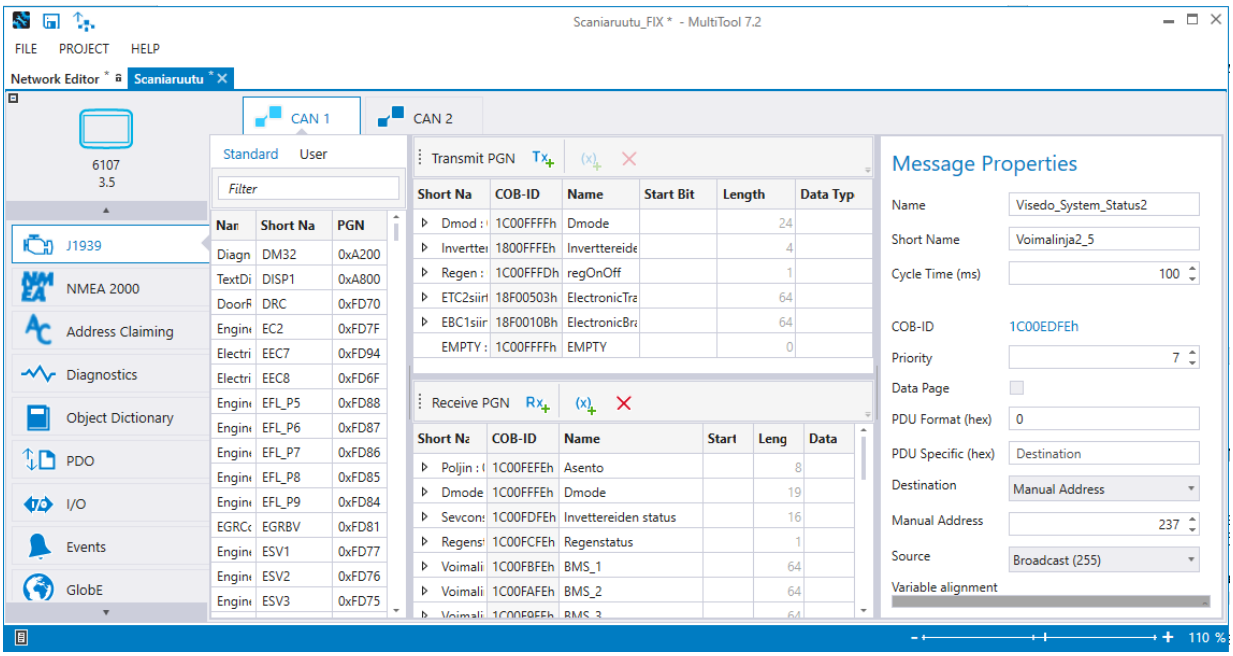
4.2 Epec Multitool

Epec Multitool on graafinen järjestelmäsuunnittelu- ja konfigurointityökalu, jonka avulla voidaan alustaa muun muassa järjestelmän CAN-, I/O- ja diagnostiikka-asetukset Codesys-ympäristöä varten. Näiden asetusten tekeminen manuaalisesti olisi aikaa vievää ja virhealtista, mutta Multitoolin avulla nämä voidaan toteuttaa automaattisesti. (EPEC, 2021). Kuvassa 18 on kuvankaappaus Multitool-projektista, jossa on näyttöyksikkö ja siihen liitetyt hallintayksiköt moottorille, jarruille ja vaihteistolle.



KUVA 18. Epec Multitool, Network Editor -näkymä.

Kuvassa 19 on esiteltynä projektissa käytetyn Epec 6107 -näyttöyksikön J1939-viestiasetukset. Asetukset koskevat moottorinhallintayksikön ja näytön välistä CAN 1-väylää. Kuvassa keskellä ylhäällä näkyvät näyttölaitteen lähettämät PGN:t ja niiden alla sen vastaanottamat PGN:t. Sivun oikealla puolella näkyvät asetukset koskevat yksittäisiä parametriryhmiä ja niiden sisältöä. Tässä näkymässä on mahdollista lisätä viestejä ja muokata niiden sisältöä.



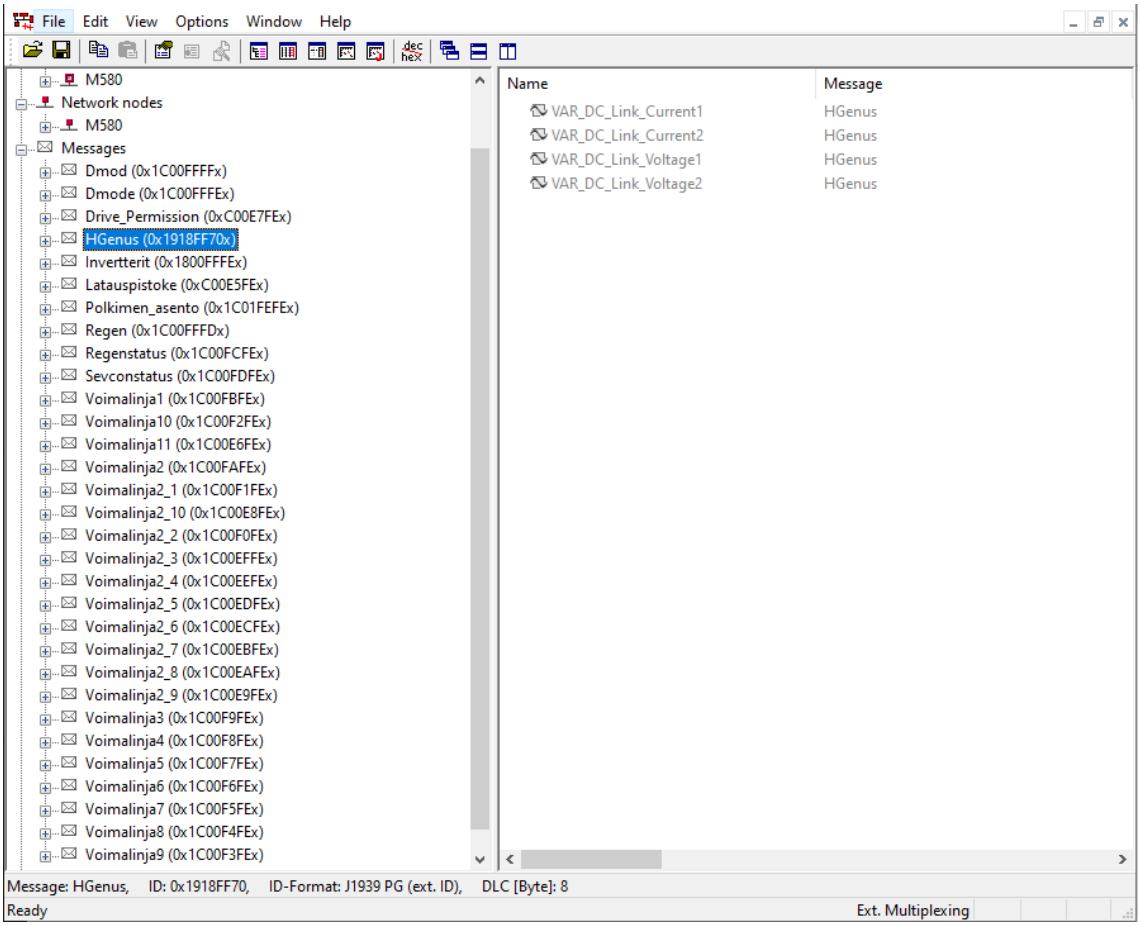
KUVA 19. Epec Multitool J1939-asetukset.

4.3 Vector CANdb++

CANdb++ on saksalaisen Vector Informatikin tuottama ohjelmisto, jonka tärkein toiminto tämän opinnäytetyön kannalta on CAN-tietokantatiedostojen (.dbc) hallinnointi (Vector, n.d.b). Nämä tiedostot sisältävät tarvittavat tiedot raa'an CAN-datan muuntamiseksi käytettävään muotoon.

Ohjelmiston käyttöliittymä on esitelty kuvassa 20. Kuvassa nähdään vasemmalla listattuna viestien nimet ja 29-bittiset CAN-ID:t. Niiden sisältämät signaalit näkyvät näkymän oikealla puolella. Näyttöä ohjelmoidessa tuli pitää huoli, että .dbc-tiedoston sisältämät viestit vastasivat Epec Multitoolin viestejä, jotta CAN-viestien tulkinta onnistuu. CODESYS:issä tehty ohjelma voi täten käyttää CAN-viestien sisältöä suoraan, eikä ohjelmoijan tarvitse kääntää viestejä erikseen.

.



KUVA 20. Vector CANdb++ käyttöliittymä.

5 Toteutus

Projekti toteutettiin yhteistyössä TAMK:n projektitiimin kanssa. Projektitiimi suoritti auton fyysiset muutostyöt ja toimitti tätä opinnäytetyötä koskevat vaatimukset. Projektin aikana järjestettiin tapaamisia myös TAMK:n autohallilla ja näiden käyntien aikana saatiin lisätietoa eri ohjelmistojen toiminnasta sekä tietoa näyttöohjelmiston vuorovaikutuksesta projektin muiden osien kanssa. Nämä olivat ohjelmistokehityksen kannalta tärkeitä käyntejä, koska niiden avulla projektin kokonaisuuden hahmottaminen helpottui ja autopajalla syntyi myös useita korjaus- ja parannusehdotuksia. Lisäksi ohjelman testaaminen käytännössä oli mahdollista vain paikan päällä. Kuvassa 21 on muutostyön kohteena oleva ajoneuvo autohallilla.

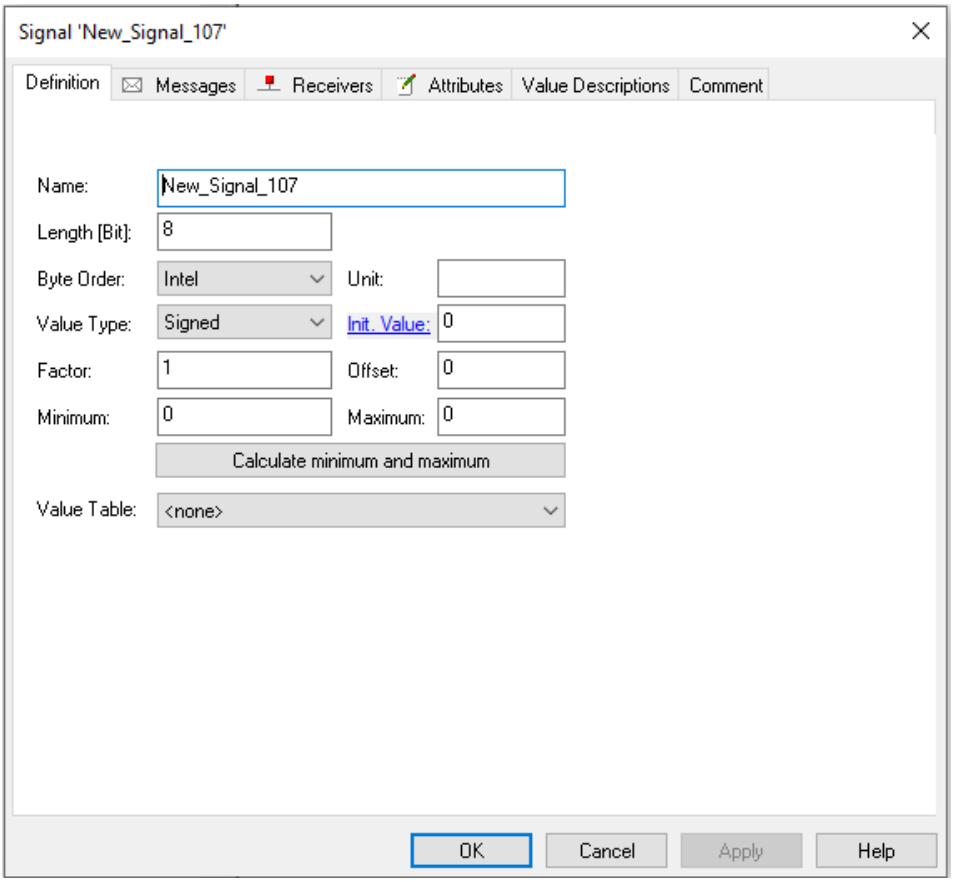
KUVA 21. Ajoneuvo hallilla marraskuussa 2022.

Tässä luvussa kuvaillaan yleisellä tasolla ohjelman luomiseen ja muutokseen tarvittavat vaiheet, joihin viittaan ohjelman muutoshistoriaa koskevassa kappaleessa. Vaiheet käydään läpi ohjelmisto kerrallaan prosessin suoritusjärjestyksessä. Versiohistoria ja visualisaation suunnittelu on esitelty kappaleessa 6.

5.1 CANdb++ -kirjasto

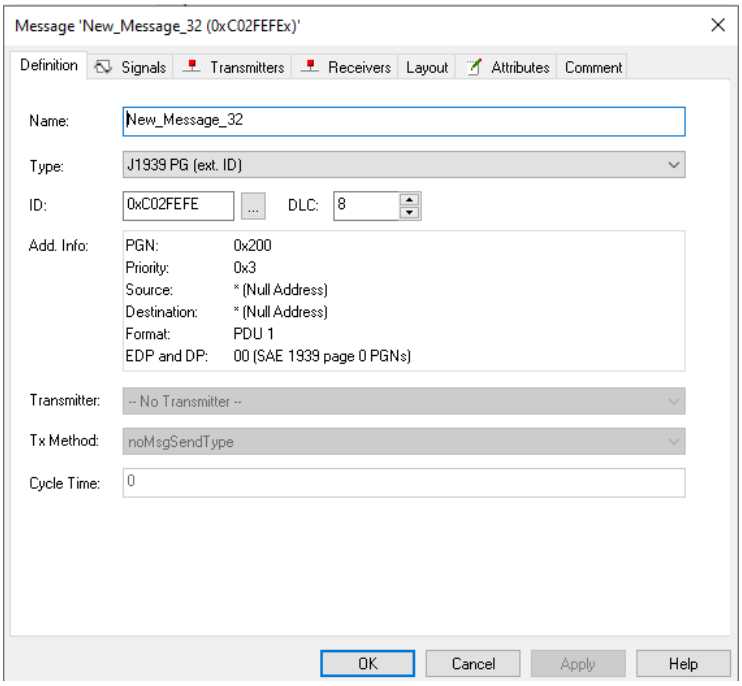
Projektin alussa saatiin käyttöön keskeneräinen CANdb++ -kirjaston *Screen.dbc*. Tämän tiedoston tulisi sisältää kaikki näyttöä koskevat signaalit. Käytännössä tämä toteutetaan luomalla tarvittavat signaalit ja niille tarvittavat viestit. Tiedot näiden luomista varten saatiin joko projektiryhmältä tai muiden .dbc-tiedostojen sisältä, esimerkiksi invertereiden tai moottorinhallintayksiköiden kirjastoista. Näiden kirjastojen ylläpito kuului periaatteessa moottorinhallintayksikön toteutukseen, mutta tämän opinnäytetyön aikana *Screen.dbc*:stä tarkistettiin PGN:ien rakenteita myöhempiä vaiheita varten ja sinne luotiin tarpeen vaatiessa uusia PGN:iä.

Kuvassa 22 on kuvattuna viestin luomiseen käytettävä ikkuna, jossa tämän opinnäytetyön kannalta kiinnostavimmat kentät ovat ”Name”, ”Length [Bit]” ja ”Value Type”. Huomattavaa on, että pituuskenttä määrää myös hieman viestirakennetta, koska CAN-kehykseen mahtuu korkeintaan 8 datatavua.



KUVA 22. Uuden signaalin luominen CANdb++:ssa.

Kuvassa 23 nähdään vastaava ikkuna viestikehyksen luomiselle. Viestin nimen lisäksi tässä ikkunassa on syytä kiinnittää huomiota ”ID”-kenttään. Tämä on 29-bittinen ID, joka määrittelee muun muassa viestin lähteen ja prioriteetin. ”ID”-kenttään tehdyt muutokset näkyvät ”Add. Info” -kentässä, joka selkeyttää viestin rakentamista ja toimii hyvänä tarkistusmetodina virheiden varalle. Signaalien lisääminen onnistuu melko helposti ”Signals”-välilehdeltä.



KUVA 23. Uuden viestikehyksen luominen CANdb++:ssa.

5.2 Multitool-projekti

Muutosten tekeminen Multitooliin on melko yksinkertaista, koska se on lähinnä CANdb++ -viestien rakenteiden kopioimista. Näytön J1939-asetuksista valitaan kuvassa 24 yläoikella näkyvä ”Receive PGN”, joka lisää uuden vastaanotettavan viestikehyksen projektiin. ”Short Name / PGN”-kenttään annettaan CANdb++:ssa luodulle viestille annettu nimi ja ”COB-ID”-kenttään sille annettu ID. ID:n säädöt tapahtuvat kuvan 19 oikeassa reunassa näkyvästä ”Message Properties”- ikkunasta. Tämän jälkeen valitaan luotu viesti ja luodaan uudet signaalit ”(x)+”-painikkeesta.

Receive PGN

Short Name / PGI	COB-ID	Name	Start Bit	Length	Data Type
Voimalinja9 : 0x0(	1C00F3FEh	Visedo_Mot_Status2		56	
Voimalinja10 : 0x(	1C00F2FEh	Visedo_Mot_Status3		48	
Voimalinja2_1 : 0x	1C00F1FEh	BMS_1		64	
Voimalinja2_2 : 0x	1C00F0FEh	BMS_2		64	
Voimalinja2_3 : 0x	1C00EFFEh	BMS_3		64	
Voimalinja2_4 : 0x	1C00EEFEh	Visedo_System_Status		16	
Voimalinja2_5 : 0x	1C00EDFEh	Visedo_System_Status		8	
Voimalinja2_6 : 0x	1C00ECFEh	Visedo_System_Status		64	
Voimalinja2_7 : 0x	1C00EBFEh	Visedo_System_Status		32	
Voimalinja2_8 : 0x	1C00EAFEh	Visedo_Mot_Status1		64	
Voimalinja2_9 : 0x	1C00E9FEh	Visedo_Mot_Status2		56	
Voimalinja2_10 : (	1C00E8FEh	Visedo_Mot_Status3		48	
Drive_Permission	1C00E7FEh	Drive_Permission_Stat		1	
Voimalinja11 : 0x(	1C00E6FEh	BMS_4		32	
Latauspistoke : 0x	1C00E5FEh	Latauspistokkeen_tila		9	
HGenus : 0x1180(	1918FF00h	HGenus		64	

KUVA 24. Vastaanotettavien PGN:ien luettelo Multitoolissa.

Kuvassa 25 on avattuna yksi luoduista PGN:istä, joka sisältää neljä signaalia. Näille annettaan nimet, jotka vastaavat *Screen.dbc*:ssä olevia tietoja. Tässä vaiheessa on oltava tarkkana, että PGN:t ovat samassa järjestyksessä, kuin *Screen.dbc*:n sisältämät PGN:t.

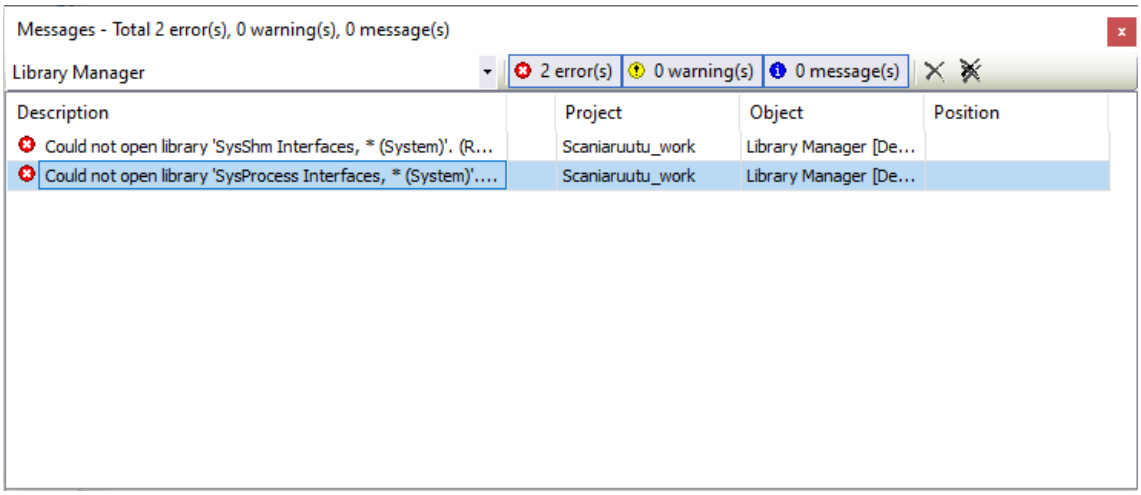
Voimalinja11 : 0x(	1C00E6FEh	BMS_4		32	
Latauspistoke : 0x	1C00E5FEh	Latauspistokkeen_tila		9	
HGenus : 0x1180(	1918FF00h	HGenus		64	
		VAR_DC_Link_Current	0	16	WORD
		VAR_DC_Link_Voltage	16	16	WORD
		VAR_DC_Link_Current	32	16	WORD
		VAR_DC_Link_Voltage	48	16	WORD

KUVA 25. PGN:n sisältö Epec Multitoolissa.

Haluttujen muutosten ollessa valmiita, voidaan palata takaisin Multitoolin ”Network Editor” -näkymään, joka on esitelty aiemmin kuvassa 18. Tässä näkymässä valitaan näyttökuvake, jonka alta avautuu pieni valikko, josta valitaan ”Create CODESYS Project”. Tämä luo Multitool-projektin pohjalta uuden CODESYS-projektin, jossa on valmiina tarvittavat alustukset. Tämä kestää yleensä muutaman minuutin, mutta säästää todella paljon aikaa ja virheitä verrattuna manuaalisesti tehtyihin alustuksiin.

5.3 CODESYS-projekti

CODESYS-projektin ensimmäinen vaihe on kirjastojen konfigurointi. Kuvassa 26 nähdään esimerkki virheilmoituksista, joita syntyy, kun projekti avataan. Kuvassa olevat virheviestit ovat projektin versiosta, jonka kirjastojen konfigurointi on suoritettu jo, joten virheitä on vain kaksi ja ne menevät pois projektin uudelleenrakentamisella. Konfiguroimattomassa projektissa näitä voi olla useita satoja.



KUVA 26. CODESYS-projektin kirjastoista johtuvia varoitusviestejä.

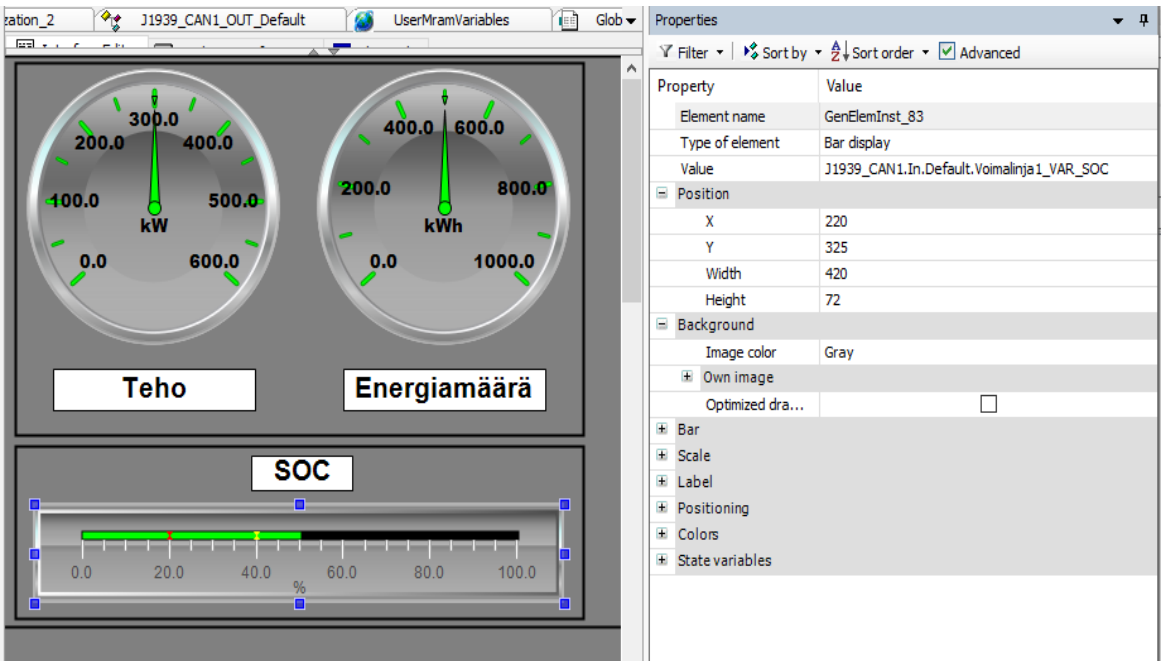
Kirjastojen konfigurointi tapahtui menemällä projektin ”Library Manager” -välilehdelle (kuva 27) ja etsimällä ensisijaisesti virheilmoitusten perusteella ongelmana olevia kirjastoja. Näitä kirjastoja muokattiin sitten joko poistamalla, lisäämällä tai vaihtamalla niiden versiota. Universaalia ratkaisua ei löytynyt, vaan jokainen kerta sisälsi suuren määrän kokeilua. Muutosten valmistuttua projekti rakennettiin uudestaan työkalurivin ”Build”-toiminnolla. Tämä prosessi toistettiin, kunnes virheitä ei enää syntynyt uudelleenrakentamisen yhteydessä.

Name	Namespace	Effective version
3SLicense = 3SLicense, 3.5.12.0 (3S - Smart Software Solutions GmbH)	_3S_LICENSE	3.5.12.0
Analyzation = Analyzation, 3.5.11.0 (System)	Analyzation	3.5.11.0
BreakpointLogging = Breakpoint Logging Functions, 3.5.5.0 (3S - Smart Software Solutions GmbH)	BPLog	3.5.5.0
CAA Device Diagnosis = CAA Device Diagnosis, 3.5.15.0 (CAA Technical Workgroup)	DED	3.5.15.0
CAA DTUtility = CAA DTUtil Extern, 3.5.12.0 (CAA Technical Workgroup)	DTU	3.5.12.0
CmpApp = CmpApp, 3.5.15.0 (System)	CmpApp	3.5.15.0
CmpErrors = CmpErrors, 3.3.1.40 (System)	CmpErrors	3.3.1.40
EPEC_6000BKL = 6000BackLightControl, 3.5.6.1 (Epec Oy)	EPEC_6000BKL	3.5.6.1
EPEC_6000BootApp = 6000BootApplication, 3.5.6.0 (Epec Oy)	EPEC_6000BootApp	3.5.6.0
EPEC_6000CANVXD = 6000CANVXD, 1.0.0.1 (Epec Oy)	EPEC_6000CANVXD	1.0.0.1
EPEC_6000INT = 6107Int, 3.2.1.0 (Epec Oy)	EPEC_6000INT	3.2.1.0
EPEC_6000IODRV = 6000IoDrv, 3.5.6.1 (Epec Oy)	EPEC_6000IODRV	3.5.6.1
EPEC_6000MRAM = 6000Mram, 3.5.6.4 (Epec Oy)	EPEC_6000MRAM	3.5.6.4
EPEC_6000Multimedia = 6000Multimedia, 3.5.13.1 (Epec Oy)	EPEC_6000Multimedia	3.5.13.1
EPEC_6000SocketCAN = 6000SocketCAN, 3.5.13.0 (Epec Oy)	EPEC_6000SOCKETCAN	3.5.13.0
EPEC_6000SPRM = 6000SystemParameters, 3.5.13.1 (Epec Oy)	EPEC_6000SPRM	3.5.13.1
EPEC_6000USB = 6000UsbDrive, 3.5.6.0 (Epec Oy)	EPEC_6000USB	3.5.6.0
EPEC_ACL = AddressClaiming, 1.0.0.1 (Epec Oy)	EPEC_ACL	1.0.0.1
EPEC_ADC = AD Conversion, 3.0.2.5 (Epec Oy)	EPEC_ADC	3.0.2.5
EPEC_AI = Analog Input, 3.0.0.5 (Epec Oy)	EPEC_AI	3.0.0.5
EPEC_AZURE = Azure, 1.1.0.4 (Epec Oy)	EPEC_AZURE	1.1.0.4
EPEC_CANL2 = CANL2 CANVXD, 4.0.0.2 (Epec Oy)	EPEC_CANL2	4.0.0.2
EPEC_CANopen = CANopen, 4.0.5.9 (Epec Oy)	EPEC_CANopen	4.0.5.9
EPEC_CANVXD = CANVXD API, 4.0.0.1 (Epec Oy)	EPEC_CANVXD	4.0.0.1
EPEC_CSVParser = CSV Parser, 3.1.1.3 (Epec Oy)	EPEC_CSVParser	3.1.1.3
EPEC_DLOG = Datalogger, 1.0.0.0 (Epec Oy)	EPEC_DLOG	1.0.0.0
EPEC_DOD = Digital Output Diagnostic, 3.0.2.1 (Epec Oy)	EPEC_DOD	3.0.2.1
EPEC_DT = DataTransfer, 3.0.3.0 (Epec Oy)	EPEC_DT	3.0.3.0

KUVA 27. "Library Manager"-välilehti ja osa kirjastoista.

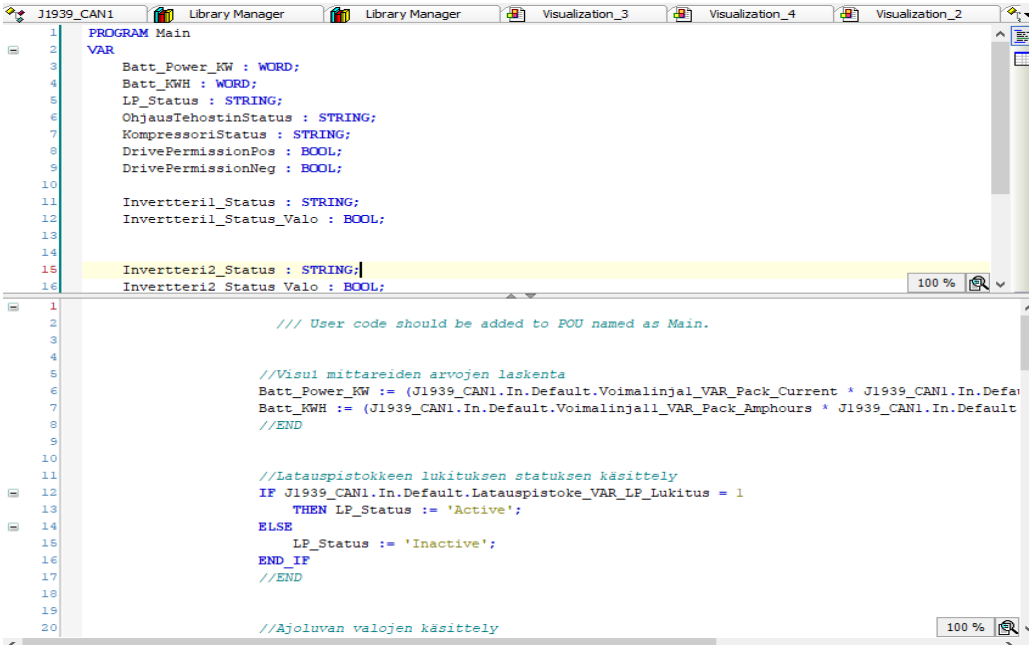
Kirjastojen korjaamisen jälkeen alkaa itse muutosten teko. Tämä tapahtuu kahdessa eri paikassa: Visualisaatiosivuilla, jotka sisältävät näytölle tulevan näkymän ja ”Main”-sivulla, joka sisältää ohjelman logiikan niiden muuttujien osalta, joita ei voi käyttää suoraan. Suurin osa muuttujista on vain lukuarvoja, joiden arvot voidaan näyttää suoraan visualisaatioelementtien avulla. Muuttujia ja niiden tarkoitusta kuvataan tarkemmin myöhemmin. CODESYS-projektin tekeminen oli työmäärältään suurin osa tästä opinnäytetyöstä ja myös teknisesti monimutkaisin.

Kuvassa 28 on nähtävissä, miten muuttujia voidaan käyttää ”suoraan”. Kuvan muuttuja koskee akuston varauksen tilaa, joka tulee prosenttilukemana näytölle. Tällainen muuttuja ei vaadi toimenpiteitä, vaan sen nimen voi syöttää elementin asetusikkunan ”Value”-kenttään, jolloin elementin arvo seuraa viestissä tullutta arvoa. Poikkeuksena tähän ovat tekstien tulostamiset eri elementteihin, jolloin elementeissä pitää määritellä %-operaattorin avulla elementtiin kirjoitettavan muuttujan tietotyyppi, esimerkiksi %s (string) tai %d (kokonaisluku).



KUVA 28. Muuttujan käyttö suoraan.

Mikäli muuttuja vaatii käsittelyä, esimerkiksi tilanteessa, jossa viestin numeroarvo pitää muuttaa tilanneviestiksi tai lampun tilaksi, täytyy se tehdä ”Main” välilehdellä. Tämä välilehti on esitelty kuvassa 29, jossa nähdään ylhäällä julistettuja muuttujia, jotka eivät tule viesteistä, vaan ovat CODESYS:issä luotuja. Nämä toimivat muuten täysin samoin, kuin viestien sisältämät muuttujat. Alempana kuvassa on ST-ohjelmointitavalla tuotettua koodia, jossa viestien arvoja käsitellään.



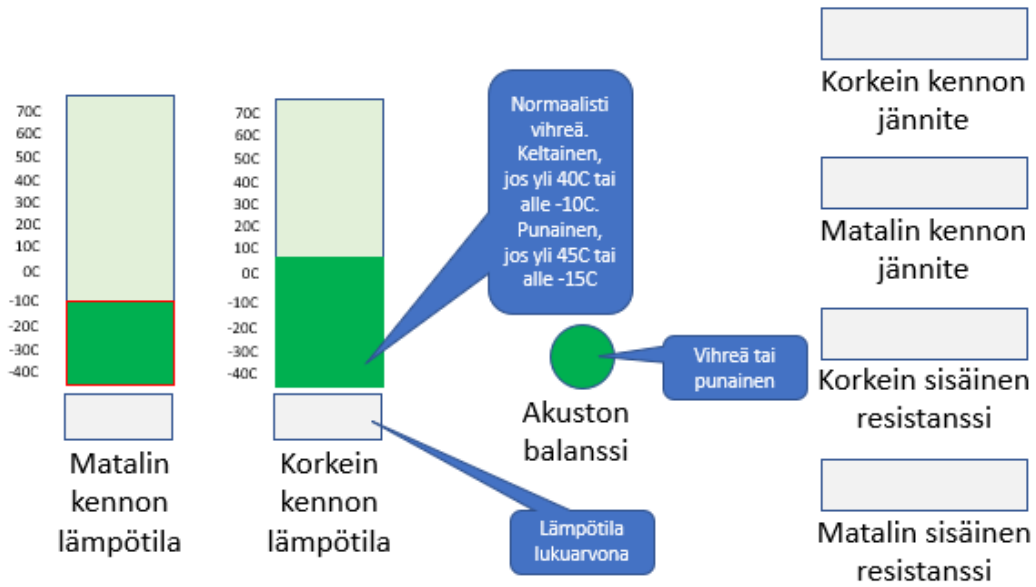
KUVA 29. "Main"-välilehti CODESYS:issä.

6 Projektin kulku ja versiot

Projektin ensimmäiset versiot olivat lähinnä CODESYS:in käytön opettelua varten ja niitä ei siitä johtuen käydä läpi kovin tarkasti. Seuraavissa luvuissa esitellyt versiot on rajattu merkittävien muutosten perusteella kahteen versioon, joista jälkimmäisen todettiin täyttävän annetut vaatimukset. Näiden sisällä on tapahtunut erilaisia korjaustoimenpiteitä ja muokkauksia, joita käsitellään tulevissa luvuissa, mikäli ne sisältävät jotain kyseisen version kannalta keskeistä.

6.1 Vaatimukset

Visualisaatioiden vaatimukset toimitettiin kuvallisessa muodossa, johon oli merkitty sivukohtaisesti halutun datan nimi, minkälaisessa muodossa se halutaan esitettäväksi sekä mahdolliset erityistoiminnallisuudet. Seurattavat muuttujat olivat erilaisia moottorinhallintayksikön ja akunvalvontajärjestelmän tietoja sekä alijärjestelmien, kuten paineilmajarrujen kompressorin tilatietoja. Yhden visualisaatiosivun vaatimukset on esitelty kuvassa 30. Esimerkiksi ”Korkein kennon lämpötila”-muuttuja tulisi esittää palkkimittarina, joka muuttaa väriään akuston kennon korkeimman mitatun lämpötilan mukaan ja tämä lämpötila tulisi esittää myös numeerisena mittarin yhteydessä.



KUVA 30. Visualisaation vaatimuksia.

Toteutuksen aikana annetuista vaatimuksista poikettiin ajoittain esimerkiksi elementtien sijoittelun suhteen. Pääosa näistä muutoksista tehtiin tilankäytön optimoinnin vuoksi. Muutama elementti jätettiin lisäämättä lopullisessa versiossa, koska koettiin, että niiden lisääminen olisi joko tarpeetonta tai senhetkisen toteutuksen puitteissa mahdotonta. Esimerkkinä tällaisesta on moottorinhallintayksikön toimintaa ilmaiseva vihreä lamppu. Mikäli yksikkö ei toimisi, ei näytölle tulisi dataa lainkaan ja täten sen toimintavirhe olisi ilmiselvä. Tämän perusteella merkkivalo koettiin tarpeettomaksi nykyisessä toteutuksessa.

Liitteessä 1 on listattu vaatimukset, jotka toimitettiin projektin alussa. Annetuista parametreistä on kirjoitettu ylös lyhyt toimintakuvaus, mahdollisia huomioita sekä niiden toteutuksen tilanne. Vihreällä merkattujen muuttujien toteutus on valmis, siniset ovat muuttujia, jotka lisätään suurella todennäköisyydellä myöhemmin tai niiden toteutus odottaa projektin etenemistä ja punaisella merkatut muuttujat on jätetty pois toteutuksesta kokonaan. ”Testattu”-sarakkeen merkintä tarkoittaa, että muuttuja on testattu CODESYS:in simulaatiotoiminnolla ja havaittu toimivaksi.

6.2 Versio 1

Versioon 1 mennessä oli saavuttu saatujen vaatimusten perusteella liitteen 2 mukaiseen toteutukseen. Neljän visualisaatiosivun muodostama valikkorakenne vaikutti hyvältä kompromissilta yksittäisen sivun luettavuuden ja valikon monimutkaisuuden suhteen. Tässä rakenteessa visualisaatiosivu 1 toimii eräänlaisena päävalikkona, josta voi siirtyä muihin sivuihin sen vasemmasta laidasta löytyvien painikkeiden avulla. Ensimmäiselle sivulle on myös pyritty keräämään ajon aikana keskeisimpiä tietoja, kuten akuston teho, energiamäärä ja varauksen tila.

Visualisaatiosivulle 2 on kerätty eri laitteiden tilatietoja, joita seurataan vain binäärisellä päällä/pois-tasolla lamppujen avulla. Sivulta 3 löytyy osittain samojen laitteiden tarkempia tilaviestejä, joiden avulla voidaan saada tarkempaa tietoa niiden toiminnasta. Sivulla 4 on akuston tilatietoja, joita seurataan useilla eri elementeillä. Huomattavaa tällä sivulla on kaksi mittaria, joiden elementtien asetukset on säädetty siten, että ne vaihtavat väriä arvonsa mukaan, jotta kuljettaja voi saada nopeasti jonkinlaisen tiedon niiden tilasta. ”Main”-välilehdellä on lyhyt

pätkä koodia, jolla lasketaan akun teho kilowatteina, sekä muutetaan kompressorin ja ajotehostimen lähettämä numeerinen tilaviesti selkokieliseksi status-viestiksi ja laitteen tilaa kuvaavan lampun bool-muuttujaksi.

6.3 Versio 2

Version 2 visualisaatiosivut ja koodi löytyvät liitteestä 3. Edellisen version jälkeen pidettiin tapaaminen pajalla projektitiimin kanssa, ja samalla testattiin koodin toimivuutta sen hetkisten mahdollisuuksien mukaan. Auto oli tässä kohtaa vielä kasaamatta, mutta näytön toimintaa ilman CAN-liikennettä pystyttiin testaamaan. Näillä testeillä selvisi, että valikkorakenne toimii hyvin, mutta visualisaatiosivut ovat liian suurikokoisia ja osa niistä menee näytön ulkopuolelle. Lisäksi kiertovesipumpun tilan seurannalle ei keksitty hyvää seurantatapaa ja ECU:n tilan seuraaminen todettiin mahdottomaksi, joten niiden seurantaan käytettävät elementit jätettiin tästä versiosta pois.

Uusi tilarajoite selvitettiin kokeellisesti skaalamaalla visualisaatiota pienemmäksi pajalla, kunnes se mahtui kokonaan ruudulle. Täten selvisi visualisaation maksimikooksi noin 700x414 pikseliä. Tämä oli huomattavasti edellistä versiota pienempi, joten siitä aiheutui suunnitellessa huomattavia ongelmia. Lopullinen näkymä ruudulla vaikuttaa skaalautuvan itsekseen, joten lopullinen koko täytyy määrittää kokeellisesti paikan päällä. Neljän visualisaatiosivun rakenne koettiin toimivaksi, joten siitä ei haluttu luopua, jos ei ole pakko, mutta sivujen pienemmän koon aiheuttamat luettavuusongelmat piti ratkaista.

Tätä lähdettiin ratkaisemaan selvittämällä jokaiselle elementille minimikoko, jonka arvioitiin säilyttävän riittävän luettavuuden. Tämän jälkeen tehtiin kehys, joka oli uuden kokorajan mukainen ja järjesteltiin elementit sen sisään siten, että kaikki mahtuvat siististi. Tämä johti melko huomattaviin muutoksiin ensimmäisellä sivulla, mutta muut sivut selvisivät vähemmillä rakenteen muutoksilla. Näistä esimerkkinä on eristevuotoilmaisimen tilavalo, joka siirrettiin sivulle 2, koska ensimmäisellä sivulla tila oli ongelma ja sivulta 2 oli poistettu aiemmin mainitut elementit.

Seuraava vaihe oli luettavuuden parantaminen. Tässä vaiheessa pyrittiin kasvattamaan tekstien fonttikokoa ja paksuntamaan niiden viivoja, jotta pienempien mittareiden arvot näkyisivät selkeästi. Mittareille valittiin myös selkeimmät tyyliversiot valmiista asetuksista ja kontrastia kasvatettiin värivalinnoilla. Sivun 1 akuston varausmittaria muokattiin siten, että mittarin väri muuttuu sen arvon mukaan. Lisäksi ajolupa-muuttujan seurantavaloista tehtiin eriväriset, jotta näistä tiedoista saisi jonkinnäköisen kuvan pelkällä vilkaisulla. Kuvassa 31 on EPEC 6107 -näyttöyksikkö ajoneuvon ohjaamossa ja samasta kuvasta on havaittavissa helppolukuisen näkymän tärkeys.

KUVA 31. EPEC 6107 -näyttöyksikön sijainti ohjaamossa.

”Main”-välilehdellä on myös tapahtunut merkittäviä muutoksia. Akuston energiamäärän laskukaava on lisätty, ajoinverttereiden tilaviestit ja -valot, latauspistokkeen tilaa ilmaisevat valot sekä ajoluvan ilmaisevat valot ovat saaneet toiminnallisuuden tässä versiossa, kun taas aiemmassa niille oli vain visualisaatioelementit.

Tämän ohjelmistoversion koettiin täyttävän työn alussa asetetut vaatimukset. Liitteessä 4 on esitelty visualisaationäkymät näyttöyksikön ruudulla. Näkymien skaalaus täytyisi vielä sovittaa näytön rajoihin paikan päällä, mutta CAN-väylältä vaikuttaisi tulevan tietoa. Hanke itsessään jatkui edelleen, joten on mahdollista, että projektiryhmälle tulee myöhemmissä vaiheissa tarve näytön toiminnallisuuden lisäämiselle. Nykyisissä näkymissä ei ole juurikaan tilaa uusille elementeille, joten jatkossa tapahtuvat lisäykset johtaisivat todennäköisesti uuteen visualisaatiosivuun tai merkittäviin muutoksiin nykyisissä näkymissä. Täten skaalauksen korjaamista nyt ei pidetty tarpeellisena, vaan projektiryhmä suorittaa sen projektin ollessa lopullisessa muodossaan.

7 Pohdinta

CAN-väylän toiminnan tuntemisesta syntyviä hyötyjä on lähes mahdotonta yliarvioida. Se on yksi yleisimmistä ajoneuvoväylistä, jonka pohjalta on luotu monia erikoistuneempia väylätekniikoita, joiden ymmärtämiseksi on hyödyllistä omata vahva käsitys CAN-väylän toiminnasta. Ajoneuvoissa olevan elektroniikan määrä vaikuttaisi olevan jatkuvassa kasvussa, joten CAN-väylä ja siitä johdetut väylätekniikat eivät ole muuttumassa tarpeettomiksi lähitulevaisuudessa.

Tämän opinnäytetyön pääasiallinen tavoite oli luoda annettujen vaatimusten pohjalta ohjelmisto kuorma-auton ohjaamoon sijoitettavaa näyttöä varten. Vaatimusten mukainen käyttöliittymä luotiin onnistuneesti, vaikkakin sen viimeistely jäikin odottamaan hankkeen edistymistä. Käyttöliittymän mahdollista, myöhemmin tapahtuvaa laajentamista ei voitu vielä sulkea pois, joten viimeistely tässä vaiheessa koettiin tarpeettomaksi.

Toissijainen tavoite oli tutusta CAN- ja J1939-protokolliin sekä CODESYS-ympäristöön. Tämä tavoite saavutettiin ja protokollista saatiin muodostettua kattava tietopaketti. Kaikki opinnäytetyössä käytetyt teknologiat ja ohjelmistot olivat itselleni uusia, joten oppimisnäkökulmasta työ tarjosi paljon mahdollisuuksia osaamiseni kehittämiseen. Työn aikana osaamiseni kehittyi yleisesti ajoneuvotekniikan saralla ja pääsin tutustumaan CODESYS:in lisäksi myös muihin ohjelmistoihin.

Opinnäytetyö ei edennyt alkuperäisen suunnitelman mukaisesti. Korona-virus aiheutti koko hankkeelle merkittäviä ongelmia, jotka heijastuivat myös opinnäytetyön toteutuksessa. Tämä näkyi pääasiallisesti jatkuvasti siirtyvän aikataulun muodossa. Näytön käyttöliittymä on luonnollisesti projektissa melko myöhäisen vaiheen asia, joten itselleni syntyi huomattavia toimettomuuden jaksoja ja täten opinnäytetyön aikataulu viivästyi. Näistä ongelmista huolimatta uskon opinnäytetyön olleen hyvä projekti oman osaamiseni kehittämisen kannalta.

LÄHTEET

CAN in Automation. n.d. About us. Verkkosivu. Viitattu 13.8.2022. https://www.can-cia.org/about-us/

CAN in Automation. n.d. History of CAN technology. Verkkosivu. Viitattu 13.8.2022. https://www.can-cia.org/can-knowledge/can/can-history/

Alanen, J. Tammikuu 2000. CAN – ajoneuvojen ja koneiden sisäinen paikallisväylä. VTT Automaatio. Verkkodokumentti. Viitattu 13.8.2022. https://docplayer.fi/10821742-Can-ajoneuvojen-ja-koneiden-sisainen-paikallis-vayla.html

Robert Bosch GmbH. 1991. CAN Specification Version 2.0. Verkkodokumentti. Viitattu 13.8.2022. http://esd.cs.ucr.edu/webres/can20.pdf

Texas Instruments. Toukokuu 2016. Introduction to the Controller Area Network (CAN). Verkkodokumentti. Viitattu 13.8.2022. https://www.ti.com/lit/an/sloa101b/sloa101b.pdf

Ixxat. n.d. SAE J1939. Verkkosivu. Viitattu 13.8.2022. https://www.ixxat.com/technologies/fieldbuses/sae-j1939

Kvaser. n.d. The CAN Bus Protocol Tutorial. Verkkosivu. Viitattu 13.8.2022. https://www.kvaser.com/can-protocol-tutorial/

Vector. 22.10.2021. Introduction to CAN (EN) | Error Tracking. Verkkosivu. Viitattu 13.8.2022. https://elearning.vector.com/mod/page/view.php?id=359

Falch, M. Maaliskuu 2022. J1939 Explained - A Simple Intro [2022]. CSS Electronics. Verkkosivu. Viitattu 13.8.2022. https://www.csselectronics.com/pages/j1939-explained-simple-intro-tutorial

Vector. n.d.a SAE J1939 Know-How. Verkkosivu. Viitattu 13.8.2022. https://www.vector.com/at/en/know-how/protocols/sae-j1939/#

Voss, W. 30.5.2019. SAE J1939 Address Management Messages (Address Claim PGNs).. Copperhill Technologies. Verkkosivu. Viitattu 13.8.2022.https://copperhilltech.com/blog/sae-j1939-address-management-messages-address-claim-pgns/

Voss, W. 12.11.2018. Guide To SAE J1939 - Address Claiming Procedure Overview. Verkkosivu. Copperhill Technologies. Viitattu 13.8.2022. https://copperhilltech.com/blog/guide-to-sae-j1939-address-claiming-procedure-overview/

CODESYS GmbH. n.d. CODESYS Development System V3. Verkkosivu. Viitattu 13.8.2022. https://store.codesys.com/en/codesys.html

EPEC Oy. 23.3.2021. Optimize Time-To-Market with Epec MultiTool. Verkkosivu. Viitattu 5.11.2022. https://epec.fi/optimize-time-to-market-with-epec-multi-tool/

Vector Informatik GmbH. n.d.b The Design Environment for CAN / CAN FD Networks. Verkkosivu. Viitattu 5.11.2022. https://www.vector.com/at/en/products/products-a-z/software/candb/#

Kuvat:

ISO. 2016. ISO 11898-2:2016(en) Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit. Verkkosivu. Viitattu 13.8.2022. https://www.iso.org/obp/ui/#iso:std:iso:11898:-2:ed-2:v1:en

Copperhill Technologies. n.d. A Brief Introduction to Controller Area Network. Verkkosivu. Viitattu 13.8.2022. https://copperhilltech.com/a-brief-introduction-to-controller-area-network/

Alanen, J. Tammikuu 2000. CAN – ajoneuvojen ja koneiden sisäinen paikallisväylä. VTT Automaatio. Verkkodokumentti. Viitattu 13.8.2022. https://docplayer.fi/10821742-Can-ajoneuvojen-ja-koneiden-sisainen-paikallis-vayla.html

Embedclogic. n.d. CAN Physical Layer. Verkkosivu. Viitattu 13.8.2022. https://embedclogic.com/can-protocol-protocol-to-broadcast-message-on-a-network/can-physical-layer/

Texas Instruments. Toukokuu 2016. Introduction to the Controller Area Network (CAN). Verkkodokumentti. Viitattu 13.8.2022. https://www.ti.com/lit/an/sloa101b/sloa101b.pdf

Falch, M. Toukokuu 2022. CAN Bus Errors Explained - A Simple Intro [2022]. CSS Electronics. Verkkosivu. Viitattu 22.11.2022. https://www.csselectronics.com/pages/can-bus-errors-intro-tutorial

Ixxat. n.d. SAE J1939. Verkkosivu. Viitattu 13.8.2022. https://www.ixxat.com/technologies/fieldbuses/sae-j1939

Voss, W. 9.4.2020. SAE J1939 Message Format And Interpretation Of PGNs. Copperhill Technologies. Verkkosivu. Viitattu 13.8.2022. https://copperhilltech.com/blog/sae-j1939-message-format-and-interpretation-of-pgns/

Falch, M. Maaliskuu 2022. J1939 Explained - A Simple Intro [2022]. CSS Electronics. Verkkosivu. Viitattu 13.8.2022. https://www.csselectronics.com/pages/j1939-explained-simple-intro-tutorial

Voss, W. 30.5.2019. SAE J1939 Address Management Messages (Address Claim PGNs). Copperhill Technologies. Verkkosivu. Viitattu 13.8.2022.https://copperhilltech.com/blog/sae-j1939-address-management-messages-address-claim-pgns/

Cpeacock. 8.4.2020. J1939 Transport Protocol Reference Example. National Instruments. Verkkosivu. https://forums.ni.com/t5/Example-Code/J1939-Transport-Protocol-Reference-Example/ta-p/3984291

LIITTEET

Liite 1. Vaatimusluettelo

(1/3)

Paineilmakompressori	Sevconstatus_VAR_Kompressori	X	Sisääntulevalle **Sevconstatus_VAR_Kompressori**:lle suoritetaan ELSIF-vertailu ja sen perusteella muodostetaan statuksen tekstimuotoinen tila **KompressoriStatus** sekä kompressorin statusvalon tilabooli **KompressoriStatusValo.** Nämä muuttujat välitetään oikeille visualisaatioelementeille.	
Ohjaustehostin	Invertterit_VAR_Tehostin	X	Sisääntulevalle muuttujalle **Sevconstatus_VAR_Tehostin** suoritetaan ELSIF-vertailu ja sen perusteella muodostetaan statuksen tekstimuotoinen tila **OhjausTehostinStatus** sekä tehostimen statusvalon tilabooli **OhjausTehostinStatusValo**. Nämä muuttujat välitetään oikeille visualisaatioelementeille.	
Ajoinvertteri 1 & 2	ST_Run	X	Ajoinverttereiltä tuleville, muotoa **VoimalinjaX_VAR_ST_Y** oleville muuttujille suoritetaan ELSIF-vertailu, jonka avulla invertereiden status määritetään. Tästä luodaan muuttujat **Invertterix_Status** ja **Invertterix_Status_Valo**, jotka välitetään oikeille visualisaatioelementeille.	
BMS	-			Elementti lisätty. (Rolling Counter-viesti olemassa BMS:ssä, pitää lisätä ketjuun BMS-softan kautta).
ECU	-			Ei tarvita
Kiertovesipumppu	-			Ei näytölle, seurataan vesijäähdytettyjen laitteiden lämpötiloja -> lähetetään korkein lämpötila- arvo ajoneuvon
Sähköinen ajotila D || P	Drive_Permission_VAR_Drive_Permission	X	Sisääntuleva muuttujalle **Drive_Permission_VAR_Drive_Permission** suoritetaan ELSIF-vertailu, jonka avulla luodut muuttujat **DrivePermissionPos** ja **DrivePermissionNeg** saavat arvoikseen 1 tai 0. Näiden muuttujien arvot välitetään sen jälkeen oikeille visualisaatioelementeille.	
Eristevuotoilmaisin	DTC Status #2, Bit #14 (0x2000): P0AA6 (High Voltage Isolation Fault)			Elementti lisätty. DTC/CAN-viesti, WIP
Akuston balanssi	Failsafe Statuses, Bit #3 (0x08): Cell balancing active (non-failsafe mode)			Elementti lisätty. DTC/CAN-viesti, WIP

(jatkuu)

(2/3)

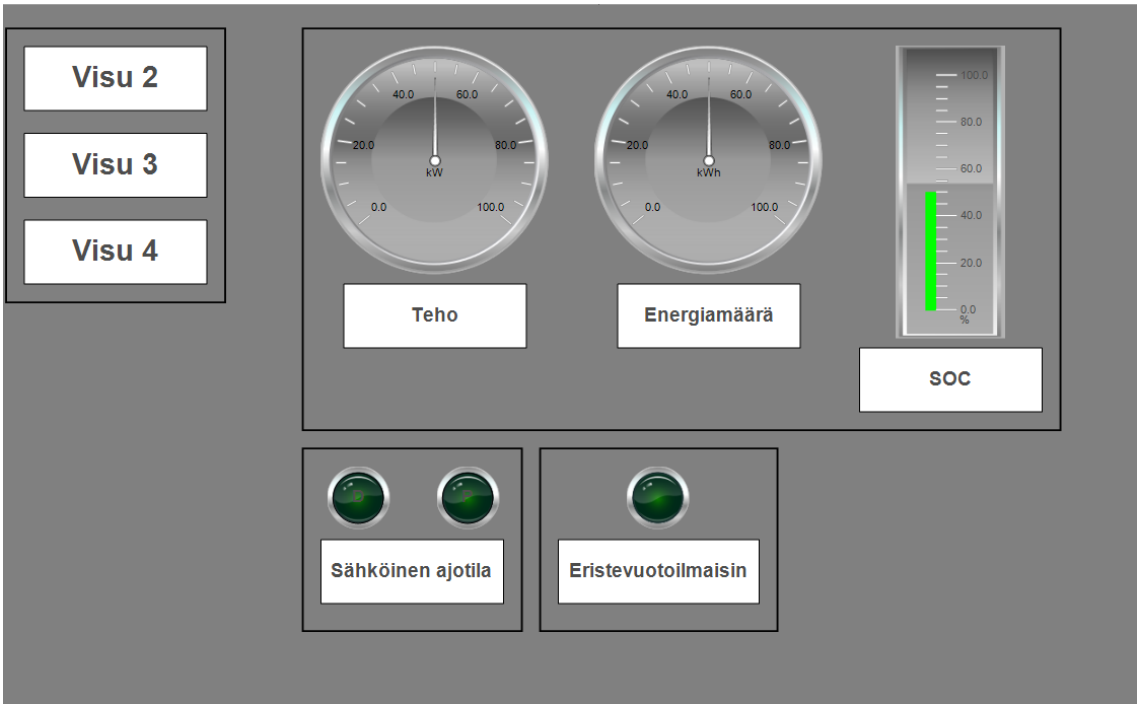
Mittarit:	Kuvaus/muutuja:	Testattu:	Toteutus:	Notes:
Teho kW	(Voimalinja1_VAR_Pack_Current * Voimalinja3_VAR_Pack_Open_Voltage)/1000	X	Toiminta sarakkeessa B olevan kaavan mukaisesti, jonka jälkeen arvo välitetään oikealle visualisaatioelementille.	
Energiamäärä kWh	(J1939_CAN1.In.Default.Voimalinja11_VAR_Pack_Amphours * J1939_CAN1.In.Default.Voimalinja3_VAR_Pack_Open_Voltage)/1000;	X	Toiminta sarakkeessa B olevan kaavan mukaisesti, jonka jälkeen arvo välitetään oikealle visualisaatioelementille.	
SoC%	Voimalinja1_VAR_SOC	X	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin **value**-kohdassa.	
Matalin kennon lämpötila	Voimalinja11_VAR_Lowest_Temperature	X	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin value-kohdassa.	
Korkein kennon lämpötila	Voimalinja11_VAR_Highest_Temperature	X	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin value-kohdassa.	

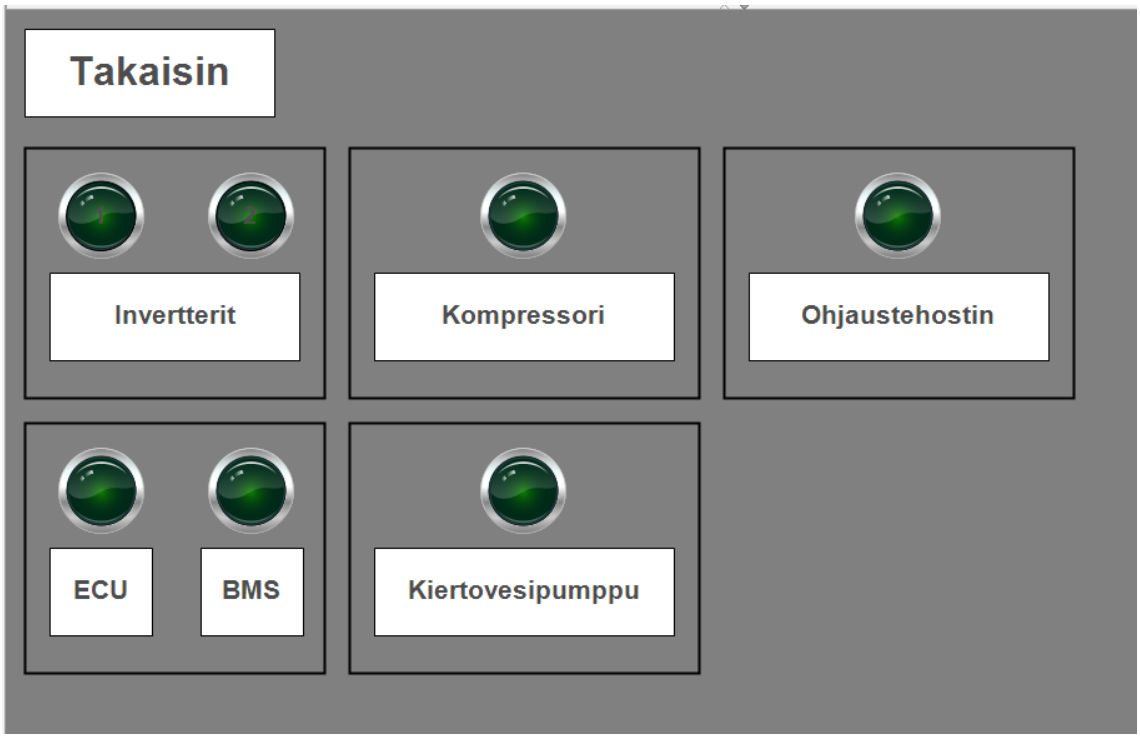
(3/3)

Ohjaustehostimen status	**VAR_Tehostin perusteella, ks. Arvot.**	**X**	Sisääntulevalle muuttujalle **Sevconstatus_VAR_Tehostin** suoritetaan ELSIF-vertailu ja sen perusteella muodostetaan statuksen tekstimuotoinen tila **OhjausTehostinStatus** sekä tehostimen statusvalon tilabooli **OhjausTehostinStatusValo**. Nämä muuttujat välitetään oikeille visualisaatioelementeille.	
Kompressorin status	**ks. Yllä, VAR_Kompressori**	**X**	Sisääntulevalle **Sevconstatus_VAR_Kompressori**:lle suoritetaan ELSIF-vertailu ja sen perusteella muodostetaan statuksen tekstimuotoinen tila **KompressoriStatus** sekä kompressorin statusvalon tilabooli **KompressoriStatusValo**. Nämä muuttujat välitetään oikeille visualisaatioelementeille.	
Ajoinvertterien status 1&2	**ST_Run, ST_Fault, ST_Warning**	**X**	Ajoinverttereiltä tuleville, muotoa **VoimalinjaX_VAR_ST_Y** oleville muuttujille suoritetaan ELSIF-vertailu, jonka avulla invertereiden status määritetään. Tästä luodaan muuttujat **Invertteri**x**_Status** ja **Invertteri**x**_Status_Valo**, jotka välitetään oikeille visualisaatioelementeille.	
Latauspistokkeen lämpötila	**Latauspistoke_VAR_LP_Temp**	**X**	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin value-kohdassa.	
Latauspistokkeen lukitus	**Latauspistoke_VAR_LP_Lukitus**	**X**	Sisääntuleva muuttujalle **Latauspistoke_VAR_LP_Lukitus** tehdään ELSIF-vertailu ja tämän tuloksena syntynyt tekstimuotoinen muuttuja **LP_Status** välitetään oikealle visualisaatioelementille.	
Korkein kennon jännite	**Voimalinja2_VAR_High_Cell_voltage**	**X**	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin value-kohdassa.	
Matalin kennon jännite	**Voimalinja2_VAR_Low_Cell_voltage**	**X**	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin value-kohdassa.	
Korkein sisäinen resistanssi	**Voimalinja3_VAR_High_Cell_Resistance**	**X**	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin value-kohdassa.	
Matalin sisäinen resistanssi	**Voimalinja3_VAR_Low_Cell_Resistance**	**X**	Sisääntulevan muuttujan arvo luetaan suoraan oikealle visualisaatioelementille elementin value-kohdassa.	

Liite 2. Version 1 visualisaationäkymät ja ”Main”-välilehden koodi.

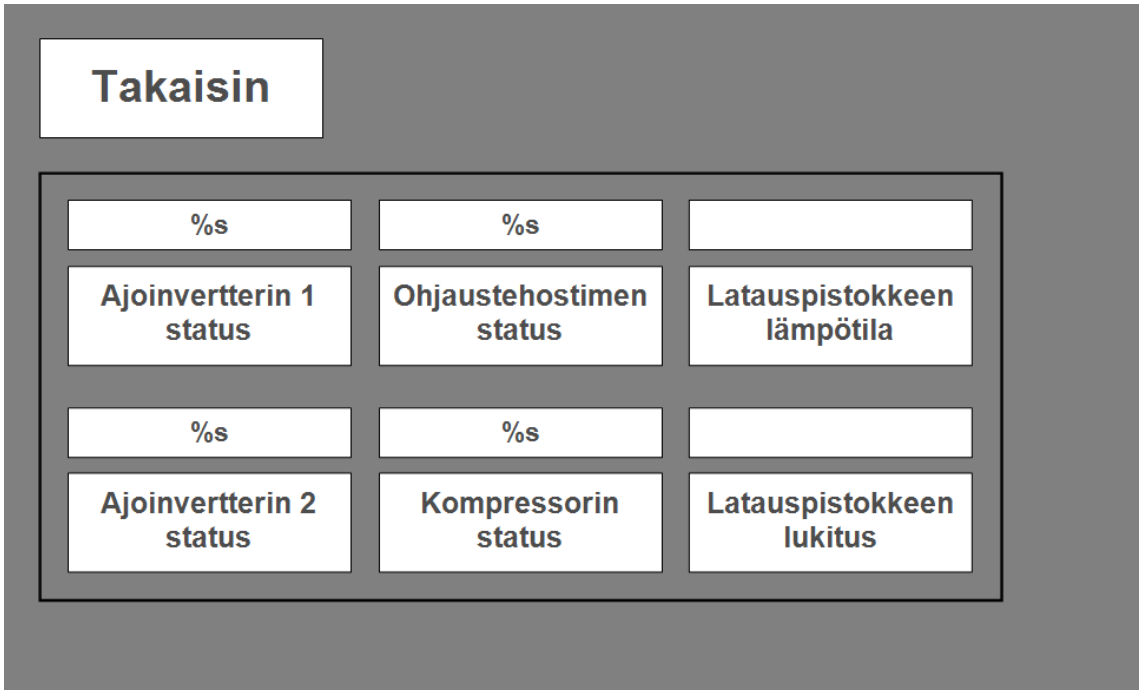
(1/3)

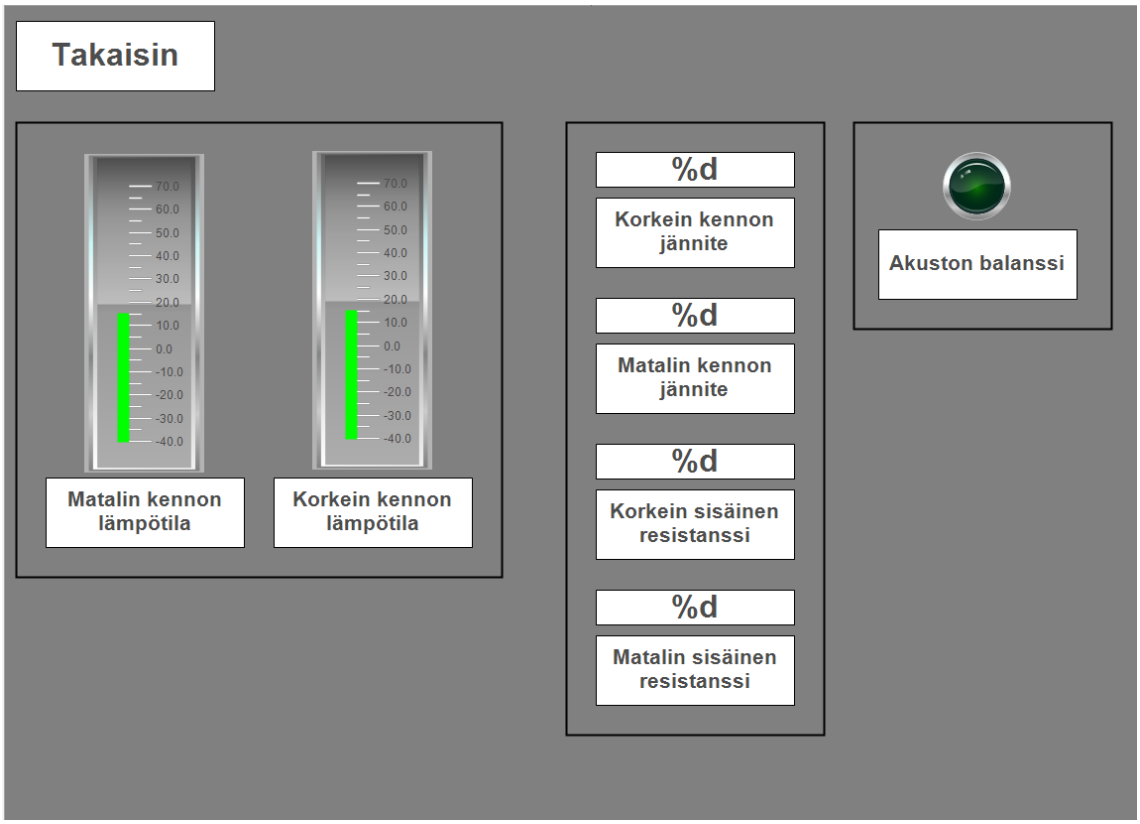




(jatkuu)

(2/3)


Takaisin
%s
%s
Ajoinvertterin 1 status
Ohjaustehostimen status
Latauspistokkeen lämpötila
%s
%s
Ajoinvertterin 2 status
Kompressorin status
Latauspistokkeen lukitus


Takaisin
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
-40.0
Matalin kennon lämpötila
Korkein kennon lämpötila
%d
Korkein kennon jännite
%d
Matalin kennon jännite
%d
Korkein sisäinen resistanssi
%d
Matalin sisäinen resistanssi
Akuston balanssi

(3/3)

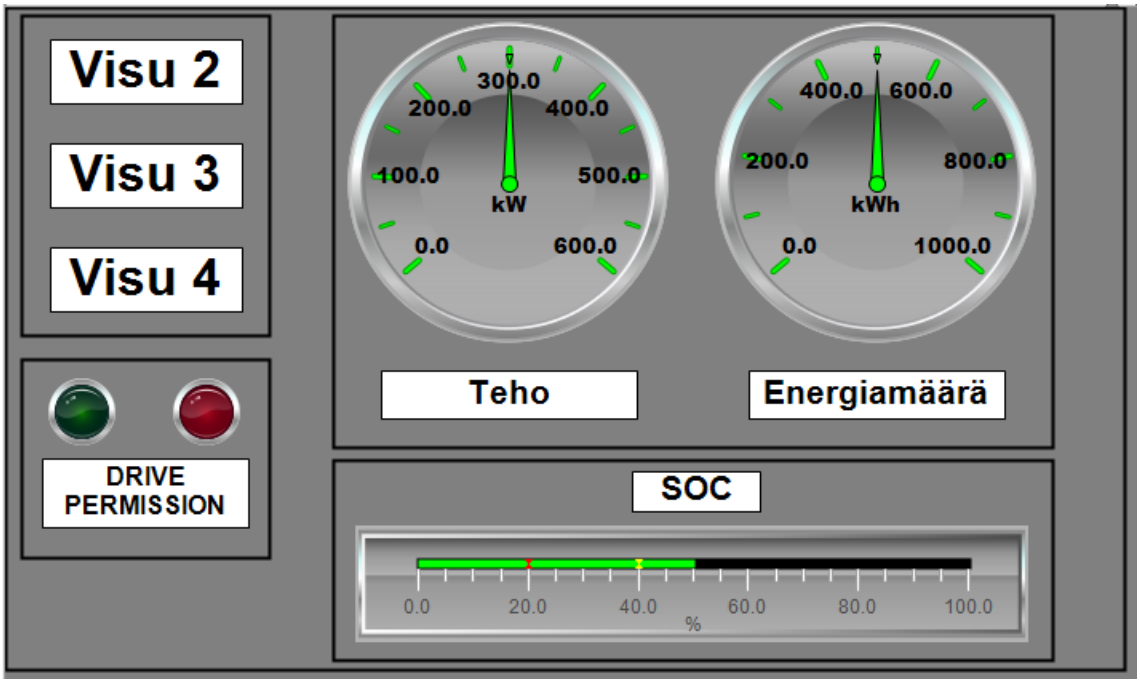
```
PROGRAM Main
VAR
    Batt_Power_KW : WORD;
    OhjausTehostinStatus : STRING;
    KompressoriStatus : STRING;

    OhjausTehostinStatusValo : BOOL := 0;
    KompressoriStatusValo : BOOL := 0;

END_VAR
```

135 %

```
        /// User code should be added to POU named as Main.

        Batt_Power_KW := (J1939_CAN1.In.Default.Voimalinja1_VAR_Pack_Current * J1939_CAN1.In.Default.Voimalinja3_VAR_Pack_Open_Voltage)/1000;


        IF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 1
          THEN OhjausTehostinStatus := 'Not ready to turn on';
        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 2
           THEN OhjausTehostinStatus := 'Shutdown';
        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 4
           THEN OhjausTehostinStatus := 'Precharge';
        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 7
           THEN OhjausTehostinStatus := 'Energised';
        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 8
           THEN OhjausTehostinStatus := 'Enabled';
        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 11
           THEN OhjausTehostinStatus := 'Fault reaction active';
        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 13
           THEN OhjausTehostinStatus := 'Fault off';
        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 14
           THEN OhjausTehostinStatus := 'Active short';
        ELSE OhjausTehostinStatus := 'N/A';
        END_IF


                        IF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 1
                          THEN KompressoriStatus := 'Not ready to turn on';
                        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 2
                           THEN KompressoriStatus := 'Shutdown';
                        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 4
                           THEN KompressoriStatus := 'Precharge';
                        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 7
                           THEN KompressoriStatus := 'Energised';
                        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 8
                           THEN KompressoriStatus := 'Enabled';
                        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 11
                           THEN KompressoriStatus := 'Fault reaction active';
                        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 13
                           THEN KompressoriStatus := 'Fault off';
                        ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 14
                           THEN KompressoriStatus := 'Active short';
                        ELSE KompressoriStatus := 'N/A';
                        END_IF

                        IF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 8
                           THEN OhjausTehostinStatusValo := TRUE;
                        ELSE OhjausTehostinStatusValo := FALSE;
                        END_IF

                        IF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 8
                           THEN KompressoriStatusValo := TRUE;
                        ELSE KompressoriStatusValo := FALSE;
                        END_IF
```

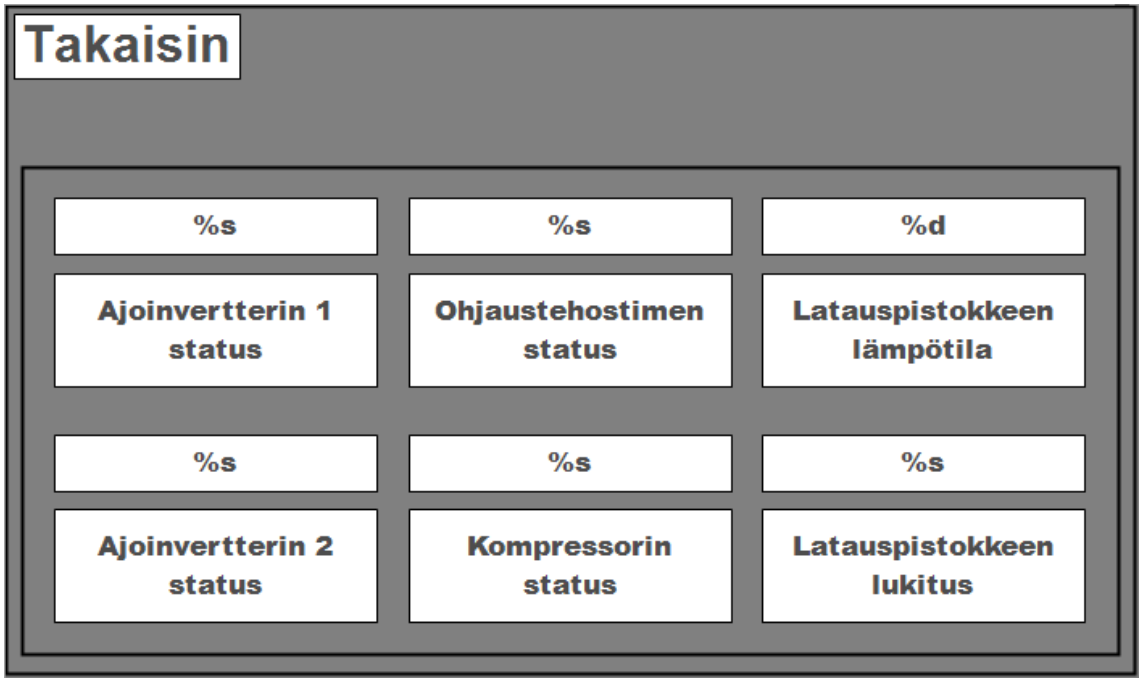
Liite 3. Version 2 visualisaationäkymät ja ”Main”-välilehden koodi.

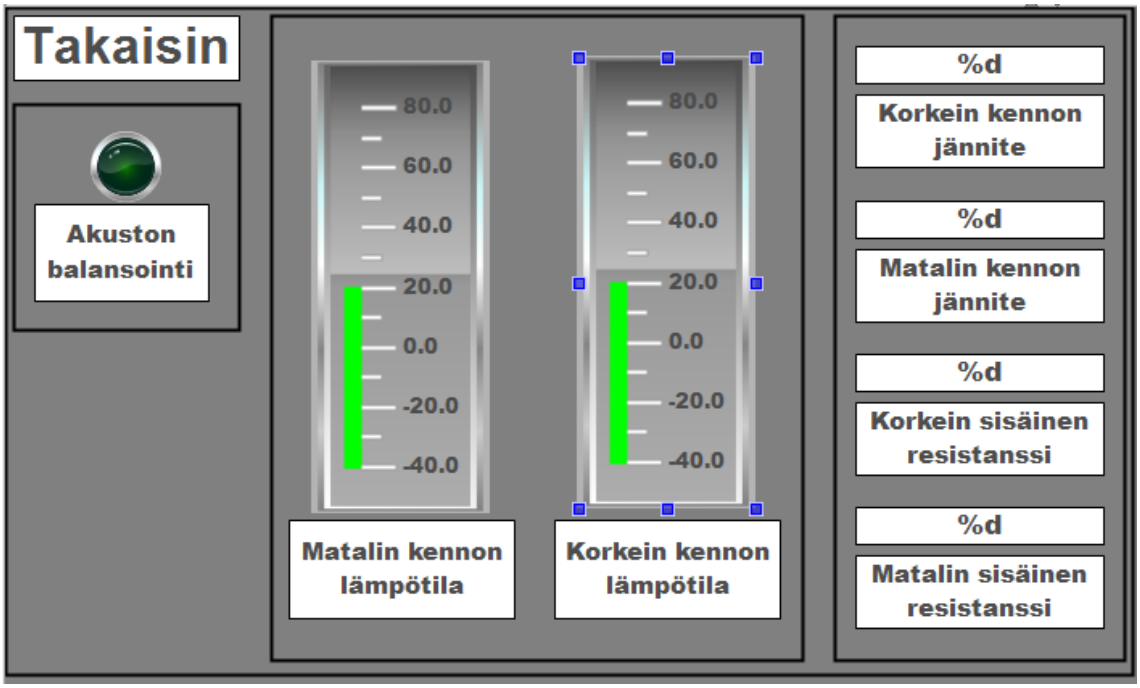
(1/6)



(jatkuu)

(2/6)


Takaisin
%s
%s
%d
Ajoinvertterin 1 status
Ohjaustehostimen status
Latauspistokkeen lämpötila
%s
%s
%s
Ajoinvertterin 2 status
Kompressorin status
Latauspistokkeen lukitus


Takaisin
Akuston balansointi
80.0
60.0
40.0
20.0
0.0
-20.0
-40.0
Matalin kennon lämpötila
80.0
60.0
40.0
20.0
0.0
-20.0
-40.0
Korkein kennon lämpötila
%d
Korkein kennon jännite
%d
Matalin kennon jännite
%d
Korkein sisäinen resistanssi
%d
Matalin sisäinen resistanssi

(3/6)

```
1   PROGRAM Main
2   VAR
3       Batt_Power_KW : WORD;
4       Batt_KWH : WORD;
5       LP_Status : STRING;
6       OhjausTehostinStatus : STRING;
7       KompressoriStatus : STRING;
8       DrivePermissionPos : BOOL;
9       DrivePermissionNeg : BOOL;
10
11      Invertteri1_Status : STRING;
12      Invertteri1_Status_Valo : BOOL;
13
14
15      Invertteri2_Status : STRING;
16      Invertteri2_Status_Valo : BOOL;
17
18      OhjausTehostinStatusValo : BOOL := 0;
19      KompressoriStatusValo : BOOL := 0;
20
21
22  END_VAR
```

```
1
2                   /// User code should be added to POU named as Main.
3
4
5               //Visu1 mittareiden arvojen laskenta
6               Batt_Power_KW := (J1939_CAN1.In.Default.Voimalinja1_VAR_Pack_Current * J1939_CAN1.In.Default.Voimalinja3_VAR_Pack_Open_Voltage)/1000;
7               Batt_KWH := (J1939_CAN1.In.Default.Voimalinja11_VAR_Pack_Amphours * J1939_CAN1.In.Default.Voimalinja3_VAR_Pack_Open_Voltage)/1000;
8               //END
9
10
11              //Latauspistokkeen lukituksen statuksen käsittely
12              IF J1939_CAN1.In.Default.Latauspistoke_VAR_LP_Lukitus = 1
13                  THEN LP_Status := 'Active';
14              ELSE
15                  LP_Status := 'Inactive';
16              END_IF
17              //END
18
19
```

(5/6)

```
20  //Ajoluvan valojen käsittely
21  IF J1939_CAN1.In.Default.Drive_Permission_VAR_Drive_Permission = 1
22      THEN DrivePermissionNeg := 0;
23      DrivePermissionPos := 1;
24  ELSE
25      DrivePermissionNeg := 1;
26      DrivePermissionPos := 0;
27  END_IF
28  //END
29
30    //Kompressorin ja ohjaustehostimen invertterein statusten käsittelyt valoineen
31    IF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 1
32      THEN OhjausTehostinStatus := 'Not ready to turn on';
33    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 2
34       THEN OhjausTehostinStatus := 'Shutdown';
35    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 4
36       THEN OhjausTehostinStatus := 'Precharge';
37    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 7
38       THEN OhjausTehostinStatus := 'Energised';
39    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 8
40       THEN OhjausTehostinStatus := 'Enabled';
41    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 11
42       THEN OhjausTehostinStatus := 'Fault reaction active';
43    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 13
44       THEN OhjausTehostinStatus := 'Fault off';
45    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 14
46       THEN OhjausTehostinStatus := 'Active short';
47    ELSE OhjausTehostinStatus := 'N/A';
48    END_IF
49
50
51    IF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 1
52      THEN KompressoriStatus := 'Not ready to turn on';
53    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 2
54       THEN KompressoriStatus := 'Shutdown';
55    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 4
56       THEN KompressoriStatus := 'Precharge';
57    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 7
58       THEN KompressoriStatus := 'Energised';
59    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 8
60       THEN KompressoriStatus := 'Enabled';
61    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 11
62       THEN KompressoriStatus := 'Fault reaction active';
63    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 13
64       THEN KompressoriStatus := 'Fault off';
65    ELSIF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 14
66       THEN KompressoriStatus := 'Active short';
67    ELSE KompressoriStatus := 'N/A';
68    END_IF
69
```

(6/6)

```
70  IF J1939_CAN1.In.Default.Sevconstatus_VAR_Tehostin = 8
71     THEN OhjausTehostinStatusValo := TRUE;
72  ELSE OhjausTehostinStatusValo := FALSE;
73  END_IF
74
75  IF J1939_CAN1.In.Default.Sevconstatus_VAR_Kompressori = 8
76     THEN KompressoriStatusValo := TRUE;
77  ELSE KompressoriStatusValo := FALSE;
78  END_IF
79  //END
80
81
82  //Ajoinvertteri 1:n statuksen asettaminen valoineen
83  IF J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Run = 1 AND J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Warning = 0 AND J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Fault = 0
84      THEN Invertteril_Status := 'Run';
85           Invertteril_Status_Valo := 1;
86  ELSIF J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Run = 0 AND J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Warning = 1 AND J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Fault = 0
87      THEN Invertteril_Status := 'Warning';
88           Invertteril_Status_Valo := 0;
89  ELSIF J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Run = 0 AND J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Warning = 0 AND J1939_CAN1.In.Default.Voimalinja4_VAR_ST_Fault = 1
90      THEN Invertteril_Status := 'Fault';
91           Invertteril_Status_Valo := 0;
92  ELSE Invertteril_Status := 'N/A';
93  END_IF
94  //END
95
96  //Ajoinvertteri 2:n statuksen asettaminen valoineen
97  IF J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Run = 1 AND J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Warning = 0 AND J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Fault = 0
98      THEN Invertteri2_Status := 'Run';
99           Invertteri2_Status_Valo := 1;
100 ELSIF J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Run = 0 AND J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Warning = 1 AND J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Fault = 0
101     THEN Invertteri2_Status := 'Warning';
102          Invertteri2_Status_Valo := 0;
103 ELSIF J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Run = 0 AND J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Warning = 0 AND J1939_CAN1.In.Default.Voimalinja2_4_VAR_ST_Fault = 1
104     THEN Invertteri2_Status := 'Fault';
105          Invertteri2_Status_Valo := 0;
106 ELSE Invertteri2_Status := 'N/A';
107 END_IF
108 //END
109
110
111
```

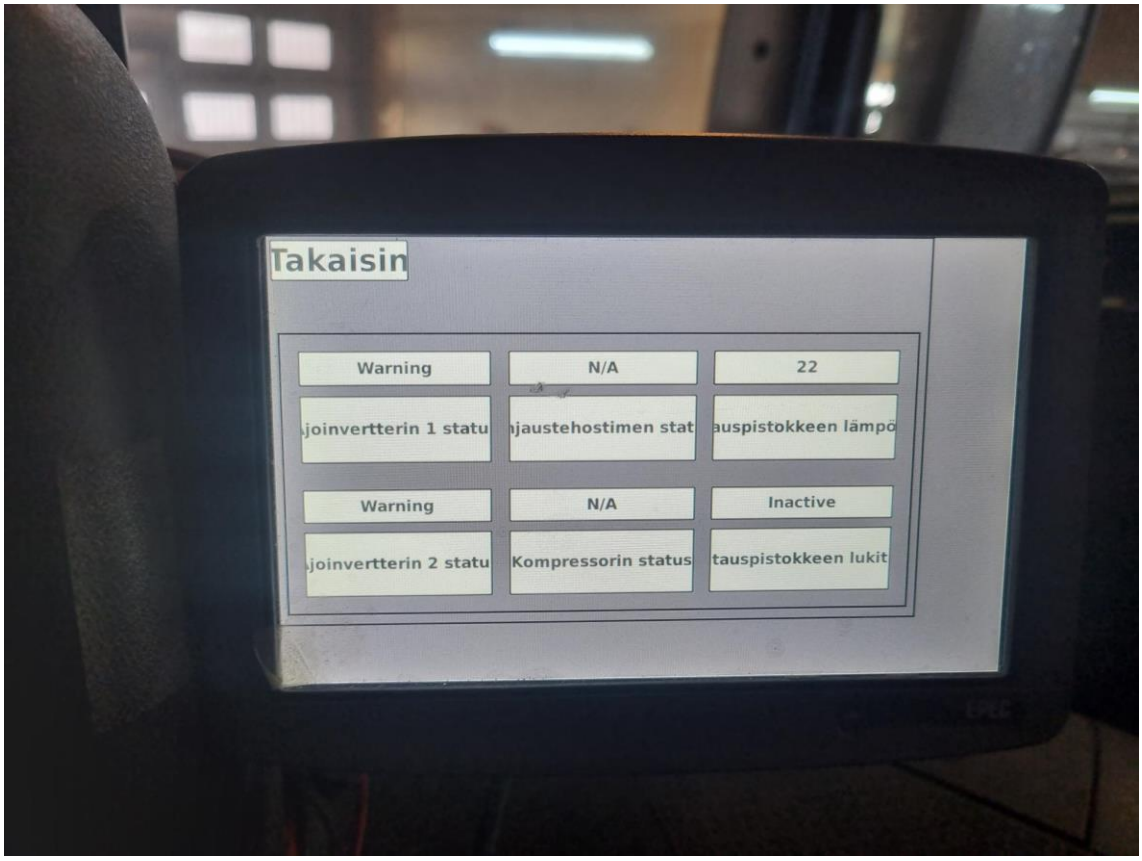
Liite 4. Visualisaationäkymät ruudulla

(1/2)

Takaisin
Invertterit
Kompressori
Ohjaustehostin
BMS
Eristevuotoilmaisin

(jatkuu)

(2/2)


Takaisin
Warning
N/A
22
joinvertterin 1 statu
jaustehostimen stat
auspistokkeen lämpö
Warning
N/A
Inactive
joinvertterin 2 statu
Kompressorin status
tauspistokkeen lukit

Takaisin
ston balansc
12
12
50.0
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
-5.0
-10.0
-15.0
-20.0
-25.0
-30.0
lin kennon lämp
ein kennon lämp
3.53
rkein kennon jänn
3.22
talin kennon jänn
20
ein sisäinen resist
20
lin sisäinen resist