Tommi Tukiainen
Jesse Pirhonen

CAN-väyläsimulaattori

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Auto- ja kuljetustekniikka

Insinöörityö

15.9.2012

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Tommi Tukiainen Jesse Pirhonen CAN-väyläsimulaattori 33 sivua + 4 liitettä 15.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Auto- ja kuljetustekniikka
Suuntautumisvaihtoehto	Autosähkötekniikka
Ohjaaja(t)	Tekninen kouluttaja Olli Ringvall Lehtori Pasi Kovanen

Tämän insinöörityön aiheena oli rakentaa CAN-väyläsimulaattori Robert Bosch Oy:n koulutuskäyttöön. Työn tilaajan projektille asettamat vaatimukset olivat laitteen realistisuus, yksinkertaisuus sekä helppo mukanakuljetettavuus.

CAN-väyläsimulaattoria on tarkoitus käyttää vikatilojen simulointiin ja siten ajoneuvomekaanikkojen täydennyskoulutukseen. Lisäksi laadittiin koulutuspaketti, johon sisältyy teoriaosuus ja CAN-väyläsimulaattorilla tehtävät harjoitustyöt.

Laite rakennettiin käyttäen kahta Arduino UNO -kehitysalustaa, joille suunniteltiin CAN-lähetin-vastaanotinpiirikortit. Ohjelmakoodit kirjoitettiin käyttäen Arduino IDE -ohjelmistoa, joka on ilmainen Arduino-kehitysalustojen ohjelmointiin suunniteltu ohjelmointiympäristö. Projektin toteuttamiseen käytettiin myös toista ilmaisohjelmaa, EAGLE PCB, jolla suunniteltiin piirikortit.

Projektin lopputuotteena syntyi toimiva CAN-väyläsimulaattori sekä sen tueksi laadittu koulutuspaketti. Työn tilaajan vaatimukset täytettiin ja laite on sijoitettu salkkuun, jota on helppo kantaa mukana. Vikatilat simuloidaan käyttäen laitteeseen asennettua näyttöä, jossa on ohjelmallisesti toteutettu valikko. Laitteita on tarkoitus toimittaa työn tilaajalle neljä kappaletta.

Avainsanat	CAN-väylä, Arduino

Author(s) Title Number of Pages Date	Tommi Tukiainen Jesse Pirhonen CAN-bus Simulator 33 pages + 4 appendices 15 May 2016
Degree	Bachelor of Engineering
Degree Programme	Automotive and Transport Engineering
Specialisation option	Automotive Electronics Engineering
Instructor(s)	Pasi Kovanen, Lecturer (B.Eng.) Olli Ringvall, Technical Support/Trainer

The final project was commissioned by Robert Bosch GmbH and the aim was to design and build a fault simulator for a vehicle can bus and produce training material for CAN bus fault diagnostics. The commissioning company set demands for the features of the fault simulator, features such as realistic behavior, easy usage and easy transportation.

The work started by getting acquainted with the theory of CAN bus, Arduino development boards and research considering CAN bus transceivers. The work was carried out by using two Arduino UNO development boards, which were programmed using the Arduino IDE development environment. The work was finalized with a self-made CAN bus module circuits, a carrying case and an LCD-display.

The result of the work was a working, realistic and easy to transport CAN bus fault simulator and training material, which can be used as tools for teaching vehicle mechanics.

In the near future it is possible that the commissioning company will order several finalized fault simulators. Helsinki Metropolia University of Applied Sciences has also shown interest in the product.

Keywords	CAN bus, Arduino

Sisällys

Lyhenteet

CAN	Controller Area Network. CAN-väylä on ajoneuvoissa ohjainlaitteiden ja antureiden väliseen kommunikointiin käytetty väylä.
ISO	International Organization Standardization. Kansainvälinen standardisoimisjärjestö, jonka tehtävänä on luoda yhdenmukaisia standardeja.
HS-CAN	High Speed CAN. CAN-väylä, jonka nopeus on väliltä 225–1000 kbit/s
LS-CAN	Low Speed CAN. CAN-väylä, jonka nopeus on maksimissaan 125 kbit/s
FT-CAN	Fault Tolerant CAN. Hidas ja vikasietoinen CAN-väylä, joka kestää katkokset ja oikosulut.
TT-CAN	Time Triggered CAN. Nopea CAN-väylä, jonka kommunikaatio tapahtuu aikaikkunoihin perustuvalla järjestelmällä.
MOST	Media Oriented System Transport. Nopea, erityisesti median siirtoon tarkoitettu ajoneuvoväylä, jossa kommunikointi tapahtuu valokuitua tai kuparikaapelia käyttäen.
LIN	Local Interconnect Network. Master-slave-periaatteeseen perustuva hidas ajoneuvoväylä.
PWM	Pulse Width Modulation. Pulssinleveysmodulaatio, pulssisuhdetta muuttamalla aikaansaatu kuormaan menevän jännitteen keskiarvo.
CAN H	CAN-väylän johdin, jossa jännitetaso nousee kun kirjoitetaan dominantti tila.
CAN L	CAN-väylän johdin, jossa jännitetaso laskee, kun kirjoitetaan dominanttitila.
I/0	Input/Output. Digitaaliset sisään ja ulostulot, joita käytettään datan lähetykseen, vastaanottamiseen ja toimilaitteiden ohjaukseen.

SSRAM	Static Random Access Memory. Digitaalinen muisti, joka on toteutettu puolijohdetekniikalla.
EEPROM	Electronically Erasable Programmable Read-Only Memory. Uudelleen kirjoitettava digitaalinen muisti.
LCD	Liquid Crystal Display. Nestekidenäyttö.
I2C	Yksinkertainen sarjamuotoinen tiedonsiirtoväylä.

1 Johdanto

Nykyajoneuvossa voi ohjainlaitteita olla jo yli sata kappaletta ja jokaisen niistä on kyettävä keskustelemaan ja vaihtamaan tietoa keskenään. Ohjainlaite on hyvin laaja-alainen käsite, jolla tarkoitetaan kaikkia ajoneuvon säätöihin osallistuvia laitteita, kuten esimerkiksi antureita ja toimilaitteita. Ajoneuvoissa käytetään useita eri väyläratkaisuja, mutta edelleen runkoverkkona toimii CAN-väylä. Muita verkkoja ovat mm. LIN, FlexRay ja MOST, minkä lisäksi myös Ethernet on yleistymässä ajoneuvokäytössä. Tästä syystä perusmekaanikonkin on hyvä tuntea edes pääsääntöisesti eri ajoneuvokäytössä olevat väylät ja niiden vianetsintä sekä vianetsintään käytettävät välineet, kuten oskilloskooppi ja yleismittari.

Projektin tarkoituksena oli suunnitella CAN-väyläsimulaattori, jota Robert Bosch Oy tulee käyttämään ajoneuvomekaanikkojen täydennyskoulutuksessa pohjoismaissa. Väyläsimulaattoria varten tuotettiin myös koulutusmateriaali. Työn tilaajan vaatimuksena oli suunnitella ja tuottaa laite, joka on mahdollisimman realistinen, helppokäyttöinen, selkeä sekä helposti mukana kuljetettava. Projektin toteuttamiseen valittiin kahden henkilön tiimi, koska vastaavissa projekteissa työ on jäänyt osaksi kesken aikataulun ja työmäärän takia.

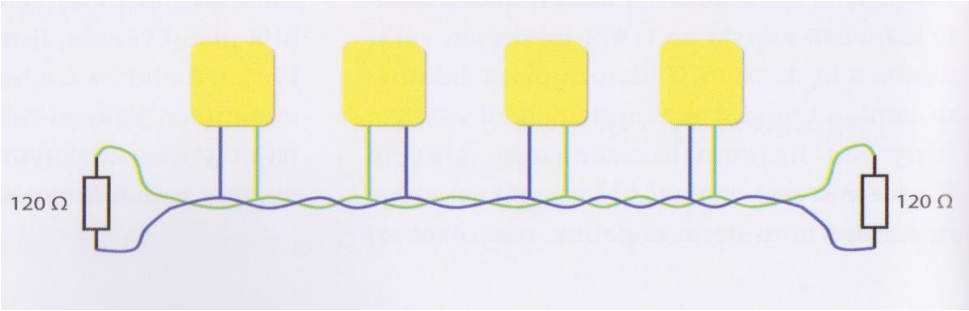
2 CAN-väylä

2.1 Perusteet

Robert Bosch GmbH aloitti CAN-väylän kehittämisen vuonna 1983 ja teollisuuskäyttöön se otettiin ensimmäistä kertaa vuonna 1989. Ensimmäistä kertaa CAN-väylä nähtiin sarjatuotanto ajoneuvossa vuonna 1991 Mercedes Benz S-sarjassa, jossa viisi ohjainlaitetta keskusteli nopeudella 500 kbit/s. (Paret 2007: 16.)

CAN-väylä koostuu pääsääntöisesti kahdesta kierretystä kuparikaapelista, CAN H- ja CAN L -kaapeleista. Väyläjohtimien molemmissa päissä on päätevastukset, poikkeuksena vikasietoinen FT-CAN-väylä. Päätevastusten väliin jäävään kierrettyyn kaapeliin eri ohjainlaitteet voidaan kytkeä käytännössä vapaasti, kunhan mini- ja maksimietäisyyksiä ei rikota (kuva 1). Tällainen ratkaisu tekee järjestelmästä modulaarisen, eli toi-

sin sanoen se mahdollistaa helpon diagnoosin ohjainlaitteiden välillä, ohjainlaitteiden helpon lisäämisen järjestelmään, sekä anturidatan helpon jakamisen eri ohjainlaitteiden välillä (Alanen 2000: 2).



Kuva 1. CAN High Speed -väylän topologia (Frei 2015: 58).

Kommunikointi CAN-väylässä tapahtuu väyläjohtimien jännitetasoa muuttamalla. Yksinkertaistettuna lähetettäessä viestiä jännitetasot CAN H -johtimessa nousevat ja CAN L -johtimessa laskevat loogisen 0:n merkiksi. Viestiä luettaessa vastaanottavat ohjainlaitteet pääsääntöisesti tulkitsevat CAN H- ja CAN L -johtimen välistä jännite-eroa.

Nykyhenkilöajoneuvosta löytyy CAN High- ja Low Speed -väylät, jotka on vielä jaoteltu siten, että nopea eli HS-CAN huolehtii voimansiirron tiedonsiirrosta ja hidas eli LS-CAN mukavuus- ja korielektroniikan tiedonsiirrosta. CAN-väylän tiedonsiirtonopeudet vaihtelevat siis ajoneuvokäytössä käyttötarkoituksen mukaan. Alla olevassa taulukossa on eri väyläratkaisut esitettynä niiden nopeuden ja käyttötarkoituksen mukaan.

Taulukko 1. Ajoneuvoväylien tiedonsiirtonopeudet ja käyttötarkoitukset lähdettä (Frei 2015: 9) mukaillen.

Väyläratkaisu	Tiedonsiirtonopeus	Käyttötarkoitus
Lin	< 25 kBit/s	Paikallisverkko, tunnistimet ja toimilaitteet
CAN (Low Speed)	25…125 kBit/s	Koriverkko
CAN (High Speed)	125…1000 kBit/s	Voimansiirto
TTCAN	1 MBit/s	Drive By Wire
Flexray	> 1 MBit/s	Multimedialaitteet
MOST	> 10 MBit/s	Multimedialaitteet

2.2 ISO-standardit

Alun perin CAN-väylän ominaisuudet määriteltiin Bosch:n julkaisemissa CAN-spesifikaatioissa. ISO-standardit korvasivat aiemmat dokumentit, määrittäen CAN-väylien ominaisuudet yhä tarkemmin. ISO-standardit laadittiin, jotta CAN-väylät voitiin tarkoin määriteltyinä ottaa käyttöön maailmanlaajuisesti eri valmistajien toimesta. (Paret 2007: 19.)

CAN Low Speed -väylän määrittelivät ensimmäisenä standardit ISO 11519-1 ja ISO 11519-2, joissa määriteltiin hidas sarjamuotoinen kommunikaatio tiekäyttöön tarkoitetuissa ajoneuvoissa. Nämä kaksi standardia on sittemmin korvattu standardilla ISO 11898-3, joka määrittää CAN Fault Tolerant -väylän fyysisen kerroksen ja kommunikaatio protokollan. (Paret 2007: 20.)

Nykyään ISO-standardit 11898-1…6 määrittävät henkilöajoneuvon eri CAN-väylät (Griffith 2013).

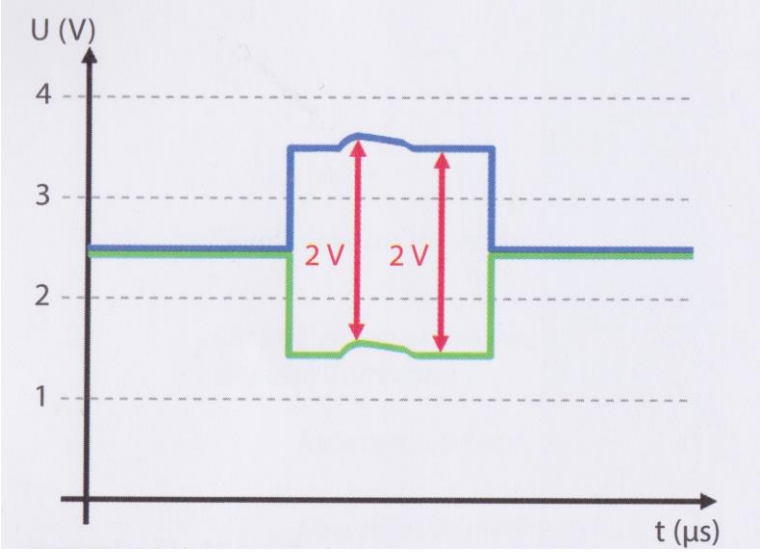
- ISO 11898-1 CAN-väylän kommunikaation
- ISO 11898-2 CAN High Speed -väylän fyysisen toteutuksen
- ISO 11898-3 CAN Low Speed- ja Fault Tolerant -väylän fyysisen toteutuksen
- ISO 11898-4 Time Triggered-CAN-väylän kommunikaation
- ISO 11898-5 CAN High Speed -väylän vaihteistoa koskevan fyysisen kerroksen, sekä lepotilaa koskevan laitteiston virransäästön
- ISO 11898-6 CAN High Speed -väylän ohjelmallisen herätyksen

2.3 CAN High Speed

Nopean CAN-väylän tiedonsiirtonopeus voi olla jopa 1 Mbit/s, mutta yleisesti käytetään nopeutta 500 kbit/s. Voimansiirron tiedonvälitykseen käytetään CAN High Speed -väylää, sillä voimansiirron ohjainyksiköiden on pystyttävä lähettämään ja vastaanottamaan dataa nopeasti. Nopea tiedonsiirto mahdollistaa lähes viiveettömän kommunikoinnin antureiden ja ohjainyksiköiden välillä, jolloin saadaan esimerkiksi ajotuntuma paremmaksi eikä ajoneuvon kuljettaja huomaa viivettä. Tämän lisäksi esimerkiksi ajonhallintajärjestelmän on kyettävä reagoimaan sekunnin sadasosissa pidon menettämisestä, jotta ajoneuvo pysyisi ajokaistalla.

2.3.1 Fyysinen toteutus

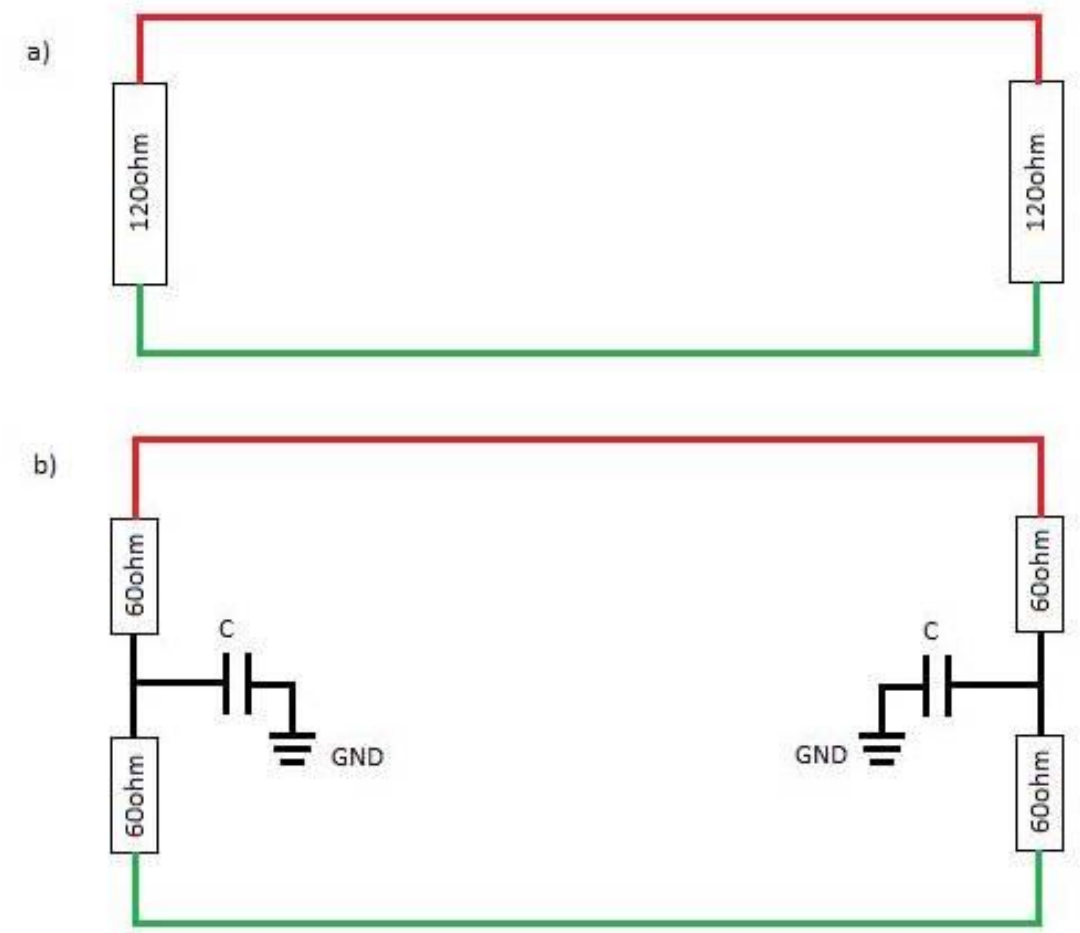
CAN High Speed -väylän fyysinen toteutus tehdään kahdella kierretyllä kuparikaapelilla, joita kutsutaan nimillä CAN-H- ja CAN-L-johdin. Johtimien päissä sijaitsevat päätevastukset eli terminointivastukset. Kierrettyä parikaapelia käytetään siitä syystä, että mahdollinen ulkopuolinen sähkömagneettinen häiriö aiheuttaa molempiin väyläjohtimiin yhtä suuren häiriöjännitteen (kuva 2). Näin ollen johtimien välinen jännite-ero pysyy häiriöstä huolimatta samana eikä vaikuta väylän toimintaan. (Frei 2015: 26–28.)



Kuva 2. Jännite-eron muutos ulkopuolisen häiriön vaikutuksesta (Frei 2015: 28).

Terminointivastusten tarkoituksena on estää signaalin heijastumista suurilla tiedonsiirtonopeuksilla. Terminointivastusten toteutuksessa käytetään pääsääntöisesti kahta

erilaista toteutustapaa. Ensimmäisessä tavassa terminointivastukset päättävät väylän 120 ohmin vastuksen avulla (kuva 3a). Toisessa tavassa 120 ohmin vastus on jaettu kahdeksi noin 60 ohmin vastuksiksi ja näiden vastusten väliin on kytketty häiriönpoisto-kondensaattori. Eri lähteissä jaetun terminointivastuksen tapauksessa, vastusten arvoina käytetään 60:tä tai 62:ta ohmia (kuva 3b).

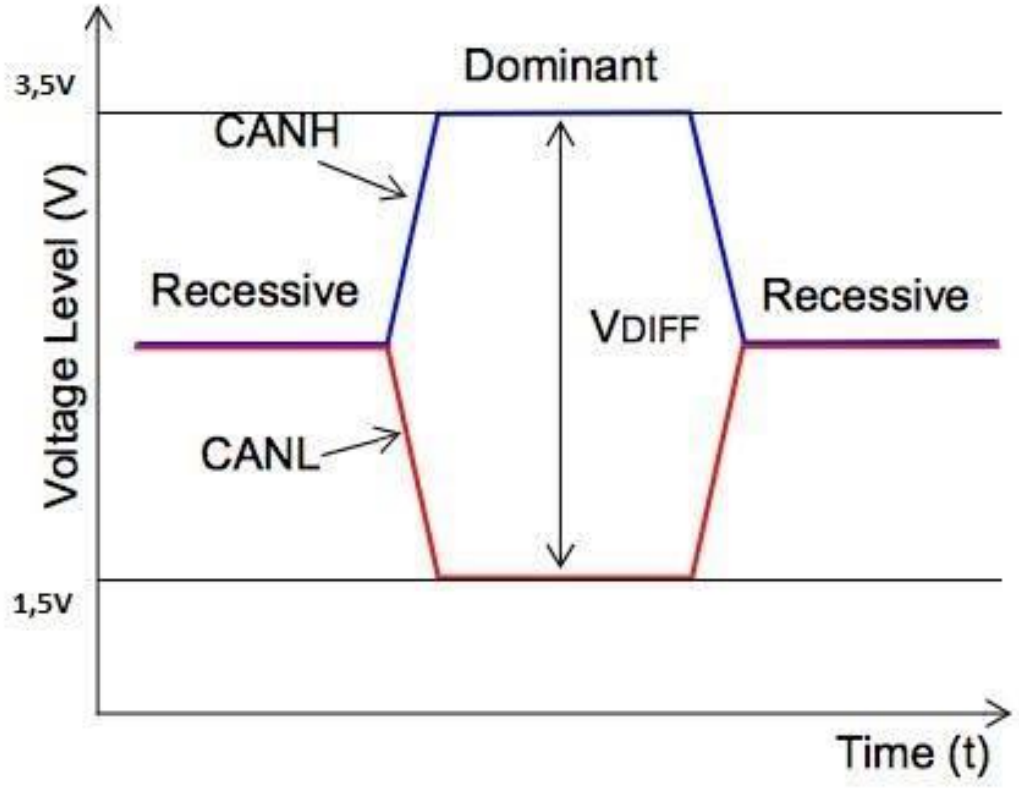


Kuva 3. High Speed CAN -väylän terminointivastusten kaksi toteutustapaa (Malmari 2016: 6).

2.3.2 Jännitetasot

CAN High Speed -väylän lähetin-vastaanotinpiiri muuntaa CAN-ohjaimelta saamansa loogiset tilat 0 ja 1 jännitetasoiksi, jotka syötetään edelleen CAN H- ja CAN L -johtimille. Jännitetasot vaihtuvat sen mukaan, onko kyseessä dominantti eli 0-tila vai resessiivinen eli 1-tila. Resessiivisessä tilassa molempien väyläjohtimien jännitetasot ovat 2,5 volttia. Dominanttitilassa CAN H -johtimen jännite nousee 2,5 voltista 3,5 volt-

tiin ja vastaavasti CAN L -johtimen jännite laskee 2,5 voltista 1,5 volttiin. Jännite-erot tilasta riippuen ovat siis joko 0 volttia tai 1 voltti. (Mischo ym. 2008: 33.)



Kuva 4. CAN High Speed -väylän jännitetasot sekä tilat (Richards 2002: 2).

2.4 CAN Low Speed

Korin- ja mukavuuselektroniikan tiedonsiirtoon käytetään hidasta CAN-väylää. Hitaaseen väylään ei sijoiteta järjestelmiä, joiden on reagoitava nopeisiin transientteihin. Tällaisia ovat esim. turvallisuuskriittiset järjestelmät, kuten ajonvakauden hallintajärjestelmät. Tästä syystä tiedonsiirtonopeudeksi on standardissa määritetty korkeintaan 125 kbit/s. Tiedonsiirtonopeuden pudottamisella voidaan saada aikaiseksi parempi elektromagneettinen yhteensopivuus, joka voi puolestaan helpottaa ajoneuvon EMC-standardien läpäisyä. Toteutettaessa hitaan väylän fyysistä kerrosta voidaan teoriassa käyttää mitä tahansa fyysistä kerrosta, kunhan se täyttää tuen dominantti- ja resessiivitiloille. (Paret 2007: 142.)

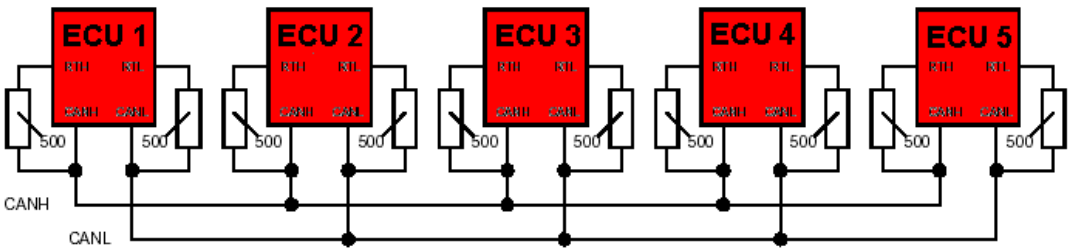
Vikasietoisen väylän tiedonsiirto kestää useita eri vikatilanteita, käytännössä väylä lopettaa toimintansa vasta kun molemmat väyläjohtimet ovat poikki tai oikosulussa maapotentiaaliin. Jos toinen johdin katkeaa, vikasietoinen väylä kommunikoi vain ehjällä johtimella, viallisen johtimen ollessa suuriohmisessa tilassa.

2.4.1 CAN High speedin fyysinen toteutus, laskettu tiedonsiirtonopeus

Hidas CAN-väylä voidaan toteuttaa myös nopean väylän fyysisellä toteutuksella, jolloin tiedonsiirtonopeus on suurimmillaan rajattu nopeuteen 125 kbit/s. Tällä tavalla toteutettu hidas CAN-väylä poikkeaa ominaisuuksiltaan vain tiedonsiirtonopeudeltaan ja näin ollen myös vikatilat sekä jännitetasot pysyvät samanlaisina (Paret 2007: 144).

2.4.2 CAN Fault Tolerantin fyysinen toteutus

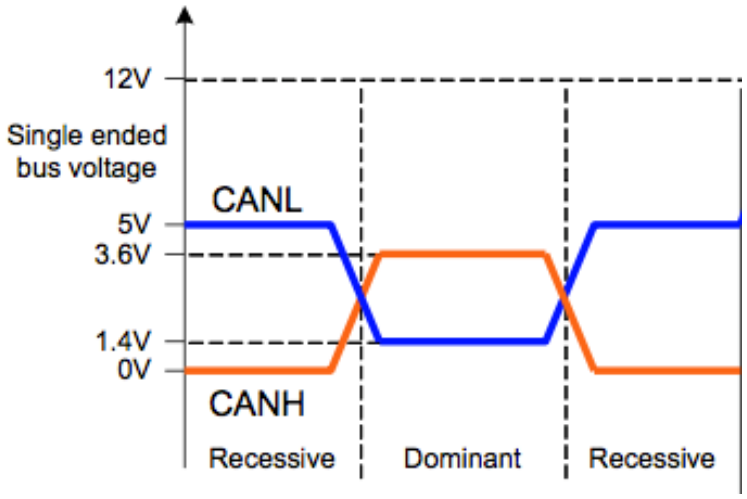
Hitaan CAN-väylän määritelmän alle luetaan myös vikasietoinen CAN-väylä. Vikasietoisen CAN-väylän fyysinen toteutus poikkeaa nopean CAN-väylän toteutuksesta siten, että väylässä terminointivastukset ovat sijoitettu erikseen jokaiseen ohjainlaitteeseen. Terminointivastukset valitaan siten, että verkon resistanssiksi tulee 100 ohmia per väyläjohdin. Mikäli verkkoon on kytketty viisi ohjainlaitetta, tulee terminointivastusten arvon olla 500 ohmia ja näin yksittäisen väyläjohtimen resistanssin arvoksi tulee 100 ohmia. (Malmari 2016: 11.)



Kuva 5. CAN Fault Tolerant -väylän topologia (Malmari 2016: 11).

2.4.3 Jännitetasot

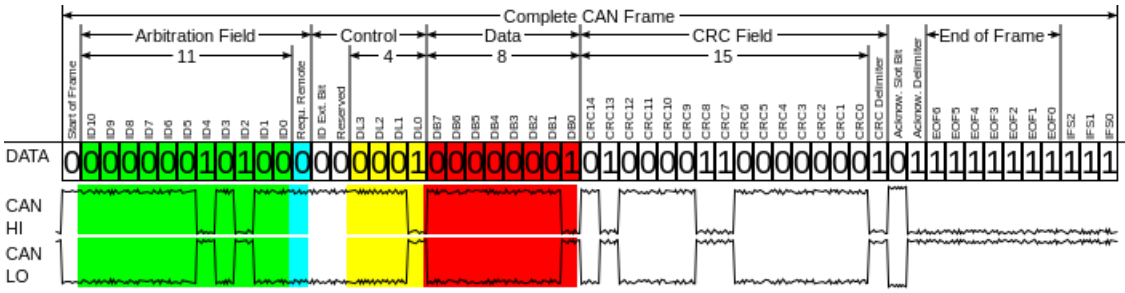
CAN Fault Tolerant -väylän ohjainpiiri sekä lähetin-vastaanotin käyttäytyvät periaate tasolla samalla tavalla kuin CAN High Speed -väylän tapauksessa, mutta jännitetasot ovat erilaiset. Resessiivisessä tilassa CAN L -johtimen jännitetaso on 5 volttia ja CAN H -johtimen 0 volttia. Dominanttitilassa CAN L -johtimen jännitetaso laskee 1,4 volttiin ja vastaavasti CAN H -johtimen jännite kasvaa 3,6 volttiin. Jännite-erot tilasta riippuen ovat siis 5 volttia tai 2,2 volttia.



Kuva 6. CAN Fault Tolerant -väylän jännitetasot sekä tilat (Schade & Muth 2016: 18).

2.5 Viestikehys

CAN High- ja Low Speed -väylissä käytetty viestikehys on tarkoin määritetty standardissa ISO-11898-1. Alla olevassa kuvassa on CAN 2.0A:n mukainen viestikehys esitetty sen eri osioiden ja jännitetasojen kanssa (kuva 7). Kuvan alla on myös listamuotoisesti esitetty eri osioiden nimet, pituudet ja tärkeimmät tehtävät. Tässä luvussa käydään vielä läpi yleisesti käytettyjen CAN 2.0A- ja CAN 2.0B -viestikehysten erot, viestin priorisointi, Bit Stuffing -menetelmä sekä virheidenhallinta.



Kuva 7. Standard-muotoinen viestikehys sekä viestiä vastaava jännitesignaali (CAN bus 2003).

- Kehyksen alku (Start of Frame)
    - 1 bitti, looginen 0, aloittaa jokaisen viestin
- Tunnistekenttä
    - 11 tai 29 bittiä (Arbitration Field), kertoo viestin tunnisteen, jonka muut ohjainlaitteet voivat tarvittaessa lukea
- Kontrollikenttä (Control)
    - 6 bittiä, kertoo datan määrän ja viestin tunnistekentän pituuden
- Datakenttä (Data)
    - 0-64 bittiä, viestin varsinainen sisältö, jota muut ohjainlaitteet hyödyntävät
- CRC-tarkistussumma/virheentarkistus (CRC Field)
    - 17 bittiä, algoritmeilla suoritettu virheentarkistus
- Vahvistuskenttä
    - 2 bittiä, viestin saaneet ohjainlaitteet kuittaavat viestin saapuneeksi
- Lopetus (End of Frame)

- 7 bittiä, looginen 0, viestin perään lisätään katkos, jollain mikään ohjainlaite ei voi lähettää

CAN 2.0A ja CAN 2.0B, Standard ja Extended

CAN-väylän tuotantoon ottamisen alkuvaiheilla huomattiin, että tunnistekentän 11 bitin osoiteavaruus voisi aiheuttaa jossain tapauksissa ongelmia. Tästä syystä kehitettiin CAN 2.0B, jonka tunnistekentän pituudeksi määritettiin 29 bittiä. Pääasialliset erot CAN 2.0A:n ja CAN 2.0B:n välillä ovat tunnistekentän pituudessa ja kontrollikentän rakenteessa. Huomattavaa on myös, että molempia CAN 2.0A:ta ja CAN 2.0B:tä voidaan käyttää samassa järjestelmässä rinnakkain. Kontrollikentän rakennetta muutettiin lähinnä siitä syystä, että tunnistekentän pituutta muutettiin. ISO-standardiin lisättiin samalla myös kohta, joka velvoittaa, että CAN 2.0B:tä tukevien komponenttien on tuettava myös CAN 2.0A:ta (Paret 2007: 76–81).

2.6 Viestin priorisointi

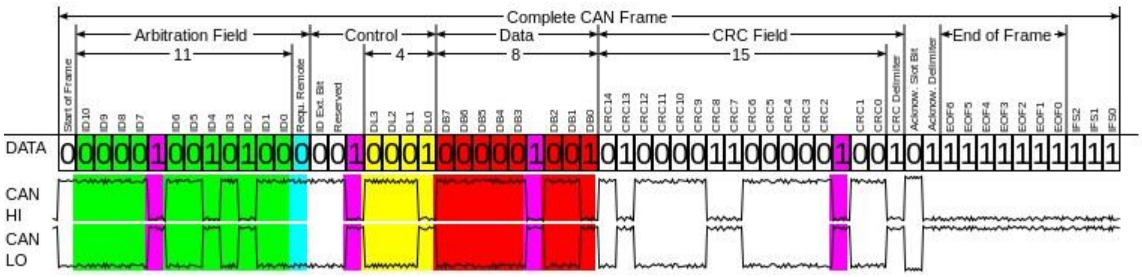
CAN-väylä perustuu niin kutsuttuun Multimaster-periaatteeseen, mikä tarkoittaa, että kaikki järjestelmään liitetyt yksiköt on asetettu toimimaan samoilla valtuuksilla (Frei 2015: 12). Ohjainlaitteet lähettävät viestejä väylälle käytännössä tietämättä toisistaan, mikä väistämättä aiheuttaa jossain vaiheessa tilanteen, jossa kaksi tai useampi ohjainlaitetta alkavat lähettämään viestiä samalla hetkellä. Viestien välistä konfliktia varten on kehitetty viestikehyksen yksilöidyn ID:n eli tunnistekentän ”arvoon” perustuva priorisointijärjestelmä. Käytännössä pienimmän tunnistekentän omaava viesti lähetetään ensin. Usein pienellä ID:llä lähetettyjen viestien data on tärkeämpää kuin isolla ID:llä. Heti kun pienimmän tunnistekentän omaava viesti on lähetetty, muut ohjainlaitteet saavat aloittaa uudelleenlähetyksen, jossa priorisointi suoritetaan uudelleen. Kuvassa 8 on annettu esimerkki viestin priorisoinnista kolmen eri viestin tunnistekentän perusteella. Kuvassa tunnistekenttää lähdetään tarkastelemaan vasemmalta oikealle. Toisen sarakkeen kohdalla moottorin lämpötila häviää priorisoinnin kahdelle muulle ohjainlaitteelle, koska sen tunnistekentän toinen bitti on 1 ja muiden ohjainlaitteiden tunnistekentän toinen bitti on 0. Viidennen sarakkeen kohdalla ajonopeuden lähettävän ohjainlaitteen bitti on 1 ja vastaavasti pyörintänopeuden lähettävän ohjainlaitteen bitti on 0. Priorisoinnin voittaa siis 177 eli moottorin pyörintänopeus, ja tämän ohjainlaitteen viesti pääsee väylälle. Esimerkkikuvassa punainen tarkoittaa häviämistä priorisoinnissa ja vihreävoittoa.

ECU 1	0	0	0	1	0	1	1	0	0	0	1	= 177 (Moottorin pyörintänopeus)
ECU 2	0	1	1	0	0	1	0	1	1	0	1	= 813 (Moottorin lämpötila)
ECU 3	0	0	0	1	1	0	1	1	0	1	0	= 218 (Ajonopeus)
Väylä	0	0	0	1	0	1	1	0	0	0	1	= 177 (Moottorin pyörintänopeus)

Kuva 8. Viestien priorisointi (Malmari 2016: 5)

2.7 Bit Stuffing -menetelmä

Aiemmin esitetyn viestikehyksen esimerkkikuvasta on selkeyden vuoksi jätetty pois Bit Stuffing -menetelmän lisäämät bitit (kuva 7). CAN-väylän viestit koodataan NRZ-menetelmällä binäärisiin jännitetasoihin, looginen 0- ja 1-tila. NRZ-menetelmällä koodattua signaalia tarkastellessa, eri osioiden rajapintaa huomioon ottamatta, tullaan väistämättä tilanteeseen, jossa looginen tila ei vaihdu yli viiden bitin välein. Viestin vastaanotto perustuu bittien kestoajan erotteluun, ja tilanne, jossa looginen tila ei vaihdu tarpeeksi usein saattaa aiheuttaa ohjainlaitteiden välistä epäsynkronisaatiota. Bit Stuffing -menetelmä lisää jokaista viittä peräkkäistä samaa loogista tilaa kohden yhden loogiselta tilaltaan vastakkaisen ylimääräisen bitin, jotta ohjainlaitteiden välinen synkronointi pysyy yllä. Todellisuudessa viestikehys saattaa loogisten tilojen mukaan poiketa ”normaalista” pituudestaan muutaman bitin verran (kuva 9).



Kuva 9. Viestikehys, johon lisätty Stuff-bitit (CAN bus 2003).

2.8 Virheidenhallinta

2.8.1 Virhetilanteet

CAN-väylässä tapahtuvassa kommunikaatiossa sattuu väistämättä aika ajoin virheitä. Viestien eri virheet voidaan karkeasti jakaa neljään ryhmään:

- Bittivirhe, on signaalitasolla tapahtuva virhe. Esimerkiksi ulkoisen häiriön aiheuttama, joka muuttaa bitin arvon päinvastaiseksi kuin alun perin.

- CRC-virhe, viestien virheitä tarkastellaan algoritmeihin perustuvalla laskennalla, jonka tuloksena saadaan viestikehystä vastaava yksinkertainen luku. Lähettävä ohjainlaite ”pakkaa” viestin luvuksi, CRC-kenttään ja viestin vastaanottava ohjainlaite ”purkaa” tämän luvun. Molempien lukujen on oltava samat, mikäli luvut eivät täsmää on tiedonsiirrossa tapahtunut virhe.

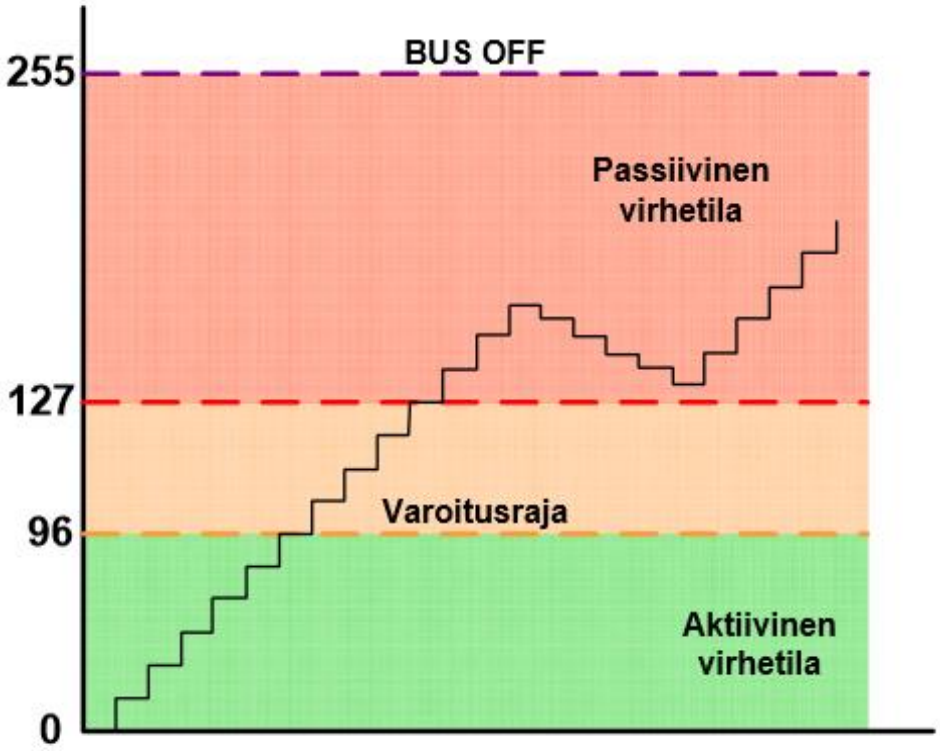
- Stuff-bittivirhe, jokaisen viiden loogisen tilan jälkeen lisätään vastakkainen looginen tila, mikäli kuusi perättäistä loogista tilaa ovat samat, tulkitaan tämä virheeksi.

- ACK-virhe, ohjainlaite ei syystä tai toisesta kuittaa vastaanotettua viestiä, tulkitaan virheeksi.

2.8.2 Virhelaskuri

Virhetilanteen sattuessa molemmat ohjainlaitteet, niin lähettävä kuin vastaanottava, kasvattavat sisäistä virhelaskurin arvoaan ja lähettävät väylälle virheviestin. Virhelaskurin avulla ohjainlaitteet pystyvät tulkitsemaan vian hetkelliseksi eli satunnaiseksi tai jatkuvaksi eli pysyväksi. Virheviestin tunnistekentän ”arvo” on hyvin pieni, ja näin ollen se voittaa priorisointitarkastelussa lähes kaikki muut viestit. Ohjainlaitteiden sisäiselle laskurille on säädetty tietyt reunaehdot, ettei väylä tukkeudu pienen tunnistekentän omaavista virheviesteistä (kuva 10). Aktiivisen virhetilan raja on 127 virheellistä viestiä. Tähän asti ohjainlaite lähettää virheviestin pienellä tunnistekentän ”arvolla” ruuhkauttaen väylän muun liikenteen lähes täysin. Virheellisten viestin määrään saavuttaessa 127 ohjainlaite siirtyy passiiviseen virhetilaan ja lähettää virheviestin suuremmalla tunnistekentän ”arvolla” eikä enää ruuhkauta väylää. Virheiden määrän saavuttaessa 255 oh-

jainlaite koodauksesta riippuen joko hiljenee täysin tai käynnistyy uudelleen ennalta määritetyn ajan kuluttua (Paret 2007: 57–59).



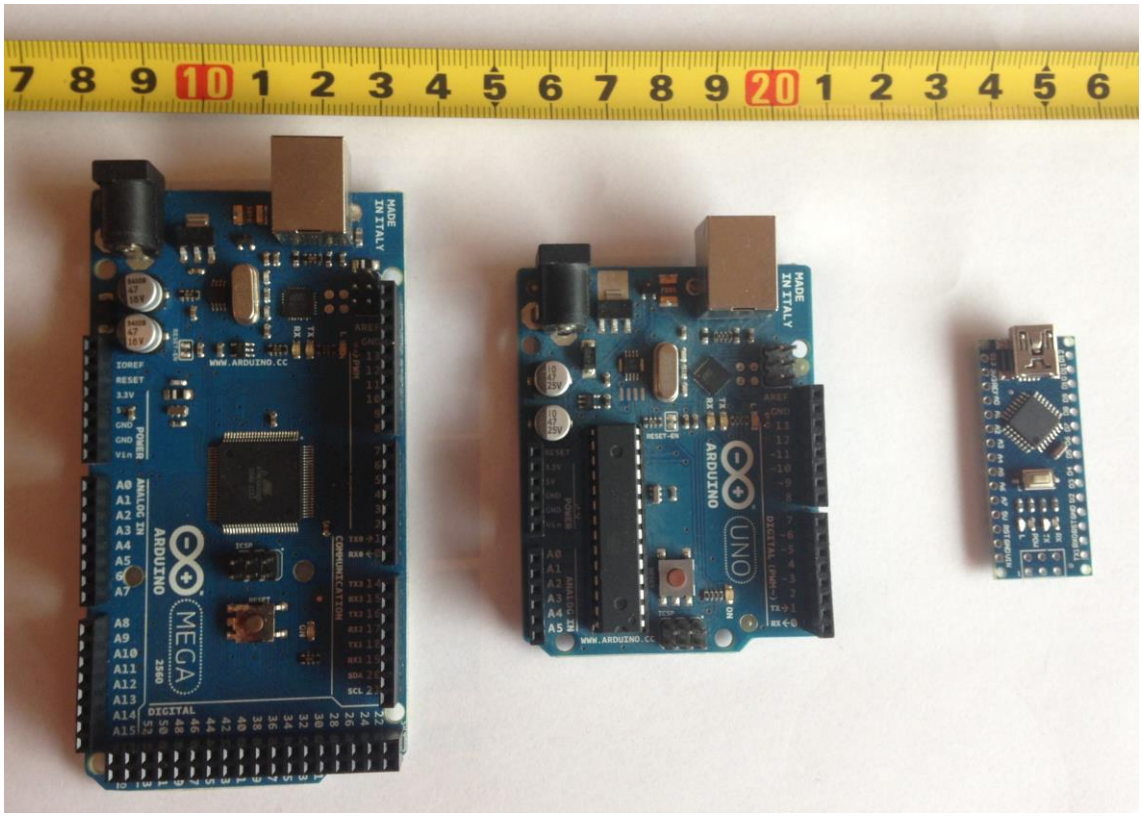
Kuva 10. Virhelaskurin rajat (Malmari 2016: 5).

3 Laitteen elektroniikka

Tässä luvussa tarkastellaan laitteen elektroniikkaa, alkaen mikrokontrollerin esittelystä ja piirin suunnittelusta ja päättyen valittuihin komponentteihin. Komponenttien kytkennät on esitetty liitteessä 1. Luvussa on myös kerrottu piirin suunnittelutyöstä, sen toiminnasta ja suunnitteluun käytetyistä ohjelmistoista. Projektia varten suunniteltiin piiri kokonaan alusta loppuun, jotta siihen valitut komponentit vastaisivat ominaisuuksiltaan työn tilaajan asettamia vaatimuksia.

3.1 Arduino

Projektissa käytettiin avoimeen laitteistoon perustuvaa Arduino UNO -kehitysalustaa. Arduino on suhteellisen halpa ja helppokäyttöinen kehitysalusta, jota ohjelmoidaan USB-portin kautta. Ohjelmointiin tarvitaan USB-kaapeli ja Arduino IDE -ohjelmointiympäristö, jonka voi ladata ilmaiseksi Arduinon kotisivuilta. Projektissa käytetty kehitysalusta verrattuna Arduino Mega- ja Arduino Nano -kehitysalustoihin (kuva 11).



Kuva 11. Arduino Mega-, Arduino UNO- ja Arduino Nano -kehitysalustat.

Erilaisia kehitysalustoja on tarjolla eri käyttötarkoituksen mukaan monia. Käyttötarkoituksia ovat muun muassa seuraavat: aloittelijalle, vaatetuksen ja internetyhteyteen pohjautuvat. Käyttötarkoituksen lisäksi kehitysalustan valintaan vaikuttavat mm. kehitysalustan muistin määrä, I/O-pinnien määrä, sekä fyysinen koko. Laajan tarjonnan vuoksi Arduino-kehitysalustoja on jo vuosia käytetty monissa eri projekteissa niin opiskelijoiden, artistien, ohjelmoijien kuin ammattilaisten keskuudessa. (Arduino Introduction.)

Arduino UNO -kehitysalusta valittiin projektiin sen suhteellisen pienen fyysisen koon, teknisten ominaisuuksien ja hintansa vuoksi. Hinta ja fyysinen koko olivat suuressa osassa siksi, että kehitysalustoja tarvittiin kaksi kappaletta laitetta kohden. Alla olevassa taulukossa on esitetty valintaan vaikuttaneet tekniset ominaisuudet (taulukko 2). Käytännössä valintaan vaikutti muistin määrä, I/O-pinnien lukumäärä, sekä käyttöjännite. Näiden lisäksi otettiin huomioon myös kytkennässä käytettyjen komponenttien virrantarve, joka pysyi alle kehitysalustan oman jännitteensäätimen maksimirajojen alapuolella.

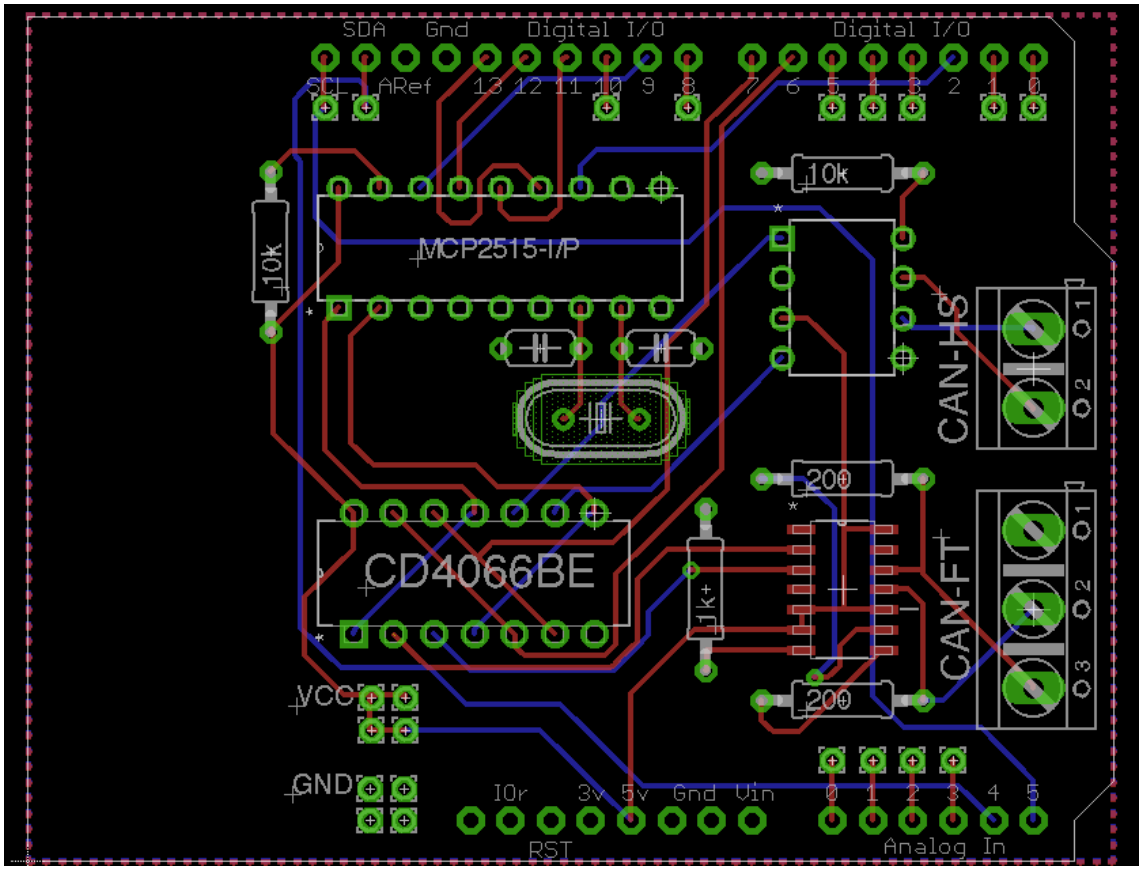
Taulukko 2. Arduino UNO -kehitysalustan tekniset ominaisuudet.

Arduino UNO	
Mikrokontrolleri	ATmega328P
Digitaaliset I/O-pinnit	14 kpl, joista 6 kpl PWM
Analogiset I/O-pinnit	6 kpl
Flash-muisti	32 kbit
SRAM	2 kbit
EEPROM	1 kbit
Kellotaajuus	16 MHz
Käyttöjännite	6–20 V

Vuodesta 2015 eteenpäin USA:ssa myydyt kehitysalustat kulkevat nimellä Arduino ja Euroopassa myydyt nimellä Genuino. Genuino-alustoja on kuitenkin verrattain pienempi tarjonta kuin Arduinoja. Käyttäjäyhteisö puhuu kuitenkin kaikista edellä mainituista kehitysalustoista toistaiseksi hyvin usein nimellä Arduino. (Landoni 2015.)

3.2 Piirikortti

Projektia varten suunniteltiin Arduino-kehitysalustan prototyyppikortin pohjalle oma piirikortti (kuva 12). Kehitysalustan lisäkorttivalikoimasta löytyy valmiitakin CAN-piirikortteja, mutta markkinoilla ei ole tarjolla piirikorttia, joka sisältäisi myös CAN Fault Tolerant -väylän. Tämän vuoksi projektia varten suunniteltiin piirikortti, joka sisältää CAN High Speed -väylän lisäksi myös CAN Fault Tolerant -väylän.



Kuva 12. Projektia varten suunniteltu piirikortti.

3.2.1 EAGLE PCB

EAGLE PCB on ilmainen piirikorttien ja kytkentäkaavioiden suunnitteluun kehitetty ohjelmisto. Lyhenne EAGLE tulee englannin kielen sanoista Easy, Applicable, Graphical, Layout, Editor. Nimensä mukaan ohjelmisto on helppokäyttöinen ja silmälle mielekäs ohjelmisto, johon on helppo päästä sisälle, vaikka olisikin vasta-alkaja. Ohjelmistoa on kehitetty jo yli 25 vuoden ajan, ja eri komponentteja myyvät yritykset tarjoavat myymilleen piireille komponenttikirjastoja, jotka mahdollistavat itse piirikortin vaivattoman suunnittelun.

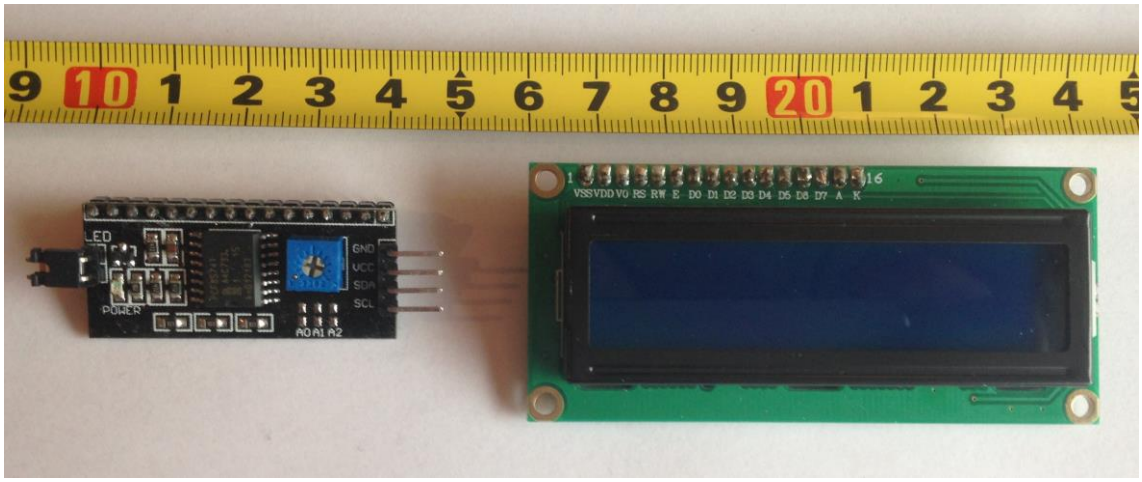
3.2.2 ITEAD Studio

Piirikortit tilattiin Kiinassa toimivalta ITEAD Studio piirikorttivalmistajalta. Yritys on hyvin tunnettu elektroniikkaharrastajien, insinöörien sekä opiskelijoiden keskuudessa. Piirikortit ovat halpoja ja niitä saa pienissä erissä. Piirikortin tilaamista, sekä suunnittelun yhteydessä huomioon otettavia asioita käydään erittäin hyvin läpi internetoppaassa alle euron piirilevyt (Jämsä 2011). Piirikorttia suunniteltaessa otettiin oppaassa ilmoitetut kohdat huomioon, muun muassa lisäämällä maatason ja johtimien välistä etäisyyttä.

3.3 LCD-näyttö ja I2C-moduuli

Väyläsimulaattorin valikko tulostetaan 16 x 2 -LCD-näytölle I2C-kommunikointiprotokollaa hyväksi käyttäen (kuva 13). LCD-näytölle on mahdollista tulostaa 16 merkkiä riviä kohden, kahdelle riville. Itse näyttö ei osallistu valikon toimintoihin muuten kuin orjalaitteena, eli valikko tulostetaan näytölle, mutta näytöllä ei näy kursoreita tai vastaavia indikaattoreita valikon tilasta vaan valikossa navigointi tapahtuu ohjelmallisesti.

I2C-moduuli lisättiin projektiin, jotta näytön ohjaamista varten tarvittujen mikrokontrollerin I/O-pinnien määrä pystyttiin laskemaan neljään. Ilman I2C-moduulia pelkän LCD-näytön ohjaamiseen tarvitaan keskimäärin kytkennästä riippuen noin 11 I/O-pinniä. Valitun kehitysalustan tapauksessa tämä olisi tarkoittanut melkein puolta käytettävissä olevista I/O-pinneistä.



Kuva 13. I2C-moduuli ja LCD-näyttö.

3.4 CAN-ohjain MCP2515

CAN-lähetin-vastaanotinpiirit tarvitsevat toimiakseen erillisen ohjainpiirin, joka muuttaa mikrokontrollerin lähettämän viestin CAN-viestikehykseksi ja purkaa vastaanotetun viestin. Tämän lisäksi viestien priorisointi tapahtuu CAN-ohjainpiirissä. Projektia varten valittiin Arduino käyttäjäyhteisön keskuudessa suosituksi noussut CAN-ohjainpiiri MCP2515, lähinnä siitä syystä, että piirille löytyy valmiita ohjelmakirjastoja. Piiriä valmistaa Microchip Technology Inc. (Stand-Alone CAN Controller with SPI Interface 2012.)

3.5 CAN-lähetin-vastaanotin MCP2551

CAN High Speed -väylän lähetin-vastaanotinpiirin valitseminen aloitettiin tutkimalla eri valmistajien ja käyttäjäkunnan suosittelemia spesifikaatioita. Nopeasti huomattiin, että MCP255-CAN-lähetin-vastaanotin oli erittäin suosittu halvan hintansa, saatavuutensa sekä yhteensopivuutensa saman valmistajan MCP2515-CAN-ohjainpiirin kanssa. (High Speed CAN Tranceiver 2010.)

3.6 CAN-lähetin-vastaanotin TJA1055

Vikasietoisen CAN Low Speed -väylän lähetin-vastaanotinpiiriksi valittiin NPX:n valmistama TJA1055. Piirin valintaan vaikutti se, että sen kanssa voitiin käyttää jo valittua MCP2515-CAN-ohjainpiiriä. Tämän lisäksi piiri toimii myös 5 voltin käyttöjännitteellä, minkä vuoksi ei ollut tarpeen käyttää piirin sisäistä jännitesäädintä vaan käyttöjännite voitiin ottaa suoraan kehitysalustasta. (TJA1055 2013.)

3.7 CD4066BE-digitaalikytkin

Texas Instruments Inc:n valmistama CD4066BE on yleiskäyttöinen digitaalikytkin, jossa on neljä eri kytkintä. Kytkimiä voidaan ohjata mikrokontrollerin lähdöillä. Digitaalikytkimen käyttöjännite on 5—20 volttia, ja se on erittäin kykenevä nopeillakin taajuuksilla. Näistä syistä projektin CAN-ohjaimen ja kahden lähetin-vastaanotinpiirin väliset kommunikointilinjat vaihtuvat tämän kytkimen kautta. (CD4066B 2003.)

4 Ohjelmointi

Projektia varten kirjoitettua ohjelmakoodia ei ole mielekästä, sen pituuden vuoksi, käydä kokonaan läpi tai liittää edes liitteeksi. Tässä luvussa kuvataan ohjelmakoodin kokonaisuuden kannalta tärkeimmät kohdat läpi, esitellään CAN-väylien ohjaamisen kannalta olennaiset funktiot sekä esitetään yksinkertainen malli ohjelman rakenteesta. Laitteen toiminta yleisellä tasolla on esitetty toimintakaaviossa (liite 2).

4.1 Arduino IDE

Arduino-kehitysalustan ohjelmointi tapahtuu avoimeen lähdekoodiin perustuvan Arduino IDE -ohjelmointiympäristön avulla. Ohjelmointiympäristö sisältää tekstieditorin, kääntäjän, sekä valmiudet kirjoittaa käännetty ohjelma mikrokontrollerin Flash-muistiin (kuva 14).

Kuva 14. Arduino IDE -ohjelmiston aloitusnäyttö.

Yksinkertaisuutensa vuoksi ohjelmointiympäristöstä ei löydy harjaantuneelle ohjelmoijalle tuttuja ominaisuuksia, kuten koodin simulointia, automaattista koodin täyttöä tai tarkkaa virhetilojen tarkastelua. Kehitysalustalle löytyy myös useita vaihtoehtoisia ohjelmointiympäristöä, joista mainittakoon Visual Studio 2015 Visual Micro -lisäosalla.

4.2 Ohjelmakoodin rakenne

Arduino-kehitysalustaa varten kirjoitetut ohjelmakoodit voidaan jakaa kolmeen osa-alueeseen: rakenne, arvot (muuttujat ja vakiot) sekä funktiot. Rakenne sisältää setup- ja loop-funktiot, joista ensimmäisessä julistetaan ohjelmakoodin globaalit osat. Loop-funktiota, nimensä mukaisesti, käydään koko ajan läpi, jollei toisin määritetä. Loop-funktiota voidaan myös kutsua pääohjelmaksi, sillä se kykenee kutsumaan käyttäjän muita funktioita. Setup-funktion jälkeen julkaistaan muuttujat sekä vakiot. Viimeisenä kirjoitetaan käyttäjän muut tarvitsevat funktiot. Ohjelmakoodien tärkeimmät osuudet esitellään kehitysalustakohtaisesti sekä CAN-viestin lähetys- ja vastaanotto-funktioille on omat kappaleensa.

4.2.1 Lähettävän kehitysalustan rakenne

Pääohjelma pyrittiin pitämään mahdollisimman yksinkertaisena, koska ohjelmakoodista tuli suhteellisen monimutkainen ja pitkä (esimerkkikoodi 1). Pääohjelman tehtävänä on tarkkailla valikon vallitsevaa tilaa ja sitä, onko käyttäjä painanut painikkeita. Current-funktio ohjaa tapahtumapohjaisen valikon toimintoja sille määritettyjen muuttujien arvojen mukaan. Tapahtuman mukaan valikossa kutsutaan eri funktioita, joista tärkeimmät on esitetty myöhemmissä kappaleissa. Painikkeiden painaminen muuttaa valikolle määritettyjen muuttujien arvoja, joiden avulla valikossa liikutaan. Painikkeiden tilaa valvotaan läpi ohjelman ja tästä vastaa nimensä mukaisesti analogButtons.checButtons-funktio.

```
void loop()
{
  analogButtons.checkButtons();
  current();
}
```

Esimerkkikoodi 1. Lähettävän kehitysalustan loop-funktio.

4.2.2 Vastaanottavan kehitysalustan rakenne

Viestejä vastaanottavan kehitysalustan ohjelmakoodista tuli huomattavasti lyhyempi ja yksinkertaisempi (esimerkkikoodi 2). Pääohjelman tehtävänä on tarkkailla, saapuuko lähettävän kehitysalustan kautta pyyntöä aloittaa kommunikaatio, vastaanottaa lähetetty CAN-viestikehys sekä vilkuttaa LED-valoa viestin saapumisen merkiksi. Condition-funktio tarkkailee kehitysalustojen välistä sarjakommunikaatiota sekä asettaa valitun lähetin-vastaanotinpiirin ja tiedonsiirtonopeuden. Message-funktion tehtävänä on tarkkailla saapuuko CAN-ohjaimelle viestejä, ja kirjoittaa saapuvan viestin tunniste sekä data niille varattujen muuttujien arvoihin. Check-funktio tarkkailee saapuvan viestin muuttujien arvoa ja oikean arvon kohdalla kirjoittaa LED-valoa ohjaavan ulostulon hetkellisesti päälle.

```
void loop()
{
  condition();
  message();
  check();
}
```

Esimerkkikoodi 2. Vastaanottavan kehitysalustan loop-funktio.

4.3 Kirjastot

Arduinon käyttöliittymä sisältää valmiina suuren joukon kirjastoja ja aktiivisen käyttäjäyhteisön kautta kirjastoja luodaan koko ajan lisää eri tarkoituksiin. Kirjastojen tarkoituksena on suorittaa todellisuudessa monimutkaisempaa ohjelmakoodia yksinkertaisemmilla valmiiksi määritetyillä komennoilla. Kirjastot julkaistaan heti ohjelmakoodin alussa (esimerkkikoodit 3 ja 4).

```
#include <SoftwareSerial.h>
#include "AnalogButtons.h"
#include <MenuBackend.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <mcp_can.h>
#include <SPI.h>
```

Esimerkkikoodi 3. Viestin lähettävän kehitysalustan kirjastot.

```
#include <SoftwareSerial.h>
#include <mcp_can.h>
#include <SPI.h>
```

Esimerkkikoodi 4. Viestin vastaanottavan kehitysalustan kirjastot.

SoftwareSerial-kirjasto mahdollistaa kahden Arduino-kehitysalustan sarjamuotoisen kommunikoinnin käyttäen laitteen loppukäyttäjältä piilossa olevia johtimia. Analogbuttons on painikkeiden rinnankytkentään, I/O-pinnien säästämiseksi, kehitetty kirjasto, joka hoitaa muun muassa painikkeiden debonkkauksen ja painikkeiden vallitsevan tilan tarkkailun. Menubackend on kirjasto, joka on luotu helpottamaan valikon luomista ja sen sisällä liikkumista yhdessä Analogbuttons-kirjaston kanssa. LCD-näyttö ja siihen juotettu I2C-moduuli vaativat toimiakseen Liquidcrystal_I2C- sekä wire-kirjastot. Liquidcrystal_I2C-kirjasto vastaa syötteen kirjoittamisesta itse LCD-näytölle ja wire-kirjasto vastaavasti siitä, missä muodossa kehitysalustalta lähtevä syöte saavuttaa I2C-moduulin. CAN-väylän lähetin-vastaanotinpiirien ohjaimena toimivalle MCP2151-piirille on kirjoitettu myös oma kirjastonsa, aktiivisen käyttäjäyhteisön toimesta, joka hoitaa käytännössä kaiken viestimisen lähetin-vastaanottopiirien ja ohjaimen välillä. Arduino ohjaa CAN-väylän ohjainpiiriä sarjamuotoisella SPI-kommunikaatiolla, ja tästä vastaa SPI-kirjasto.

4.4 CAN-viestin lähetys- ja vastaanotto -funktiot

4.4.1 Lähettävän kehitysalustan message-funktio

Funktion rakenne on toteutettu if-lauseilla, joiden tehtävänä on suorittaa lähetinpiirin vaihto, tiedonsiirtonopeuden määrittely sekä varsinaisen CAN-viestikehyksen lähetys. Eri tilat toteutuvat sen mukaan, mitä käyttäjä on valikossa valinnut (esimerkkikoodi 5).

```
void message()
{
  if(canmode == canhson)
 {
   digitalWrite(6,LOW); // LOW = HS CAN / HIGH = FT CAN
   digitalWrite(7,HIGH); // LOW = FT CAN / HIGH = HS CAN
```

```
    CAN0.begin(CAN_500KBPS) == CAN_OK;
    CAN0.sendMsgBuf(0x01, 0, 8, stmp);
    delay(30);
    CAN0.sendMsgBuf(0x02, 0, 8, stmp1);
    delay(17);
    CAN0.sendMsgBuf(0x03, 0, 8, stmp2);
    delay(10);
  }
  else if (canmode == canlson)
  .
  .
  .
  }
```

Esimerkkikoodi 5. Lähettävän kehitysalustan message-funktio.

Käyttäjän valikosta valitsema toiminto muuttaa muuttujan canmode arvoa. Esimerkkikoodin 5 kohdalla käyttäjä on valinnut CAN High Speed -väylän, jota muuttujan arvo canhson vastaa. Seuraavaksi kytketään valittua väylää vastaava lähetinpiiri digitalWrite-komentoja käyttäen. CAN-ohjain käynnistetään lähetinpiirin valinnan jälkeen komennolla CAN0.begin(CAN_500KBPS) == CAN_OK. Käynnistyskomennolla määritellään myös tiedonsiirtonopeus, joka on esimerkkitapauksessa 500 kbit/s. Varsinaiset viestit lähetetään CAN0.sendMsgBuf(0x01, 0, 8, stmp)-komennolla, jolla määritetään viestin tunnistekentän ”arvo”, tunnistekentän pituus, datakentän pituus sekä ennalta määritetty viestin sisältö. Funktion sisällä on samaa kaavaa noudattavat koodit CAN Low Speed- ja Fault Tolerant -väylille. Valitun väylän konfiguraatio siirretään SoftwareSerial-kirjastoa hyväksi käyttäen vastaanottavalle kehitysalustalle, jotta kehitysalustojen väliset CAN-väylät pystytään asettamaan vastaamaan toisiaan.

4.4.2 Vastaanottavan kehitysalustan message-funktio

SoftwareSerial-kirjaston, sekä condition-funktion avulla vastaanottavalle kehitysalustalle saapuva väyläkonfiguraatio asetetaan vastamaan lähettävän kehitysalustan konfiguraatiota. Vastaanottavan kehitysalustan viestin vastaanotto perustuu if-lauseeseen, joka tarkastelee CAN-ohjaimen INT-sisääntuloa. Sisääntulon looginen tila vaihtuu riippuen siitä, onko CAN-ohjaimelle saapunut viestejä vai ei (esimerkkikoodi 6).

```
void message()
{
    if(!digitalRead(2))
        {
          CAN0.readMsgBuf(&len, rxBuf);
          rxId = CAN0.getCanId();
          Serial.print("ID: ");
          Serial.print(rxId, HEX);
          Serial.print("  Data: ");
          for(int i = 0; i<len; i++)
        {
          if(rxBuf[i] < 0x10)
          {
            Serial.print("0");
          }
        Serial.print(rxBuf[i], HEX);
        Serial.print(" ");
        }
        }
}
```

Esimerkkikoodi 6. Vastaanottavan kehitysalustan message-funktio.

Viestin saapuessa komento CAN0.readMsgBuf(&len, rxBuf) purkaa ja lukee viestikehyksen. Komennolla rxId = CAN0.getCanId() määritetään muuttujan rxId-arvo saapuneen viestin tunnistekentän mukaiseksi. Serial.print-komennot tulostavat viestikehysten tunniste- ja datakentät Arduino IDE:n sarjamonitorille. Sarjamonitorin avulla voidaan tarvittaessa tarkastaa väylän toiminta, kehitysalustan USB-porttiin liitettävällä tietokoneella.

5 Väyläsimulaattorin toiminta

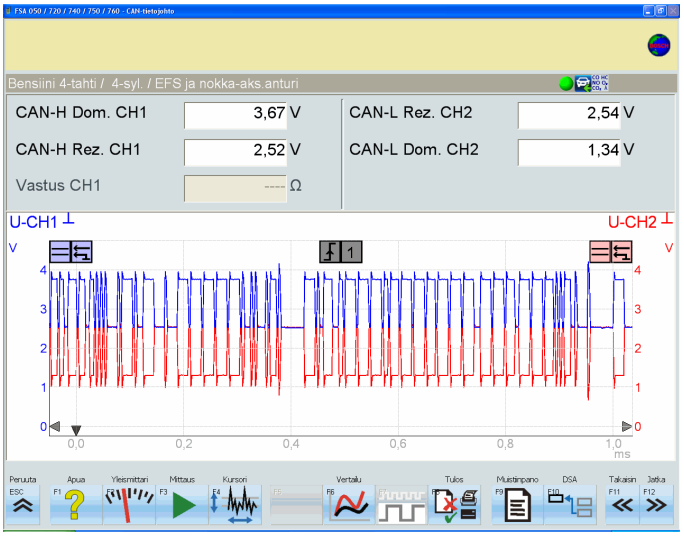
5.1 Yhteneväisyystarkastelu

Rakennetun laitteen väylien jännitetasoa ja yleistä toimintaa verrattiin Metropolian Volvo XC90 -merkkisen ajoneuvon väylien toimintaan. Vertailu suoritettiin, jotta laite käyt-

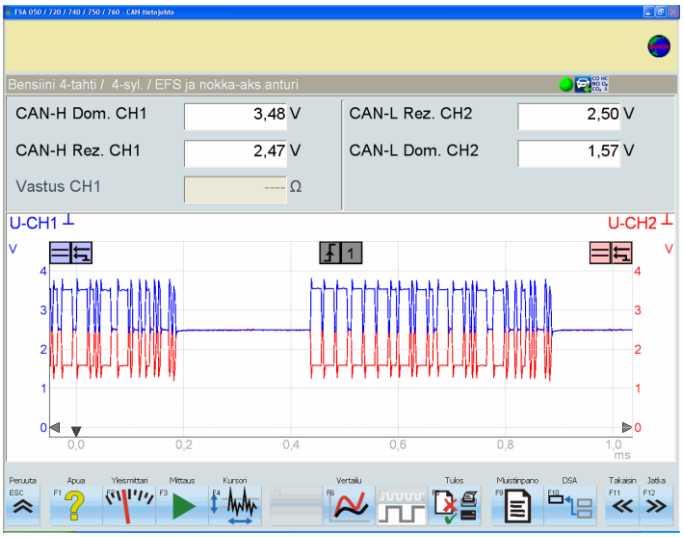
täytyisi mahdollisimman realistisesti, niin normaalissa tilanteessa kuin vikatilanteessakin. Samalla ajoneuvosta pystyttiin kartoittamaan millä tasolla esim. CAN L-johtimen poikkeaminen vaikutti ajoneuvon mukavuuslaitteiden toimintaan. Työn tilaaja toivoi, että vikasimulaattoria varten tehdyissä rastitehtävissä mainittaisiin miten eri vikatilanteet vaikuttavat ajoneuvon toimilaitteiden toimintaan.

Ajoneuvo Volvo XC90 valittiin siitä syystä, että siinä oli valmiiksi mittapisteet ja vikatilat CAN-väylän mittauksia varten. Vikatiloina olivat erilaiset oikosulut ja johtimien katkokset. Tässä osiossa käsitellään normaalitilan lisäksi vain yksi vikatila ja loput vikatilat on esitetty liitteessä 3. On kuitenkin huomattava, että ko. ajoneuvosta ei löydy vikasietoista väylää, vaan CAN Low Speed -väylä on toteutettu CAN High Speed -väylän fyysisellä rakenteella.

Normaalitilanteessa jännitetasoissa ei ole juurikaan mainittavia eroja mitatun ajoneuvon CAN-väylän (kuva 15) ja rakennetun CAN-väyläsimulaattorin välillä (kuva 16). Pieniä eroja voidaan kuitenkin huomata jännitetason muuttuessa, mutta myös ajoneuvovalmistajasta riippuen väylien jännitetasot vaihtelevat hieman. Jännitetasoihin vaikuttavat esimerkiksi järjestelmän ohjainlaiteiden määrä, valmistajan komponenttivalinnat sekä väylän rakenne.

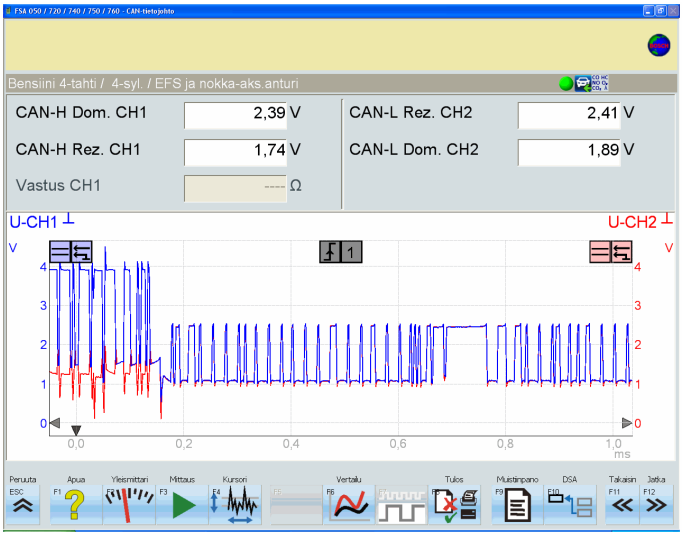


Kuva 15. Volvo XC90:n CAN High Speed -väylän viesti ja jännitetasot.

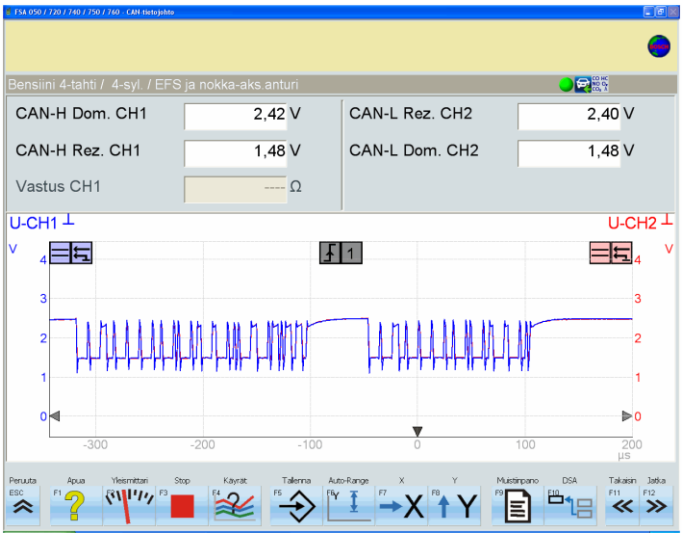


Kuva 16. Simulaattorin CAN High Speed -väylän viestikehys ja jännitetasot.

Laitteen realistisuuden kannalta oli myös tärkeää, että vikatilatkin näyttävät oskilloskoopilla mitattuina samoilta kuin oikeassa ajoneuvossa. Mittaustulokset osoittivat laitteen väylien käyttäytyvän realistisesti (kuvat 17 ja 18). Ajoneuvon CAN High Speed-väylän ”ehjällä” puolella olevat laitteet aiheuttavat kuvan alussa näkyvän korkean jännitetason osuuden. Tätä ei kuitenkaan voida nähdä CAN-väyläsimulaattorin väylän vikatilassa, koska väylään lähettäviä laitteita on yksi ja sen takia heijastumista ei tapahdu.



Kuva 17. Ajoneuvon CAN High Speed -väylän jännitetasot H-johtimen ollessa poikki.



Kuva 18. Laitteen CAN High Speed -väylän jännitetasot H-johtimen ollessa poikki.

5.2 Kommunikointi ajoneuvon väylän kanssa

Väyläsimulaattorilla pystyttiin myös kytkeytymään ajoneuvon CAN-väylään ja tätä kautta esimerkiksi ohjaamaan keskuslukitusta, ikkunoiden moottoreita, sekä tarkkailemaan eri tilatietoja. Tämä kuvaa hyvin laiteen viestien ja väylien toiminnan realistisuutta. Ajoneuvon väylään kytkeytyminen vaati kuitenkin eri ohjelmakoodin ajamisen kehitysalustaan, ja näin ollen tätä ominaisuutta ei itse lopputuotteessa ole ollenkaan.

Väylää kuunneltiin ensin yhdellä ohjelmakoodilla, jolla saatiin selville eri ohjainlaitteiden tunnistekentät sekä viestin sisältämä data. Viestikehyksen talteenoton jälkeen kehitysalustaan ajettiin ohjelmakoodi, jolla väylälle pystyttiin lähettämään valitulla tunnistekentällä haluttua toimintoa vastaava datakenttä.

Havaitut ongelmat

Lähetettäessä useampaa viestiä kerrallaan ajoneuvon väylään ilman viivettä viestien välillä ajoneuvon toiminta oli erittäin satunnaista. Lähetettyjä viestejä alettiin tarkastella siten, että väyläsimulaattorin toinen kehitysalusta lähetti neljä kappaletta viestejä toiselle kehitysalustalle, ja huomattiin, että viestien välissä pitää olla lyhyt viive. Lyhyt viive piti lisätä, jotta lähetin-vastaanotin ja CAN-ohjain ehtivät lähettää viestin ilman priorisointitarkastelua. Toisin sanoen CAN-ohjain yritti lähettää kaikkia neljää viestiä samaan

aikaan, ja tästä syystä pienimmän tunnistekentän sisältämä viesti pääsi väylälle ensin, sotkien viestien alkuperäisen lähetysjärjestyksen.

6 Koulutuspaketti

Työn tilaajan toiveesta väyläsimulaattorin tueksi suunniteltiin myös koulutuspaketti. Materiaali laadittiin aiempien kurssien materiaalin, tämän projektin teoriaosuuden sekä suoritettujen CAN-väylä mittausten pohjalta. Koulutuspaketti on esitetty liitteessä 4 (vain työntilaajan käyttöön).

6.1 Teoriaosuus

Koulutuspakettia varten laadittu teoriaosuus on suunniteltu pitäen mielessä koulutettavien asentajien eri lähtötasot. Työn tilaaja toivoi, että teoriaosuudessa pysyttäisiin mahdollisimman käytännönläheisissä asioissa. Näistä syistä teoriaosuus on suhteellisen niukka eikä kuvaa esimerkiksi viestikehystä kovinkaan tarkasti, sillä normaalissa korjaamotyöskentelyssä ei viestikehyksestä tarvitse olla niin syvällistä käsitystä.

6.2 Tehtävät vikatiloista

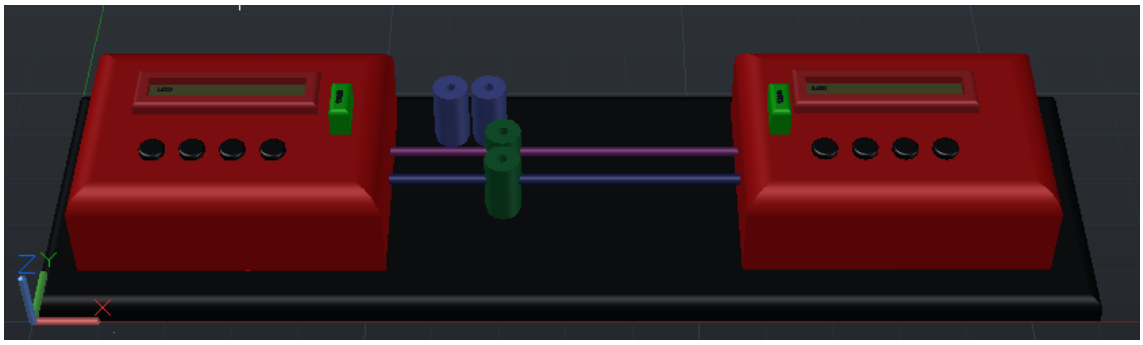
Vikatilojen mallintaminen noudattaa suunnilleen samaa kaavaa kuin mitatun ajoneuvon yhteneväisyysmittauksissa käytettiin. Väyläsimulaattorilla tehtävistä vikatiloista asentajat piirtävät tehtäväpaperille oskilloskooppikuvaa vastaavat jännitekuvaajat värikynillä. Jännitekuvaajien piirtämisen lisäksi tehtäviin laadittiin joitakin käytännön kysymyksiä mahdollisesti vastaantulevista vikatilanteista ja siitä, miten ne vaikuttavat ajoneuvon käyttäytymiseen.

7 Yhteenveto

Projektin tarkoituksena oli tuottaa Robert Bosch Oy:lle CAN-väylän vikaantumisia simuloiva laite ja sen tueksi laatia koulutusmateriaali. Projekti saatiin kokonaisuudessaan valmiiksi ja kaikki tilaajan vaatimukset täyttyivät. Projektin tilaajan tarkoituksena on tilata projektin tekijöiltä neljä kappaletta valmiita laitteita. Näiden laitteiden lisäksi myös

Metropolian ajoneuvotekniikan tutkinto-ohjelma on kiinnostunut tilamaan yhden kappaleen, opetuskäyttöön.

Työssä jouduttiin paneutumaan CAN-väylätekniikkaan niin kirjallisuuden kuin mittausten avulla. Ensimmäisenä projektin tekijät toteuttivat yksinkertaisen CAN -väylän lähetin-vastaanotinpiiriä mallintavan elektroniikkapiirin, joka koostui operaatiovahvistimesta, kahdesta diodista, kuudesta vastuksesta ja kolmesta bipolaaritransistorista. Rakennetun prototyypin tarkoituksena oli selventää projektin tekijöille CAN-väylän toimintaperiaate yksinkertaisimmillaan. Ensimmäisen prototyypin jälkeen suunniteltiin Arduino-kehitysalustan prototyyppikortille CAN High Speed -väylä ja lasketulla nopeudella toimiva CAN Low Speed -väylä. Laitteen mahdollisesta lopputuloksesta piirrettiin myös työn tilaajalle 3D-mallinnos, jotta työn tilaajalle tulisi mahdollisimman tarkka kuva lopputuotteesta (kuva 19). Lisää kirjallisuutta tutkimalla päätettiin protokortille rakentaa myös CAN Fault Tolerant -väylä. Prototyypin valmistuttua tilattiin suunnitellut piirikortit ITEAD-studiolta.



Kuva 19. Mahdollisen lopputuotteen 3D-mallinnos.

Tekijöiden aikaisemmissa projekteissa on huomattu ohjelmoinnin vievän suurimman osan projektiin käytetystä ajasta, joten tärkeänä pidettiin, että ohjelmointi päästäisiin aloittamaan mahdollisimman aikaisessa vaiheessa. Molemmille kehitysalustoille kirjoitettiin omat ohjelmakoodit. Lähettävän kehitysalustan mikrokontrollerille kertyi runsas määrä ohjelmakoodia, koska kehitysalusta pyörittää valikkoa, LCD-näyttöä ja huolehtii CAN-viestien lähettämisestä väylään. Ohjelmointityötä helpotti Arduino käyttäjäyhteisön laatimat kirjastot ja sen projektit. Tälläkin kertaa ohjelmointiin käytettiin yli puolet projektille varatusta ajasta.

Prototyypin valmistuttua laite esiteltiin projektin tilaajalle, joka antoi vielä muutaman pienen kehitysehdotuksen. Työn tilaajan toivomat kehityskohteet toteutettiin nopeassa aikataulussa ja laite sai hyväksynnän (kuva 20).

Kuva 20. Valmis laite.

CAN-väyläsimulaattori rakennettiin, jotta mekaanikkojen täydennyskoulutusta varten saadaan luokkatiloihin soveltuva laite. Tällä hetkellä käytössä olevat laitteet ovat suurikokoisia ja niitä ei ole tarkoituskaan kuljettaa mukana. Tämän lisäksi ne maksavat yleensä tuhansia euroja. Projektia varten tehdyn CAN-väyläsimulaattorin osalta tämä ei tuota ongelmaa, sillä se kulkee mukana salkussa ja sen hinta on huomattavasti halvempi verrattuna vastaaviin.

Lähteet

Alanen, Jarmo. 2000. CAN-ajoneuvojen ja koneiden sisäinen paikallisväylä. Verkkodokumentti. VVT AUTOMAATIO. <http://www.oamk.fi/~eeroko/Opetus/Ohjausjarjestelmat/CAN/CAN-perusteet_AlasenMateriaalia.pdf>. Luettu 20.3.2016.

Arduino Introduction. Verkkojulkaisu. <https://www.arduino.cc/en/Guide/Introduction>. Luettu 11.2.2016

CAN bus. 2003. Muokattu 29.5.2016. Verkkodokumentti. Wikipedia. <https://en.wikipedia.org/wiki/CAN_bus>. Luettu 1.5.2016.

CD4066BE, CMOS QUAD BILATERAL SWITCH. 1998. Muokattu 2003. Verkkodokumentti. Texas Instruments Inc. <http://www.ti.com/lit/ds/symlink/cd4066b.pdf>. Luettu 15.3.2016.

Frei, Martin. 2015. Verkotettujen järjestelmien vikadiagnoosi. 3. laajennettu painos. Helsinki: Autoalan koulutuskeskus.

Griffith, John. 2013. The ISO 11898 CAN Standard. Verkkodokumentti. Texas Instruments Inc. <https://e2e.ti.com/support/interface/industrial_interface/w/industrial_interface/2613.the-iso-11898-can-standard>. Luettu 25.3.2016.

High Speed CAN Tranceiver. 2012. Verkkodokumentti. Microchip Technology Inc. <http://ww1.microchip.com/downloads/en/DeviceDoc/21667f.pdf>. Luettu 20.2.2016.

Jämsä, Lauri 2011. Alle euron piirilevyt. Verkkodokumentti. < http://www.ruuvipenkki.fi/2011/05/26/alle-euron-piirilevyt >. Luettu 25.2.2016.

Landoni, Boris. 2015. From Arduino to Genuino, the reason for a choice. Verkkodokumentti. Open Electronics. <http://www.open-electronics.org/from-arduino-to-genuino-the-reasons-for-a-choice/>. Luettu 8.4.2016.

Malmari, Frans. 2016. Ajoneuvojen verkkotekniikka. Luentokalvot. AEL.

Mischo, S., Powolny, S., Zündel, H., Löchel, N., Stuphorn, J., Constapel, R., Häußermann, P., Leonhardi, A & Holtkamp, H. 2008. Ajoneuvojen verkottuminen. Helsinki: Autoalan koulutuskeskus.

Paret, Dominique. 2007. Multiplexed Networks for Embedded Systems. West Sussex: John Wiley & sons, Ltd.

Richards, Pat. 2002. A CAN Physical Layer Discussion. Verkkodokumentti. Microchip Technology Inc. <http://ww1.microchip.com/downloads/en/AppNotes/00228a.pdf>

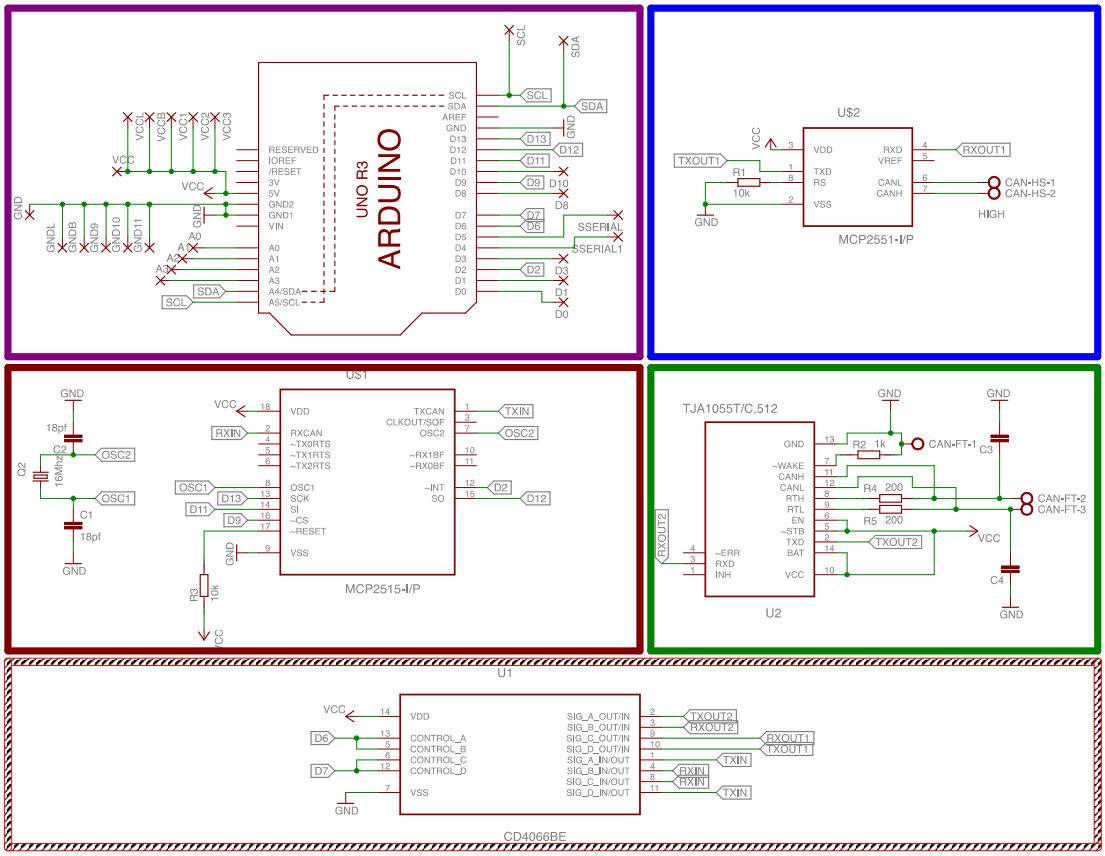
Schade, Frank & Muth, Matthias. 2016. TJA1055T – Fault-tolerant CAN transceiver. Verkkodokumentti. NXP Semiconductors. <http://www.nxp.com/documents/application_note/AH0801.pdf>. Luettu 1.4.2016.

Stand-Alone CAN Controller with SPI Interface. 2012. Verkkodokumentti. Microchip Technology Inc.<http://ww1.microchip.com/downloads/en/DeviceDoc/21801G.pdf>. Luettu 20.2.2016.

TJA1055, Enhanced fault-tolerant CAN transceiver. 2013. Verkkodokumentti. NPX Semiconductors. <http://www.nxp.com/documents/data_sheet/TJA1055.pdf>. Luettu 24.02.2016.

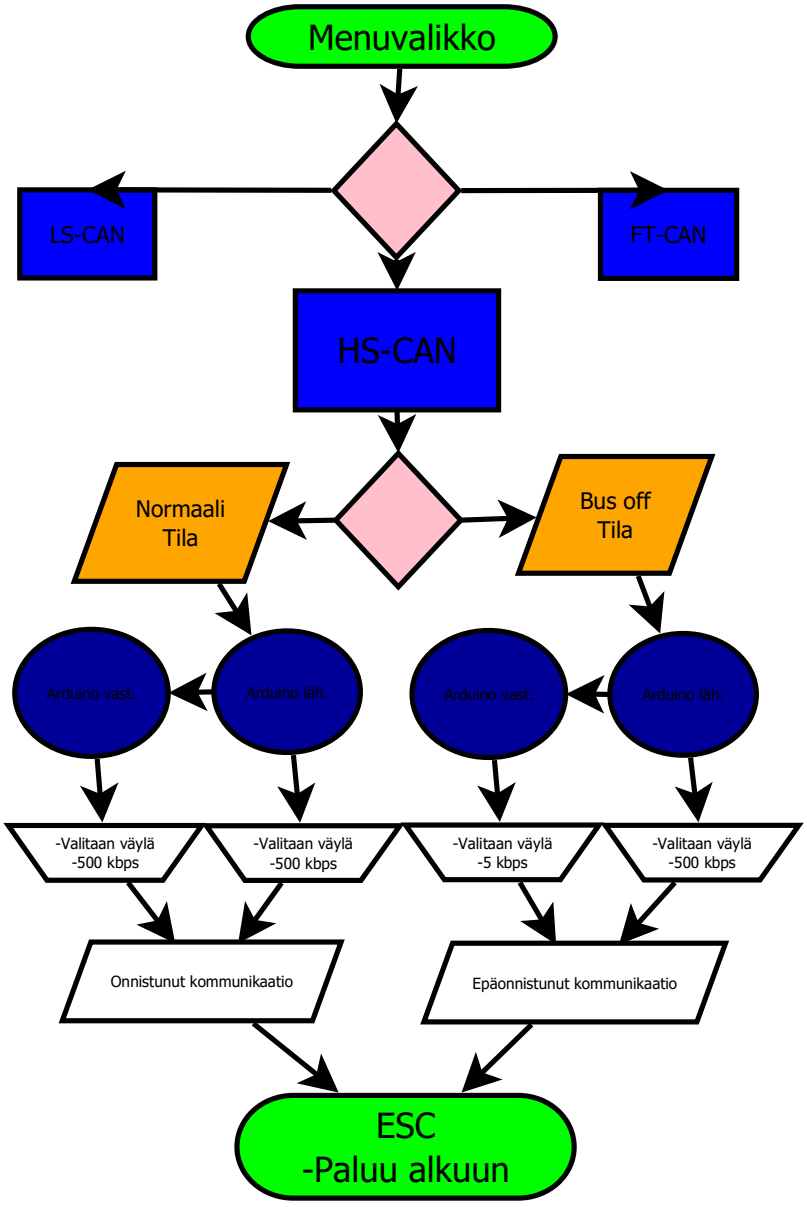
Piirikaavio

Yhden kehitysalustan piirikortin piirikaavio, jossa eri komponenttikokonaisuudet esitetty eri värisillä rajauksilla. Piirikortti on esitetty violetilla, CAN-ohjain ruskealla, CAN High Speed lähetin-vastaanotinpiiri sinisellä, CAN Fault Tolerant lähetin-vastaanotinpiiri vihreällä, sekä digitaalikytkin punavalkoisella rajauksella.



Toimintakaavio

Laitteen toimintakaavio esitettynä yleisellä tasolla, yhden fyysisen toteutuksen osalta.



Mittaustulokset

VOLVO

HS-CAN

1. **Normaalitilanne**
2. **CAN H -johdin poikki**
3. **CAN L -johdin poikki**
4. **CAN L -johdin maadoitettu 20 ohmin vastuksen kautta**
5. **CAN H -johdin maadoitettu 20 ohmin vastuksen kautta**
6. **CAN H- ja CAN L -johtimet yhdessä**

CAN-väyläsimulaattori

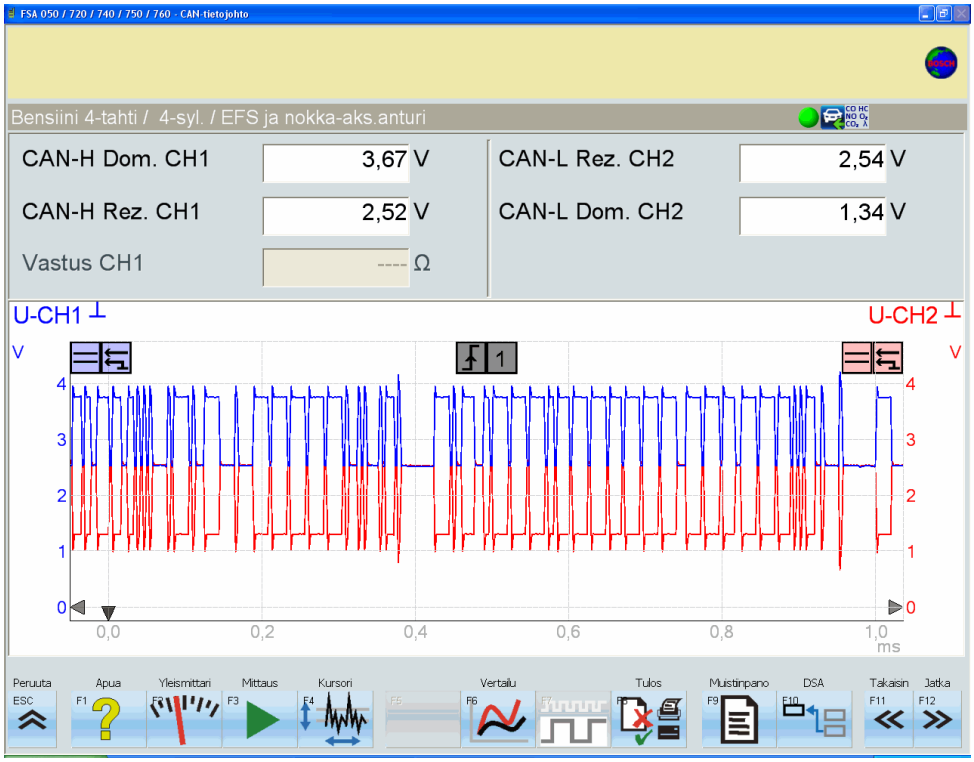
HS-CAN

7. **Normaalitilanne**
8. **CAN H -johdin poikki**
9. **CAN L -johdin poikki**
10. **CAN L -johdin maadoitettu 20 ohmin vastuksen kautta**
11. **CAN H -johdin maadoitettu 20 ohmin vastuksen kautta**
12. **CAN H- ja CAN L -johtimet yhdessä**
13. **Yksi terminointivastus irrotettuna**
14. **Ilman terminointivastuksia**

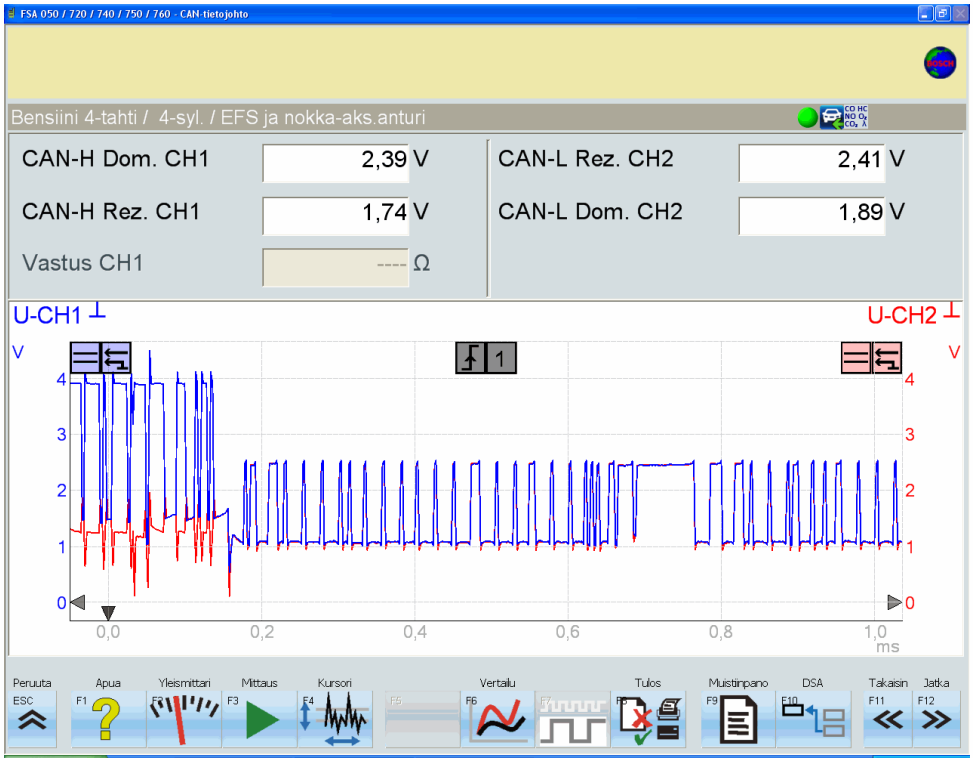
FT-CAN

15. **Normaalitilanne**
16. **CAN H -johdin poikki**
17. **CAN L -johdin poikki**
18. **CAN L -johdin maadoitettu 20 ohmin vastuksen kautta**
19. **CAN H -johdin maadoitettu 20 ohmin vastuksen kautta**
20. **CAN H- ja CAN L -johtimet yhdessä**

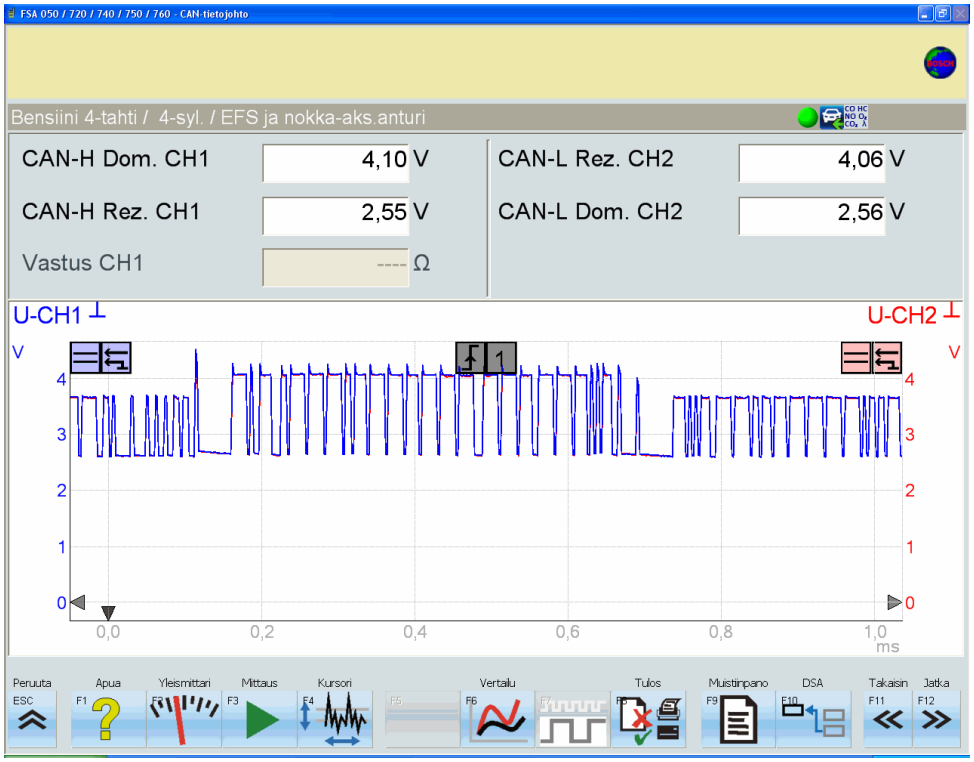
Mittaus 1. Normaalitilanne



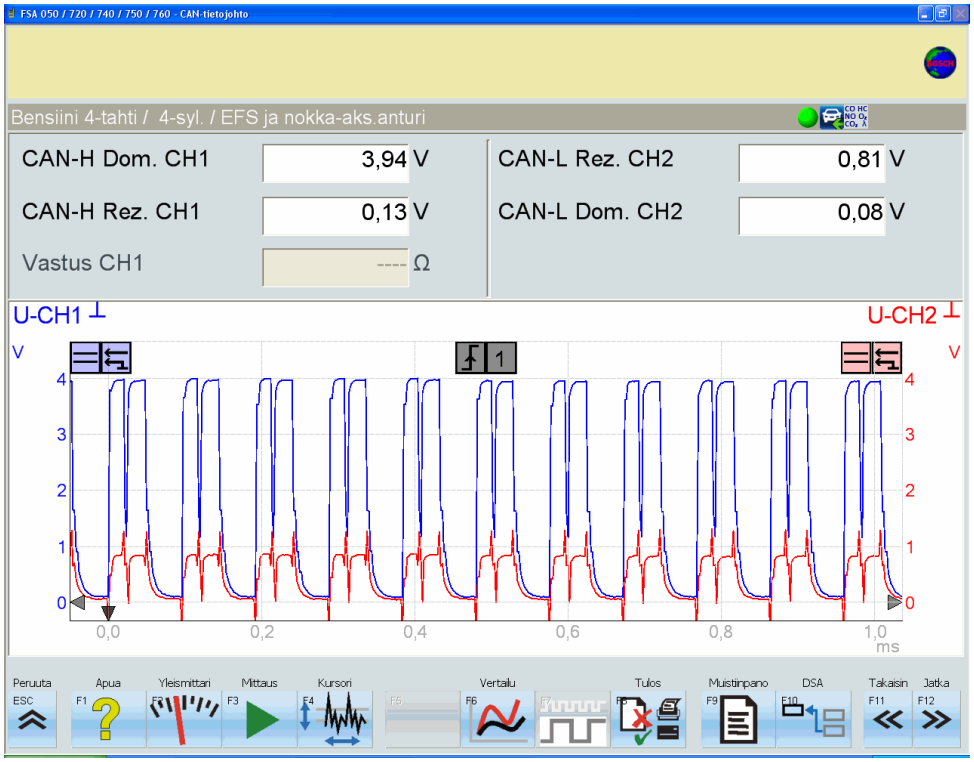
Mittaus 2. CAN H -johdin poikki



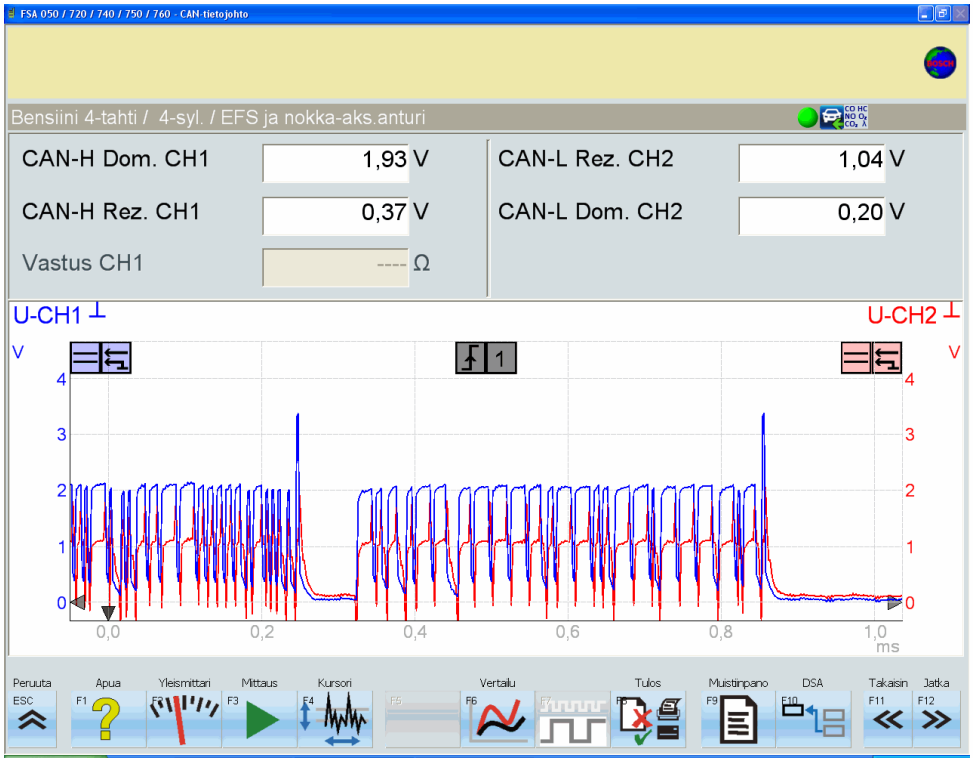
Mittaus 3. CAN L -johdin poikki



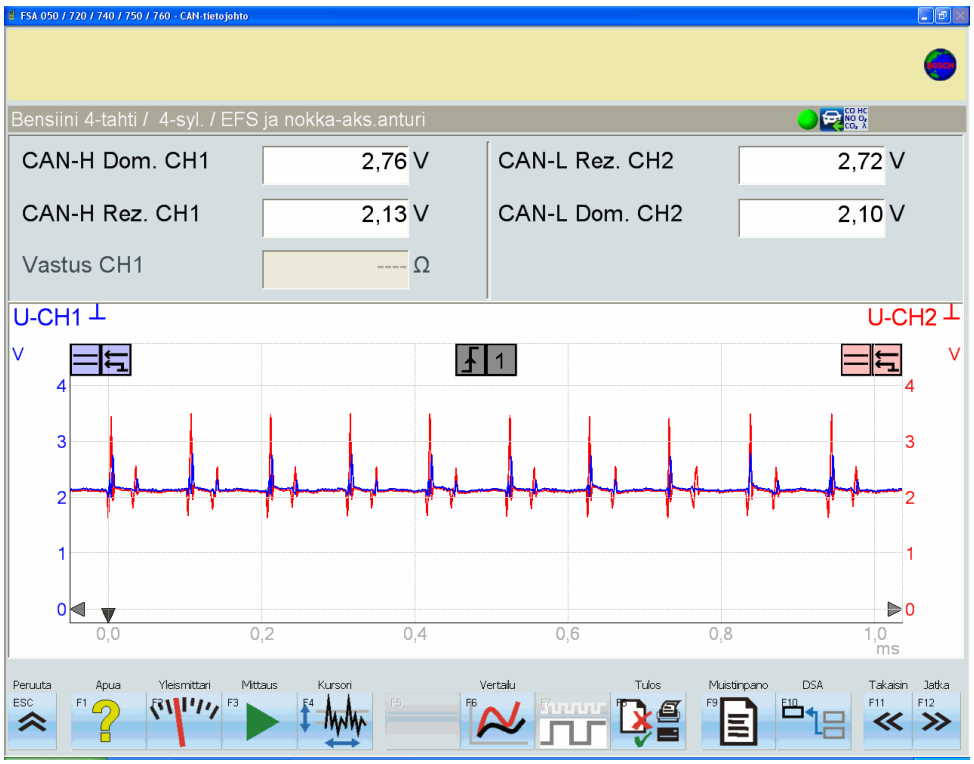
Mittaus 4. CAN L -johdin maadoitettu 20 ohmin vastuksen kautta



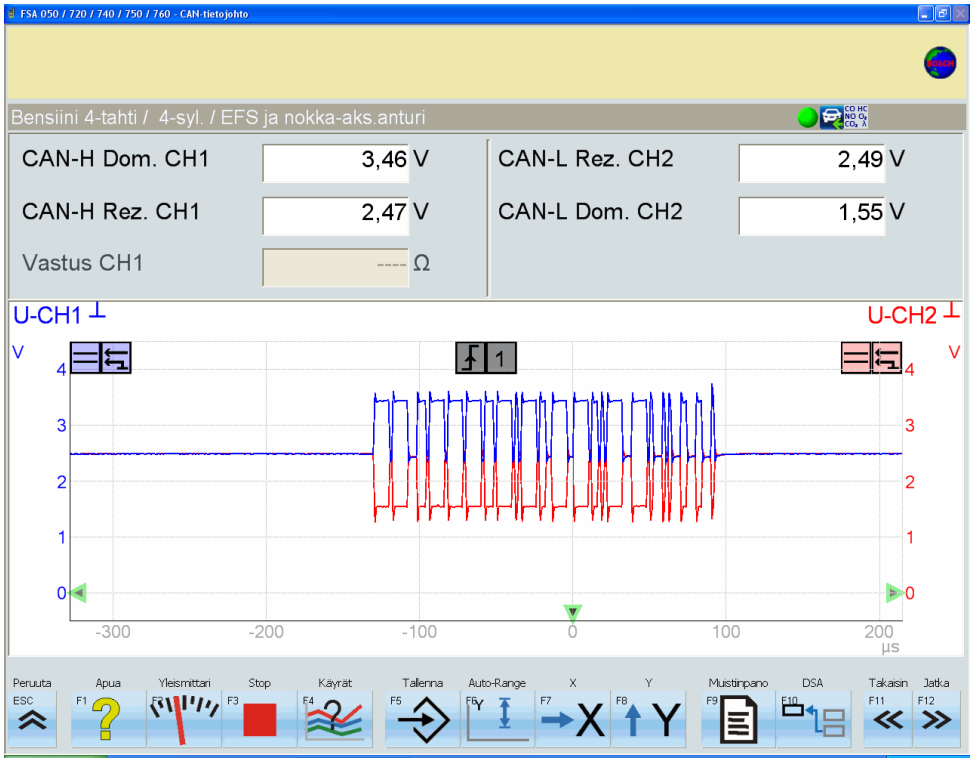
Mittaus 5. CAN H -johdin maadoitettu 20 ohmin vastuksen kautta



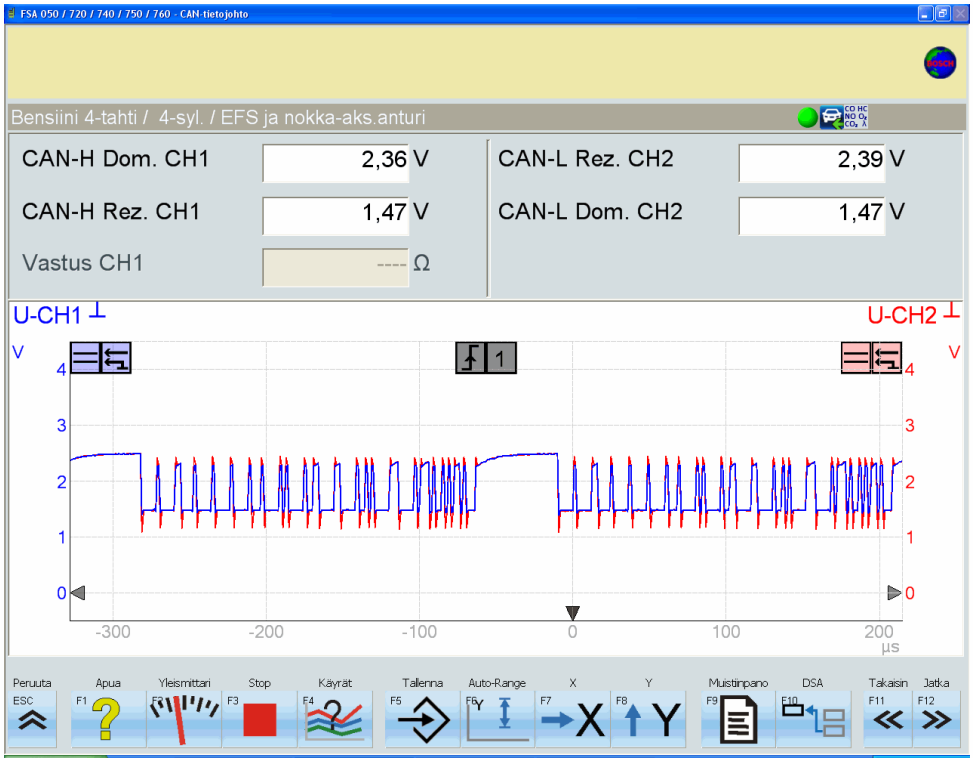
Mittaus 6. CAN H- ja CAN L -johtimet yhdessä



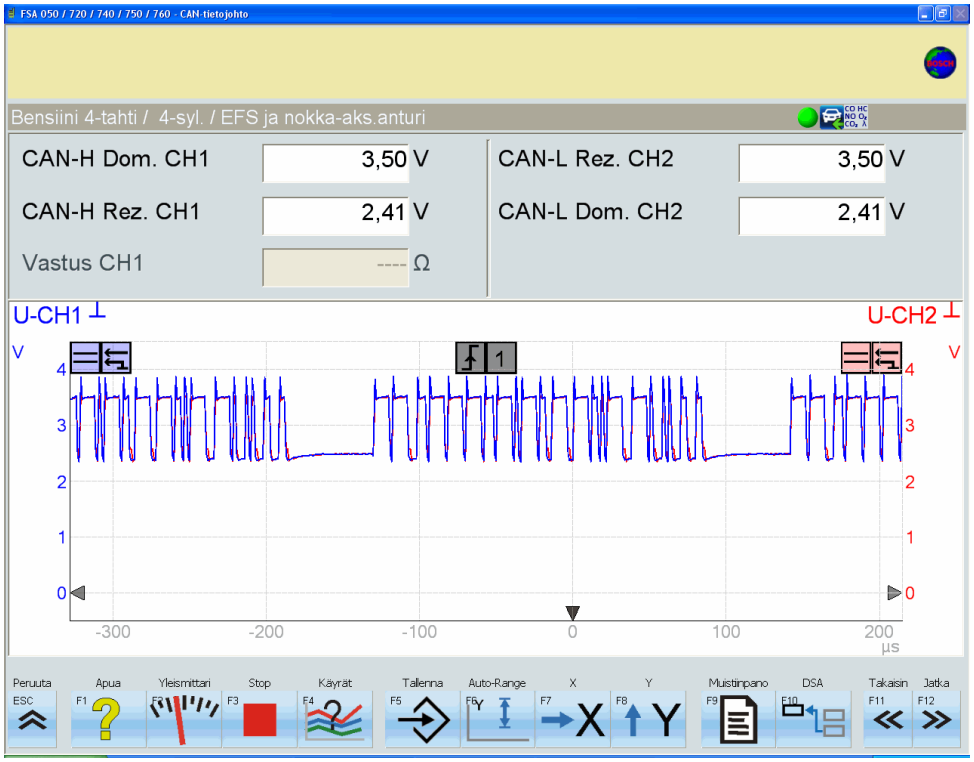
Mittaus 7. Normaalitilanne



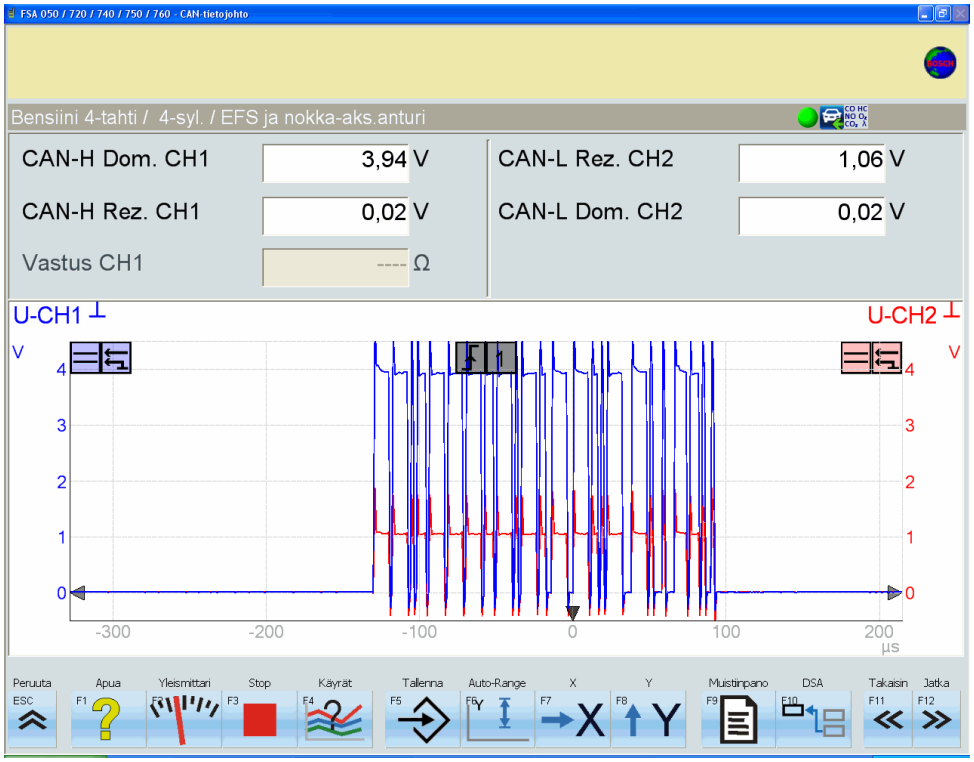
Mittaus 8. CAN H -johdin poikki



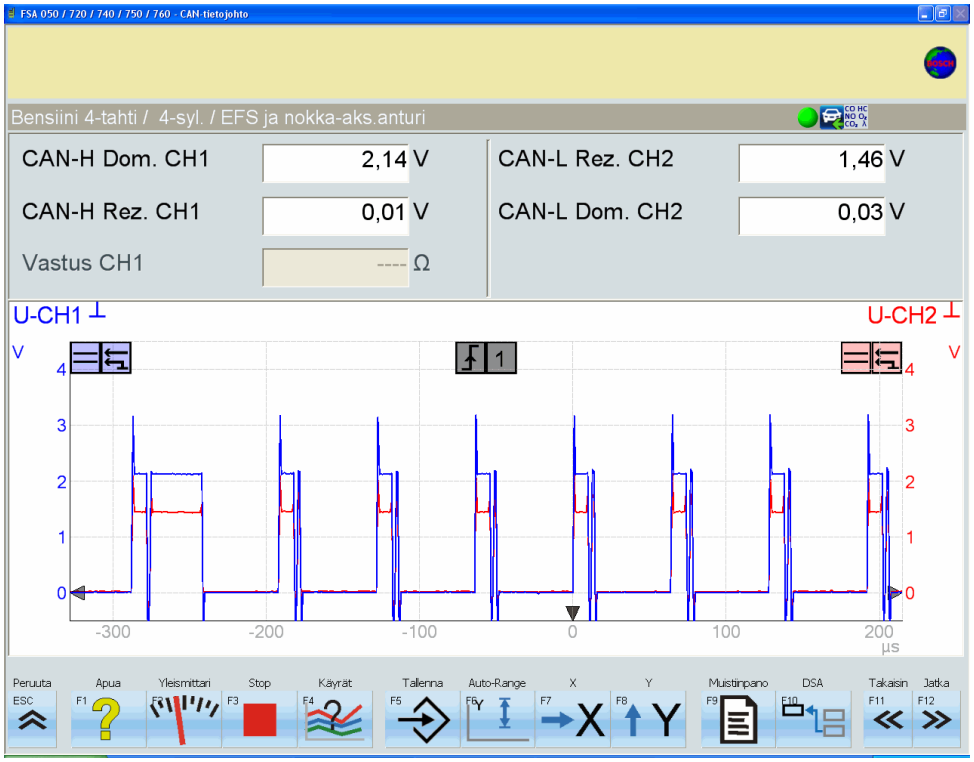
Mittaus 9. CAN L -johdin poikki



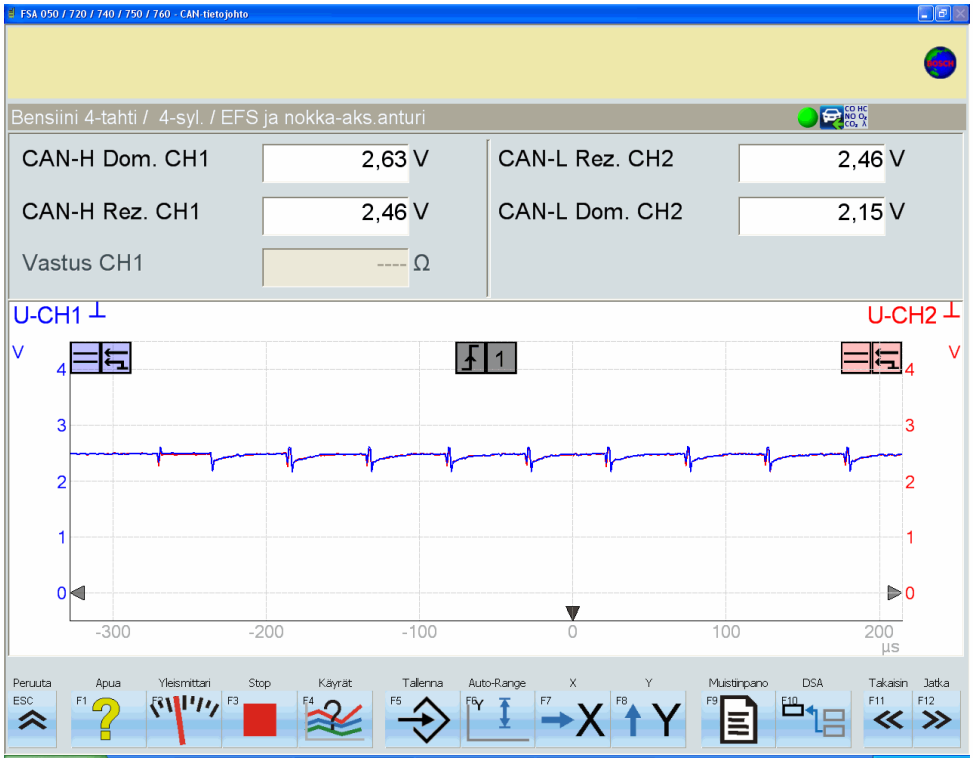
Mittaus 10. CAN L -johdin maadoitettu 20 ohmin vastuksen kautta



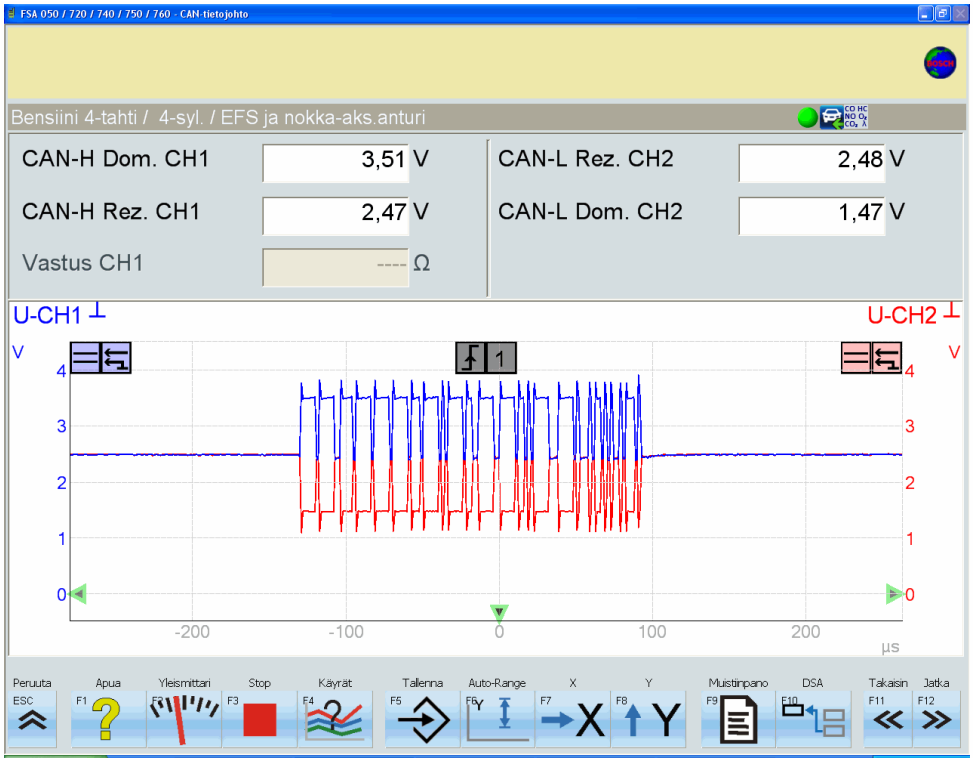
Mittaus 11. CAN H -johdin maadoitettu 20 ohmin vastuksen kautta



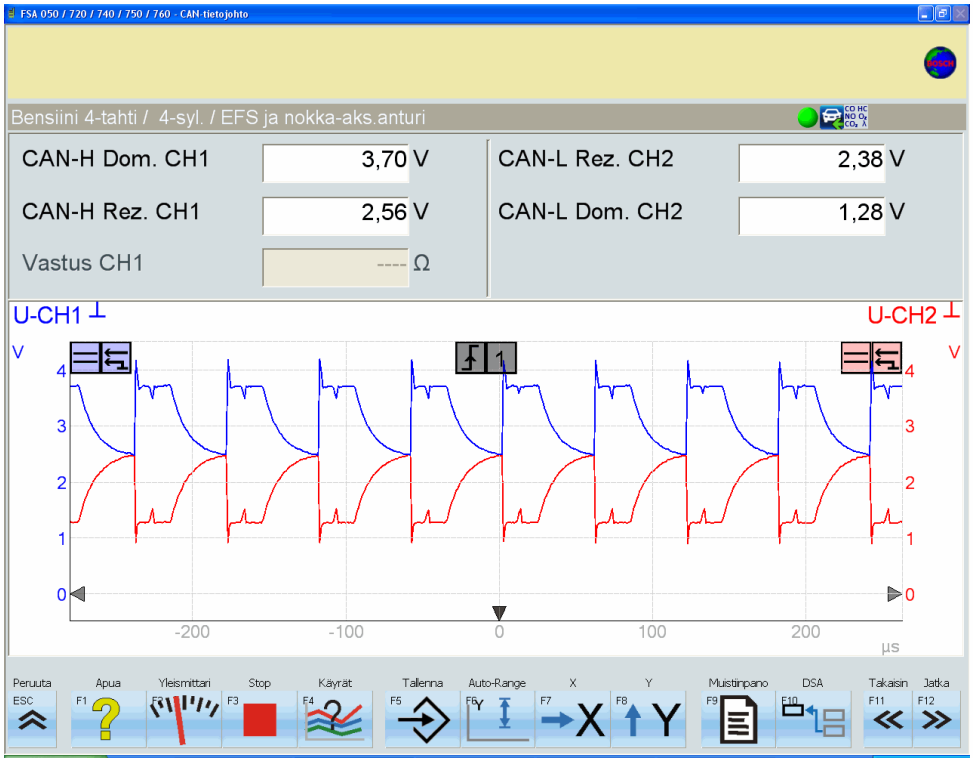
Mittaus 12. CAN H- ja CAN L -johtimet yhdessä



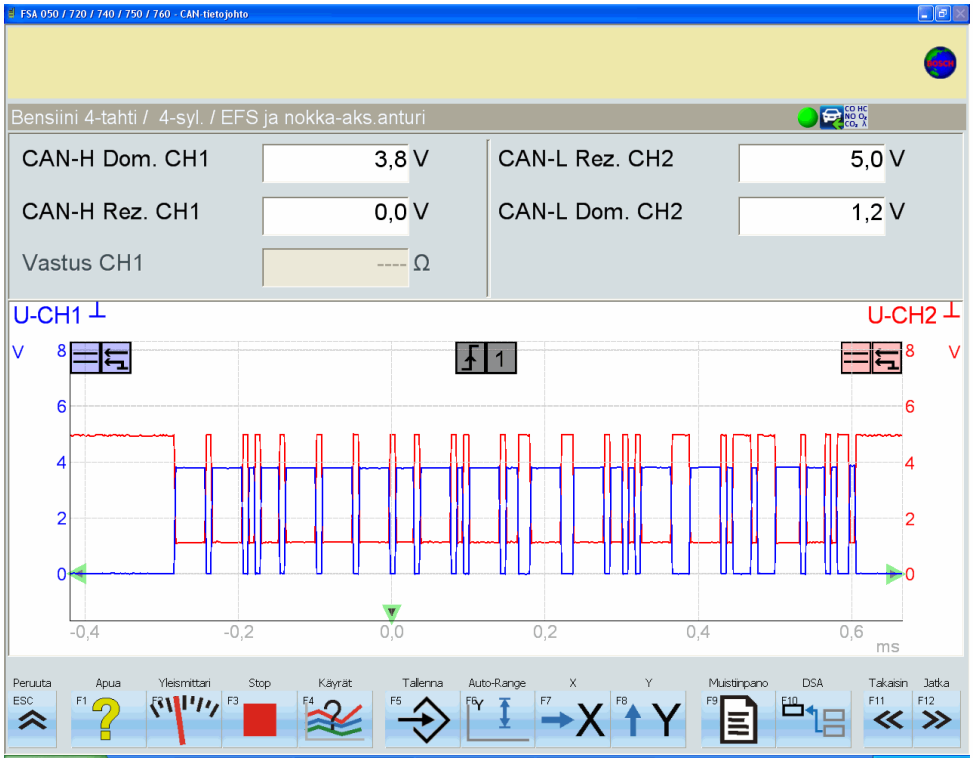
Mittaus 13. Yksi terminointivastus irrotettuna



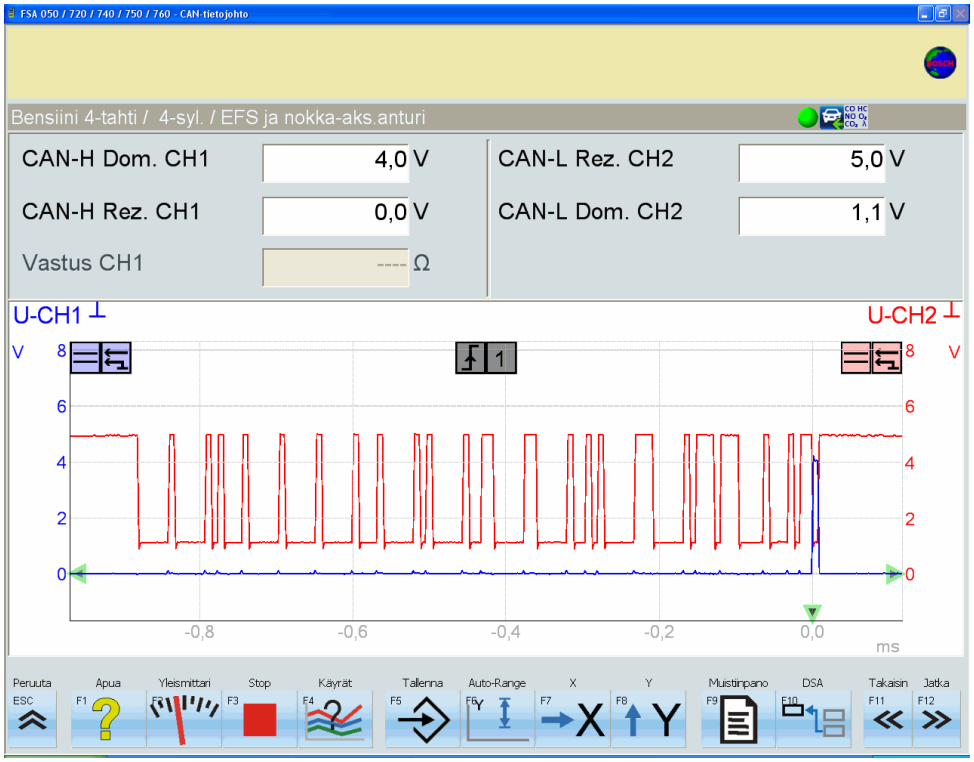
Mittaus 14. Ilman terminointivastuksia



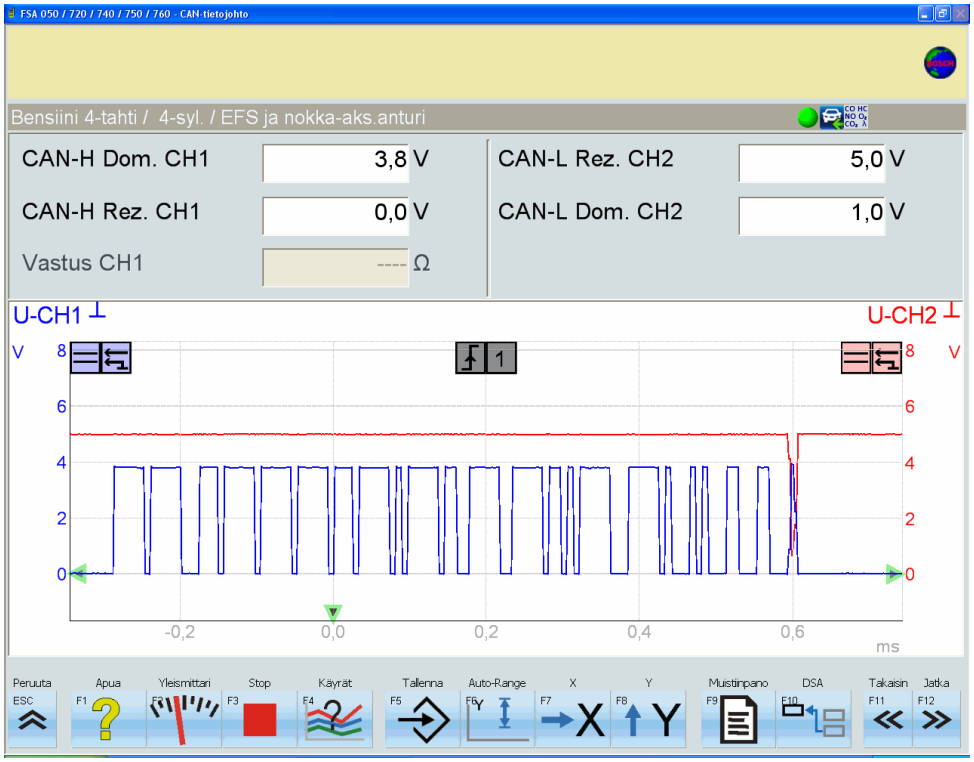
Mittaus 15. Normaalitilanne



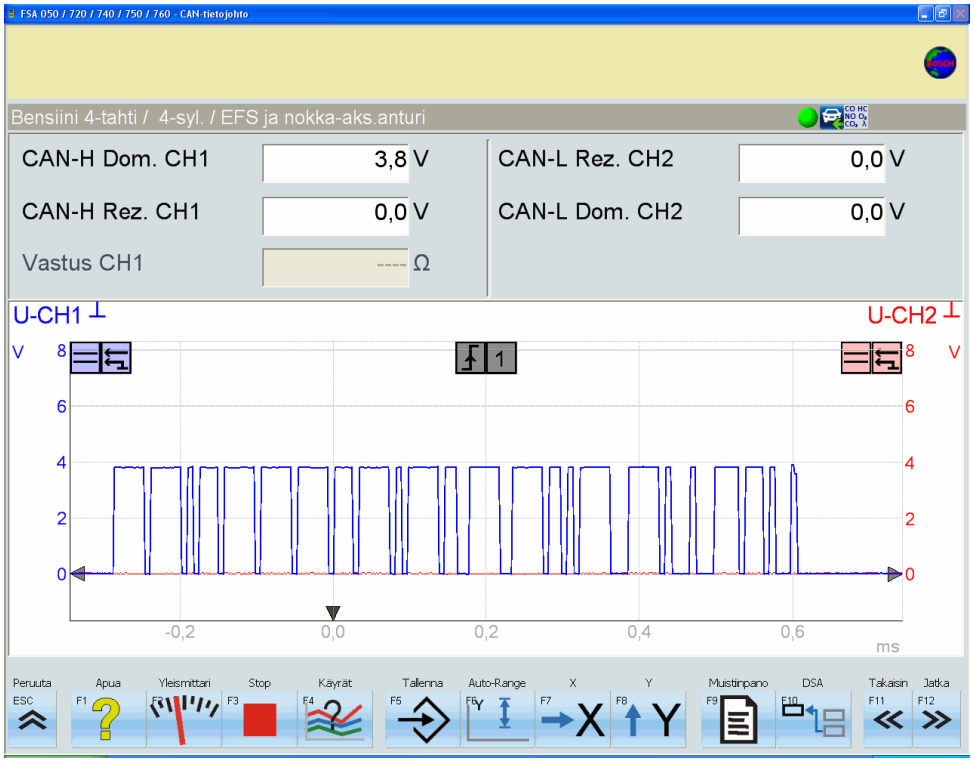
Mittaus 16. CAN H -johdin poikki



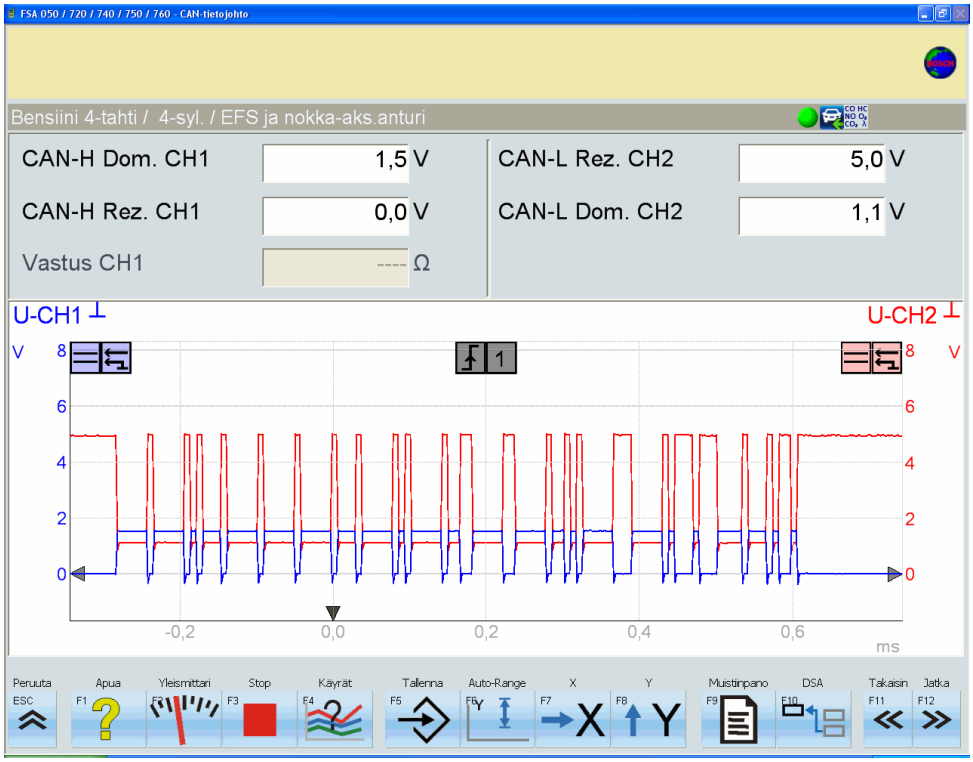
Mittaus 17. CAN L -johdin poikki



Mittaus 18. CAN L -johdin maadoitettu 20 ohmin vastuksen kautta



Mittaus 19. CAN H -johdin maadoitettu 20 ohmin vastuksen kautta



Mittaus 20. CAN H- ja CAN L -johtimet yhdessä